# Active-set Newton-MR methods for nonconvex optimization problems with bound constraints*

Ernesto G. Birgin† Geovani N. Grapiglia‡ Diaulas S. Marcondes§

May 28, 2025

**Abstract**

This paper presents active-set methods for minimizing nonconvex, twice continuously differentiable functions subject to bound constraints. Within the faces of the feasible set, we employ descent methods with Armijo line search, utilizing approximated Newton directions obtained through the Minimum Residual (MINRES) method. To escape the faces, we investigate the use of the Spectral Projected Gradient (SPG) method and a tailored variant of the Cubic Regularization of Newton's method for bound-constrained problems. We provide theoretical guarantees, demonstrating that when the objective function has a Lipschitz continuous gradient, the SPG-based method requires no more than $\mathcal{O}(n\epsilon^{-2})$ oracle calls to find $\epsilon$-approximate stationary points. Furthermore, if the objective function also has a Lipschitz continuous Hessian, we show that the method based on cubic regularization requires no more than $\mathcal{O}\left(n|\log_2(\epsilon)|\epsilon^{-3/2}\right)$ oracle calls to achieve the same goal. Numerical experiments are conducted to compare the proposed methods with existing active-set methods, highlighting the potential benefits of using MINRES instead of the Conjugate Gradient (CG) method for approximating Newton directions.

## 1 Introduction

The optimization problem entails finding a solution that satisfies given constraints while minimizing a specified objective function. This type of problem has far-reaching applications in various fields, including psychology, medicine, economics, engineering, physics, and biology, to name a few. In this work, we focus on cases where the functions that define the feasible region and the objective function are continuous and differentiable. Among existing methods for addressing such problems, augmented Lagrangian methods warrant special attention. Consider a simple problem with no constraints or straightforward constraints, for which we have an efficient solution method. Now, suppose that we want to introduce an additional constraint, rendering our initial method inapplicable. A natural approach would be to transform this new constraint into a penalty that increases the value of the objective function when the constraint is violated. This concept gives rise to penalty methods [20],

*This work has been partially supported by the Brazilian agencies FAPESP (grants 2013/07375-0, 2022/05803-3, 2023/08706-1, and 2024/22384-0) and CNPq (grant 302073/2022-1).

†Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil. e-mail: egbirgin@ime.usp.br

‡Université Catholique de Louvain, ICTEAM/INMA, Avenue Georges Lemaître, 4-6/L4.05.01, B-1348, Louvain-la-Neuve, Belgium. e-mail: geovani.grapiglia@uclouvain.be

§Department of Applied Mathematics, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil. e-mail: diaulas@ime.usp.br

which involve solving a sequence of subproblems that minimize the penalized function subject to simple constraints. We could utilize our initial efficient method to solve these subproblems. However, penalty methods rely on a penalty parameter that tends to increase with each iteration. If this parameter becomes too large, the subproblems become ill-conditioned and difficult [40, Ch.17]. To address this issue, augmented Lagrangian methods [42, 30, 43] penalize shifted constraints [7, Ch.4], aiming to achieve the same results as penalty methods but with better-conditioned subproblems.

The authors of this paper are dedicated to developing and implementing robust and efficient augmented Lagrangian methods, encompassing both theoretical advancements and the provision of free software. Recent studies [34, 35, 36] have highlighted the theoretical properties and practical performance of Newtonian methods for unconstrained minimization, which leverage the minimal residual method (MINRES) [41] to solve linear Newtonian systems. These methods utilize the Hessian matrix of the objective function to compute Hessian vector products without performing matrix factorizations. In practice, large problems with dense Hessians are scarce, as they typically exhibit some structure. However, subproblems in augmented Lagrangian methods may have dense Hessians or Hessians with dense factorizations, as the Hessian of an augmented Lagrangian function combines the Hessian of the objective function and the Jacobians and Hessians of the constraints. In this context, it is intriguing to explore the method proposed in [36] for solving augmented Lagrangian subproblems. Specifically, our objective is to evaluate this method for solving subproblems in Algencan [1, 2, 7, 9], a particular implementation of augmented Lagrangian methods. Since Algencan's subproblems involve bound constraints, our initial objective is to extend the method proposed in [36] to accommodate minimization with bound constraints. This objective is the main focus of the present work.

The development of methods for minimization with bound constraints is a vibrant and dynamic field. During the past three decades, numerous methods have been devised [3, 5, 6, 14, 16, 17, 18, 22, 23, 24, 25, 28, 29, 31, 33, 39, 45, 38]. A comprehensive numerical comparison of freely available software packages was presented in [4].

The extension proposed here builds on the methods introduced in [36] and follows the active set paradigm. Specifically, we propose two distinct extensions of the Newton-MR method from [36] for minimization with bound constraints. In [36], the method achieves a worst-case complexity of $\mathcal{O}(\epsilon^{-3/2})$ to find a point $x$ such that $\|\nabla f(x)\| \leq \epsilon$ when minimizing $f(x)$ over $x \in \mathbb{R}^n$. Our first proposed extension, which introduces greater flexibility in algorithmic choices, takes no more than $\mathcal{O}(n\epsilon^{-2})$ calls to the oracle to find a point $x$ such that $\|P_\Omega(x - \nabla f(x)) - x\| \leq \epsilon$ when applied to the problem of minimizing $f(x)$ subject to $x \in \Omega$, where $\Omega = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\}$ with $\ell, u \in \mathbb{R}^n$. The second extension takes no more than $\mathcal{O}(n|\log_2(\epsilon)|\epsilon^{-3/2})$ calls to the oracle to achieve the same goal. Numerical experiments demonstrate that the first option is more robust and efficient compared to the second, using both unconstrained and bound-constrained minimization problems from the CUTEst collection [27]. Additional experiments also show that the first version outperforms the method currently used by Algencan to solve augmented Lagrangian subproblems, particularly when second derivatives are available but matrix factorizations are not.

The remainder of this paper is organized as follows. The two Newton-MR-based methods for bound-constrained minimization and their theoretical properties are described in Section 2. Numerical experiments are presented and analyzed in Section 3. Conclusions and directions for future work are presented in the final section.

## 2 Problem definition and new methods

Consider the problem

$$\text{Minimize } f(x) \text{ subject to } x \in \Omega, \tag{1}$$

where $\Omega = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$, with $\ell_i < u_i$ for $i = 1, \ldots, n$, and $f : \mathbb{R}^n \to \mathbb{R}$ being a twice continuously differentiable function. Specifically, let us assume that:

**A1.** $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is $L_g$-Lipschitz continuous.

**A2.** $f(\,\cdot\,)$ is bounded from below by $f_{\text{low}}$.

Given $\boxed{x \in \Omega}$, let us define the following subsets of $\{1, \ldots, n\}$:

$$\begin{aligned}
\mathcal{A}_0(x) &= \{j \in \{1, \ldots, n\} : x_j = \ell_j\}, \\
\mathcal{A}_1(x) &= \{j \in \{1, \ldots, n\} : x_j = u_j\}, \\
\mathcal{A}(x) &= \mathcal{A}_0(x) \cup \mathcal{A}_1(x), \\
\mathcal{I}(x) &= \{1, \ldots, n\} \setminus \mathcal{A}(x).
\end{aligned}$$

The *face* to which $x \in \Omega$ belongs is the set

$$\mathcal{F}(x) = \{z \in \Omega : z_i = \ell_i \text{ if } i \in \mathcal{A}_0(x),\ z_i = u_i \text{ if } i \in \mathcal{A}_1(x),\ \ell_i < z_i < u_i \text{ otherwise}\}.$$

The reduced gradient of $f$ at $x$ with respect to $\Omega$ is defined by

$$\nabla_\Omega f(x) = x - P_\Omega(x - \nabla f(x)),$$

where $P_\Omega(\,\cdot\,)$ is the projection operator onto $\Omega$. Let $\nabla^I_\Omega f(x) \in \mathbb{R}^n$ be the vector defined by

$$\left[\nabla^I_\Omega f(x)\right]_i = \begin{cases} [\nabla_\Omega f(x)]_i, & \text{if } i \in \mathcal{I}(x), \\ 0, & \text{otherwise.} \end{cases}$$

Considering the enumeration $\mathcal{I}(x) = \left\{i_1, \ldots, i_{|\mathcal{I}(x)|}\right\}$, with $i_j < i_{j+1}$ for $j = 1, \ldots, |\mathcal{I}(x)| - 1$ when $|\mathcal{I}(x)| \geq 2$, let us define

$$Q(x) = \left[e_{i_1} \ \ldots \ e_{i_{|\mathcal{I}(x)|}}\right] \in \mathbb{R}^{n \times |\mathcal{I}(x)|},$$

where $e_i$ denotes the $i$-th vector of the canonical basis of $\mathbb{R}^n$. In what follows, we will consider the function $f_x : \mathbb{R}^{|\mathcal{I}(x)|} \to \mathbb{R}$ given by

$$f_x(y) = f(x + Q(x)y). \tag{2}$$

We have

$$\nabla f_x(y) = Q(x)^T \nabla f(x + Q(x)y) \quad \text{and} \quad \nabla^2 f_x(y) = Q(x)^T \nabla^2 f(x + Q(x)y) Q(x),$$

and so

$$\nabla f_x(0) = Q(x)^T \nabla f(x) \quad \text{and} \quad \nabla^2 f_x(0) = Q(x)^T \nabla^2 f(x) Q(x). \tag{3}$$

## 2.1 Method that employs SPG for leaving faces

Our first method (Algorithm 1) is inspired by Algorithm 4.1 in [8]. In its $k$th iteration, we check whether the norm of the gradient in the current face, $\|\nabla^I_\Omega f(x^k)\|$, is greater than or equal to a multiple of the norm of the reduced gradient, $\|\nabla_\Omega f(x^k)\|$. If this condition holds, then $x^{k+1}$ is computed using Algorithm 2, which performs one iteration of a descent method on $f_{x^k}(\,\cdot\,)$, where the descent direction is obtained by MINRES. Otherwise, $x^{k+1}$ is computed by Algorithm 3, which applies a monotone iteration of the Spectral Projected Gradient method to the minimization of $f(\,\cdot\,)$ in $\Omega$.

**Algorithm 1.** Based on Algorithm 4.1 in [8]

**Step 0.** Given $x^0 \in \Omega$, $\epsilon > 0$, $\theta \in (0,1]$, $\rho \in (0,1)$, $m \in \mathbb{N}$, $a_1 \geq 1 > a_2 > 0$, set $k := 0$.

**Step 1.** If $\|\nabla_\Omega f(x^k)\| \leq \epsilon$, STOP.

**Step 2.** If $\left\|\nabla^I_\Omega f(x^k)\right\| \geq \theta \|\nabla_\Omega f(x^k)\|$, find $x^{k+1}$ by applying Algorithm 2. Otherwise, find $x^{k+1}$ by applying Algorithm 3.

**Step 3.** Set $k := k + 1$ and go to Step 1.

Within faces, we apply Algorithm 2, which is a descent method with Armijo line search and extrapolation. To obtain the search direction, we first compute an approximate solution $d^k_1$ to the minimum residual problem associated with the Newton system. This solution is then corrected to produce a direction $d^k$ satisfying

$$\|d^k\| \leq a_1 \|\nabla f_{x^k}(0)\| \quad \text{and} \quad \langle \nabla f_{x^k}(0), d^k \rangle \leq -a_2 \|\nabla f_{x^k}(0)\|^2$$

as we will see in Lemma 2.1.

**Algorithm 2.** Descent Method with Armijo Line Search

**Inputs:** $x^k \in \Omega$, $\rho \in (0,1)$, $a_1 \geq 1 > a_2 > 0$, $m \in \mathbb{N}$.

**Step 1.** Using MINRES, compute an approximate solution $d^k_1$ of the subproblem

$$\min_{s \in \mathbb{R}^{|\mathcal{I}(x^k)|}} \|\nabla^2 f_{x^k}(0) s + \nabla f_{x^k}(0)\|_2^2.$$

Define

$$d^k_2 = \beta_1 d^k_1, \tag{4}$$

where

$$\beta_1 = \begin{cases} 1, & \text{if } \|d^k_1\| \leq a_1 \|\nabla f_{x^k}(0)\|, \\ \dfrac{a_1 \|\nabla f_{x^k}(0)\|}{\|d^k_1\|}, & \text{otherwise,} \end{cases} \tag{5}$$

and then set

$$d^k = \beta_2 d^k_2 + (1 - \beta_2)\left(-\nabla f_{x^k}(0)\right), \tag{6}$$

where

$$\beta_2 = \begin{cases} 1, & \text{if } \left\langle \nabla f_{x^k}(0), d_2^k \right\rangle \leq -a_2 \|\nabla f_{x^k}(0)\|^2, \\ \dfrac{1 - a_2}{\left(1 + \frac{\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle}{\|\nabla f_{x^k}(0)\|^2}\right)}, & \text{otherwise.} \end{cases} \tag{7}$$

**Step 2.**

**Step 2.1.** If $x^k + Q(x^k)d^k \in \mathcal{F}(x^k)$, set $\alpha_0 = 1$ and go to Step 3.

**Step 2.2.** If $f(P_\Omega(x^k + Q(x^k)d^k)) \leq f_{x^k}(0)$, then using at most $m$ oracle calls, find $x^{k+1} \in \partial\mathcal{F}(x^k)$ such that $f(x^{k+1}) \leq f(P_\Omega(x^k + Q(x^k)d^k))$ and STOP.

**Step 2.3.** Compute $t_{\max} = \max\left\{t \in (0,1] : x^k + tQ(x^k)d^k \in \Omega\right\}$. If $f_{x^k}(t_{\max}d^k) \leq f_{x^k}(0)$, using at most $m$ oracle calls, find $x^{k+1} \in \partial\mathcal{F}(x^k)$ such that $f(x^{k+1}) \leq f_{x^k}(t_{\max}d^k)$ and STOP. Otherwise, set $\alpha_0 = t_{\max}$.

**Step 3.** Find the smallest nonnegative integer $\ell_k$ such that

$$f_{x^k}((0.5)^{\ell_k}\alpha_0 d^k) \leq f_{x^k}(0) + \rho(0.5)^{\ell_k}\alpha_0\langle \nabla f_{x^k}(0), d^k\rangle. \tag{8}$$

**Step 4.** If $\ell_k > 0$, define $x^{k+1} = x^k + (0.5)^{\ell_k}\alpha_0 Q(x^k)d^k$. Otherwise, using at most $m$ oracle calls, find $x^{k+1} \in \overline{\mathcal{F}}(x^k)$ such that $f(x^{k+1}) \leq f_{x^k}(\alpha_0 d^k)$.

As a leaving-face algorithm, we use a monotone variant of the Spectral Projected Gradient (SPG) method, which applies Armijo line search with the search direction defined as

$$v^k = P_\Omega\left(x^k - \frac{1}{\lambda_k^{\text{spg}}}\nabla f(x^k)\right) - x^k,$$

where $\lambda_k^{\text{spg}}$ is the projection of the Barzilai-Borwein stepsize onto the interval $[\lambda_{\min}^{\text{spg}}, \lambda_{\max}^{\text{spg}}]$.

**Algorithm 3.** SPG (Algorithm 4.3 in [9])

**Inputs:** $x^k \in \Omega$, $\rho \in (0,1)$, $\lambda_{\max}^{\text{spg}} \geq \lambda_{\min}^{\text{spg}} > 0$.

**Step 1.** Choose $\lambda_k^{\text{spg}} \in \left[\lambda_{\min}^{\text{spg}}, \lambda_{\max}^{\text{spg}}\right]$ and compute $v^k$ as the solution to

$$\min_{v \in \Omega \setminus \{x^k\}} \nabla f(x^k)^T v + \frac{\lambda_k^{\text{spg}}}{2}\|v\|^2.$$

**Step 2.** Find the smallest nonnegative integer $j_k$ such that

$$f(x^k + (0.5)^{j_k}v^k) \leq f(x^k) + \rho(0.5)^{j_k}\langle \nabla f(x^k), v^k\rangle. \tag{9}$$

**Step 3.** Set $x^{k+1} = x^k + (0.5)^{j_k}v^k$.

Let us begin our analysis by focusing on the properties of Algorithm 2. Our first lemma establishes the descent properties of the search direction $d^k$.

**Lemma 2.1.** *Let $d^k$ be defined by (6) and (7). Then*

$$\|d^k\| \leq a_1\|\nabla f_{x^k}(0)\| \quad \textit{and} \quad \langle \nabla f_{x^k}(0), d^k\rangle \leq -a_2\|\nabla f_{x^k}(0)\|^2. \tag{10}$$

*Proof.* By (4) and (5), we have

$$\|d_2^k\| \leq a_1 \|\nabla f_{x^k}(0)\|. \tag{11}$$

Let us now analyse $d^k$ defined by (6) and (7). If $\boxed{\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle \leq -a_2 \|\nabla f_{x^k}(0)\|^2}$ then by (7) we have $\beta_2 = 1$. Thus, in view of (11), we obtain

$$\|d^k\| = \|d_2^k\| \leq a_1 \|\nabla f_{x^k}(0)\| \quad \text{and} \quad \left\langle \nabla f_{x^k}(0), d^k \right\rangle = \left\langle \nabla f_{x^k}(0), d_2^k \right\rangle \leq -a_2 \|\nabla f_{x^k}(0)\|^2,$$

that is (10) holds. Suppose that $\boxed{\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle > -a_2 \|\nabla f_{x^k}(0)\|^2}$. Then

$$1 + \frac{\left\langle \nabla f_{x^k}(0), d^k \right\rangle}{\|\nabla f_{x^k}(0)\|^2} > 1 - a_2 > 0,$$

and so, by (7),

$$0 < \beta_2 = \frac{1 - a_2}{\left(1 + \frac{\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle}{\|\nabla f_{x^k}(0)\|^2}\right)} < 1.$$

Consequently, it follows from (6) and (11) that

$$\|d^k\| \leq \beta_2 \|d_2^k\| + (1 - \beta_2)\|\nabla f_{x^k}(0)\| \leq a_1 \|\nabla f_{x^k}(0)\|.$$

In addition, we also have

$$\begin{aligned}
\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle &= \beta_2 \left\langle \nabla f_{x^k}(0), d_2^k \right\rangle - (1 - \beta_2)\|\nabla f_{x^k}(0)\|^2 \\
&= \frac{(1 - a_2)\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle - \left(1 + \frac{\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle}{\|\nabla f_{x^k}(0)\|^2} - (1 - a_2)\right)\|\nabla f_{x^k}(0)\|^2}{\left(1 + \frac{\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle}{\|\nabla f_{x^k}(0)\|^2}\right)} \\
&= \frac{-a_2\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle - a_2\|\nabla f_{x^k}(0)\|^2}{\left(1 + \frac{\left\langle \nabla f_{x^k}(0), d_2^k \right\rangle}{\|\nabla f_{x^k}(0)\|^2}\right)} \\
&= \frac{-a_2\left(\|\nabla f_{x^k}(0)\|^2 + \left\langle \nabla f_{x^k}(0), d_2^k \right\rangle\right)}{\left(\|\nabla f_{x^k}(0)\|^2 + \left\langle \nabla f_{x^k}(0), d_2^k \right\rangle\right)}\|\nabla f_{x^k}(0)\|^2 \\
&= -a_2\|\nabla f_{x^k}(0)\|^2.
\end{aligned}$$

Therefore, (10) also holds in this case. □

Combining Lemma 2.1 and the Lipschitz continuity of $\nabla f(\,\cdot\,)$ (assumption A1), the next lemma provides a positive lower bound for stepsizes that do not satisfy the Armijo condition.

**Lemma 2.2.** *Suppose that A1 holds. Given $\alpha > 0$, if*

$$f_{x^k}(\alpha d^k) > f_{x^k}(0) + \rho\alpha\langle \nabla f_{x^k}(0), d^k \rangle \tag{12}$$

*then*

$$\alpha > \frac{2(1 - \rho)a_2}{L_g a_1^2}. \tag{13}$$

*Proof.* Using the definition of $f_{x^k}(\cdot)$ in (2), we see that inequality (12) is equivalent to

$$\rho\alpha\langle\nabla f_{x^k}(0), d^k\rangle + f(x^k) < f\left(x^k + \alpha Q(x^k)d^k\right). \tag{14}$$

Notice that $Q(x^k)^T Q(x^k) = Id$. Thus, by assumption A1 we also have

$$\begin{aligned} f\left(x^k + \alpha Q(x^k)d^k\right) &\leq f(x^k) + \alpha\left\langle\nabla f(x^k), Q(x^k)d^k\right\rangle + \frac{L_g}{2}\alpha^2\|Q(x^k)d^k\|^2 \\ &= f(x^k) + \alpha\left\langle Q(x^k)^T\nabla f(x^k), d^k\right\rangle + \frac{L_g}{2}\alpha^2\|d^k\|^2 \\ &= f(x^k) + \alpha\left[\langle\nabla f_{x^k}(0), d^k\rangle + \frac{L_g}{2}\alpha\|d^k\|^2\right] \end{aligned} \tag{15}$$

Combining (14) and (15), it follows that

$$\rho\langle\nabla f_{x^k}(0), d^k\rangle < \langle\nabla f_{x^k}(0), d^k\rangle + \frac{L_g}{2}\alpha\|d^k\|^2.$$

Therefore, by Lemma 2.1, we conclude that

$$\alpha > \frac{2(1-\rho)}{L_g}\left(-\frac{\langle\nabla f_{x^k}(0), d^k\rangle}{\|d^k\|^2}\right) \geq \frac{2(1-\rho)a_2}{L_g a_1^2}.$$

□

In view of Lemmas 2.1 and 2.2, if $x^{k+1}$ is computed by Algorithm 2, then the objective function decreases by at least a multiple $\|\nabla_\Omega^I f(x^k)\|^2$.

**Lemma 2.3.** *Suppose that A1 holds. Then, whenever $x^{k+1}$ is computed by Step 4 of Algorithm 2, we have*

$$f(x^k) - f(x^{k+1}) \geq \rho\min\left\{a_2, \frac{(1-\rho)a_2^2}{L_g a_1^2}\right\}\left\|\nabla_\Omega^I f(x^k)\right\|^2. \tag{16}$$

*Moreover, the number of evaluations of $f(\cdot)$ necessary to guarantee the fulfillment of (8) is bounded from above by*

$$\left|\log_2\left(\min\left\{\frac{1}{2}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}\right)\right|. \tag{17}$$

*Proof.* First, let us show that

$$(0.5)^{\ell_k}\alpha_0 \geq \min\left\{\alpha_0, \frac{(1-\rho)a_2}{L_g a_1^2}\right\}. \tag{18}$$

If $\ell_k = 0$, then (18) is clearly true. Thus, let us assume that $\ell_k > 0$. In this case, by the definition of $\ell_k$, it follows that

$$f_{x^k}\left((0.5)^{\ell_k-1}\alpha_0 d^k\right) > f_{x^k}(0) + \rho(0.5)^{\ell_k-1}\alpha_0\langle\nabla f_{x^k}(0), d^k\rangle.$$

Thus, by Lemma 2.1, we have

$$(0.5)^{\ell_k-1}\alpha_0 > \frac{2(1-\rho)a_2}{L_g a_1^2},$$

which implies that (18) also holds when $\ell_k > 0$. Now, let us refine the lower bound in (18). In view of Steps 2.1 and 2.3 of Algorithm 2, we have $\boxed{\alpha_0 = 1 \text{ or } \alpha_0 = t_{\max} \leq 1}$. The latter occurs only if $f_{x^k}(t_{\max} d^k) > f_{x^k}(0)$. In particular, this means that inequality (12) holds for $\alpha = t_{\max}$. Consequently, by Lemma 2.2, in this case we must have

$$\alpha_0 = t_{\max} > \frac{2(1-\rho)a_2}{L_g a_1^2}.$$

Therefore, in any case, we have

$$\alpha_0 \geq \min\left\{1, \frac{2(1-\rho)a_2}{L_g a_1^2}\right\}. \tag{19}$$

Combining (18) and (19), we obtain

$$(0.5)^{\ell_k}\alpha_0 \geq \min\left\{1, \frac{(1-\rho)a_2}{L_g a_1^2}\right\}. \tag{20}$$

Now, from Steps 3 and 4 of Algorithm 2, we obtain

$$\begin{aligned} f(x^k) - f(x^{k+1}) &\geq \rho(0.5)^{\ell_k}\alpha_0\left(-\langle \nabla f_{x^k}(0), d^k\rangle\right) \\ &\geq \rho(0.5)^{\ell_k}\alpha_0 a_2 \|\nabla f_{x^k}(0)\|^2 \\ &= \rho(0.5)^{\ell_k}\alpha_0 a_2 \left\|Q(x^k)^T \nabla f(x^k)\right\|^2 \\ &\geq \rho(0.5)^{\ell_k}\alpha_0 a_2 \left\|\nabla^I_\Omega f(x^k)\right\|^2 \\ &\geq \rho \min\left\{1, \frac{(1-\rho)a_2}{L_g a_1^2}\right\} a_2 \left\|\nabla^I_\Omega f(x^k)\right\|^2, \end{aligned}$$

that is, (16) is true. Finally, notice that the number of evaluations of $f(\,\cdot\,)$ performed at Step 3 of Algorithm 2 is equal to $\ell_k + 1$. From (20), we have

$$(0.5)^{\ell_k+1} \geq (0.5)^{\ell_k+1}\alpha_0 \geq \min\left\{\frac{1}{2}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}.$$

Then, taking the logarithm on both sides, it follows that

$$\ell_k + 1 \leq \left|\log_2\left(\min\left\{\frac{1}{2}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}\right)\right|.$$

□

If $x^{k+1}$ is computed by Algorithm 3, then the objective function decreases by at least a constant multiple of $\|\nabla_\Omega f(x^k)\|^2$.

**Lemma 2.4.** *(Theorem 4.2 in [9]) Suppose that A1 holds. Then, whenever $x^{k+1}$ is computed by Algorithm 3, we have*

$$f(x^k) - f(x^{k+1}) \geq \frac{\rho(1-2\rho)}{8L_g} \min\left\{\lambda^{\text{spg}}_{\min}, \frac{\lambda^{\text{spg}}_{\min}}{\lambda^{\text{spg}}_{\max}}\right\}^2 \left\|\nabla_\Omega f(x^k)\right\|^2. \tag{21}$$

*Moreover, the number of evaluations of $f(\,\cdot\,)$ necessary to guarantee the fulfillment of (8) is bounded from above by*

$$\left|\log_2\left(\min\left\{1, \frac{(1-2\rho)\lambda^{\text{spg}}_{\min}}{4L_g}\right\}\right)\right| + 1. \tag{22}$$

Let $\{x^k\}_{k\geq 0}$ be a sequence generated by Algorithm 1. Denote by

$$T(\epsilon) = \inf\left\{k \in \mathbb{N} \,:\, \|\nabla_\Omega f(x^k)\| \leq \epsilon\right\}, \tag{23}$$

the first hitting time to the set of $\epsilon$-approximate stationary points of $f(\,\cdot\,)$ with respect to $\Omega$, and consider the sets

$$\mathcal{S}_{T(\epsilon)-1} = \left\{k \in \{0,\ldots,T(\epsilon)-1\} \,:\, x^{k+1} \text{ is computed in Step 4 of Algorithm 2 or by Algorithm 3}\right\}$$

$$\mathcal{U}_{T(\epsilon)-1} = \left\{k \in \{0,\ldots,T(\epsilon)-1\} \,:\, x^{k+1} \text{ is computed in Step 2.2 or Step 2.3 of Algorithm 2}\right\}.$$

The next theorem establishes that $T(\epsilon) \leq \mathcal{O}\left(n\epsilon^{-2}\right)$, that is, Algorithm 1 needs no more than $\mathcal{O}(n\epsilon^{-2})$ iterations to find $\epsilon$-approximate stationary points.

**Theorem 2.5.** *Suppose that A1-A2 hold, and let $\{x^k\}_{k=0}^{T(\epsilon)}$ be generated by Algorithm 1. Then*

$$T(\epsilon) \leq 1 + (n+1)\left(\frac{f(x^0) - f_{\text{low}}}{\kappa_1}\right)\epsilon^{-2}, \tag{24}$$

*where*

$$\kappa_1 = \min\left\{\rho a_2\theta^2, \frac{\rho(1-\rho)}{L_g}\left(\frac{\theta a_2}{a_1}\right)^2, \frac{\rho(1-2\rho)}{8L_g}\min\left\{\lambda_{\min}^{\text{spg}}, \frac{\lambda_{\min}^{\text{spg}}}{\lambda_{\max}^{\text{spg}}}\right\}^2\right\}. \tag{25}$$

*Proof.* If $T(\epsilon) \leq 1$, then (24) is clearly true. Thus, suppose that $T(\epsilon) \geq 2$ and let $k \in \{0,\ldots,T(\epsilon)-1\}$. By the definition of $T(\epsilon)$ we have

$$\|\nabla_\Omega f(x^k)\| > \epsilon. \tag{26}$$

If

$$\left\|\nabla_\Omega^I f(x^k)\right\| \geq \theta\|\nabla_\Omega f(x^k)\| \tag{27}$$

then, by Step 2 of Algorithm 1, the next iterate $x^{k+1}$ is computed by Algorithm 2. Specifically, if $\boxed{x^{k+1} \text{ is obtained from Step 4 of Algorithm 2}}$, it follows from Lemma 2.3, (27), (26) and (25) that

$$\begin{aligned}
f(x^k) - f(x^{k+1}) &\geq \rho\min\left\{a_2, \frac{(1-\rho)a_2^2}{L_g a_1^2}\right\}\left\|\nabla_\Omega^I f(x^k)\right\|^2 \\
&\geq \rho\min\left\{a_2, \frac{(1-\rho)a_2^2}{L_g a_1^2}\right\}\theta^2\|\nabla_\Omega f(x^k)\|^2 \\
&> \rho\min\left\{a_2, \frac{(1-\rho)a_2^2}{L_g a_1^2}\right\}\theta^2\epsilon^2 \\
&\geq \kappa_1\epsilon^2.
\end{aligned} \tag{28}$$

On the other hand, if $\boxed{x^{k+1} \text{ is computed by Algorithm 3}}$, it follows from Lemma 2.3, (27) and (25) that

$$f(x^k) - f(x^{k+1}) \geq \frac{\rho(1-2\rho)}{8L_g}\min\left\{\lambda_{\min}^{\text{spg}}, \frac{\lambda_{\min}^{\text{spg}}}{\lambda_{\max}^{\text{spg}}}\right\}^2\epsilon^2 \geq \kappa_1\epsilon^2. \tag{29}$$

In view of (28) and (29), for any $k \in \mathcal{S}_{T(\epsilon)-1}$ we have

$$\boxed{f(x^k) - f(x^{k+1}) \geq \kappa_1 \epsilon^2.}$$

In addition, for any $k \in \mathcal{U}_{T(\epsilon)-1}$, we have $f(x^{k+1}) \leq f(x^k)$. Thus, by A1,

$$\begin{aligned} f(x^0) - f_{\text{low}} &\geq f(x^0) - f\left(x^{T(\epsilon)}\right) = \sum_{k=0}^{T(\epsilon)-1} f(x^k) - f(x^{k+1}) \\ &\geq \sum_{k \in \mathcal{S}_{T(\epsilon)-1}} f(x^k) - f(x^{k+1}) \\ &\geq |\mathcal{S}_{T(\epsilon)-1}| \kappa_1 \epsilon^2, \end{aligned}$$

which implies that

$$|\mathcal{S}_{T(\epsilon)-1}| \leq \left(\frac{f(x^0) - f_{\text{low}}}{\kappa_1}\right) \epsilon^{-2}. \tag{30}$$

Notice that if $k \in \mathcal{U}_{T(\epsilon)-1}$, then $x^{k+1} \in \partial\mathcal{F}(x^{k+1})$, and so $|\mathcal{I}(x^{k+1})| \leq |\mathcal{I}(x^k)| - 1$. Consequently, there can be at most $n$ consecutive iterations of Algorithm 1 with $x^{k+1}$ being computed by Step 2.3 of Algorithm 2. In the worst-case, each iteration in $\mathcal{S}_{T(\epsilon)-1}$ would be followed by $n$ consecutive iterations in $\mathcal{U}_{T(\epsilon)-1}$. Therefore,

$$|\mathcal{U}_{T(\epsilon)-1}| \leq n|\mathcal{S}_{T(\epsilon)-1}|. \tag{31}$$

Finally, combining (30) and (31), we conclude that

$$T(\epsilon) = |\mathcal{S}_{T(\epsilon)-1}| + |\mathcal{U}_{T(\epsilon)-1}| \leq (1+n)\left(\frac{f(x^0) - f_{\text{low}}}{\kappa_1}\right) \epsilon^{-2},$$

which means that (24) also holds when $T(\epsilon) \geq 2$. □

**Remark 2.6.** *If $k \in \mathcal{S}_{T(\epsilon)-1}$ then it follows from Lemmas 2.3 and 2.4 that the number of evaluations of $f(\cdot)$ at the $k$-th iteration is bounded from above by*

$$(m+3) + \max\left\{\left|\log_2\left(\min\left\{\frac{1}{2}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}\right)\right|, \left|\log_2\left(\min\left\{1, \frac{(1-2\rho)\lambda_{\min}^{\text{spg}}}{4L_g}\right\}\right)\right|\right\}.$$

*Additionally, there will be one gradient computation and one Hessian computation. On the other hand, if $k \in \mathcal{U}_{T(\epsilon)-1}$, then at the $k$-th iteration there will be one gradient computation, one Hessian computation and, at most, $m+1$ function evaluations. In summary, each iteration of Algorithm 1 requires at most*

$$(m+5) + \max\left\{\left|\log_2\left(\min\left\{\frac{1}{2}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}\right)\right|, \left|\log_2\left(\min\left\{1, \frac{(1-2\rho)\lambda_{\min}^{\text{spg}}}{4L_g}\right\}\right)\right|\right\}$$

*calls to the oracle. Then, in view of Theorem 2.5, Algorithm 1 needs no more than $\mathcal{O}\left(n\epsilon^{-2}\right)$ calls to the oracle to find $x^k$ such that $\|\nabla_\Omega f(x^k)\| \leq \epsilon$.*

## 2.2 Method that employs cubic regularization for leaving faces

In what follows, we consider a method that uses an adaptation of the Newton MINRES method [36] within the faces, while it employes the Cubic Regularization of the Newton's method [8] to leave the faces. Regarding the MINRES method for approximately solving

$$\min_{s\in\mathbb{R}^p} \|Hs+g\|_2^2,$$

we use Algorithm 1 from [36], which is called by:

$$\left[s, \mathtt{D_{type}}\right] = \mathtt{MINRES}\left(H, g, \eta\right).$$

Denote the residual by $r = -(Hs+g)$. In view of Lemmas 11–13 in [36], we have the following cases:

- If $\mathtt{D_{type}} = \text{`}\mathtt{SOL}\text{'}$ then

$$\langle g, s\rangle + \langle Hs, s\rangle \leq 0, \tag{32}$$

$$\langle Hs, s\rangle > 0, \quad \langle Hr, r\rangle > 0, \tag{33}$$

  and

$$\|Hr\| \leq \eta\|Hs\|. \tag{34}$$

- If $\mathtt{D_{type}} = \text{`}\mathtt{NPC}\text{'}$ then

$$\langle g, r\rangle = -\|r\|^2, \tag{35}$$

  and

$$\langle Hr, r\rangle < 0. \tag{36}$$

Algorithm 4 follows a structure similar to that of Algorithm 1, but with some important differences. In addition to using different methods within faces and for leaving faces, a key distinction is the switching mechanism between these methods. Specifically, whenever Newton-MR neither produces a functional decrease of order $\mathcal{O}(\epsilon^{3/2})$ nor returns a point on the boundary of the face that results in a simple decrease of the objective function, we switch to the Cubic Regularization method for the next iterate, which is guaranteed to produce a functional decrease of $\mathcal{O}(\epsilon^{3/2})$.

**Algorithm 4.** Based on Algorithm 4.1 in [8]
**Step 0.** Given $x^0 \in \Omega$, $\epsilon > 0$, $\theta, \eta \in (0,1]$, $\tau \in (0,1]$, $\rho \in (0, 1/2)$, $m \in \mathbb{N}$, $a_1 \geq 1 > a_2 > 0$, and $M, \alpha, \gamma > 0$, set $\eta_0 \geq \eta$, $M_0 = M$, $\sigma_0 = 0$ and $k := 0$.
**Step 1.** If $\|\nabla_\Omega f(x^k)\| \leq \epsilon$, STOP.
**Step 2.** If $\sigma_k \in \{0, 2\}$ and $\left\|\nabla^I_\Omega f(x^k)\right\| \geq \theta\|\nabla_\Omega f(x^k)\|$, call Algorithm 5 as

$$\left(x^{k+1}, \sigma_{k+1}, \eta_{k+1}\right) = \mathtt{NewtonMR}\left(x^k, \rho, \eta_k, \eta, \tau, m, a_1, a_2\right),$$

and set $M_{k+1} = M_k$. Otherwise, call Algorithm 6 as

$$\left(x^{k+1}, M_{k+1}\right) = \mathtt{CubicRegularization}\left(x^k, M_k, M, \alpha, \gamma\right),$$

and set $\sigma_{k+1} = 0$ and $\eta_{k+1} = \eta_k$.
**Step 3.** Set $k := k+1$ and go to Step 1.

In what follows, we describe in detail the Newton-MR algorithm.

**Algorithm 5.** $\left(x^{k+1}, \sigma_{k+1}, \eta_{k+1}\right) = \texttt{NewtonMR}(x^k, \rho, \eta_k, \eta, \tau, m, a_1, a_2)$

**Inputs:** $x^k \in \Omega$, $\rho \in (0, 1/2)$, $\eta_k \in [\eta, 1]$, $\eta \in (0, 1]$, $\tau \in (0, 1]$, $m \in \mathbb{N}$, $a_1 \geq 1 > a_2 > 0$.

**Step 1.** Call Algorithm 1 from [36] (MINRES) as

$$[s^k, \mathtt{D}^k_{\mathtt{type}}] = \texttt{MINRES}\left(\nabla^2 f_{x^k}(0), \nabla f_{x^k}(0), \eta_k\right), \tag{37}$$

and set

$$r^k = -(\nabla^2 f_{x^k}(0) s^k + \nabla f_{x^k}(0)). \tag{38}$$

Define

$$d_1^k = \begin{cases} s^k, & \text{if } \mathtt{D}^k_{\mathtt{type}} = \text{`}\mathtt{SOL}\text{'} \\ r^k, & \text{if } \mathtt{D}^k_{\mathtt{type}} = \text{`}\mathtt{NPC}\text{'} \end{cases}. \tag{39}$$

Set

$$\mathtt{Flag} = \begin{cases} 1, & \text{if } \mathtt{D}^k_{\mathtt{type}} = \text{`}\mathtt{SOL}\text{'} \text{ and } \min\left\{\dfrac{\langle \nabla^2 f_{x^k}(0) d_1^k, d_1^k\rangle}{\|d_1^k\|^2}, \dfrac{\langle \nabla^2 f_{x^k}(0) r^k, r^k\rangle}{\|r^k\|^2}\right\} < \eta_k, \\ 1, & \text{if } \mathtt{D}^k_{\mathtt{type}} = \text{`}\mathtt{NPC}\text{'} \text{ and } \|\nabla^2 f_{x^k}(0) r^k\| \leq \eta_k \|\nabla^2 f_{x^k}(0) s^k\|, \\ 0, & \text{otherwise.} \end{cases} \tag{40}$$

and define

$$\eta_{k+1} = \begin{cases} \eta_k, & \text{if } \mathtt{Flag} = 0, \\ \max\{\tau\ \eta_k, \eta\}, & \text{if } \mathtt{Flag} = 1. \end{cases} \tag{41}$$

If $\mathtt{Flag} = 0$, define $d^k = d_1^k$. Otherwise, define

$$d_2^k = \beta_1 d_1^k, \tag{42}$$

where

$$\beta_1 = \begin{cases} 1, & \text{if } \|d_1^k\| \leq a_1 \|\nabla f_{x^k}(0)\|, \\ \dfrac{a_1 \|\nabla f_{x^k}(0)\|}{\|d_1^k\|}, & \text{otherwise,} \end{cases} \tag{43}$$

and then set

$$d^k = \beta_2 d_2^k + (1 - \beta_2)\left(-\nabla f_{x^k}(0)\right), \tag{44}$$

where

$$\beta_2 = \begin{cases} 1, & \text{if } \left\langle \nabla f_{x^k}(0), d_2^k \right\rangle \leq -a_2 \|\nabla f_{x^k}(0)\|^2, \\ \dfrac{1 - a_2}{\left(1 + \frac{\langle \nabla f_{x^k}(0), d_2^k\rangle}{\|\nabla f_{x^k}(0)\|^2}\right)}, & \text{otherwise.} \end{cases} \tag{45}$$

**Step 2.**

**Step 2.1.** If $x^k + Q(x^k) d^k \in \mathcal{F}(x^k)$, set $\alpha_0 = 1$ and go to Step 3.

**Step 2.2.** If $f(P_\Omega(x^k + Q(x^k) d^k)) \leq f_{x^k}(0)$, using at most $m$ oracle calls, find $x^{k+1} \in \partial\mathcal{F}(x^k)$ such that $f(x^{k+1}) \leq f(P_\Omega(x^k + Q(x^k) d^k))$, set $\sigma_{k+1} = 2$ and STOP.

**Step 2.3.** Compute $t_{\max} = \max\left\{t \in (0,1] : x^k + tQ(x^k)d^k \in \Omega\right\}$. If $f_{x^k}(t_{\max}d^k) \leq f_{x^k}(0)$, using at most $m$ oracle calls, find $x^{k+1} \in \partial\mathcal{F}(x^k)$ such that $f(x^{k+1}) \leq f_{x^k}(t_{\max}d^k)$, set $\sigma_{k+1} = 2$ and STOP. Otherwise, set $\alpha_0 = t_{\max}$.

**Step 3.** Find the smallest nonnegative integer $\ell_k$ such that

$$f_{x^k}((0.5)^{\ell_k}\alpha_0 d^k) \leq f_{x^k}(0) + \rho(0.5)^{\ell_k}\alpha_0\langle\nabla f_{x^k}(0), d^k\rangle. \tag{46}$$

**Step 4.** If $\ell_k > 0$, define $x^{k+1} = x^k + (0.5)^{\ell_k}\alpha_0 Q(x^k)d^k$, set $\sigma_{k+1} = \texttt{Flag}$ and STOP.

**Step 5.** If $\texttt{Flag} = 1$, using at most $m$ oracle calls, find $x^{k+1} \in \overline{\mathcal{F}}(x^k)$ such that $f(x^{k+1}) \leq f_{x^k}(\alpha_0 d^k)$. If $x^{k+1} \in \partial\mathcal{F}(x^k)$, set $\sigma_{k+1} = 2$. Otherwise, set $\sigma_{k+1} = 1$. Then STOP.

**Step 6.** Find the smallest nonnegative integer $j_k$ such that

$$x^k + 2^j\alpha_0 Q(x^k)d^k \in \Omega \quad \text{and} \quad f_{x^k}(2^j\alpha_0 d^k) \leq f_{x^k}(0) + \rho\left(2^j\right)\alpha_0\langle\nabla f_{x^k}(0), d^k\rangle \tag{47}$$

holds with $j = j_k$ and (47) does not hold with $j = j_k + 1$.

**Step 6.1.** If $x^k + 2^{j_k+1}\alpha_0 Q(x^k)d^k \in \Omega$, define $x_{k+1} = x_k + 2^{j_k}\alpha_0 Q(x^k)d^k$, $\sigma_{k+1} = 0$ and STOP.

**Step 6.2.** Compute $t_{\max} = \max\left\{t \in (0,1] : x^k + tQ(x^k)d^k \in \Omega\right\}$. If $f_{x^k}(t_{\max}d^k) > f_{x^k}(0)$, define $x_{k+1} = x_k + 2^{j_k}\alpha_0 Q(x^k)d^k$, $\sigma_{k+1} = 0$ and STOP.

**Step 6.3.** Using at most $m$ oracle calls, find $x^{k+1} \in \partial\mathcal{F}(x^k)$ such that $f(x^{k+1}) \leq f_{x^k}(t_{\max}d^k)$, and set $\sigma_{k+1} = 2$.

In Step 1 of Algorithm 5, the variable `Flag` characterizes the quality of the search direction $d_1^k$ obtained via MINRES. As it will be established in Lemmas 2.9, 2.12 and 2.13, when $\texttt{Flag} = 0$, we can obtain a functional decrease of $\mathcal{O}(\epsilon^{3/2})$ along the search direction $d_1^k$, which is then chosen as the search direction $d^k$. On the other hand, when $\texttt{Flag} = 1$, the descent properties of $d_1^k$ are less clear. In this case, we modify $d_1^k$ by the same procedure used in Algorithm 2, which produces a direction $d^k$ along which a functional decrease of $\mathcal{O}(\epsilon^2)$ is guaranteed (Lemma 2.9). In Step 2, we check whether a unity step along $d^k$ results in a point that remains within the face $\mathcal{F}(x^k)$. If it does, we proceed to Step 3, where a suitable stepsize is determined using an Armijo line-search starting from $\alpha_0 = 1$. If not, we attempt to find a point on the boundary of the face that results in at least a simple decrease in the objective function, in which case we set $\sigma_{k+1} = 2$. If this fails, we move to Step 3 and begin the Armijo line-search with the stepsize $\alpha_0 = t_{\max}$, corresponding to the point along $d^k$ that lies on the boundary of $\mathcal{F}(x^k)$. If the Armijo condition is not satisfied with the stepsize $\alpha_0$, we define $x^{k+1}$ as the resulting point and set $\sigma_{k+1} = \texttt{Flag}$, stopping at Step 4. In contrast, if the Armijo condition is satisfied with $\alpha_0$, we attempt to obtain a larger functional decrease using extrapolation[1]. The result is a point $x^{k+1}$ that either:

- belongs to the boundary of the current face and produces at least a simple decrease in the objective function, where we set $\sigma_{k+1} = 2$ (Steps 5 and 6.3); or

[1] A precise way to implement the extrapolation procedure is described in Section 3.

- produces a functional decrease of $\mathcal{O}(\epsilon^{3/2})$, in which case we set $\sigma_{k+1} = 0$ (Steps 6.1 and 6.2).

Finally, it is worth noticing that when $\Omega = \mathbb{R}^n$ and $f(\,\cdot\,)$ is $\mu$-strongly convex, Algorithm 5 with $\eta_k = \eta = \mu$ essentially reduces to Algorithm 4 in [36], as we will have $\texttt{Flag} = 0$ and Steps 2.2, 2.3, 5, 6.2, and 6.3 will be unnecessary.

Let us now present the Cubic Regularization Method for bound-constrained minimization proposed in [8]. For that, given $x \in \mathbb{R}^n$, we define $T_2(x,\,\cdot\,) : \mathbb{R}^n \to \mathbb{R}$ by

$$T_2(x,s) = \langle \nabla f(x), s\rangle + \frac{1}{2}\langle \nabla^2 f(x)s, s\rangle.$$

**Algorithm 6.** $(x^{k+1}, M_{k+1}) = \texttt{CubicRegularization}\left(x^k, M_k, M, \alpha, \gamma\right)$

**Inputs:** $x^k \in \Omega$, and constants $M_k \geq M > 0$, $\alpha, \gamma > 0$.

**Step 1.** Set $\ell := 0$.

**Step 2.** Compute an approximate solution $s^\ell$ of the subproblem

$$\begin{array}{ll} \min_{s\in\mathbb{R}^n} & T_2(x,s) + (2^\ell M_k)\|s\|^3, \\ \text{s.t.} & x^k + s \in \Omega, \end{array}$$

such that

$$T_2(x^k, s^\ell) + (2^\ell M_k)\|s^\ell\|^3 \leq 0$$

and

$$\left\| \nabla_\Omega \left[ T_p(x^k, x - x^k) + (2^\ell M_k)\|x - x^k\|^3 \right]\Big|_{x = x^k + s^\ell} \right\| \leq \gamma \|s^\ell\|^2.$$

**Step 3.** If

$$f(x^k) - f(x^k + s^\ell) \geq \alpha \|s^\ell\|^3, \tag{48}$$

set $\ell_k = \ell$, and go to Step 4. Otherwise, set $\ell := \ell + 1$ and go to Step 2.

**Step 4.** Define $x^{k+1} = x^k + s^{\ell_k}$ and $M_{k+1} = \max\left\{2^{\ell_k - 1} M_k, M\right\}$.

In addition to A1 and A2, in what follows we will consider the following assumption:

**A3.** $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is $L_H$-Lipschitz continuous.

We begin our analysis by examining the properties of Algorithm 5. The next two lemmas provide positive lower bounds for stepsizes that do not satisfy the Armijo condition.

**Lemma 2.7.** *Suppose that A1 and A3 hold and let the pair $(x^k, \sigma_k)$ be generated by Algorithm 4 such that $\sigma_k = 0$ and $\|\nabla^I_\Omega f(x^k)\| \geq \theta \|\nabla_\Omega f(x^k)\|$. In addition, suppose that $\texttt{D}^k_{\texttt{type}} =$ 'SOL' and $\texttt{Flag} = 0$. Given $\alpha \in (0, 1]$, if*

$$f_{x^k}(\alpha d^k) > f_{x^k}(0) + \rho\alpha\langle \nabla f_{x^k}(0), d^k\rangle, \tag{49}$$

*then*

$$\alpha > \max\left\{ \sqrt{\frac{3(1-2\rho)\eta}{L_H\|d^k\|}}, \frac{2(1-\rho)\eta}{L_g} \right\}. \tag{50}$$

*Proof.* Since $\sigma_k = 0$ and $\|\nabla^I_\Omega f(x^k)\| \geq \theta\|\nabla_\Omega f(x^k)\|$, it follows from Step 2 of Algorithm 4 that Algorithm 5 is called. Then, as $\mathtt{D}^k_{\mathtt{type}} = \text{`}\mathtt{SOL}\text{'}$ and $\mathtt{Flag} = 0$, it follows from Step 2.1 of Algorithm 5 and definition (39) that $d^k = d^k_1 = s^k$. Consequently, by (32), $\mathtt{Flag} = 0$ and (40), we have

$$\langle \nabla f_{x^k}(0), d^k\rangle + \langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle \leq 0, \tag{51}$$

and

$$\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle \geq \eta_k\|d^k\|^2 \geq \eta\|d^k\|^2. \tag{52}$$

It follows from A3, (2), (3), (51), and (52) that

$$\begin{aligned}
f_{x^k}(\alpha d^k) &= f(x^k + \alpha Q(x^k)d^k)\\
&\leq f(x^k) + \alpha\langle \nabla f(x^k), Q(x^k)d^k\rangle + \frac{\alpha^2}{2}\langle \nabla^2 f(x^k)Q(x^k)d^k, Q(x^k)d^k\rangle + \frac{L_H}{6}\alpha^3\|Q(x^k)d^k\|^3\\
&= f(x^k) + \alpha\langle Q(x^k)^T\nabla f(x^k), d^k\rangle + \frac{\alpha^2}{2}\langle Q(x^k)^T\nabla^2 f(x^k)Q(x^k)d^k, d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3\\
&= f_{x^k}(0) + \alpha\nabla f_{x^k}(0), d^k\rangle + \frac{\alpha^2}{2}\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3\\
&\leq f_{x^k}(0) + \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{\alpha}{2}\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3.
\end{aligned} \tag{53}$$

Now, combining (51), (49) and (53), we get

$$\begin{aligned}
(1-\rho)\alpha\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle &\leq (1-\rho)\alpha\left(-\langle \nabla f_{x^k}(0), d^k\rangle\right)\\
&= \rho\alpha\langle \nabla f_{x^k}(0), d^k\rangle - \alpha\langle \nabla f_{x^k}(0), d^k\rangle\\
&< f_{x^k}(\alpha d^k) - f_{x^k}(0) - \alpha\langle \nabla f_{x^k}(0), d^k\rangle\\
&\leq \frac{\alpha}{2}\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3.
\end{aligned}$$

Thus,

$$\left(\frac{1}{2} - \rho\right)\alpha\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle < \frac{L_H}{6}\alpha^3\|d^k\|^3,$$

and so, by (52),

$$\alpha > \sqrt{\frac{3(1-2\rho)}{L_H\|d^k\|}\frac{\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle}{\|d^k\|^2}} \geq \sqrt{\frac{3(1-2\rho)\eta}{L_H\|d^k\|}}. \tag{54}$$

On the other hand, by A1 and (3), we have

$$\begin{aligned}
f_{x^k}(\alpha d^k) &= f(x^k + \alpha Q(x^k)d^k)\\
&\leq f(x^k) + \alpha\langle \nabla f(x^k), Q(x^k)d^k\rangle + \frac{L_g}{2}\alpha^2\|Q(x^k)d^k\|^2\\
&= f(x^k) + \alpha\langle Q(x^k)^T\nabla f(x^k), d^k\rangle + \frac{L_g}{2}\alpha^2\|d^k\|^2\\
&= f_{x^k}(0) + \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{L_g}{2}\alpha^2\|d^k\|^2.
\end{aligned} \tag{55}$$

Combining (55) and (49), we obtain

$$\rho\alpha\langle \nabla f_{x^k}(0), d^k\rangle < \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{L_g}{2}\alpha^2\|d^k\|^2,$$

which together with (51) and (52) gives

$$\alpha > \frac{2(1-\rho)}{L_g}\left(-\frac{\langle \nabla f_{x^k}(0), d^k\rangle}{\|d^k\|^2}\right) \geq \frac{2(1-\rho)}{L_g}\frac{\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle}{\|d^k\|^2} \geq \frac{2(1-\rho)\eta}{L_g}. \tag{56}$$

Finally, from (54) and (56) we see that (50) is true. □

**Lemma 2.8.** *Suppose that A1 and A3 hold and let the pair $(x^k, \sigma_k)$ be generated by Algorithm 4 such that $\sigma_k = 0$ and $\|\nabla^I_\Omega f(x^k)\| \geq \theta\|\nabla_\Omega f(x^k)\|$. In addition, suppose that $D^k_{type} =$ 'NPC' and $\texttt{Flag} = 0$. Given $\alpha \in (0, 1]$, if*

$$f_{x^k}(\alpha d^k) > f_{x^k}(0) + \rho\alpha\langle \nabla f_{x^k}(0), d^k\rangle \tag{57}$$

*then*

$$\alpha > \max\left\{\sqrt{\frac{6(1-\rho)}{L_H\|d^k\|}}, \frac{2(1-\rho)}{L_g}\right\}. \tag{58}$$

*Proof.* Since $\sigma_k = 0$ and $\|\nabla^I_\Omega f(x^k)\| \geq \theta\|\nabla_\Omega f(x^k)\|$, it follows from Step 2 of Algorithm 4 that Algorithm 5 is called. Then, as $\mathtt{D}^k_{\mathtt{type}} =$ 'NPC' and $\texttt{Flag} = 0$, it follows from Step 2.1 of Algorithm 5 and definition (39) that $d^k = d^k_1 = r^k$. Consequently, by (35) and (36) we have

$$\langle \nabla f_{x^k}(0), d^k\rangle = -\|d^k\|^2 \tag{59}$$

and

$$\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle < 0. \tag{60}$$

It follows from A3, (2), (3) and (60) that

$$\begin{aligned} f_{x^k}(\alpha d^k) &= f(x^k + \alpha Q(x^k)d^k) \qquad (61)\\ &\leq f(x^k) + \alpha\langle \nabla f(x^k), Q(x^k)d^k\rangle + \frac{\alpha^2}{2}\langle \nabla^2 f(x^k)Q(x^k)d^k, Q(x^k)d^k\rangle + \frac{L_H}{6}\alpha^3\|Q(x^k)d^k\|^3\\ &= f(x^k) + \alpha\langle Q(x^k)^T\nabla f(x^k), d^k\rangle + \frac{\alpha^2}{2}\langle Q(x^k)^T\nabla^2 f(x^k)Q(x^k)d^k, d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3\\ &= f(x^k) + \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{\alpha^2}{2}\langle \nabla^2 f_{x^k}(0)d^k, d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3\\ &\leq f_{x^k}(0) + \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3. \qquad (62) \end{aligned}$$

Combining (57) and (62), it follows that

$$\rho\alpha\langle \nabla f_{x^k}(0), d^k\rangle < \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{L_H}{6}\alpha^3\|d^k\|^3.$$

Thus, by (59),

$$\alpha^2 > \frac{6(1-\rho)\left(-\langle \nabla f_{x^k}(0), d^k\rangle\right)}{L_H\|d^k\|^3} = \frac{6(1-\rho)}{L_H\|d^k\|},$$

which implies that

$$\alpha > \sqrt{\frac{6(1-\rho)}{L_H\|d^k\|}}. \tag{63}$$

On the other hand, by A1, (2) and (3) we also have

$$\begin{aligned} f_{x^k}(\alpha d^k) &= f(x^k + \alpha Q(x^k)d^k) \\ &\leq f(x^k) + \alpha\langle \nabla f(x^k), Q(x^k)d^k\rangle + \frac{L_g}{2}\alpha^2\|Q(x^k)d^k\|^2 \\ &= f(x^k) + \alpha\langle Q(x^k)^T\nabla f(x^k), d^k\rangle + \frac{L_g}{2}\alpha^2\|d^k\|^2 \\ &= f(x^k) + \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{L_g}{2}\alpha^2\|d^k\|^2. \end{aligned} \tag{64}$$

Then, combining (57) and (64), it follows that

$$\rho\alpha\langle \nabla f_{x^k}(0), d^k\rangle < \alpha\langle \nabla f_{x^k}(0), d^k\rangle + \frac{L_g}{2}\alpha^2\|d^k\|^2.$$

Therefore, by (59), we get

$$\alpha > \frac{2(1-\rho)(-\langle \nabla f_{x^k}(0), d^k\rangle)}{L_g\|d^k\|^2} = \frac{2(1-\rho)}{L_g}. \tag{65}$$

In view of (63) and (65), we conclude that (58) is true. □

In view of Lemmas 2.7 and 2.8, we can now derive lower bounds for the functional decrease $f(x^k) - f(x^{k+1})$ that occurs when $x^{k+1}$ is computed in Step 4 of Algorithm 5.

**Lemma 2.9.** *Suppose that A1 and A3 hold. Then, whenever $x^{k+1}$ is computed at Step 4 of Algorithm 5, we have*

$$f(x^k) - f(x^{k+1}) \geq \begin{cases} \frac{\rho(1-\rho)}{L_g}\left(\frac{a_2}{a_1}\right)^2\|\nabla^I_\Omega f(x^k)\|^2, & \text{if } \texttt{Flag} = 1, \\ \frac{\rho}{2^{5/2}}\left(\frac{\eta}{L_g}\right)^{3/2}\sqrt{\frac{3(1-2\rho)}{L_H}}\|\nabla^I_\Omega f(x^k)\|^{3/2}, & \text{if } \texttt{Flag} = 0 \text{ and } D^k_{type} = \text{`SOL'}, \\ \frac{\rho}{2\left[\left(\frac{L_g}{\eta}\right)+1\right]^{3/2}}\sqrt{\frac{6(1-\rho)}{L_H}}\|\nabla^I_\Omega f(x^k)\|^{3/2}, & \text{if } \texttt{Flag} = 0 \text{ and } D^k_{type} = \text{`NPC'}. \end{cases} \tag{66}$$

*Moreover, the number of evaluations of $f(\cdot)$ required to guarantee the fulfilment of (46) is bounded from above by*

$$\left|\log_2\left(\min\left\{\frac{(1-\rho)\eta}{2L_g}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}\right)\right|$$

*Proof.* Since $x^{k+1}$ is computed by Step 4 of Algorithm 5, it follows that $\ell_k > 0$, and so

$$f_{x^k}((0.5)^{\ell_k-1}\alpha_0 d^k) > f_{x^k}(0) + \rho(0.5)^{\ell_k-1}\alpha_0\langle \nabla f_{x^k}(0), d^k\rangle. \tag{67}$$

Let us analyse separately the possible cases.

**Case 1:** $\texttt{Flag} = 1$.

By Step 1 of Algorithm 5, $d^k$ is defined by (42)-(45). Consequently, by Lemma 2.1 we have

$$\|d^k\| \leq a_1\|\nabla f_{x^k}(0)\| \quad \text{and} \quad \langle \nabla f_{x^k}(0), d^k\rangle \leq -a_2\|\nabla f_{x^k}(0)\|^2. \tag{68}$$

Then, by A1, Lemma 2.2 also applies to $d^k$. In particular, it follows from (67) that

$$(0.5)^{\ell_k-1}\alpha_0 > \frac{2(1-\rho)a_2}{L_g a_1^2}. \tag{69}$$

Now, combining (2), (46), the second inequality in (68), and (69), we obtain

$$\begin{aligned} f(x^k) - f(x^{k+1}) &= f_{x^k}(0) - f_{x^k}((0.5)^{\ell_k}\alpha_0 d^k) \\ &\geq \rho(0.5)^{\ell_k}\alpha_0\left(-\langle\nabla f_{x^k}(0), d^k\rangle\right) \\ &\geq \frac{\rho}{2}(0.5)^{\ell_k-1}\alpha_0 a_2\|\nabla f_{x^k}(0)\|^2 \\ &\geq \frac{\rho(1-\rho)}{L_g}\left(\frac{a_2}{a_1}\right)^2\|\nabla f_{x^k}(0)\|^2 \\ &\geq \frac{\rho(1-\rho)}{L_g}\left(\frac{a_2}{a_1}\right)^2\|\nabla_\Omega^I f(x^k)\|^2, \end{aligned}$$

that is, (66) holds in this case.

**Case 2:** $\texttt{Flag} = 0$ and $\texttt{D}^k_{\texttt{type}} = \texttt{'SOL'}$.

In this case, it follows from (67) and Lemma 2.7 that

$$(0.5)^{\ell_k-1}\alpha_0 > \max\left\{\sqrt{\frac{3(1-2\rho)\eta}{L_H\|d^k\|}}, \frac{2(1-\rho)\eta}{L_g}\right\}. \tag{70}$$

In addition, by (40), (32)-(34), and $\eta_k \geq \eta$, we also have

$$\langle\nabla f_{x^k}(0), d^k\rangle + \langle\nabla^2 f_{x^k}(0)d^k, d^k\rangle \leq 0, \tag{71}$$

$$\langle\nabla^2 f_{x^k}(0)d^k, d^k\rangle \geq \eta\|d^k\|^2, \tag{72}$$

$$\langle\nabla^2 f_{x^k}(0)r^k, r^k\rangle \geq \eta\|r^k\|^2, \tag{73}$$

and

$$\|\nabla^2 f_{x^k}(0)r^k\| \leq \eta_k\|\nabla^2 f_{x^k}(0)d^k\|. \tag{74}$$

Combining (2), (46), (71) and (72), it follows that

$$\begin{aligned} f(x^k) - f(x^{k+1}) &= f_{x^k}(0) - f_{x^k}((0.5)^{\ell_k}\alpha_0 d^k) \\ &\geq \rho(0.5)^{\ell_k}\alpha_0\left(-\langle\nabla f_{x^k}(0), d^k\rangle\right) \\ &\geq \frac{\rho}{2}(0.5)^{\ell_k-1}\alpha_0\langle\nabla^2 f_{x^k}(0)d^k, d^k\rangle \\ &\geq \frac{\eta\rho}{2}\sqrt{\frac{3(1-2\rho)\eta}{L_H\|d^k\|}}\|d^k\|^2. \end{aligned} \tag{75}$$

On the other hand, by (73) we have

$$\|r^k\|^2 = \frac{\|r^k\|^2}{\langle\nabla^2 f_{x^k}(0), r^k\rangle}\langle\nabla^2 f_{x^k}(0)r^k, r^k\rangle \leq \frac{\|\nabla^2 f_{x^k}(0)r^k\|\|r^k\|}{\eta_k}.$$

Then, dividing both sides by $\|r^k\|$ and using (74) and A1 we get

$$\|r^k\| \leq \frac{\|\nabla^2 f_{x^k}(0) r^k\|}{\eta_k} \leq \|\nabla^2 f_{x^k}(0) d^k\| \leq \|\nabla^2 f_{x^k}(0)\| \|d^k\| \leq L_g \|d^k\|.$$

Thus,

$$\begin{aligned} \|\nabla f_{x^k}(0)\| &\leq \|\nabla f_{x^k}(0) + \nabla^2 f_{x^k}(0) s^k\| + \|\nabla^2 f_{x^k}(0) s^k\| \\ &= \|r^k\| + \|\nabla^2 f_{x^k}(0) d^k\| \\ &\leq 2L_g \|d^k\|. \end{aligned} \tag{76}$$

Combining (75) and (76), it follows that

$$\begin{aligned} f(x^k) - f(x^{k+1}) &\geq \frac{\eta\rho}{2} \sqrt{\frac{3(1-2\rho)\eta}{L_H \|d^k\|}} \frac{\|\nabla f_{x^k}(0)\|^{3/2}}{(2L_g)^{3/2}} \\ &\geq \frac{\rho}{2^{5/2}} \left(\frac{\eta}{L_g}\right)^{3/2} \sqrt{\frac{3(1-2\rho)}{L_H}} \|\nabla^I_\Omega f(x^k)\|^{3/2}, \end{aligned}$$

that is, (66) holds in this case.

**Case 3:** $\texttt{Flag} = 0$ and $\texttt{D}^k_{\texttt{type}} = \texttt{'NPC'}$.

In this case, it follows from (67) and Lemma 2.8 that

$$(0.5)^{\ell_k - 1} \alpha_0 > \max\left\{ \sqrt{\frac{6(1-\rho)}{L_H \|d^k\|}}, \frac{2(1-\rho)}{L_g} \right\}. \tag{77}$$

In addition, by (40) and (35) we have

$$\langle \nabla f_{x^k}(0), d^k \rangle = -\|d^k\|^2, \tag{78}$$

and

$$\|\nabla^2 f_{x^k}(0) d^k\| > \eta_k \|\nabla^2 f_{x^k}(0) s^k\| \geq \eta \|\nabla^2 f_{x^k}(0) s^k\|. \tag{79}$$

Combining (2), (8), (78) and (77), it follows that

$$\begin{aligned} f(x^k) - f(x^{k+1}) &= f_{x^k}(0) - f_{x^k}((0.5)^{\ell_k} \alpha_0 d^k) \\ &\geq \rho (0.5)^{\ell_k} \alpha \left( -\langle \nabla f_{x^k}(0), d^k \rangle \right) \\ &= \frac{\rho}{2} (0.5)^{\ell_k - 1} \alpha_0 \|d^k\|^2 \\ &\geq \frac{\rho}{2} \sqrt{\frac{6(1-\rho)}{L_H}} \|d^k\|^{3/2}. \end{aligned} \tag{80}$$

On the other hand, by (79) and A1, we have

$$\begin{aligned}
\|\nabla f_{x^k}(0)\| &\leq \|\nabla f_{x^k}(0) + r^k\| + \|r^k\| \\
&= \|\nabla f_{x^k}(0) - (\nabla f_{x^k}(0) + \nabla^2 f_{x^k}(0)s^k)\| + \|r^k\| \\
&= \|\nabla^2 f_{x^k}(0)s^k\| + \|d^k\| \\
&< \frac{\|\nabla^2 f_{x^k}(0)d^k\|}{\eta} + \|d^k\| \\
&\leq \frac{\|\nabla^2 f_{x^k}(0)\|\|d^k\|}{\eta} + \|d^k\| \\
&\leq \left[\left(\frac{L_g}{\eta}\right) + 1\right]\|d^k\|. \qquad (81)
\end{aligned}$$

Combining (80) and (81), it follows that

$$\begin{aligned}
f(x^k) - f(x^{k+1}) &\geq \frac{\rho}{2}\sqrt{\frac{6(1-\rho)}{L_H}}\frac{\|\nabla f_{x^k}(0)\|^{3/2}}{\left[\left(\frac{L_g}{\eta}\right) + 1\right]^{3/2}} \\
&\geq \frac{\rho}{2\left[\left(\frac{L_g}{\eta}\right) + 1\right]^{3/2}}\sqrt{\frac{6(1-\rho)}{L_H}}\|\nabla^I_\Omega f(x^k)\|^{3/2},
\end{aligned}$$

that is, (66) also holds in this case.

To conclude, notice that the number of evaluations of $f(\,\cdot\,)$ performed at Step 3 of Algorithm 5 is equal to $\ell_k + 1$. Since $\alpha_0 \in (0, 1]$, it follows from (68), (70) and (77) that

$$(0.5)^{\ell_k+1} \geq (0.5)^{\ell_k+1}\alpha_0 > \min\left\{\frac{(1-\rho)\eta}{2L_g}, \frac{(1-\rho)a_2}{2L_g a_1^2}, \frac{(1-\rho)}{2L_g}\right\}.$$

Then, taking the logarithm on both sides and using the fact that $\eta \in (0, 1]$, we obtain

$$\ell_k + 1 \leq \left|\log_2\left(\min\left\{\frac{(1-\rho)\eta}{2L_g}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}\right)\right|.$$

□

The lemma below provides an upper bound on the maximum stepsize that satisfies the Armijo condition when the direction $d^k$ is certified with $\texttt{Flag} = 0$. This result will be used in the sequel to derive an upper bound on the number of function evaluations required in the extrapolation procedure in Step 6 of Algorithm 5.

**Lemma 2.10.** *Suppose that A1-A3 hold and let $x^k$ be an iterate generated by Algorithm 4 such that $\|\nabla^I_\Omega f(x^k)\| \geq \theta\|\nabla_\Omega f(x^k)\|$. Suppose that $d^k$ is computed by Algorithm 5 and that $\texttt{Flag} = 0$. Given $\alpha > 0$, if $\|\nabla_\Omega f(x^k)\| \geq \epsilon$ and*

$$f_{x^k}(\alpha d^k) \leq f_{x^k}(0) + \rho\alpha\langle\nabla f_{x^k}(0), d^k\rangle, \qquad (82)$$

*then*

$$\alpha \leq \frac{\max\left\{2\left(\frac{L_g}{\eta}\right), \left[\left(\frac{L_g}{\eta}\right) + 1\right]^2\right\}(f(x^0) - f_{low})}{\rho\theta^2}\epsilon^{-2} \qquad (83)$$

*Proof.* If $\mathtt{D}^k_{\mathtt{type}} = \texttt{'SOL'}$, then it follows from (82), (71), (72) and (76) that

$$\begin{aligned} f(x^0) - f_{low} &\geq f_{x^k}(0) - f_{x^k}(\alpha d^k) \geq \rho\alpha\left(-\langle\nabla f_{x^k}(0), d^k\rangle\right) \\ &\geq \rho\alpha\langle\nabla^2 f_{x^k}(0)d^k, d^k\rangle \geq \rho\alpha\eta\|d^k\|^2 \\ &\geq \frac{\rho\eta}{2L_g}\alpha\|\nabla f_{x^k}(0)\|^2 = \frac{\rho}{2}\left(\frac{\eta}{L_g}\right)\alpha\|\nabla^I_\Omega f(x^k)\|^2 \\ &\geq \frac{\rho}{2}\left(\frac{\eta}{L_g}\right)\alpha\theta^2\|\nabla_\Omega f(x^k)\|^2 \\ &\geq \frac{\rho\theta^2}{2}\left(\frac{\eta}{L_g}\right)\epsilon^2\alpha. \end{aligned}$$

Thus,

$$\alpha \leq \frac{2\left(\frac{L_g}{\eta}\right)\left(f(x^0) - f_{low}\right)}{\rho\theta^2}\epsilon^{-2},$$

which means that (83) holds in this case.

Now, suppose that $\mathtt{D}^k_{\mathtt{type}} = \texttt{'NPC'}$. Then, by (82), (78) and (81) we have

$$\begin{aligned} f(x^0) - f_{low} &\geq \rho\alpha\|d^k\|^2 \geq \frac{\rho\alpha}{\left[\left(\frac{L_g}{\eta}\right)+1\right]^2}\|\nabla f_{x^k}(0)\|^2 \\ &= \frac{\rho\alpha}{\left[\left(\frac{L_g}{\eta}\right)+1\right]^2}\|\nabla^I_\Omega f(x^k)\|^2 \geq \frac{\rho\alpha}{\left[\left(\frac{L_g}{\eta}\right)+1\right]^2}\theta^2\|\nabla_\Omega f(x^k)\|^2 \\ &\geq \frac{\rho\theta^2}{\left[\left(\frac{L_g}{\eta}\right)+1\right]^2}\epsilon^2\alpha. \end{aligned}$$

Therefore,

$$\alpha \leq \frac{\left[\left(\frac{L_g}{\eta}\right)+1\right]^2\left(f(x^0) - f_{low}\right)}{\rho\theta^2}\epsilon^{-2},$$

which means that (83) also holds in this case. □

The next lemma gives an upper bound of $\mathcal{O}\left(|\log_2(\epsilon)|\right)$ for the number of evaluations of $f(\cdot)$ required by the extrapolation procedure in Step 6 of Algorithm 5.

**Lemma 2.11.** *Suppose that assumptions A1-A3 hold, and that $x^{k+1}$ is computed in Step 6 of Algorithm 5. Then, the number of evaluations of $f(\cdot)$ required to guarantee the fulfilment of (47) is bounded from above by*

$$1 + \log_2\left(\frac{\max\left\{\left(\frac{L_g}{\eta}\right), \left[\left(\frac{L_g}{\eta}\right)+1\right]^2\right\}\left(f(x^0) - f_{low}\right)}{\rho\theta^2}\epsilon^{-2}\right). \tag{84}$$

*Proof.* The execution of Step 6 of Algorithm 5 implies that $f_{x^k}(\alpha_0 d^k) \leq f_{x^k}(0)$. Therefore, $\alpha_0 = 1$. Indeed, if we had $\alpha_0 = t_{\max} < 1$, then we would have had $f_{x^k}(t_{\max} d^k) \leq f_{x^k}(0)$, and the method would have stopped at Step 2.3, which was not the case. Notice that the number of evaluations of $f(\cdot)$ required to fulfill (47) is equal to $j_k + 1$. Since $x^{k+1}$ is computed by Algorithm 5, we must have had $\|\nabla^I_\Omega f(x^k)\| > \theta \|\nabla_\Omega f(x^k)\|$, and $\|\nabla_\Omega f(x^k)\| > \epsilon$. Moreover, by design, we also have $f(x^k) \leq f(x^0)$. Then, the definition of $j_k$, $\alpha_0 = 1$, and Lemma 2.10 imply that

$$2^{j_k} = 2^{j_k}\alpha_0 \leq \frac{\max\left\{\left(\frac{L_g}{\eta}\right), \left[\left(\frac{L_g}{\eta}\right)+1\right]^2\right\}(f(x^0) - f_{low})}{\rho\theta^2}\epsilon^{-2}. \tag{85}$$

and so, taking the logarithm, we conclude that $j_k+1$ is bounded from above by the number in (84). □

The following two lemmas establish lower bounds of $\mathcal{O}\left(\|\nabla^I_\Omega f(x^k)\|^{3/2}\right)$ for the functional decrease $f(x^k) - f(x^{k+1})$ obtained when $x^{k+1}$ is computed in Steps 6.1 and 6.2 of Algorithm 5.

**Lemma 2.12.** *Suppose that A1 and A3 hold. Then, whenever $x^{k+1}$ is computed in Step 6.1 of Algorithm 5, we have*

$$f(x^k) - f(x^{k+1}) \geq \begin{cases} \frac{\rho}{2^{5/2}}\left(\frac{\eta}{L_g}\right)^{3/2}\sqrt{\frac{3(1-2\rho)}{L_H}}\|\nabla^I_\Omega f(x^k)\|^{3/2}, & \text{if } D^k_{type} = \text{`SOL'}, \\ \frac{\rho}{2\left[\left(\frac{L_g}{\eta}\right)+1\right]^{3/2}}\sqrt{\frac{6(1-\rho)}{L_H}}\|\nabla^I_\Omega f(x^k)\|^{3/2}, & \text{if } D^k_{type} = \text{`NPC'}. \end{cases} \tag{86}$$

*Proof.* As $x^{k+1}$ is computed in Step 6.1, this means that $\boxed{\texttt{Flag} = 0}$; otherwise, the Algorithm 5 would have stopped at Step 5. Since $x^k + 2^{j_k+1}\alpha_0 Q(x^k)d^k \in \Omega$, it follows from the definition of $j_k$, that

$$f_{x^k}(2^{j_k+1}\alpha_0 d^k) > f_{x^k}(0) + \rho 2^{j_k+1}\alpha_0 \langle \nabla f_{x^k}(0), d^k\rangle. \tag{87}$$

If $\boxed{\texttt{D}^k_{\texttt{type}} = \text{`SOL'}}$, then it follows from Lemma 2.7 that

$$2^{j_k+1}\alpha_0 > \sqrt{\frac{3(1-2\rho)\eta}{L_H\|d^k\|}}.$$

Thus, by the same reasoning used in the proof of Lemma 2.9 (Case 2), we see that

$$f(x^k) - f(x^{k+1}) \geq \frac{\rho}{2^{5/2}}\left(\frac{\eta}{L_g}\right)^{3/2}\sqrt{\frac{3(1-2\rho)}{L_H}}\|\nabla^I_\Omega f(x^k)\|^{3/2}.$$

On the other hand, if $\boxed{\texttt{D}^k_{\texttt{type}} = \text{`NPC'}}$, then it follows from (87) and Lemma 2.8 that

$$2^{j_k+1}\alpha_0 > \sqrt{\frac{6(1-\rho)}{L_H\|d^k\|}}.$$

Thus, as in the proof of Lemma 2.9 (Case 3), it follows that

$$f(x^k) - f(x^{k+1}) \geq \frac{\rho}{2\left[\left(\frac{L_g}{\eta}\right)+1\right]^{3/2}}\sqrt{\frac{6(1-\rho)}{L_H}}\|\nabla^I_\Omega f(x^k)\|^{3/2}.$$

□

**Lemma 2.13.** *Suppose that A1 and A3 hold. Then, whenever $x^{k+1}$ is computed in Step 6.2 of Algorithm 5, we have*

$$f(x^k) - f(x^{k+1}) \geq \begin{cases} \frac{\rho}{2^{5/2}} \left(\frac{\eta}{L_g}\right)^{3/2} \sqrt{\frac{3(1-2\rho)}{L_H}} \|\nabla_\Omega^I f(x^k)\|^{3/2}, & \text{if } \mathtt{D}_{type}^k = \text{`SOL'}, \\ \frac{\rho}{2\left[\left(\frac{L_g}{\eta}\right)+1\right]^{3/2}} \sqrt{\frac{6(1-\rho)}{L_H}} \|\nabla_\Omega^I f(x^k)\|^{3/2}, & \text{if } \mathtt{D}_{type}^k = \text{`NPC'}. \end{cases} \tag{88}$$

*Proof.* As $x^{k+1}$ is computed in Step 6.2, this means that $\boxed{\texttt{Flag} = 0}$ and

$$f_{x^k}(t_{\max} d^k) > f_{x^k}(0) > f_{x^k}(0) + \rho t_{\max} \langle \nabla f_{x^k}(0), d^k \rangle. \tag{89}$$

Since

$$x^k + 2^{j_k} \alpha_0 Q(x^k) d^k \in \Omega \quad \text{and} \quad x^k + 2^{j_k+1} \alpha_0 Q(x^k) d^k \notin \Omega,$$

we also have $2^{j_{k+1}} \alpha_0 > t_{\max}$. Then, it follows from (89) and Lemmas 2.7 and 2.8 that

$$2^{j_k} \alpha_0 > t_{\max} \geq \begin{cases} \sqrt{\frac{3(1-2\rho)}{L_H \|d^k\|}}, & \text{if } \mathtt{D}_{\mathtt{type}}^k = \text{`SOL'}, \\ \sqrt{\frac{6(1-\rho)}{L_H \|d^k\|}}, & \text{if } \mathtt{D}_{\mathtt{type}}^k = \text{`NPC'}. \end{cases}$$

Thus, by following the same reasoning in the proof of Lemma 2.9 (Cases 2 and 3), we conclude that (88) is true. □

As shown in the lemma below, if $x^{k+1}$ is computed by Algorithm 6, the objective function decreases by at least a constant multiple of $\|\nabla_\Omega f(x^k)\|^{3/2}$.

**Lemma 2.14.** *Suppose that A1 and A3 hold. Then, whenever $x^{k+1}$ is computed by Algorithm 6, we have*

$$f(x^k) - f(x^{k+1}) \geq \alpha \left( \frac{\|\nabla_\Omega f(x^{k+1})\|}{L_H + 6(L_H + \alpha) + \gamma} \right)^{\frac{3}{2}}. \tag{90}$$

*Moreover, the number of evaluations of $f(\cdot)$ required to guarantee the fulfillment of (48) is bounded from above by*

$$1 + \log_2 \left( \max \left\{ 1, \frac{L_H + \alpha}{M} \right\} \right). \tag{91}$$

*Proof.* Inequality (90) follows from Lemma 3.4 in [8]. Regarding the upper bound (91), note that Algorithm 6 requires $\ell_k + 1$ evaluations of $f(\cdot)$. Thus, let us first show that

$$2^{\ell_k} M_k \leq \max \{ M_k, 2(L_H + \alpha) \}, \tag{92}$$

where, by design, $\ell_k$ is the smallest nonnegative integer for which (48) is satisfied. If $\ell_k = 0$, then $2^{\ell_k} M_k = M_k$ and then (92) clearly holds. Suppose that $\ell_k > 0$. By Lemma 3.2 in [8], (48) is satisfied whenever $2^\ell M_k \geq L_H + \alpha$. This means that we must have

$$2^{\ell_k - 1} M_k < L_H + \alpha,$$

because, otherwise, (48) would had been satisfied by some $\ell$ with $0 \leq \ell \leq \ell_k - 1$, contradicting the definition of $\ell_k$. Therefore,

$$2^{\ell_k} M_k = 2\left(2^{\ell_k - 1} M_k\right) < 2(L_H + \alpha) \leq \max\left\{M_k, 2(L_H + \alpha)\right\},$$

concluding the proof of (92). Finally, from (92) and $M_k \geq M$, it follows that $\ell_k + 1$ is bounded from above by the number in (91). □

Let $\left\{x^k\right\}_{k \geq 0}$ be generated by Algorithm 4, and denote by $T(\epsilon)$ the first hitting time to the set of $\epsilon$-approximate stationary points of $f(\,\cdot\,)$ with respect to $\Omega$, as in (23). Given $j \in \{0, \ldots, T(\epsilon) - 1\}$, consider the following sets:

$$\begin{aligned} \mathcal{S}_j^{(1)} &= \left\{k \in \{0, \ldots, j\} \,:\, \sigma_{k+1} = 0\right\}, \\ \mathcal{S}_j^{(2)} &= \left\{k \in \{0, \ldots, j\} \,:\, \sigma_{k+1} = 1\right\}, \\ \mathcal{U}_j &= \left\{k \in \{0, \ldots, j\} \,:\, \sigma_{k+1} = 2\right\}. \end{aligned}$$

The next theorem establishes that $T(\epsilon) \leq \mathcal{O}\left(n\epsilon^{-3/2}\right)$, i.e., Algorithm 4 needs no more than $\mathcal{O}(n\epsilon^{-3/2})$ iterations to find $\epsilon$-approximate stationary points.

**Theorem 2.15.** *Suppose that A1-A3 hold, and let $\left\{x^k\right\}_{k=0}^{T(\epsilon)}$ be generated by Algorithm 4. Then*

$$T(\epsilon) \leq (n+1) + 2(n+1)\left(\frac{f(x^0) - f_{\text{low}}}{\kappa_2}\right)\epsilon^{-3/2}, \tag{93}$$

*where*

$$\kappa_2 = \min\left\{\frac{\rho\theta^{3/2}}{2\left[\frac{L_g}{\eta} + 1\right]^{3/2}}\sqrt{\frac{6(1-\rho)}{L_H}}, \frac{\rho\theta^{3/2}}{2^{5/2}}\left(\frac{\eta}{L_g}\right)^{3/2}\sqrt{\frac{3(1-2\rho)}{L_H}}, \frac{\alpha}{\left[L_H + 6(L_H + \alpha) + \gamma\right]^{3/2}}\right\}. \tag{94}$$

*Proof.* If $T(\epsilon) \leq 1$, then (93) is true. Thus, suppose that $T(\epsilon) \geq 2$ and let $k \in \{0, \ldots, T(\epsilon) - 1\}$. By (23) we have

$$\|\nabla_\Omega f(x^k)\| > \epsilon. \tag{95}$$

If $k \in \mathcal{S}_{T(\epsilon)-1}^{(1)}$, then $\sigma_{k+1} = 0$. This means that $x^{k+1}$ was computed either by Algorithm 6, at Step 4 of Algorithm 5 with `Flag` $= 0$, or at Steps 6.1 or 6.2 of Algorithm 5. In either case, Lemmas 2.9, 2.12 and 2.13 together with (95) imply that

$$f(x^k) - f(x^{k+1}) \geq \kappa_2 \epsilon^{3/2}, \tag{96}$$

where $\kappa_2$ is defined in (94). Moreover, we have

$$f(x^{\ell+1}) \leq f(x^\ell), \quad \ell = 0, \ldots, T(\epsilon) - 1. \tag{97}$$

Thus, combining A2, (97) and (96), it follows that

$$f(x^0) - f_{\text{low}} \geq f(x^0) - f(x^{T(\epsilon)}) \geq \sum_{k \in \mathcal{S}_{T(\epsilon)-1}^{(1)}} f(x^k) - f(x^{k+1}) \geq \left|\mathcal{S}_{T(\epsilon)-1}^{(1)}\right| \kappa_2 \epsilon^{3/2},$$

and so

$$\left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right| \leq \left(\frac{f(x^0)-f_{\text{low}}}{\kappa_2}\right)\epsilon^{-3/2}. \tag{98}$$

On the other hand, if $k \in \mathcal{S}^{(2)}_{T(\epsilon)-2}$, then $\sigma_{k+1}=1$. Consequently, by Step 2 of Algorithm 4, $x^{k+2}$ is computed by Algorithm 6, and so $\sigma_{k+2}=0$. This means that every iteration in $\mathcal{S}^{(2)}_{T(\epsilon)-2}$ is followed by one iteration in $\mathcal{S}^{(1)}_{T(\epsilon)-1}$. Thus

$$\left|\mathcal{S}^{(2)}_{T(\epsilon)-2}\right| \leq \left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right|,$$

and so

$$\left|\mathcal{S}^{(2)}_{T(\epsilon)-1}\right| \leq 1 + \left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right|. \tag{99}$$

Finally, if $k \in \mathcal{U}_{T(\epsilon)-1}$, then $x^{k+1}$ is computed at Step 2.2, Step 2.3, or Step 6.3 of Algorithm 5. This means that $x^{k+1} \in \partial\mathcal{F}(x^k)$ and so $\left|\mathcal{I}(x^{k+1})\right| \leq \left|\mathcal{I}(x^k)\right| - 1$. Therefore, there can be at most $n$ consecutive iterations of Algorithm 4 with $k \in \mathcal{U}_{T(\epsilon)-1}$. In the worst case, each iteration in $\mathcal{S}^{(1)}_{T(\epsilon)-1} \cup \mathcal{S}^{(2)}_{T(\epsilon)-1}$ would be followed by $n$ consecutive iterations in $\mathcal{U}_{T(\epsilon)-1}$. Then, we have

$$\left|\mathcal{U}_{T(\epsilon)-1}\right| \leq n\left|\mathcal{S}^{(1)}_{T(\epsilon)-1} \cup \mathcal{S}^{(2)}_{T(\epsilon)-1}\right| = n\left(\left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right| + \left|\mathcal{S}^{(2)}_{T(\epsilon)-1}\right|\right). \tag{100}$$

Then, combining (98), (99) and (100), we conclude that

$$\begin{aligned}
T(\epsilon) &= \left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right| + \left|\mathcal{S}^{(2)}_{T(\epsilon)-1}\right| + \left|\mathcal{U}_{T(\epsilon)-1}\right| \\
&\leq \left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right| + 1 + \left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right| + n\left(\left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right| + \left|\mathcal{S}^{(2)}_{T(\epsilon)-1}\right|\right). \\
&\leq (n+1) + 2(n+1)\left|\mathcal{S}^{(1)}_{T(\epsilon)-1}\right| \\
&\leq (n+1) + 2(n+1)\left(\frac{f(x^0)-f_{\text{low}}}{\kappa_2}\right)\epsilon^{-3/2}.
\end{aligned}$$

□

**Remark 2.16.** *In view of Lemmas 2.9 and 2.11, the computation of $x^{k+1}$ by Algorithm 5 requires no more than*

$$m + 5 + \max\left\{\left|\log_2\left(\min\left\{\frac{(1-\rho)\eta}{2L_g}, \frac{(1-\rho)a_2}{2L_g a_1^2}\right\}\right)\right|, \left|\log_2\left(\frac{\max\left\{\left(\frac{L_g}{\eta}\right), \left[\left(\frac{L_g}{\eta}\right)+1\right]^2\right\}(f(x^0)-f_{low})}{\rho\theta^2}\epsilon^{-2}\right)\right|\right\}$$

*evaluations of $f(\cdot)$. On the other hand, by Lemma 2.14, the computation of $x^{k+1}$ by Algorithm 6 requires no more than*

$$1 + \log_2\left(\max\left\{1, \frac{L_H+\alpha}{M}\right\}\right)$$

*evaluations of $f(\cdot)$. Additionally, each execution of Algorithm 5 or Algorithm 6 requires the evaluation of one gradient and one Hessian of $f(\cdot)$. In summary, each iteration of Algorithm 4 requires at most $\mathcal{O}\left(|\log_2(\epsilon)|\right)$ calls to the oracle. Therefore, it follows from Theorem 2.15 that Algorithm 5 takes no more than $\mathcal{O}\left(n|\log_2(\epsilon)|\epsilon^{-3/2}\right)$ calls to the oracle to find an $x^k$ such that $\|\nabla_\Omega f(x^k)\| \leq \epsilon$.*

# 3 Numerical experiments

In this section, we present numerical results to evaluate the performance of the introduced algorithms. Hereafter, we will call Algorithm P the algorithm comprising Algorithms 1, 2 and 3 and Algorithm T the algorithm comprising Algorithms 4, 5 and 6. The letter P indicates the practical appeal of the first one, with worst-case complexity $\mathcal{O}(n\epsilon^{-2})$, while the letter T indicates the theoretical concern behind the development of the second algorithm, with worst-case complexity $\mathcal{O}(n|\log_2(\epsilon)|\epsilon^{-3/2})$. For both algorithms, we used as stop criterion $\|\nabla_\Omega f(x)\|_\infty \leq \epsilon = 10^{-8}$. Other stopping criteria related to maximum iterations and lack of progress exist, which are identical in the two algorithms. We also consider a CPU time limit of 10 minutes for each pair method/problem. In the following we describe some implementation details.

We implemented Algorithms P and T and MINRES in Fortran 90. Codes are available for download at `http://www.ime.usp.br/~egbirgin/`. The two methods were evaluated using all unconstrained and bound-constrained problems from the most updated version of the CUTest collection [27] (version 2.4.0). There are 313 unconstrained problems and 162 bound-constrained problems in this release, for a total of 475 problems. We considered all the problems with their default dimension and the given starting point $x^0$. The smallest problem has 1 variable, the largest problem has 192,627 variables, and the quartiles of the number of variables are $Q_1 = 4$, $Q_2 = 50$, and $Q_3 = 5{,}000$. All experiments were performed on a computer with a 5.2 GHz Intel Core i9-12900K and 5.1 GHz Intel Core i9-12900K processor and 9 128 GB of 32000 MHz DDR4 RAM, running Ubuntu 23.04. The codes were compiled by the GNU Fortran compiler GCC (version 12.3.0) with the -O3 optimization directive enabled.

## 3.1 Implementation details

### 3.1.1 When MINRES encounters a non-positive curvature direction

When MINRES is used in Step 1 of Algorithms 2 and 5, it returns an approximate solution $s^k$ plus $\mathtt{D}^k_{\mathtt{type}}$ = 'NPC' or $\mathtt{D}^k_{\mathtt{type}}$ = 'SOL'. The second case means that the Newtonian linear system has been solved with the desired tolerance (which will be detailed later). If this happens, $d^k_1$ in Algorithms 2 and 5 gets the computed solution $s^k$. On the other hand, $\mathtt{D}^k_{\mathtt{type}}$ = 'NPC' means that MINRES found a non-positive curvature direction while solving the linear system. If this happened when being called by Algorithm 5, $d^k_1$ gets the residue of the linear system, i.e. $d^k_1 = -(\nabla^2 f_{x^k}(0)s^k + \nabla f_{x^k}(0))$. In the implementation of Algorithm 2, we evaluated two possibilities for the case where MINRES returns $\mathtt{D}^k_{\mathtt{type}}$ = 'NPC'. The first is the one used in Algorithm 5, which is to consider $d^k_1 = -(\nabla^2 f_{x^k}(0)s^k + \nabla f_{x^k}(0))$. The second is to consider $d^k_1 = s^k$ if $s^k \neq 0$, i.e. the approximate solution itself, and $d^k_1 = -\nabla f_{x^k}(0)$ if $s^k = 0$. (Note that $s^k = 0$ only when MINRES detects a non-positive curvature direction in its first iteration). Both options will be evaluated numerically below.

### 3.1.2 Tolerance in solving Newtonian systems using MINRES

When MINRES is used in Step 1 of Algorithm 5, the required accuracy is dynamically determined by $\eta_k \geq \eta \equiv \epsilon$, updated according to (41), where its initial value $\eta_0$ and its update factor $\tau \in (0, 1)$ are given parameters. In Algorithm 2, we use a dynamic tolerance $\epsilon^{\text{MR}}_k$ borrowed from [6, p.113]. For $k = 1$ the tolerance is $\epsilon^{\text{MR}}_1 = \epsilon^{\text{MR}}_{\text{ini}}$, where $\epsilon^{\text{MR}}_{\text{ini}} \geq \epsilon^{\text{MR}}_{\text{end}} \equiv \epsilon$ is a given parameter, and the idea is that in

the last iteration the tolerance will be $\epsilon_{\text{end}}^{\text{MR}}$. To do this, at iteration $k \geq 1$ we use a tolerance whose value varies linearly with $\log_{10}(\|\nabla_\Omega f(x^k)\|)$, i.e.,

$$\epsilon_k^{\text{MR}} = \sqrt{10^{a \log_{10}(\|\nabla_\Omega f(x^k)\|^2)+b}}, \tag{101}$$

where

$$a = \frac{\log_{10}(\epsilon_{\text{end}}^{\text{MR}}/\epsilon_{\text{ini}}^{\text{MR}})}{\log_{10}(\epsilon/\|\nabla_\Omega f(x^0)\|)} \quad \text{and} \quad b = 2\log_{10}(\epsilon_{\text{ini}}^{\text{MR}}) - a\log_{10}(\|\nabla_\Omega f(x^0)\|^2). \tag{102}$$

### 3.1.3 Optional extrapolations

Algorithm P (Steps 2.2, 2.3, and 4 of Algorithm 2) and Algorithm T (Steps 2.2, 2.3, 5, and 6.3 of Algorithm 5) attempt to improve the current point by extrapolation. These attempts are by definition of limited effort, so they do not affect the order of the complexity of the algorithms. For this reason, their practical influence on the performance of the methods must be determined numerically. The extrapolation consists of

**(i)** set $u_k = 1$;

**(ii)** while $u_k \leq m$ and $f(P_\Omega(x^k + 2^{u_k}\alpha_0 Q(x^k)d^k)) \leq f(P_\Omega(x^k + 2^{u_k-1}\alpha_0 Q(x^k)d^k))$ set $u_k = u_k + 1$;

**(iii)** define $x^{k+1} = P_\Omega(x^k + 2^{u_k-1}\alpha_0 Q(x^k)d^k)$.

### 3.1.4 Backtracking, Barzilai-Borwein stepsize, and other details

In Step 3 of Algorithm 2, Step 2 of Algorithm 3 and Step 3 of Algorithm 5, in practice, we use quadratic interpolation with safeguards to find a step $t_k > 0$ such that the corresponding sufficient descent condition holds. The description of the algorithms refers to a step that is a power of 2 just to simplify the description. This change has no significance on the theoretical results.

In Algorithm 3, for $k \geq 1$ and whenever $(x^k - x^{k-1})^T(\nabla f_{x^k}(0) - \nabla f_{x^{k-1}}(0)) > 0$, we compute

$$\lambda_k^{\text{spg}} = \max\left\{\lambda_{\min}^{\text{spg}}, \min\left\{\frac{(x^k - x^{k-1})^T(x^k - x^{k-1})}{(x^k - x^{k-1})^T(\nabla f_{x^k}(0) - \nabla f_{x^{k-1}}(0))}, \lambda_{\max}^{\text{spg}}\right\}\right\}.$$

In the other cases, $\lambda_k^{\text{spg}} \in [\lambda_{\min}^{\text{spg}}, \lambda_{\max}^{\text{spg}}]$ is arbitrary and we considered

$$\lambda_k^{\text{spg}} = \max\left\{\lambda_{\min}^{\text{spg}}, \min\left\{\frac{\max\{1, \|x^k\|_\infty\}}{\|\nabla_\Omega f(x^k)\|_\infty}, \lambda_{\max}^{\text{spg}}\right\}\right\}.$$

In Algorithm 6, the approximate solution to the subproblem in Step 2 is calculated using the projected gradient method [26, 32]. In addition, the representation of the regularization parameter with the term $2^\ell M$ in the iteration $\ell$, where $M > 0$ is a parameter of Algorithm 6, is only a simplification for the presentation of the algorithm. In practice, the regularization parameter is represented by $\omega$. When $\ell = 0$, we consider $\omega = 0$. In Step 3, if sufficient decent is not obtained, together with the operation $\ell := \ell + 1$, we update $\omega$ by making $\omega := \max\{\omega_{\min}, \zeta\omega\}$, where $\omega_{\min} > 0$ and $\zeta > 1$ are parameters of the algorithm. In practice, we consider $\omega_{\min} = 10^{-6}$ and $\zeta = 10$, which are common values in regularized methods.

### 3.2 Evaluation of Algorithm P and Algorithm T and their alternatives

This section compares different variants of Algorithms P and T, as well as the best variant of each algorithm. When comparing two algorithms, we first compare their robustness. In this study, we examine both unconstrained and bound-constrained problems. The algorithms we consider produce feasible iterates. Thus, we associate the robustness of a method with the quality of its solutions, i.e., the value of the objective function of the solution it delivers. For a given problem, let $f_1$ and $f_2$ be the functional value found by methods $M_1$ and $M_2$. Given a tolerance $f_{\text{tol}} > 0$, we say that $f_i$ is equivalent to the best value found if

$$f_i \leq f_{\min} + f_{\text{tol}} \max\{1, |f_{\min}|\}$$

where $f_{\min} = \{f_1, f_2\}$, or if $f_i \leq -10^{-12}$. If $f_i$ is equivalent to the best value found, we say that the method $M_i$ was successful. Otherwise, we say that it failed. The greater the number of successes of a method, the greater its robustness. We only analyze the efficiency of methods in problems where the solutions computed by both methods are considered equivalent. We use CPU time as a measure of efficiency and present the comparison of efficiency in the form of performance profiles [21].

Given $f_{\text{tol}} > 0$, let $p$ be the number of problems in which methods $M_1, \ldots, M_q$ being compared found equivalent solutions and let $t_{ij}$ be the CPU time of method $M_i$ when applied to problem $j$. In a performance profile, the curve $\Gamma_i(\tau)$ associated with method $M_i$ is given by

$$\Gamma_i(\tau) = \frac{\#\left\{j \in \{1, \ldots, p\} \mid t_{ij} \leq \tau \min_{s=1,\ldots,q}\{t_{sj}\}\right\}}{p}$$

for $\tau \geq 1$. The value of $\Gamma_i(1)$ corresponds to the proportion of problems in which method $M_i$ was the fastest (including ties). Since only problems in which the methods find equivalent solutions are considered, for all $i$ there exists a finite value of $\tau$ such that $\Gamma_i(\tau) = 1$. Another option would be to include problems in which the methods fail, considering $t_{ij} = +\infty$ if the method $M_i$ failed on problem $j$. In that case, for each $i$, there exists a finite $\bar{\tau}$ such that $\Gamma_i(\tau)$ is constant for all $\tau \geq \bar{\tau}$ and the value of $\Gamma_i(\bar{\tau})$ can be understood as a measure of robustness of the method $M_i$. In this work, we evaluate the robustness first and restrict the performance profiles to evaluate the efficiency only.

#### 3.2.1 Evaluation of alternatives in Algorithm P

In Algorithm P, we considered the standard values $\theta = 0.1$, $\rho = 10^{-4}$, $a_1 = 10^8$, $a_2 = 10^{-16}$, $\lambda_{\min}^{\text{spg}} = 10^{-16}$, and $\lambda_{\max}^{\text{spg}} = 10^{16}$ from the literature. See, for example, [6]. For the parameter $m$ that limits the effort of the optional extrapolations, we considered $m \in \{0, 5, 10, 15, 20\}$. The case $m = 0$ corresponds to no optional extrapolations at all. For the parameter $\epsilon_{\text{ini}}^{\text{MR}}$ that determines the tolerance for the solution of Newtonian linear systems, we considered $\epsilon_{\text{ini}}^{\text{MR}} \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, \epsilon\}$. Note that when $\epsilon_{\text{ini}}^{\text{MR}} = \epsilon_{\text{end}}^{\text{MR}} = \epsilon$ all Newtonian linear systems are solved to full precision.

The five options for values of $m$ plus the five options for $\epsilon_{\text{ini}}^{\text{MR}}$ and the two options for the choice of direction $d_1^k$ when MINRES detects a non-positive curvature direction (see Section 3.1.1) leave a total of fifty parameter combinations for Algorithm P. The best combination of parameters was found by employing irace [37]. The irace package implements the Iterated Race method for the automatic tuning of optimization algorithms, given a set of instances of an optimization problem. We considered 20% of the 475 instances as a training set, selecting one in every five when ordered from smallest to largest by the number of variables. When choosing the best combination of parameters, we ignored the final value of the objective function and considered CPU time as a performance

metric, considering a time of 10 minutes if the stopping criterion of a small projected continuous gradient was not reached. The combination that was identified as the best by irace was $m = 20$, $\epsilon_{\text{ini}}^{\text{MR}} = 0.1$ and taking $d_1^k$ as the approximate solution $s^k$ (instead of the residue $r^k$) when MINRES detectes a non-positive curvature direction. The values of $m$ and $\epsilon_{\text{ini}}^{\text{MR}}$ coincide with values reported in the literature for similar situations [6]. The choice of $d_1^k$ coincides with the results of the preliminary experiments carried out to define Algorithm P.

It is important to discuss the sensitivity of the method in relation to its parameter and algorithmic choice options. When comparing the $m = 0$ and $m = 20$ options, we see that the latter finds values of $f$ smaller than $-10^{12}$ (suggesting that the objective function may be unbounded from below) in three more problems (eight versus eleven) and finds better function values in twenty-nine problems when considering tolerance $f_{\text{tol}} = 0.1$. The variant with $m = 0$ finds values of $f$ equivalent to the best one in 441 problems, while the variant with $m = 20$ does the same in 470 problems. Of the problems in which the two variants found equivalent values of $f$ with tolerance $f_{\text{tol}} = 0.1$, the variant with $m = 0$ is faster in 49% of the problems, while the variant with $m = 20$ is faster in 56%. In conclusion, extrapolations increase the effectiveness of the method but have little impact on its average efficiency. Now, examine the options for choosing a search direction when MINRES identifies a non-positive curvature direction. The options are to use the approximate solution found by MINRES as the search direction or to use the residue as the search direction. Both options identify eleven values of $f$ that are smaller than $-10^{12}$. However, the first option identifies 469 values of $f$ that are considered equivalent to the best, while the second option identifies 440. Considering problems in which both options identify equivalent function values, the first option is faster in 58%, while the second option is faster 45%. In summary, choosing the approximate solution of the linear system as the search direction is a more robust and efficient option. This practical observation contrasts with the fact that the second option guarantees a functional decrease of order $\mathcal{O}(\epsilon^{3/2})$. Regarding the tolerance required to solve linear Newtonian systems, we highlight the difference between the more relaxed option, $\epsilon_{\text{ini}}^{\text{MR}} = 0.1$, and the more stringent option, $\epsilon_{\text{ini}}^{\text{MR}} = \epsilon_{\text{end}}^{\text{MR}} = \epsilon = 10^{-8}$. Surprisingly, the two options produced very similar results. The two variants identified eleven cases in which $f$ appears to be unbounded below and found 470 and 464 better function values, respectively. Considering the cases in which they identified equivalent function values, the former variant was faster in 54% of the cases, while the latter variant was faster in 51% of the cases. In short, the two variants were very similar, with a slight advantage in robustness and efficiency for the variant in which the Newtonian systems are solved with increasing accuracy.

### 3.2.2 Evaluation of alternatives in Algorithm T

In Algorithm T we considered $\theta = 0.1$, $\eta = 10^{-8}$, $\rho = 10^{-4}$, $\alpha = 10^{-8}$, and $\gamma = 1$. For the parameter $m$ that limits the effort of the optional extrapolations, we considered $m \in \{0, 5, 10, 15, 20\}$. For the parameters $\eta_0$ and $\tau$ that determine the tolerance for the solution of Newtonian linear systems, we considered $\eta_0 \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, \eta\}$ and $\tau \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We found the best combination of parameters for Algorithm T by using irace the same way we used to calibrate the parameters of Algorithm P. The best configuration returned by irace was $m = 20$, $\eta_0 = \eta$, and $\tau = 0.9$. Note that since $\eta_0 = \eta$ means that the Newtonian linear systems are solved to full precision by MINRES, the value of $\tau$ has no effect in Algorithm T.

The influence of the extrapolations in Algorithm T was very similar to that already reported for Algorithm P. On the other hand, the tolerance required in the solution of linear Newtonian systems deserves special mention. Here, we consider the variants with $\eta_0 = 0.1$ and $\eta_0 = \eta = \epsilon = 10^{-8}$.

It is worth noting that when the Newton direction is computed with higher precision, there is a higher chance of the search direction producing a functional descent $\mathcal{O}(\epsilon^{3/2})$. Conversely, when the Newton direction is computed with lower precision, there is a higher chance that the search direction will guarantee only a functional descent of order $O(\epsilon^2)$. Consequently, Algorithm T is forced to select a regularized Newton iteration as the next iteration (Algorithm 6). These iterations are computationally expensive because they require approximately solving a regularized model to compute the search direction. As a consequence, on the one hand, the variant with $\eta_0 = 0.1$ proved to be more robust, finding 464 better function values against 426 of the other variant. On the other hand, in the cases where the two variants found function values considered equivalent, the first one was faster in 30% of the cases while the second one was faster in 70% of the cases. In other words, the variant that solves linear Newtonian systems in a relaxed way makes more use of Newton iterations with regularization. This method is considered more robust, though less efficient. Because efficiency is prioritized when choosing parameters with irace, the variant with $\eta_0 = \eta$ was selected.

### 3.2.3 Algorithm P versus Algorithm T

We end this section by comparing Algorithms P and T. Considering the 475 problems, Algorithms P and T stopped at the CPU time limit in 26 and 38 problems, found a function value less than or equal to $-10^{12}$ in 11 and 13 problems, and found a point with a gradient sup-norm less than or equal to $\epsilon$ in 400 and 338 problems, respectively. Regardless of this, considering all the 475 problems, Table 1 shows the comparison of the function values found, and Figure 1 compares the efficiency of Algorithms P and T in those problems where both found equivalent function values with $f_{\text{tol}} = 0.1$. The table shows that Algorithm P is substantially more robust than Algorithm T, since it finds a significantly larger number of better solutions, regardless of the tolerance considered to determine that functional values are equivalent. The figure shows that when both methods find equivalent functional values, Algorithm P is substantially more efficient than Algorithm T.

| | $f_{\text{tol}}$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0.1 | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ | $10^{-7}$ | $10^{-8}$ |
| Algorithm P | 461 | 453 | 450 | 449 | 445 | 445 | 443 | 437 |
| Algorithm T | 389 | 375 | 365 | 357 | 354 | 354 | 351 | 349 |

Table 1: Number of solutions equivalent to the best solution found by Algorithms P and T, as a function of the tolerance $f_{\text{tol}} \in \{10^{-1}, 10^{-2}, \ldots, 10^{-8}\}$, considering all the 475 unconstrained problems and bound-constrained problems from the CUTest collection.

## 3.3 Comparision of Algorithm P and Gencan

In this section we compare Algorithm P with Gencan (included in Algencan 3.1.1 and freely available at `http://www.ime.usp.br/~tango/`). Gencan is an active set method for bound-constrained minimization, introduced in [6]. Algencan [1, 2, 7, 9], an augmented Lagrangian method for nonlinear programming, uses Gencan to solve its subproblems. Gencan is an active set method whose general framework is exactly the same as that described by Algorithm 1. That is, it uses exactly the same criteria as Algorithm P to decide whether the next iteration should be within the current face or whether the current face should be abandoned. If the current face should be abandoned, Gencan

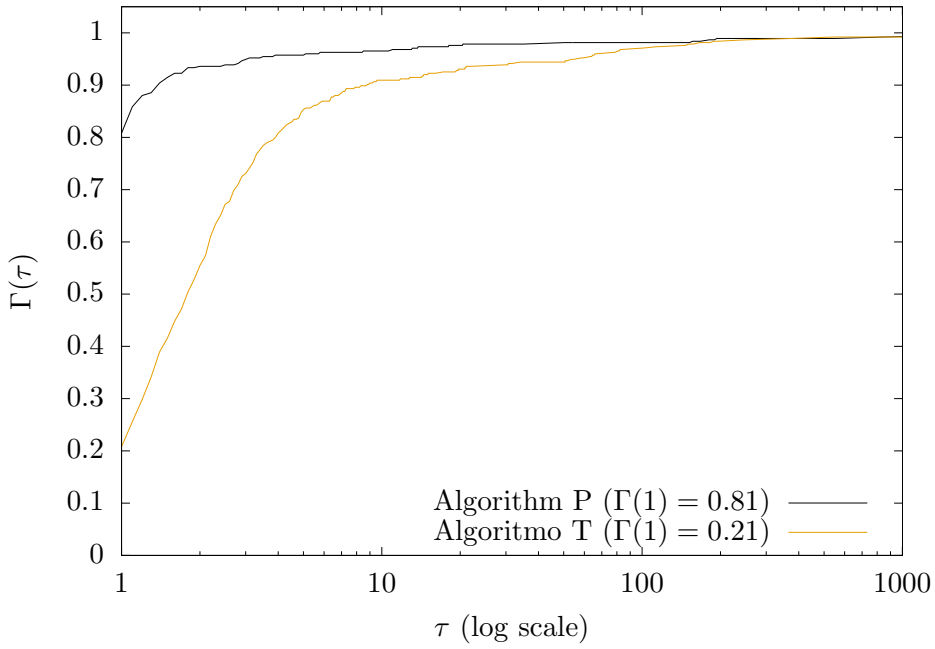


Figure 1: Performance profiles comparing the efficiency of Algorithms P and T on the 375 problems where the two methods found equivalent objective function values with tolerance $f_{\text{tol}} = 0.1$, taking into account all the 475 unconstrained problems and bound-constrained problems from the CUTEst collection.

also uses an SPG iteration as described in Algorithm 3. For iterations within a face, when matrix-factorizations are not allowed, Gencan, like Algorithm P, uses truncated Newton with line search. The difference is that Newtonian linear systems are solved with conjugate gradients. In conjugate gradients, if Hessians are not available, the Hessian vector products are approximated by differences of gradients. In the present work we are assuming that Hessians are available and, therefore, Gencan as well as Algorithms P and T use true Hessian vector products in conjugate gradients and MINRES. Gencan's truncated Newton inspired and shares with Algorithm 2 the way to calculate the tolerance with which linear systems should be solved, what to do when a non-positive curvature direction is detected, and how to decide whether to attempt extrapolations or not. That is, the only relevant difference between Gencan and Algorithm P is that the former uses conjugate gradients and the latter uses MINRES to solve Newtonian linear systems. It is important to mention that the comparison presented in [4] ranked Gencan among the most efficient and robust methods for bound-constrained minimization, in a comparison that included ASA-CG [28], Ipopt [44], Lancelot B [19], L-BFGS-B [14, 45, 38], SPG [10, 11, 12, 13] and fmincon [15, 16].

We run Gencan with all its default parameters and the same stopping criterion already mentioned for Algorithms P and T, i.e., $\|\nabla_\Omega f(x)\|_\infty \leq \epsilon$ with $\epsilon = 10^{-8}$. Considering the 475 problems, Algorithm P and Gencan stopped at the CPU time limit in 26 and 33 problems, found a function value less than or equal to $-10^{12}$ in 11 and 12 problems, and found a point with a gradient sup-norm less than or equal to $\epsilon$ in 400 and 364 problems, respectively. Regardless of this, considering all the 475 problems, Table 2 shows the comparison of the function values found, and Figure 2 compares

the efficiency of the two variants in those problems where both found equivalent function values with $f_{\text{tol}} = 0.1$. The table and the figure show that Algorithm P is slightly more robust and significantly more efficient than Gencan. When we consider only the 313 unconstrained problems in the CUTEst collection, the results are qualitatively equivalent to those shown in [36]. In that study, the Newton-MR method, from which Algorithms P and T originated, was found to be more robust and efficient than several variations of Newton's method that use conjugate gradients to solve Newtonian linear systems.

| | $f_{\text{tol}}$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0.1 | $10^{-2}$ | $10^{-3}$ | $10^{-4}$ | $10^{-5}$ | $10^{-6}$ | $10^{-7}$ | $10^{-8}$ |
| Algorithm P | 461 | 452 | 447 | 442 | 436 | 434 | 433 | 426 |
| Gencan | 448 | 446 | 438 | 432 | 429 | 425 | 424 | 418 |

Table 2: Number of solutions equivalent to the best solution found by Algorithm P and Gencan, as a function of the tolerance $f_{\text{tol}} \in \{10^{-1}, 10^{-2}, \ldots, 10^{-8}\}$, considering all the 475 unconstrained problems and bound-constrained problems from the CUTest collection.

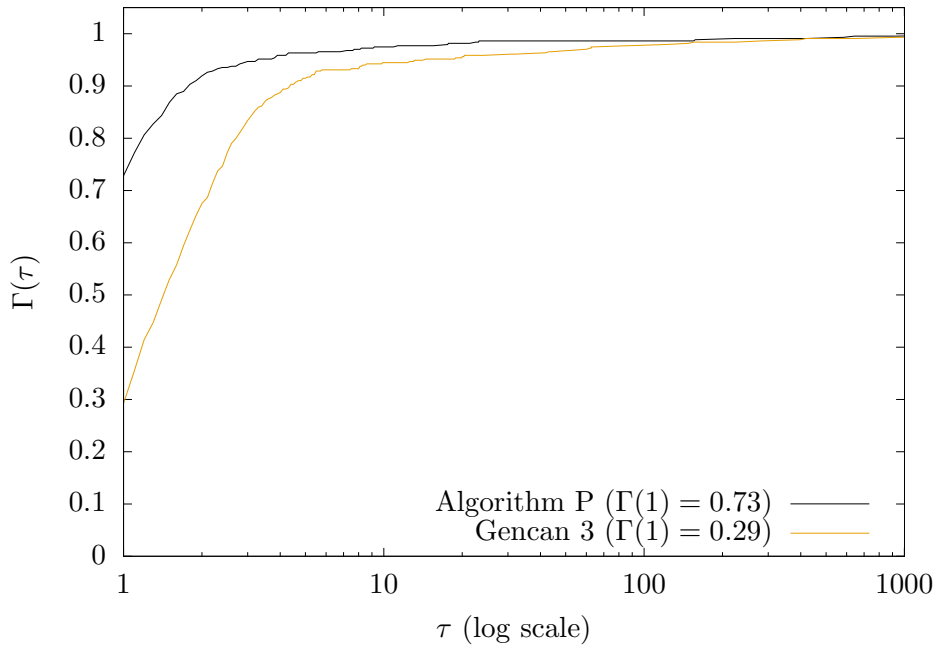


Figure 2: Performance profiles comparing the efficiency of Algorithm P and Gencan on the 434 problems where the two methods found equivalent objective function values with tolerance $f_{\text{tol}} = 0.1$, taking into account all the 475 unconstrained problems and bound-constrained problems from the CUTEst collection.

# 4 Conclusion

Recent work has analyzed the practical and theoretical properties of the well-known MINRES method for solving linear systems, particularly in the context of a truncated Newton method (Newton-MR) for unconstrained minimization. In this paper, we extended the Newton-MR method in two distinct ways. In one approach, we preserved the worst-case complexity of $\mathcal{O}(\epsilon^{-3/2})$ exhibited by Newton-MR for unconstrained minimization. In the other approach, inspired by Gencan and guided by numerical evaluations of various alternatives, we developed an extension of Newton-MR for bound-constrained minimization with worst-case complexity of $\mathcal{O}(\epsilon^{-2})$. Numerical experiments demonstrated that the latter approach is more robust and efficient than the former, when considering both unconstrained and bound-constrained problems from the CUTEst collection. A similar conclusion is reached when only unconstrained problems are considered. On the one hand, it can be argued that worst-case complexity does not always accurately reflect a method's practical performance. On the other hand, it is important to note that the method with lower complexity requires much stronger assumptions than the method with higher complexity. These stronger assumptions are difficult to verify in practice. The best of the two methods was also compared with Gencan, a method with similar characteristics but that solves linear systems using conjugate gradients. The new method proved to be more robust and efficient. As future work, it remains to be seen whether this advantage holds when the method is used to solve subproblems in an augmented Lagrangian method. Another avenue for future research would be to explore a different paradigm, such as nonlinear conjugate gradients, to solve augmented Lagrangian subproblems.

# References


[1] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. Augmented Lagrangian methods under the constant positive linear dependence constraint qualification. *Mathematical Programming*, 111(1-2):5–32, 2006.

[2] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. On augmented Lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18(4):1286–1309, 2008.

[3] M. Andretta, E. G. Birgin, and J. M. Martínez. Practical active-set euclidian trust-region method with spectral projected gradients for bound-constrained minimization. *Optimization*, 54(3):305–325, 2005.

[4] E. G. Birgin and J. M. Gentil. Evaluating bound-constrained minimization software. *Computational Optimization and Applications*, 53(2):347–373, 2012.

[5] E. G. Birgin and J. M. Martínez. A box-constrained optimization algorithm with negative curvature directions and spectral projected gradients. In G. Alefeld and X. Chen, editors, *Topics in Numerical Analysis. Computing Supplementa, vol 15*, pages 49–60. Springer, Vienna, 2001.

[6] E. G. Birgin and J. M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Computational Optimization and Applications*, 23(1):101–125, 2002.

[7] E. G. Birgin and J. M. Martínez. *Practical Augmented Lagrangian Methods for Constrained Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[8] E. G. Birgin and J. M. Martínez. On regularization and active-set methods with complexity for constrained optimization. *SIAM Journal on Optimization*, 28(2):1367–1395, 2018.

[9] E. G. Birgin and J. M. Martínez. Complexity and performance of an augmented Lagrangian algorithm. *Optimization Methods and Software*, 35(5):885–920, 2020.

[10] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

[11] E. G. Birgin, J. M. Martínez, and M. Raydan. Algorithm 813: SPG–Software for convex-constrained optimization. *ACM Transactions on Mathematical Software*, 27(3):340–349, 2001.

[12] E. G. Birgin, J. M. Martínez, and M. Raydan. Inexact spectral projected gradient methods on convex sets. *IMA Journal of Numerical Analysis*, 23(4):539–559, 2003.

[13] E. G. Birgin, J. M. Martínez, and M. Raydan. Spectral projected gradient methods: Review and perspectives. *Journal of Statistical Software*, 60(3), 2014.

[14] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[15] T. F. Coleman and Y. Li. On the convergence of interior-reflective Newton methods for nonlinear minimization subject to bounds. *Mathematical Programming*, 67(1–3):189–224, 1994.

[16] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):418–445, 1996.

[17] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433–460, 1988.

[18] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, 50(182):399–430, 1988.

[19] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Lancelot: A Fortran package for large scale nonlinear optimization*. Springer-Verlag, Berlin, Heidelberg, 1992.

[20] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–23, 1943.

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

[22] F. Facchinei, J. Júdice, and J. Soares. An active set Newton algorithm for large-scale nonlinear programs with box constraints. *SIAM Journal on Optimization*, 8(1):158–186, 1998.

[23] F. Facchinei and S. Lucidi. A class of penalty functions for optimization problema with bound constraints. *Optimization*, 26(3–4):239–259, 1992.

[24] F. Facchinei, S. Lucidi, and L. Palagi. A truncated Newton algorithm for large scale box constrained optimization. *SIAM Journal on Optimization*, 12(4):1100–1125, 2002.

[25] A. Friedlander, J. M. Martínez, and S. A. Santos. A new trust region algorithm for bound constrained minimization. *Applied Mathematics & Optimization*, 30(3):235–266, 1994.

[26] A. A. Goldstein. Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, 70(5):709–710, 1964.

[27] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2014.

[28] W. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIAM Journal on Optimization*, 17(2):526–557, 2006.

[29] M. Heinkenschloss, M. Ulbrich, and S. Ulbrich. Superlinear and quadratic convergence of affine-scaling interior-point Newton methods for problems with simple bounds without strict complementarity assumption. *Mathematical Programming*, 86(3):615–635, 1999.

[30] M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[31] M. Lescrenier. Convergence of trust region algorithms for optimization with bounds when strict complementarity does not hold. *SIAM Journal on Numerical Analysis*, 28(2):476–495, 1991.

[32] E. S. Levitin and B. T. Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966.

[33] C.-J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

[34] Y. Liu and F. Roosta. Convergence of Newton-MR under inexact hessian information. *SIAM Journal on Optimization*, 31(1):59–90, 2021.

[35] Y. Liu and F. Roosta. MINRES: From negative curvature detection to monotonicity properties. *SIAM Journal on Optimization*, 32(4):2636–2661, 2022.

[36] Y. Liu and F. Roosta. A Newton-MR algorithm with complexity guarantees for nonconvex smooth unconstrained optimization. Technical report, arXiv:2208.07095v2, 2023.

[37] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[38] J. L. Morales and J. Nocedal. Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software*, 38(1):1–4, 2011.

[39] Q. Ni and Y. Yuan. A subspace limited memory quasi-Newton algorithm for large-scale nonlinear bound constrained optimization. *Mathematics of Computation*, 66(220):1509–1520, 1997.

[40] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, 1st edition, 2006.

[41] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[42] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, NY, 1969.

[43] R. T. Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973.

[44] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2005.

[45] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.