J. Nonlinear Var. Anal. 10 (2026), No. 2, pp. 435-470
Available online at http://jnva.biemdas.com
https://doi.org/10.23952/jnva.10.2026.2.10

# CONVERGENCE ANALYSIS AND COMPETITIVE NUMERICAL RESULTS OF A TRUST REGION-LINE SEARCH PROJECTED EXACT PENALTY ALGORITHM

HANI AHMADZADEH, NEZAM MAHDAVI-AMIRI*

*Department of Mathematical Sciences, Sharif University of Technology, Tehran 11155-9415, Iran*

**Abstract.** We propose a trust region-line search projected exact penalty algorithm for solving constrained nonlinear optimization problems (NLPs). We make use of the well-known relationship between the solutions of NLP and the minimizers of the $\ell_1$-exact penalty function to design the algorithm. In our algorithm, whenever the current iterate is far from the feasible region, either a potential infeasible stationary point is identified, or the penalty parameter is decreased to navigate the iterates towards the feasible region. After updating the penalty parameter, a descent direction for the penalty function is computed using an approximate minimizer of a projected quadratic model of the penalty function over a trust region. In nearly feasible or feasible iterations, the step direction is determined by a combination of a horizontal step and a vertical step directions. The horizontal step direction is computed to reduce the penalty function, while the vertical step direction is calculated to preserve feasibility. Near stationarity, the Lagrange multipliers are computed by solving a linear least squares problem. If the computed Lagrange multipliers satisfy the first-order optimality conditions, a Newton step direction is computed to obtain a fast rate of convergence to a first-order point of NLP. Otherwise, a dropping step is calculated to decrease the penalty function value. The step length along the dropping step is computed using a backtracking line search strategy. We use the BFGS updating formula to update the approximate projected Hessian in local iterations. Here, we establish the global convergence of the proposed trust region-line search algorithm under conditions less stringent than those required by other available exact penalty methods. We also demonstrate a two-step superlinear local rate of convergence of the algorithm. We implement the algorithm in the MATLAB environment. Comparative numerical experiments on some test problems from the CUTEst library confirm the competitiveness of the proposed algorithm in comparison with a number of well-developed NLP solvers.

**Keywords.** Exact penalty function; Global convergence; Inexact method; Projected Hessian; Rate of convergence; Trust region.

## 1. INTRODUCTION

Design and analysis of algorithms for solving nonlinear optimization problems (NLPs) are of practical importance in various fields of science and technology. A variety of algorithms for solving NLPs employ penalty functions in various manners. A popular penalty function is the $\ell_1$-exact penalty function. The relationship between the optimal solutions of an NLP and

*Corresponding author.
E-mail address: hani.ahmadzadeh@gmail.com (H. Ahmadzadeh), nezamm@sharif.edu (N. Mahdavi-Amiri).
Received 23 April 2025; Accepted 4 September 2025; Published online 1 January 2026.

the local minimizers of its associated $\ell_1$-exact penalty function has been extensively investigated [16, 39, 51]. Different algorithms for solving NLPs have been developed based on the minimization of the $\ell_1$-exact penalty function [8, 9, 17, 18, 19, 20, 32].

Augmented Lagrangian methods offer alternative approaches for solving NLPs [4, 49]. In these methods, a sequence of unconstrained, bound-constrained, or linearly constrained optimization subproblems are approximately solved to approach a stationary point of the NLP [21, 22, 41, 52]. Unlike the exact penalty function, the augmented Lagrangian function is smooth but requires an estimate of Lagrange multipliers. Both the $\ell_1$-exact penalty function and the augmented Lagrangian function serve as merit functions for evaluating a trial step in some constrained optimization algorithms [10, 30, 33, 34, 40, 52, 53].

Coleman and Conn [17, 18, 19] devised an $\ell_1$-exact penalty method for solving constrained nonlinear optimization problems. Furthermore, Mahdavi-Amiri and Bartels [43] introduced an exact penalty algorithm featuring projected structured quasi-Newton updates, aimed at addressing general constrained nonlinear least squares problems. Their approach was inspired by the work of Nocedal and Overton [50] on projected Hessian approximation. More recently, Ansari and Mahdavi-Amiri [3] proposed a competitive combined trust region-line search exact penalty projected structured algorithm specifically tailored for solving constrained nonlinear least squares problems. The approach was also extended to embody the Broyden family update formulas for the structured projected Hessians; see Bidabadi and Mahdavi-Amiri [5].

Here, we design a trust region-line search projected exact penalty method for solving NLPs based on the approach of Coleman and Conn. Our algorithm employs a reliable and resilient approach to updating the penalty parameter, achieved simply by computing Cauchy points of the trust region subproblems. Furthermore, the step direction is consistently computed in a structured manner across all iterations. Moreover, we propose a new strategy for evaluating and updating the iterates. We establish the global convergence of the proposed algorithm using a different approach as compared to the global convergence analysis of [18] for the Coleman and Conn method. Moreover, our approach relies on weaker assumptions. Also, we show that our proposed algorithm exhibits a two-step superlinear local rate of convergence.

The proposed algorithm is implemented in the MATLAB environment and tested on some test problems from CUTEst library [35]. The experimental results confirm the efficiency of the proposed algorithm as compared to the four well-known software packages: KNITRO/Active-Set [8, 11], SNOPT [30], filterSQP [29] and LANCELOT [21, 22].

The remainder of our work is summarized as follows. In Section 2, our algorithm is presented. The investigation into the global convergence of the algorithm is conducted in Section 3. In Section 4, we establish a two-step superlinear local rate of convergence for our algorithm. We report some comparative numerical results in Section 5 to show the competitiveness of our algorithm. Finally, we conclude in Section 6. In the remainder of this section, we present the required notations for describing the algorithm and analysing its convergence.

**Notations:** Throughout the paper, $\mathbb{R}$ is the set of real numbers. The cardinality of a set $\mathscr{S}$ is denoted by $|\mathscr{S}|$. The $j$th component of a vector $a \in \mathbb{R}^n$ is shown by $[a]_j$. An identity matrix is shown by $I$, and the vector $e_j$ denotes the $j$th column of an identity matrix. An array with all entries equal to zero is represented by $\mathbf{0}$, while an array with all entries being equal to one is shown by $\mathbf{1}$. The size of arrays $\mathbf{0}$, $\mathbf{1}$, and an identity matrix will be clear from the context. Considering two vectors $a$ and $b$ in the same Euclidean space, the notation $a \leq b$

$(a < b)$ means that $[a]_j \leq [b]_j$ ($[a]_j < [b]_j$), for all $j$. Moreover, the $j$th entry of vectors $c = \max\{a,b\}$ and $d = \min\{a,b\}$ are defined as $[c]_j = \max\{[a]_j,[b]_j\}$ and $[d]_j = \min\{[a]_j,[b]_j\}$, respectively. The positive part of a vector $a$ is denoted by $[a]^+ = \max\{a,\mathbf{0}\}$ and its negative part is $[a]^- = \max\{-a,\mathbf{0}\}$. The $p$-norm of a vector and the induced $p$-norm of a matrix are denoted by $\|\cdot\|_p$, where $p = 1,2,\infty$. The sign function, $\mathrm{sgn}(x)$, returns 1 when $x$ is non-negative ($x \geq 0$), and $-1$ when $x$ is negative ($x < 0$). For two vanishing sequences $\{a_k\}$ and $\{b_k\}$ in $\mathbb{R}^n$, the notation $b_k = \mathscr{O}(a_k)$ means that there exists a constant $c > 0$, such that $\|b_k\|_2 \leq c\|a_k\|_2$, for all sufficiently large $k$. Moreover, the notation $b_k = o(a_k)$ signifies that $\|b_k\|_2/\|a_k\|_2 \to 0$.

## 2. ALGORITHM

Here, we present a trust region projected exact penalty algorithm for solving the constrained nonlinear optimization problem,

$$\begin{aligned} &\min_x \ f(x) \\ &\text{s.t. } \ c_i(x) \geq 0, \quad i = 1,\ldots,m, \end{aligned} \tag{NLP}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1,\ldots,m$, are twice continuously differentiable functions. The optimality conditions of (NLP) could be expressed using its *Lagrangian function* defined as $\mathscr{L}(x,y) := f(x) - \sum_{i=1}^m y_i c_i(x)$, where $y = \begin{bmatrix} y_1 & \cdots & y_m \end{bmatrix}^T \in \mathbb{R}^m$ is the Lagrange multiplier vector. By defining the *KKT (Karush-Kuhn-Tucker) residual* of (NLP) as

$$\Psi(x,y) := \begin{bmatrix} \nabla_x \mathscr{L}(x,y) \\ \min\{c(x),y\} \end{bmatrix},$$

if $\Psi(x,y) = \mathbf{0}$, then $(x,y)$ is termed a *KKT pair* of (NLP), and $x$ is called a *KKT point* of (NLP). The *active set* at a point $x \in \mathbb{R}^n$ is defined as

$$\mathrm{AC}(x) := \{i = 1,\ldots,m \ : \ c_i(x) = 0\}.$$

It is known that if $x$ is a local solution of (NLP) satisfying a constraint qualification, then $x$ is a KKT point of (NLP). We use the following practical constraint qualification throughout this paper.

**Definition 2.1** (LICQ)**.** The *linear independence constraint qualification* (LICQ) holds at a point $x$ if the set of active constraint gradients at that point, $\{\nabla_x c_i(x) \ : \ i \in \mathrm{AC}(x)\}$, is linearly independent.

The *constraint violation function*, defined as $h(x) := \sum_{i=1}^m \max\{-c_i(x),0\}$, measures the infeasibility of point $x$. Then, the $\ell_1$-*exact penalty function* for (NLP) is denoted as $p(x;\mu) := \mu f(x) + h(x)$, with $\mu > 0$ representing the *penalty parameter*. It is a well-established result that if $x^*$ is a strong local solution of (NLP) and satisfies the LICQ, then there exists a threshold value $\mu^* > 0$ such that $x^*$ remains a local minimizer of $p(x;\mu)$ for all $\mu \in (0,\mu^*]$; see [51]. Under suitable conditions, an appropriate choice for the threshold $\mu^*$ would be $\mu^* = 1/\|y^*\|_\infty$, where $y^*$ denotes the Lagrange multiplier vector such that $\Psi(x^*,y^*) = \mathbf{0}$ [14, 15, 42]. On the other hand, suppose $x^*$ is a stationary point of $p(x;\mu)$ for all $\mu \in (0,\mu^*]$, with $\mu^* > 0$. If $h(x^*) = 0$, then $x^*$ is a KKT point for problem (NLP). However, if $h(x^*) > 0$, then $x^*$ is an infeasible stationary point of $h(x)$; see [12, Theorem 4.1].

**Definition 2.2.** [49, Definition 17.1] A point $x^*$ is termed an *infeasible stationary point* of the constraint violation function $h(x)$ if $h(x^*) > 0$ and it serves as a stationary point for the function $h(x)$, meaning that

$$D_d h(x^*) := \lim_{t \downarrow 0^+} \frac{h(x^* + td) - h(x^*)}{t} \geq 0,$$

for any unit vectors $d \in \mathbb{R}^n$.

Considering the connection between solutions of (NLP) and minimizers of its associated penalty function, various algorithms were designed to solve constrained optimization problems [17, 18, 19, 20, 23, 32, 33, 36]. Additionally, several exact penalty methods were adapted to solve constrained nonlinear least squares problems [3, 6, 7, 43, 44, 45, 46, 47].

In practice, for a point $x \in \mathbb{R}^n$ and an activity tolerance parameter $\varepsilon \geq 0$, the *$\varepsilon$-active* and *$\varepsilon$-violated index sets* are specified by

$$\mathrm{AC}(x, \varepsilon) := \{i = 1, \ldots, m \ : \ |c_i(x)| \leq \varepsilon\},$$
$$\mathrm{VC}(x, \varepsilon) := \{i = 1, \ldots, m \ : \ c_i(x) < -\varepsilon\},$$

respectively.

Introducing the *$\varepsilon$-constraint violation function* as $h_\varepsilon(x) := -\sum_{i \in \mathrm{VC}(x,\varepsilon)} c_i(x)$, the *$\varepsilon$-active penalty function* [3, 17, 43] is defined as $p_\varepsilon(x; \mu) := \mu f(x) + h_\varepsilon(x)$. The subsequent proposition encapsulates some important properties of the $\varepsilon$-active and $\varepsilon$-violated index sets, the $\varepsilon$-constraint violation and $\varepsilon$-active penalty functions.

**Proposition 2.1.** *Given $x \in \mathbb{R}^n$ and $\mu > 0$, the following assertions hold.*

*(1) There exists a thereshold value $\hat{\varepsilon} > 0$ so that $\mathrm{AC}(x, \varepsilon) = \mathrm{AC}(x, 0) = \mathrm{AC}(x)$, and $\mathrm{VC}(x, \varepsilon) = \mathrm{VC}(x, 0)$, for all $\varepsilon \in [0, \hat{\varepsilon}]$.*
*(2) There exists $\delta > 0$ such that $\mathrm{VC}(x + s, \varepsilon) = \mathrm{VC}(x, \varepsilon)$, for all $\varepsilon > 0$ and all $s$ with $\|s\|_2 \leq \delta$.*
*(3) We have $h_0(x) = h(x)$ and $p_0(x; \mu) = p(x; \mu)$.*
*(4) Functions $p_\varepsilon(x; \mu)$ and $h_\varepsilon(x)$ are twice continuously differentiable, for all $\varepsilon \geq 0$.*
*(5) If $\bar{\varepsilon} \geq \hat{\varepsilon} \geq 0$, then $\mathrm{AC}(x, \hat{\varepsilon}) \subseteq \mathrm{AC}(x, \bar{\varepsilon})$.*
*(6) If $\bar{\varepsilon} \geq \hat{\varepsilon} \geq 0$, then $\mathrm{VC}(x, \bar{\varepsilon}) \subseteq \mathrm{VC}(x, \hat{\varepsilon})$.*
*(7) If $\bar{\varepsilon} \geq \hat{\varepsilon} \geq 0$, then $h_{\bar{\varepsilon}}(x) \leq h_{\hat{\varepsilon}}(x)$ and $p_{\bar{\varepsilon}}(x; \mu) \leq p_{\hat{\varepsilon}}(x; \mu)$.*

*Proof.* The proof directly stems from the definitions of $\varepsilon$-active and $\varepsilon$-violated index sets, the continuity of both functions $f$ and $c$, alongside the definitions of $h_\varepsilon(x)$ and $p_\varepsilon(x; \mu)$. □

The first-order necessary optimality conditions as articulated by Coleman and Conn [16] assert that if $x^*$ serves as a local minimizer of $p(x; \mu)$ for a given $\mu > 0$, and it adheres to the LICQ, then there exist multipliers $\lambda_i^*$, $i \in \mathrm{AC}(x^*)$, satisfying

$$\nabla_x p_0(x^*; \mu) = \sum_{i \in \mathrm{AC}(x^*)} \lambda_i^* \nabla_x c_i(x^*), \tag{2.1}$$
$$0 \leq \lambda_i^* \leq 1, \quad i \in \mathrm{AC}(x^*). \tag{2.2}$$

A point $x^*$ that fulfills (2.1) for certain multipliers $\lambda_i^*$, $i \in \mathrm{AC}(x^*)$ is referred to as a *stationary point* of $p(x; \mu)$. Furthermore, if the multipliers $\lambda_i^*$, $i \in \mathrm{AC}(x^*)$ also conform to (2.2), then $x^*$ is referred to as a *first-order point of* $p(x; \mu)$. If $x^*$ is a first-order point of $p(x; \mu)$ with the strict satisfaction of inequality (2.2), then it is denoted as a *strict first-order point* of $p(x; \mu)$ [18].

Additionally, in accordance with the second-order necessary conditions of Coleman and Conn [16], if $x^*$ serves as a local minimizer of $p(x;\mu)$, which satisfies LICQ, then, for all $d \in \mathbb{R}^n$ satisfying

$$d^T \nabla_x c_i(x^*) = 0, \quad \forall i \in \mathrm{AC}(x^*), \tag{2.3}$$

we have

$$d^T \left[ \nabla^2_{xx} p_0(x^*;\mu) - \sum_{i \in \mathrm{AC}(x^*)} \lambda_i^* \nabla^2_{xx} c_i(x^*) \right] d \geq 0, \tag{2.4}$$

where $\lambda_i^*$, $i \in \mathrm{AC}(x^*)$, represent specific scalars that fulfill both (2.1) and (2.2). The point $x^*$ is designated as a *second-order point* of $p(x;\mu)$ if there exist multipliers $\lambda_i^*$, $i \in \mathrm{AC}(x^*)$, for which (2.1), (2.2), and (2.4) hold for every $d$ satisfying (2.3). Moreover, $x^*$ is called a *strict second-order point* of $p(x;\mu)$ if it is a strict first-order point of $p(x;\mu)$ and inequality (2.4) holds strictly for every $d \neq \mathbf{0}$ satisfying (2.3) [18].

Now, we are ready to start the description of our algorithm. Let $x_k$ be the acquired estimate for a local minimizer of $p(x;\mu_{k-1})$ at the $k$th iteration of the algorithm, where $\mu_{k-1} > 0$ is a penalty parameter. During the $k$th iteration, with $\varepsilon_k > 0$ representing the activity tolerance, we determine the $\varepsilon$-active index set $\mathrm{AC}(x_k, \varepsilon_k)$, the $\varepsilon$-violated index set $\mathrm{VC}(x_k, \varepsilon_k)$, and construct the normal matrix $A_k := \begin{bmatrix} \cdots & \nabla_x c_i(x_k) & \cdots \end{bmatrix}_{i \in \mathrm{AC}(x_k, \varepsilon_k)}$. Utilizing the QR decomposition of $A_k$, it is possible to compute normal orthogonal (orthonormal) bases for both the null space of $A_k^T$, $N(A_k^T)$, and the range space of $A_k$, $R(A_k)$. Actually, considering the QR decomposition of the full column rank matrix $A_k \in \mathbb{R}^{n \times t_k}$ as

$$A_k = \begin{bmatrix} Y_k & Z_k \end{bmatrix} \begin{bmatrix} R_k \\ \mathbf{0} \end{bmatrix}, \tag{2.5}$$

where $t_k$ is the cardinality of $\mathrm{AC}(x_k, \varepsilon_k)$, $Y_k \in \mathbb{R}^{n \times t_k}$, $Z_k \in \mathbb{R}^{n \times (n-t_k)}$, $Y_k^T Y_k = I$, $Z_k^T Z_k = I$, $Y_k^T Z_k = \mathbf{0}$, and $R_k \in \mathbb{R}^{t_k \times t_k}$ is a nonsingular uper triangular matrix, we have

$$N(A_k^T) = R(Z_k) := \left\{ Z_k d \ : \ d \in \mathbb{R}^{n-t_k} \right\}, \tag{2.6}$$

$$R(A_k) = R(Y_k) := \left\{ Y_k d \ : \ d \in \mathbb{R}^{t_k} \right\}. \tag{2.7}$$

Furthermore, in the vicinity of a first-order point of (NLP), it is possible to approximate the Lagrange multipliers by solving the following linear least squares problem:

$$\min \quad \tfrac{1}{2} \| A_k \lambda - \nabla_x p_{\varepsilon_k}(x_k;\mu_k) \|_2^2 \ , \tag{2.8}$$

using the QR decomposition of $A_k$.

Iterations of the algorithm are categorized into "infeasible", "almost feasible", and "local" iterations. he $k$th iteration is deemed to be an *infeasible iteration* whenever the set of violated constraints, $\mathrm{VC}(x_k, \varepsilon_k)$, is non-empty, or, in other words, when $h_{\varepsilon_k}(x_k)$ is positive. The $k$th iteration is designated as an *almost feasible iteration* if $h_{\varepsilon_k}(x_k) = 0$ and $\| Z_k^T \nabla_x p_{\varepsilon_k}(x_k;\mu_k) \|_2 > \tau$, where $\tau \geq 0$ is a stationary tolerance parameter. Moreover, the $k$th iteration is termed a *local iteration* if $h_{\varepsilon_k}(x_k) = 0$ and $\| Z_k^T \nabla_x p_{\varepsilon_k}(x_k;\mu_k) \|_2 \leq \tau$. Our proposed algorithm employs distinct strategies for computing the search direction based on whether the $k$th iteration is infeasible, almost feasible, or local.

In infeasible iterations, we have

$$h(x_k) = h_0(x_k) \geq h_{\varepsilon_k}(x_k) > \Gamma_k \varepsilon_k > 0, \tag{2.9}$$

where $\Gamma_k = |\mathrm{VC}(x_k, \varepsilon_k)| > 0$ represents the count of violated constraints. In light of (2.9), we can infer that $x_k$ is considerably distant from the feasible region. Hence, in the course of infeasible iterations, our aim is to identify whether (NLP) is locally infeasible or to diminish the value of the constraint violation function.

In a manner similar to the methods outlined in [1, 2, 9, 10, 33], our algorithm identifies (NLP) as locally infeasible by recognizing an *infeasible stationary point*. Later, Lemma 3.1 articulates the conditions for recognizing an infeasible stationary point being employed in our algorithm. When $\nabla_x h_{\varepsilon_k}(x_k) = \mathbf{0}$ and $\mathrm{VC}(x_k, \varepsilon_k) = \mathrm{VC}(x_k, 0) \neq \emptyset$, $x_k$ is identified as an infeasible stationary point, leading to the termination of the algorithm. Conversely, if $\nabla_x h_{\varepsilon_k}(x_k) = \mathbf{0}$ but $\mathrm{VC}(x_k, \varepsilon_k) \neq \mathrm{VC}(x_k, 0)$, reducing the value of $\varepsilon_k$ can lead to $\nabla_x h_{\varepsilon_k}(x_k) \neq \mathbf{0}$, or eventually, $\mathrm{VC}(x_k, \varepsilon_k) = \mathrm{VC}(x_k, 0)$. It should be noted that, if $\nabla_x h_{\varepsilon_k}(x_k) \neq \mathbf{0}$ but $Z_k^T \nabla_x h_{\varepsilon_k}(x_k) = \mathbf{0}$, we could drop certain indices from the active-set to ensure $Z_k^T \nabla_x h_{\varepsilon_k}(x_k) \neq \mathbf{0}$. Whenever $Z_k^T \nabla_x h_{\varepsilon_k}(x_k) \neq \mathbf{0}$, it is possible to decrease the constraint violation. To accomplish this, the *feasibility projected trust region subproblem* is considered as follows:

$$\begin{aligned} \min_{w} \quad & \check{h}_{\varepsilon_k}(w; D_k, x_k) := \frac{1}{2} w^T D_k w + w^T g_k^{\mathrm{h}} + h_{\varepsilon_k}(x_k) \\ \text{s.t.} \quad & \|w\|_2 \leq \delta_k, \end{aligned} \tag{hTR}$$

where $\delta_k > 0$ is a trust region radius, $g_k^{\mathrm{h}} := Z_k^T \nabla_x h_{\varepsilon_k}(x_k)$, and $D_k$ is a symmetric approximation of $Z_k^T \nabla_{xx}^2 h_{\varepsilon_k}(x_k) Z_k$. Then, the *Cauchy point* of (hTR), $w_k^{\mathrm{FC}}$, is calculated as

$$\begin{aligned} w_k^{\mathrm{FC}} &:= \alpha_k^{\mathrm{FC}} w_k^{\mathrm{FS}}, \\ w_k^{\mathrm{FS}} &:= -\frac{\delta_k}{\|g_k^{\mathrm{h}}\|_2} g_k^{\mathrm{h}}, \\ \alpha_k^{\mathrm{FC}} &:= \arg\min_{\alpha \in [0,1]} \check{h}_{\varepsilon_k}\left(\alpha w_k^{\mathrm{FS}}; D_k, x_k\right) \\ &= \begin{cases} 1, & \left(g_k^{\mathrm{h}}\right)^T D_k g_k^{\mathrm{h}} \leq 0, \\ \min\left\{1, \frac{\|g_k^{\mathrm{h}}\|_2^3}{\delta_k \left(\left(g_k^{\mathrm{h}}\right)^T D_k g_k^{\mathrm{h}}\right)}\right\}, & \left(g_k^{\mathrm{h}}\right)^T D_k g_k^{\mathrm{h}} > 0. \end{cases} \end{aligned}$$

Moreover, given a penalty parameter $\mu > 0$, the *penalty projected trust region subproblem* is formulated as

$$\begin{aligned} \min_{w} \quad & \check{p}_{\varepsilon_k}(w; H_k(\mu), x_k, \mu) := \frac{1}{2} w^T H_k(\mu) w + w^T g_k^{\mathrm{p}}(\mu) + p_{\varepsilon_k}(x_k; \mu) \\ \text{s.t.} \quad & \|w\|_2 \leq \delta_k. \end{aligned} \tag{pTR($\mu$)}$$

In (pTR($\mu$)), we have $g_k^{\mathrm{p}}(\mu) := Z_k^T \nabla_x p_{\varepsilon_k}(x_k; \mu)$ and $H_k(\mu) := \mu B_k + D_k$, where $B_k$ is a symmetric approximation of $Z_k^T \nabla_{xx}^2 f(x_k) Z_k$. The Cauchy point of (pTR($\mu$)), $w_k^{\mathrm{PC}}(\mu)$, is computed as

follows:

$$
\begin{aligned}
w_k^{\mathrm{PC}}(\mu) &:= \alpha_k^{\mathrm{PC}}(\mu) w_k^{\mathrm{PS}}(\mu),\\
w_k^{\mathrm{PS}}(\mu) &:= -\frac{\delta_k}{\|g_k^{\mathrm{p}}(\mu)\|_2} g_k^{\mathrm{p}}(\mu),\\
\alpha_k^{\mathrm{PC}}(\mu) &:= \arg\min_{\alpha\in[0,1]} \check{p}_{\varepsilon_k}\left(\alpha w_k^{\mathrm{PS}}(\mu); H_k(\mu), x_k, \mu\right)\\
&= \begin{cases} 1, & \left(g_k^{\mathrm{p}}(\mu)\right)^T H_k(\mu) g_k^{\mathrm{p}}(\mu) \le 0,\\ \min\left\{1, \dfrac{\|g_k^{\mathrm{p}}(\mu)\|_2^3}{\delta_k\left(\left(g_k^{\mathrm{p}}(\mu)\right)^T H_k(\mu) g_k^{\mathrm{p}}(\mu)\right)}\right\}, & \left(g_k^{\mathrm{p}}(\mu)\right)^T H_k(\mu) g_k^{\mathrm{p}}(\mu) > 0. \end{cases}
\end{aligned}
$$

At the $k$th iteration of our algorithm, we intentionally select the penalty parameter $\mu_k > 0$ to be sufficiently small, ensuring the satisfaction of the following conditions:

$$\Delta\check{h}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu_k); D_k, x_k\right) \ge \eta_1 \Delta\check{h}_{\varepsilon_k}\left(w_k^{\mathrm{FC}}; D_k, x_k\right), \tag{2.10a}$$

$$\Delta\check{p}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu_k); H_k(\mu_k), x_k, \mu_k\right) \ge \eta_2 \Delta\check{h}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu_k); D_k, x_k\right), \tag{2.10b}$$

where $\eta_1, \eta_2 \in (0,1)$ are predetermined constants. Here,

$$\Delta\check{h}_{\varepsilon_k}(v; D_k, x_k) := h_{\varepsilon_k}(x_k) - \check{h}_{\varepsilon_k}(v; D_k, x_k), \tag{2.11}$$

$$\Delta\check{p}_{\varepsilon_k}(v; H_k(\mu_k), x_k, \mu_k) := p_{\varepsilon_k}(x_k; \mu_k) - \check{p}_{\varepsilon_k}(v; H_k(\mu_k), x_k, \mu_k), \tag{2.12}$$

are quadratic predicted changes in the constraint violation and penalty functions, respectively. In Lemma 3.2, we can see that conditions (2.10a) and (2.10b) hold for sufficiently small positive values of $\mu_k$. Two similar conditions were proposed by Ansari and Mahdavi-Amiri [3], in which the Cauchy points in (2.10a) and (2.10b) are replaced with the optimal solutions of trust region subproblems (hTR) and (pTR($\mu$)). Also, similar conditions are used in several penalty successive linear-quadratic programming (SLQP) algorithms [8, 9, 12, 33]. In these SLQP algorithms, solving several linear programs may be required, which turn to be very costly.

Once a suitable penalty parameter $\mu_k > 0$ fulfilling conditions (2.10a) and (2.10b) has been determined, the penalty projected trust region subproblem (pTR($\mu$)) with $\mu = \mu_k$ is approximately solved to obtain a step $w_k$ satisfying

$$\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) \ge \Delta\check{p}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu_k); H_k(\mu_k), x_k, \mu_k\right). \tag{2.13}$$

An approximate solution $w_k$ of (pTR($\mu$)) with $\mu = \mu_k$ meeting the condition (2.13) could be derived using various methods such as the Dogleg strategy of [54], the two-dimensional subspace minimization technique of [13], the truncated conjugate gradient method of [55, 56], or the Lanczos method of [37].

In an infeasible iteration, let $\mu_k$ be the penalty parameter satisfying conditions (2.10a) and (2.10b), and let $w_k$ be an approximate solution of (pTR($\mu$)) with $\mu = \mu_k$ satisfying (2.13). Then, the step direction is determined as $d_k := Z_k w_k + \beta_k d_k^{\mathrm{v}}$, where the vertical step direction $d_k^{\mathrm{v}}$ is simply set to zero, and $\beta_k = 1$. When $h_{\varepsilon_k}(x_k) = 0$ and $\|Z_k^T \nabla_x p_{\varepsilon_k}(x_k; \mu_k)\|_2 > \tau$, the $k$th iteration is deemed almost feasible. In such iterations, given that $\mathrm{VC}(x_k, \varepsilon_k) = \emptyset$, the matrix $D_k$ is treated as zero, leading to $\check{h}_{\varepsilon_k}(w; D_k, x_k) = 0$, for all $w$. As a consequence, there is no necessity to decrease the penalty parameter in almost feasible iterations; it remains unchanged ($\mu_k \leftarrow \mu_{k-1}$). Then, $w_k$ satisfying (2.13) is calculated as an approximate solution of (pTR($\mu$)) with $\mu = \mu_k$,

and the horizontal step direction is set to be $d_k^{\mathrm{h}} := Z_k w_k$. Additionally, the vertical step direction is determined via a single Newton step applied to the system of nonlinear equations given by

$$[c(x_k + Z_k w_k + d)]_{\mathrm{AC}(x_k,\varepsilon_k)} = \mathbf{0}.$$

This involves computing the minimum norm solution of the following linear system of equations:

$$A_k^T d = -[c(x_k + Z_k w_k)]_{\mathrm{AC}(x_k,\varepsilon_k)}. \tag{2.14}$$

Assuming that $A_k$ is a full column rank matrix, the minimum norm solution of (2.14) is computed as:

$$\begin{aligned} d_k^{\mathrm{v}} &:= -A_k \left(A_k^T A_k\right)^{-1} [c(x_k + Z_k w_k)]_{\mathrm{AC}(x_k,\varepsilon_k)} \\ &= -Y_k R_k^{-T} [c(x_k + Z_k w_k)]_{\mathrm{AC}(x_k,\varepsilon_k)}. \end{aligned} \tag{2.15}$$

Subsequently, the step direction is determined as $d_k := Z_k w_k + \beta_k d_k^{\mathrm{v}}$, with

$$\beta_k := \begin{cases} \min\left\{\frac{\delta_k}{\|d_k^{\mathrm{v}}\|_2}, 1\right\}, & \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}} \le 0, \\ \min\left\{\eta \frac{\Delta \check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k)}{\nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}}}, \frac{\delta_k}{\|d_k^{\mathrm{v}}\|_2}, 1\right\}, & \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}} > 0, \end{cases} \tag{2.16}$$

where $\eta \in (0,1)$ is a prescribed constant. It is important to highlight that choosing $\beta_k$ as given in (2.16) ensures that $\|d_k\|_2 \le 2\delta_k$ and the "predicted reduction in the penalty function" is positive; see Lemma 3.4.

In infeasible and almost feasible iterations of our algorithm, after computing the step direction as $d_k = Z_k w_k + \beta_k v_k$ with $w_k \neq \mathbf{0}$ as an approximate solution of (pTR($\mu$)) and $v_k \in \{\mathbf{0}, d_k^{\mathrm{v}}\}$, the trial point is established to be $x_k^+ := x_k + d_k$. To evaluate the step direction $d_k$, or equivalently, the trial point, it is necessary to define an estimation of the changes in the penalty function. In light of Lemma 3.3, we define the *predicted reduction* as

$$\mathrm{Pred}_k := \Delta \check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) - \beta_k \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) v_k, \tag{2.17}$$

where $\Delta \check{p}_{\varepsilon_k}(\,\cdot\,; H_k(\mu_k), x_k, \mu_k)$ is defined as (2.12). Lemma 3.4 states that selecting $\beta_k$ as (2.16) assures that $\mathrm{Pred}_k > 0$. Also, we define the ratio of the actual reduction to the predicted reduction as:

$$\rho_k := \frac{p(x_k;\mu_k) - p(x_k + d_k;\mu_k)}{\mathrm{Pred}_k}, \tag{2.18}$$

to evaluate the trial point.

For a predetermined constant $\rho^{\mathrm{s}} \in (0,1)$, the $k$th iteration is called *successful* if $\rho_k \ge \rho^{\mathrm{s}}$; otherwise ($\rho_k < \rho^{\mathrm{s}}$), it is called *failed*. More specifically, the $k$th iteration is called *very successful*, whenever $\rho_k \ge \rho^{\mathrm{vs}}$, where $\rho^{\mathrm{vs}} \ge \rho^{\mathrm{s}}$ is a predefined constant. In successful iterations, the trial point is incorporated into the next iterate; i.e., $x_{k+1} \leftarrow x_k^+$. In a very successful iteration, the trust region is expanded by the factor $\gamma_e > 1$; i.e., $\delta_{k+1} \leftarrow \gamma_e \delta_k$. If the $k$th iteration is failed, then the trial point $x_k^+$ is rejected ($x_{k+1} \leftarrow x_k$), and both the trust region radius and the activity tolerance parameter are decreased by a factor $\gamma_c < 1$.

The $k$th iteration is termed a local iteration if $h_{\varepsilon_k}(x_k) = 0$ and $\|Z_k^T \nabla_x p_{\varepsilon_k}(x_k;\mu_k)\|_2 \le \tau$. In such cases, it is signified that $x_k$ resides in proximity of a stationary point. During a local iteration,

the penalty parameter is taken from the previous iteration ($\mu_k \leftarrow \mu_{k-1}$). Then, the linear least squares problem (2.8) is solved to obtain an estimate of the Lagrange multipliers as

$$\lambda_k := \left(A_k^T A_k\right)^{-1} A_k^T \nabla_x p_{\varepsilon_k}(x_k;\mu_k) = R_k^{-1} Y_k^T \nabla_x p_{\varepsilon_k}(x_k;\mu_k), \tag{2.19}$$

provided that the columns of $A_k$ are linearly independent.

If $[\lambda_k]_r \notin [0,1]$ for some $r \in \mathrm{AC}(x_k,\varepsilon_k)$, then the index $r$ is dropped from the active set and a dropping step direction is computed as:

$$d_k^{\mathrm{d}} := -\mathrm{sgn}([\lambda_k]_r) A_k \left(A_k^T A_k\right)^{-1} e_r = -\mathrm{sgn}([\lambda_k]_r) Y_k R_k^{-T} e_r, \tag{2.20}$$

where $e_r$ is the $r$th unit vector. This dropping step direction serves as the minimum norm solution of the following linear system of equations:

$$A_k^T d = -\mathrm{sgn}([\lambda_k]_r) e_r.$$

In Lemma 3.9, we will see that $d_k^{\mathrm{d}}$ is a descent direction for the penalty function. In alignment with other cases, the search direction in this case can be expressed as $d_k = Z_k w_k + \beta_k d_k^{\mathrm{d}}$, with $w_k = \mathbf{0}$ and $\beta_k = 1$. Then, employing a backtracking line search strategy, a step-length $\alpha_k \in (0,1]$ is determined such that the following Armijo-like condition is satisfied for $\alpha = \alpha_k$:

$$p(x_k + \alpha d_k^{\mathrm{d}};\mu_k) \le p(x_k;\mu_k) + c_1 e_{\lambda_k} \alpha, \tag{2.21}$$

where

$$e_{\lambda_k} := \begin{cases} [\lambda_k]_r, & \text{if } [\lambda_k]_r < 0, \\ 1 - [\lambda_k]_r, & \text{if } [\lambda_k]_r > 1, \end{cases} \tag{2.22}$$

and $c_1 \in (0,1)$ is a predetermined constant. It is worth noting that $e_{\lambda_k} < 0$. Lemma 3.9 guarantees that condition (2.21) holds for sufficiently small values of $\alpha > 0$.

If $[\lambda_k]_i \in [0,1]$ for all $i \in \mathrm{AC}(x_k,\varepsilon_k)$, then the horizontal step direction is determined as $d_k^{\mathrm{h}} := Z_k w_k$, where $w_k$ is a solution of

$$H_k(\mu_k) w = -g_k^{\mathrm{p}}(\mu_k), \tag{2.23}$$

with $H_k(\mu_k)$ as a symmetric and positive definite approximation of

$$G_k := Z_k^T \left( \mu_k \nabla_{xx}^2 f(x_k) - \sum_{i \in \mathrm{AC}(x_k,\varepsilon_k)} [\lambda_k]_i \nabla_{xx}^2 c_i(x_k) \right) Z_k. \tag{2.24}$$

Moreover, the vertical step direction $d_k^{\mathrm{v}}$ is calculated as (2.15). Afterward, the 'Newton step direction' is computed as $d_k := Z_k w_k + d_k^{\mathrm{v}}$. If we have

$$p(x_k + d_k;\mu_k) \le p(x_k;\mu_k) - c_2 \left( \|g_k^{\mathrm{p}}(\mu_k)\|_2^2 + \left\| [c(x_k)]_{\mathrm{AC}(x_k,\varepsilon_k)} \right\|_1 \right), \tag{2.25}$$

for a prespecified constant $c_2 \in (0,1)$, then the iterate is updated as $x_{k+1} \leftarrow x_k + d_k$. Otherwise, if (2.25) does not hold, then the Newton step is rejected (setting $x_{k+1} \leftarrow x_k$) and the stationary tolerance $\tau$ is reduced so that $\|g_k^{\mathrm{p}}(\mu_k)\|_2 > \tau$.

Now, we are in a position to present our new trust region-line search projected exact penalty algorithm for solving (NLP) as Algorithm 1. In our implementation of Algorithm 1, all initial approximate projected Hessian matrices are chosen to be the identity matrix. Moreover, the

trust region subproblems are solved using the CG-Steihaug method [55]. To avoid the underflow phenomenon, as in [3] we use the following functions:

$$\text{reference}_1(\varsigma, \text{val}) := \varsigma^2 + 0.1\varsigma \cdot \text{val}, \tag{2.26}$$

$$\text{reference}_2(\varsigma, x) := \text{reference}_1\left(\varsigma, \frac{\sum_{i=1}^{m} |c_i(x)|}{m+1}\right). \tag{2.27}$$

For example, the assessment of the smallness of $\|\nabla_x h_{\varepsilon_k}(x_k)\|_2$ against $\theta > 0$ in line 12 of Algorithm 1 can be made as follows:

$$\|\nabla_x h_{\varepsilon_k}(x_k)\|_2 \leq \text{reference}_1\left(\theta, \|\nabla_x h_{\varepsilon_k}(x_k)\|_2\right). \tag{2.28}$$

Indeed, condition (2.28) serves as a relative-absolute error test, determining whether $\|\nabla_x h_{\varepsilon_k}(x_k)\|_2$ exceeds $\theta$. Similarly, for determining the active sets and the violated sets, the scalar $\varepsilon_k$ is replaced by $\text{reference}_2(\varepsilon_k, x_k)$. Note that the "test optimality" and the "test feasibility" steps in line 40 of the algorithm are expected to set the logical variables *optimal* and *feasible* to **true** or **false**, appropriately. The binary variable *feasible* is set to **true** if

$$\text{VC}\left(x_k, \text{reference}_2(\gamma, x_k)\right) = \emptyset,$$

where $\gamma > 0$ is a predetermined constant ($\gamma = 10^{-14}$). Otherwise, *feasible* is set to **false**. As in [3], the optimality is detected whenever the following conditions hold:
(1) $\|g_k^{\text{p}}(\mu_k)\|_2 \leq \text{reference}_1\left(\theta, \|\nabla_x p_{\varepsilon_k}(x_k; \mu_k)\|_2\right)$,
(2) $-\theta \leq [\lambda_k]_i \leq 1 + \theta$, $\forall i \in \text{AC}(x_k, \varepsilon_k)$,
(3) $|p_{\varepsilon_k}(x_{k+1}; \mu_k) - p_{\varepsilon_k}(x_k; \mu_k)| \leq \text{reference}_1\left(\gamma, |p_{\varepsilon_k}(x_{k+1}; \mu_k)|\right)$,
(4) $\|x_{k+1} - x_k\|_2 \leq \text{reference}_1(\theta, \|x_{k+1}\|_2)$, where $\theta$ is a predesignated positive constant; for example, $\theta = 10^{-6}$.

When $\varepsilon_k > 0$ is not small enough, it is possible that $\text{AC}(x_k, \varepsilon_k) \neq \text{AC}(x_k, 0)$, $\text{VC}(x_k, \varepsilon_k) \neq \text{VC}(x_k, 0)$, and consequently $p_{\varepsilon_k}(x; \mu_k)$ may not be an adequate predictor of the behavior of $p(x; \mu_k)$. Note that the horizental step direction is computed by minimizing a quadratic model of $p_{\varepsilon_k}(x; \mu_k)$ over a trust region. Moreover, according to Lemma 3.1 to identify an infeasible stationary point, we need to have $\text{VC}(x_k, \varepsilon_k) = \text{VC}(x_k, 0)$. Therefore, at the beginning of each iteration of the algorithm we reduce $\varepsilon_k$ to have $\text{VC}(x_k, \varepsilon_k) = \text{VC}(x_k, 0)$ and $\text{AC}(x_k, \varepsilon_k) = \text{AC}(x_k, 0)$.

Our algorithm uses the idea of Nocedal and Overton [50] for approximating the "two-sided projected Hessian" matrices in lines 18, 28 and 44 of Algorithm 1. Let $B_{k-1}$, $D_{k-1}$ and $H_{k-1}(\mu_{k-1})$ be approximated projected Hessians from the previous iteration. If the set of active constraints is changed, we simply reset $B_k$, $D_k$, and $H_k(\mu_k)$ as appropriate identity matrices. Until stated otherwise, assume that the set of active constraints does not change. Since the columns of $\begin{bmatrix} Y_k & Z_k \end{bmatrix}$ form an orthonormal basis for $\mathbb{R}^n$, the vectors $q_{k-1}$ and $s_{k-1}$ could be found so that

$$x_k - x_{k-1} = Y_k q_{k-1} + Z_k s_{k-1}. \tag{2.29}$$

Consequently, from (2.29) we have:

$$s_{k-1} = Z_k^T (x_k - x_{k-1}),$$
$$q_{k-1} = Y_k^T (x_k - x_{k-1}).$$

Equality (2.29) implies that $Z_k s_{k-1} = x_k - x_{k-1} - Y_k q_{k-1}$. Provided that $q_{k-1}$ is negligible as compared to $s_{k-1}$, using the first order Taylor's approximation of $\nabla_x f(\cdot)$, $\nabla_x h_{\varepsilon_k}(\cdot)$ and

**Algorithm 1** A trust region-line search projected exact penalty algorithm.

**Parameters:** $\theta, \gamma, \mu_{\min}, \eta, \eta_1, \eta_2 > 0$, $0 < \rho^{\mathrm{s}} \leq \rho^{\mathrm{vs}} < 1$, $c_1, c_2 \in (0,1)$, and $0 < \gamma_c < 1 < \gamma_e$.
Give $x_0 \in \mathbb{R}^n$, $\delta_0 \geq 0$, $\varepsilon_0 \in (0, \delta_0^2]$, $\tau > 0$, and $\mu_0 > 0$;
set *terminated* $\leftarrow$ **false**, $k \leftarrow 0$;
**while** (**not** *terminated*) **do**
  set *updated* $\leftarrow$ **false**;
  Decrease the value of $\varepsilon_k$, if necessary, to have $\mathrm{AC}(x_k, \varepsilon_k) = \mathrm{AC}(x_k, 0)$, $\mathrm{VC}(x_k, \varepsilon_k) = \mathrm{VC}(x_k, 0)$, and $\varepsilon_k \leq \delta_k^2$;
  compute $h_{\varepsilon_k}(x_k)$ and $f(x_k)$, $\nabla_x h_{\varepsilon_k}(x_k)$ and $\nabla_x f(x_k)$;
  form the normal matrix as $A_k := \left[\cdots \quad \nabla_x c_i(x_k) \quad \cdots\right]_{i \in \mathrm{AC}(x_k, \varepsilon_k)}$;
  calculate the QR decomposition of $A_k$ as $A_k = \begin{bmatrix} Y_k & Z_k \end{bmatrix} \begin{bmatrix} R_k^T & \mathbf{0}^T \end{bmatrix}^T$;
  set $g_k^{\mathrm{h}} = Z_k^T \nabla_x h_{\varepsilon_k}(x_k)$, $g_k^{\mathrm{f}} = Z_k^T \nabla_x f(x_k)$, $g_k^{\mathrm{p}}(\mu_k) = \mu_k g_k^{\mathrm{f}} + g_k^{\mathrm{h}}$;
  **if** ($h_{\varepsilon_k}(x_k) > 0$) **then** ▷ Infeasible iteration
    **if** ($\|\nabla_x h_{\varepsilon_k}(x_k)\|_2$, tested against $\theta$, is small) **then**
      set $\lambda_k = \mathbf{0}$ and *terminated* = **true**;
      ▷ An infeasible stationary point is found
    **else if** ($\|g_k^{\mathrm{h}}\|_2$, tested against $\theta$, is small) **then**
      drop indices from the $\varepsilon$-active index set until the condition $\|g_k^{\mathrm{h}}\|_2 > 0$ is met.
    **end if**
    **if** (**not** *terminated*) **then**
      determine the approximation of projected Hessian matrices as $D_k \approx Z_k^T \nabla_{xx}^2 h_{\varepsilon_k}(x_k) Z_k$ and $B_k \approx Z_k^T \nabla_{xx}^2 f(x_k) Z_k$;
      choose $\mu_k$ sufficiently small so that conditions (2.10a) and (2.10b) hold;
      **if** ($\mu_k < \mu_{\min}$) **then**
        set $\lambda_k = \mathbf{0}$ and *terminated* $\leftarrow$ **true**;
      **else**
        set $H_k(\mu_k) \leftarrow \mu_k B_k + D_k$ and $g_k^{\mathrm{p}}(\mu_k) = \mu_k g_k^{\mathrm{f}} + g_k^{\mathrm{h}}$;
        compute $w_k$ as an approximate solution of (pTR($\mu$)) for $\mu = \mu_k$, satisfying (2.13), set $v_k \leftarrow \mathbf{0}$, $\beta_k = 1$ and $d_k \leftarrow Z_k w_k + \beta_k v_k$;
      **end if**
    **end if**
  **else if** ($\|g_k^{\mathrm{p}}(\mu_k)\|_2$, tested against $\tau$, is not small) **then**
    ▷ Almost feasible iteration
    compute the approximate projected Hessian matrix $B_k \approx Z_k^T \nabla_{xx}^2 f(x_k) Z_k$, and set $D_k = \mathbf{0}$ and $H_k(\mu_k) \leftarrow \mu_k B_k$;
    compute $w_k$ as an approximate solution of (pTR($\mu$)) for $\mu = \mu_k$ satisfying (2.13);
    calculate the vertical step direction $v_k = d_k^{\mathrm{v}}$ using (2.15), determine $\beta_k$ as given in (2.16), and set $d_k \leftarrow Z_k w_k + \beta_k v_k$;

**Algorithm 1** A new trust region-line search projected exact penalty algorithm (continued).

**else** ▷ Local iteration
  compute $\lambda_k$ as (2.19);
  **if** (there exists an index $r$ so that $[\lambda_k]_r \notin [-\theta, 1+\theta]$) **then**
    choose an index $r$ so that $[\lambda_k]_r \notin [-\theta, 1+\theta]$, and compute $e_{\lambda_k}$ as in (2.22);
    calculate the dropping step direction $d_k^{\mathrm{d}}$ as in (2.20) and set $d_k \leftarrow d_k^{\mathrm{d}}$;
    using a backtracking line search strategy, find $\alpha_k \in (0,1]$ that meets the Armijo-like condition (2.21) for $\alpha = \alpha_k$;
    set $x_{k+1} \leftarrow x_k + \alpha_k d_k$ and *updated* $\leftarrow$ **true**;
    drop index $r$ from the active set;
  **else**
    test optimality and feasibility;
    **if** ( *optimal* **and** *feasible* ) **then**
      set *terminated* $\leftarrow$ **true**; ▷ A first order optimal solution is found
    **else**
      determine $H_k(\mu_k)$ as a symmetric and positive definite approximation of (2.24);
      compute $w_k$ by solving (2.23);
      calculate the vertical step direction $v_k = d_k^{\mathrm{v}}$ using (2.15);
      set $d_k = Z_k w_k + v_k$;
      **if** (2.25) hold **then**
        set $x_{k+1} \leftarrow x_k + d_k$ and *updated* $\leftarrow$ **true**;
      **else**
        reduce $\tau$ so that $\|g_k^{\mathrm{p}}(\mu_k)\|_2$, tested against $\tau$, is not small;
        set $x_{k+1} \leftarrow x_k$ and *updated* $\leftarrow$ **true**;
      **end if**
    **end if**
  **end if**
**end if**
**if** (**not** *updated*) **then**
  compute the predicted reductoin as (2.17) and $\rho_k$ as (2.18);
  **if** $\rho_k \geq \rho^{\mathrm{vs}}$ **then**
    set $x_{k+1} \leftarrow x_k + d_k$ and $\delta_{k+1} \leftarrow \gamma_e \delta_k$; ▷ Very successful iteration
  **else if** $\rho_k \geq \rho^{\mathrm{s}}$ **then**
    set $x_{k+1} \leftarrow x_k + d_k$ and $\delta_{k+1} \leftarrow \delta_k$; ▷ Successful iteration
  **else**
    set $x_{k+1} \leftarrow x_k$ and $\delta_{k+1} \leftarrow \gamma_c \delta_k$; ▷ Failed iteration
  **end if**
**end if**
set $\mu_{k+1} \leftarrow \mu_k$, $\varepsilon_{k+1} \leftarrow \varepsilon_k$ and $k \leftarrow k+1$;
**end while**
set $y_k \leftarrow \lambda_k / \mu_k$;
**return** $(x_k, y_k)$;

$\mu_k \nabla_x f(\cdot) - \sum_{i \in \mathrm{AC}_k} [\lambda_k]_i \nabla_x c_i(\cdot)$, we have

$$\begin{aligned} Z_k^T \nabla_{xx}^2 f(x_k) Z_k s_{k-1} &\approx Z_k^T \left( \nabla_x f(x_k) - \nabla_x f(x_{k-1}) \right), \\ Z_k^T \nabla_{xx}^2 h_{\varepsilon_k}(x_k) Z_k s_{k-1} &\approx Z_k^T \left( \nabla_x h_{\varepsilon_k}(x_k) - \nabla_x h_{\varepsilon_{k-1}}(x_{k-1}) \right), \\ G_k s_{k-1} &\approx Z_k^T \left( \mu_k \left( \nabla_x f(x_k) - \nabla_x f(x_{k-1}) \right) + A_{k-1} \lambda_k \right), \end{aligned}$$

where $G_k$ is defined as (2.24). Therefore, matrices $B_k$, $D_k$ and $H_k(\mu_k)$ are chosen in such a way that the following secant equations are satisfied:

$$\begin{aligned} B_k s_{k-1} &= y_{k-1,f}; \qquad & y_{k-1,f} &:= Z_k^T \left( \nabla_x f(x_k) - \nabla_x f(x_{k-1}) \right), \\ D_k s_{k-1} &= y_{k-1,h}; \qquad & y_{k-1,h} &:= Z_k^T \left( \nabla_x h_{\varepsilon_k}(x_k) - \nabla_x h_{\varepsilon_{k-1}}(x_{k-1}) \right), \\ H_k(\mu_k) s_{k-1} &= y_{k-1,l}; \qquad & y_{k-1,l} &:= Z_k^T \left( \mu_k \left( \nabla_x f(x_k) - \nabla_x f(x_{k-1}) \right) + A_{k-1} \lambda_k \right). \end{aligned}$$

To update $B$ to $\bar{B}$ so that $\bar{B}s = y$, the Davidon-Fletcher-Powell (DFP) or the Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating formulas can be used as follows:

$$\bar{B} = B + \mathrm{DFP}(B, s, y),$$

$$\mathrm{DFP}(B, s, y) := \frac{(y - Bs)y^T + y(y - Bs)^T}{s^T y} - \frac{s^T (y - Bs)}{(s^T y)^2} yy^T, \tag{2.30}$$

$$\bar{B} = B + \mathrm{BFGS}(B, s, y), \quad \mathrm{BFGS}(B, s, y) := \frac{yy^T}{s^T y} - \frac{(Bs)(Bs)^T}{s^T Bs}. \tag{2.31}$$

The BFGS formula is one of the most popular methods for updating the approximation of the Hessian matrix. It is known that if $B$ is positive definite and $s^T y > 0$, then the updated matrix $\bar{B}$ by (2.31) is well-defined and also positive definite [26, Theorem 7.8]. In a local iteration, we update the approximate projected Hessian matrix $H_k := H_k(\mu_k)$ using the BFGS formula, whenever $s_{k-1}^T y_{k-1,l} > 0$. Moreover, similar to Nocedal and Overton's method [50], the approximate projected Hessian is updated as $H_k = H_{k-1} + \mathrm{BFGS}(H_{k-1}, s_{k-1}, y_{k-1,l})$, only if the condition,

$$\|q_{k-1}\|_2 < \frac{\varphi}{k^{\upsilon+1}} \|s_{k-1}\|_2, \tag{2.32}$$

holds, for some prescribed positive constants $\upsilon$ and $\varphi$. If (2.32) does not hold or $s_{k-1}^T y_{k-1,l} \leq 0$, then we set $H_k \leftarrow H_{k-1}$.

## 3. Global Convergence

Here, we investigate the global convergence of Algorithm 1. The results are established under the following assumptions.

**Assumption 3.1.** The objective function $f : \mathbb{R}^n \to \mathbb{R}$ and the constraint functions $c_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, are twice continuously differentiable over an open, bounded, convex set $\mathscr{X} \subseteq \mathbb{R}^n$ that contains the closure of the sequence of iterates $\{x_k\}$ generated by Algorithm 1.

**Assumption 3.2.** There exists $\kappa_1 \geq 0$ such that $\|H_k(\mu_k)\|_2 \leq \kappa_1$, for all $k$.

**Assumption 3.3.** The second-order derivatives of the objective function and the constraint functions are uniformly bounded on $\mathscr{X}$. Consequently, there exists $\kappa_2 \geq 1$ such that $\|\nabla_{xx}^2 p_\varepsilon(x; \mu)\|_2 \leq \kappa_2$ for all $x \in \mathscr{X}$, $\varepsilon \geq 0$ and $\mu \in [0, 1]$.

**Assumption 3.4.** At any iteration $k$, the normal matrix $A_k$ has full column rank. Additionally, for any point $x \in \mathscr{X}$, LICQ holds.

**Assumption 3.5.** For any $\mu \in [0,1]$, the set of stationary but not first-order points of $p(x;\mu)$ over $\mathscr{X}$ is finite.

It is important to note that our assumptions for proving the global convergence are less stringent than those of Coleman and Conn [18]. Specifically, we do not need the sufficient reduction in penalty function required by the line search assumption of Coleman and Conn [18]. The following lemma demonstrates the conditions for detecting an infeasible stationary point in our algorithm.

**Lemma 3.1.** *Let Assumptions 3.1-3.4 hold. Given that* $\mathrm{VC}(x_k,\varepsilon_k) = \mathrm{VC}(x_k,0) \neq \emptyset$*, the following statements are valid:*

(a) *If* $\nabla_x h_{\varepsilon_k}(x_k) = \mathbf{0}$*, then* $x_k$ *is an infeasible stationary point.*
(b) *If* $x_k$ *is an infeasible stationary point, then* $Z_k^T \nabla_x h_{\varepsilon_k}(x_k) = \mathbf{0}$.

*Proof.* The condition $\mathrm{VC}(x_k,0) \neq \emptyset$ indicates that $h(x_k) > 0$. Additionally, for a given unit vector $d \in \mathbb{R}^n$ and a sufficiently small $t > 0$, from $\mathrm{VC}(x_k,\varepsilon_k) = \mathrm{VC}(x_k,0)$ and the continuous differentiability of $c_i$, $i = 1,\ldots,m$, we can infer that

$$\begin{aligned} h(x_k+td) &= -\sum_{i\in \mathrm{VC}(x_k,0)} c_i(x_k+td) - \sum_{i\in \mathrm{AVC}(x_k,d)} c_i(x_k+td) + o(t) \\ &= -\sum_{i\in \mathrm{VC}(x_k,\varepsilon_k)} c_i(x_k+td) - \sum_{i\in \mathrm{AVC}(x_k,d)} c_i(x_k+td) + o(t), \end{aligned}$$

where

$$\mathrm{AVC}(x,d) := \left\{ i = 1,\ldots,m : c_i(x) = 0,\ d^T \nabla_x c_i(x) < 0 \right\}. \tag{3.1}$$

Therefore, for any unit vector $d \in \mathbb{R}^n$, we obtain

$$\begin{aligned} D_d h(x_k) &= \lim_{t\downarrow 0^+} \frac{h(x_k+td) - h(x_k)}{t} \\ &= \lim_{t\downarrow 0^+} \left( -\frac{\sum_{i\in \mathrm{VC}(x_k,\varepsilon_k)} \left(c_i(x_k+td) - c_i(x_k)\right)}{t} \right. \\ &\qquad \left. -\frac{\sum_{i\in \mathrm{AVC}(x_k,d)} \left(c_i(x_k+td) - c_i(x_k)\right) + o(t)}{t} \right) \\ &= -d^T \nabla_x h_{\varepsilon_k}(x_k) - \sum_{i\in \mathrm{AVC}(x_k,d)} d^T \nabla_x c_i(x_k). \end{aligned} \tag{3.2}$$

Utilizing this fact, we can substantiate both assertions (a) and (b) as outlined below.

(a) Referring to the definition of $\mathrm{AVC}(x_k,d)$ as provided in (3.1), and taking into account (3.2), we can establish that the inequality $D_d h(x_k) \geq -d^T \nabla_x h_{\varepsilon_k}(x_k)$ holds for all unit vectors $d \in \mathbb{R}^n$. Hence, $\nabla_x h_{\varepsilon_k}(x_k) = \mathbf{0}$ implies that $D_d h(x_k) \geq 0$, for all $d \in \mathbb{R}^n$ with $\|d\|_2 = 1$. Moreover, we see that $h(x_k) > 0$. As a result, according to Definition 2.2, $x_k$ is recognized as an infeasible stationary point.

(b) Firstly, it is worth noting that the inclusion $\mathrm{AC}(x_k) \subseteq \mathrm{AC}(x_k,\varepsilon_k)$ entails $Z_k^T \nabla_x c_i(x_k) = 0$ for all $i \in \mathrm{AC}(x_k)$, where the matrix $Z_k \in \mathbb{R}^{n\times(n-t_k)}$ is defined as (2.5) and (2.6). Therefore, the definition of the index set $\mathrm{AVC}(x_k,\cdot)$ as provided by (3.1) implies that

$\mathrm{AVC}(x_k, Z_k w) = \emptyset$ for all vectors $w \in \mathbb{R}^{n-t_k}$. Also, considering (3.2) for $d = Z_k w$ with $\|w\|_2 = 1$, we have

$$D_{Z_k w} h(x_k) = -w^T Z_k^T \nabla_x h_{\varepsilon_k}(x_k).$$

Now, we suppose that $x_k$ is an infeasible stationary point. As a consequence of Definition 2.2, we observe that $D_{Z_k w} h(x_k) \geq 0$, for all unit vectors $w$. If $Z_k^T \nabla_x h_{\varepsilon_k}(x_k) \neq \mathbf{0}$, then, for $\bar{w} = Z_k^T \nabla_x h_{\varepsilon_k}(x_k) / \|Z_k^T \nabla_x h_{\varepsilon_k}(x_k)\|_2$, we have $\|\bar{w}\|_2 = 1$ and

$$0 \leq D_{Z_k \bar{w}} h(x_k) = -\bar{w}^T Z_k^T \nabla_x h_{\varepsilon_k}(x_k) = -\|Z_k^T \nabla_x h_{\varepsilon_k}(x_k)\|_2 < 0,$$

which is a contradiction. Consequently, if $x_k$ is an infeasible stationary point, then we have $Z_k^T \nabla_x h_{\varepsilon_k}(x_k) = \mathbf{0}$.

□

In an infeasible iteration, defining $g_k^{\mathrm{f}} := Z_k^T \nabla_x f(x_k)$, we have

$$g_k^{\mathrm{p}}(\mu) = \mu g_k^{\mathrm{f}} + g_k^{\mathrm{h}},$$
$$\check{p}_{\varepsilon_k}(w; H_k(\mu), x_k, \mu) = \mu \check{f}(w; B_k, x_k) + \check{h}_{\varepsilon_k}(w; D_k, x_k),$$

where

$$\check{f}(w; B_k, x_k) := \frac{1}{2} w^T B_k w + w^T g_k^{\mathrm{f}} + f(x_k).$$

Moreover, we have $\Delta \check{p}_{\varepsilon_k}(w; H_k(\mu), x_k, \mu) = \mu \Delta \check{f}(w; B_k, x_k) + \Delta \check{h}_{\varepsilon_k}(w; D_k, x_k)$, where

$$\Delta \check{f}(w; B_k, x_k) := \check{f}(\mathbf{0}; B_k, x_k) - \check{f}(w; B_k, x_k),$$

and $\Delta \check{h}_{\varepsilon_k}(w; D_k, x_k)$ is defined as (2.11).

In the following lemma, we show that conditions (2.10) hold for a sufficiently small penalty parameter $\mu_k > 0$.

**Lemma 3.2.** *If Assumptions 3.1-3.4 hold, the kth iteration is an infeasible iteration, $g_k^{\mathrm{h}} \neq \mathbf{0}$, and $\eta_1, \eta_2 \in (0,1)$ are predesignated constants, then there exists $\bar{\mu} > 0$ such that*

$$\Delta \check{h}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu); D_k, x_k\right) \geq \eta_1 \Delta \check{h}_{\varepsilon_k}\left(w_k^{\mathrm{FC}}; D_k, x_k\right),$$
$$\Delta \check{p}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu); H_k(\mu), x_k, \mu\right) \geq \eta_2 \Delta \check{h}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu); D_k, x_k\right),$$

*for all $\mu \in [0, \bar{\mu}]$.*

*Proof.* Provided that $g_k^{\mathrm{h}} \neq \mathbf{0}$, the Cauchy point of (hTR) meets the following inequality (see [24, Theorem 6.3.1] for the details.)

$$\Delta \check{h}_{\varepsilon_k}(w_k^{\mathrm{FC}}; D_k, x_k) \geq \frac{1}{2} \left\|g_k^{\mathrm{h}}\right\|_2 \min\left\{ \frac{\left\|g_k^{\mathrm{h}}\right\|_2}{1 + \|D_k\|_2}, \delta_k \right\} > 0; \tag{3.3}$$

Moreover, we have

$$\lim_{\mu \downarrow 0^+} \Delta \check{h}_{\varepsilon_k}(w_k^{\mathrm{PC}}(\mu); D_k, x_k) = \Delta \check{h}_{\varepsilon_k}(w_k^{\mathrm{FC}}; D_k, x_k). \tag{3.4}$$

Due to $\eta_1 \in (0,1)$, (3.3), and (3.4), one sees that there exists a positive constant $\bar{\mu}_1$ such that, for any $\mu \in [0, \bar{\mu}_1]$,

$$\left|\Delta \check{h}_{\varepsilon_k}(w_k^{\mathrm{PC}}(\mu); D_k, x_k) - \Delta \check{h}_{\varepsilon_k}(w_k^{\mathrm{FC}}; D_k, x_k)\right| \leq (1 - \eta_1) \Delta \check{h}_{\varepsilon_k}(w_k^{\mathrm{FC}}; D_k, x_k),$$

and hence

$$\Delta\check{h}_{\varepsilon_k}(w_k^{\mathrm{PC}}(\mu);D_k,x_k) \geq \eta_1\Delta\check{h}_{\varepsilon_k}(w_k^{\mathrm{FC}};D_k,x_k), \qquad \forall\mu\in[0,\bar{\mu}_1]. \tag{3.5}$$

Now, we consider

$$\varphi(\mu) := (1-\eta_2)\Delta\check{h}_{\varepsilon_k}(w_k^{\mathrm{PC}}(\mu);D_k,x_k) - \mu\left(\frac{\delta_k^2\|B_k\|_2}{2}+\delta_k\|g_k^{\mathrm{f}}\|_2\right).$$

From $\eta_2\in(0,1)$, (3.3), and (3.4), we observe that

$$\lim_{\mu\downarrow 0^+}\varphi(\mu) = (1-\eta_2)\Delta\check{h}_{\varepsilon_k}(w_k^{\mathrm{FC}};D_k,x_k) > 0.$$

Accordingly, there exists a positive constant $\bar{\mu}_2$ such that $\varphi(\mu)>0$ for all $\mu\in[0,\bar{\mu}_2]$. Also, we have

$$\begin{aligned}
&\Delta\check{p}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu);H_k(\mu),x_k,\mu\right)\\
&=\mu\Delta\check{f}(w_k^{\mathrm{PC}}(\mu);B_k,x_k)+\Delta\check{h}(w_k^{\mathrm{PC}}(\mu);D_k,x_k)\\
&=\Delta\check{h}(w_k^{\mathrm{PC}}(\mu);D_k,x_k)-\mu\left(\frac{1}{2}\left(w_k^{\mathrm{PC}}(\mu)\right)^T B_k w_k^{\mathrm{PC}}(\mu)+\left(w_k^{\mathrm{PC}}(\mu)\right)^T g_k^{\mathrm{f}}\right)\\
&\geq\Delta\check{h}(w_k^{\mathrm{PC}}(\mu);D_k,x_k)-\mu\left(\frac{1}{2}\|B_k\|_2\|w_k^{\mathrm{PC}}(\mu)\|_2^2+\|w_k^{\mathrm{PC}}(\mu)\|_2\|g_k^{\mathrm{f}}\|_2\right)\\
&\geq\Delta\check{h}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu);D_k,x_k\right)-\mu\left(\frac{\delta_k^2\|B_k\|_2}{2}+\delta_k\|g_k^{\mathrm{f}}\|_2\right)\\
&=\varphi(\mu)+\eta_2\Delta\check{h}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu);D_k,x_k\right).
\end{aligned}$$

As a result,

$$\Delta\check{p}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu);H_k(\mu),x_k,\mu\right)\geq\eta_2\Delta\check{h}_{\varepsilon_k}\left(w_k^{\mathrm{PC}}(\mu);D_k,x_k\right), \qquad \forall\mu\in[0,\bar{\mu}_2]. \tag{3.6}$$

Letting $\bar{\mu}:=\min\{\bar{\mu}_1,\bar{\mu}_2\}$ and utilizing (3.5) and (3.6), the desired results follows immediately. □

In the subsequent lemma, we express the rationale behind defining the predicted reduction illustrated by (2.17).

**Lemma 3.3.** *Let Assumptions 3.1-3.4 hold. At the kth iteration of the algorithm, if the step direction is computed as $d_k = Z_k w_k + \beta_k v_k$, with $\beta_k\geq 0$, and $v_k\in\{\boldsymbol{0},d_k^{\mathrm{v}}\}$, then*

$$\begin{aligned}
p(x_k;\mu_k)-p(x_k+d_k;\mu_k) =&\Delta\check{p}_{\varepsilon_k}\left(w_k;H_k(\mu_k),x_k,\mu_k\right)-\beta_k v_k^T\nabla_x p_{\varepsilon_k}(x_k;\mu_k)\\
&+\mathscr{O}(\|d_k\|_2^2)+\mathscr{O}(\varepsilon_k).
\end{aligned}\tag{3.7}$$

*Proof.* Defining the index sets,

$$\begin{aligned}
M_1^k &:= M_1(x_k,d_k,\varepsilon_k) := \mathrm{AC}(x_k,\varepsilon_k)\cap\mathrm{AC}_-(x_k+d_k,\varepsilon_k),\\
M_2^k &:= M_2(x_k,d_k,\varepsilon_k) := \mathrm{AC}_-(x_k,\varepsilon_k)\cap\mathrm{AC}_+(x_k+d_k,\varepsilon_k),\\
M_3^k &:= M_3(x_k,d_k,\varepsilon_k) := \mathrm{AC}_+(x_k,\varepsilon_k)\cap\mathrm{AC}_-(x_k+d_k,\varepsilon_k),
\end{aligned}$$

where

$$\begin{aligned}
\mathrm{AC}_+(x,\varepsilon) &:= \{i=1,\ldots,m\ :\ 0\leq c_i(x)\leq\varepsilon\},\\
\mathrm{AC}_-(x,\varepsilon) &:= \{i=1,\ldots,m\ :\ -\varepsilon\leq c_i(x)<0\},
\end{aligned}$$

we have

$$\begin{aligned}
&p(x_k+d_k;\mu_k)-p(x_k;\mu_k)\\
&=p_{\varepsilon_k}(x_k+d_k;\mu_k)-p_{\varepsilon_k}(x_k;\mu_k)+\sum_{i\in \mathrm{AC}(x_k+d_k,\varepsilon_k)}\max\{-c_i(x_k+d_k),0\}\\
&\quad-\sum_{i\in \mathrm{AC}(x_k,\varepsilon_k)}\max\{-c_i(x_k),0\}\\
&=p_{\varepsilon_k}(x_k+d_k;\mu_k)-p_{\varepsilon_k}(x_k;\mu_k)-\sum_{i\in M_1^k}(c_i(x_k+d_k)-c_i(x_k))-\sum_{i\in M_2^k\cup M_3^k}c_i(x_k).
\end{aligned}\tag{3.8}$$

Given that $M_2^k\subseteq \mathrm{AC}_-(x_k,\varepsilon_k)$ and $M_3^k\subseteq \mathrm{AC}_+(x_k,\varepsilon_k)$, it follows that $-\varepsilon_k\le c_i(x_k)\le 0$, $i\in M_2^k$, and $0\le c_i(x_k)\le\varepsilon_k$, $i\in M_3^k$. These observations imply

$$-|M_k^2|\varepsilon_k\le\sum_{i\in M_2^k\cup M_3^k}c_i(x_k)\le|M_k^3|\varepsilon_k\Rightarrow\sum_{i\in M_2^k\cup M_3^k}c_i(x_k)=\mathscr{O}(\varepsilon_k).\tag{3.9}$$

Consequently, from (3.8) and (3.9), we arrive at

$$\begin{aligned}
p(x_k+d_k;\mu_k)-p(x_k;\mu_k)&=p_{\varepsilon_k}(x_k+d_k;\mu_k)-p_{\varepsilon_k}(x_k;\mu_k)\\
&\quad-\sum_{i\in M_1^k}(c_i(x_k+d_k)-c_i(x_k))+\mathscr{O}(\varepsilon_k).
\end{aligned}\tag{3.10}$$

Furthermore, by applying Taylor's theorem and relying on the Lipschitz continuity assumption for $\nabla_x f(\cdot)$ and $\nabla_x c_i(\cdot)$, $i=1,\dots,m$, along with the boundedness assumption on the Hessian of the penalty function, and its approximations $\{H_k(\mu_k)\}_{k=1}^{\infty}$, in (3.10), we obtain

$$\begin{aligned}
&p(x_k+d_k;\mu_k)-p(x_k;\mu_k)\\
&=d_k^T\nabla_x p_{\varepsilon_k}(x_k;\mu_k)+\frac{1}{2}w_k^TH_k(\mu_k)w_k-\sum_{i\in M_1^k}d_k^T\nabla_x c_i(x_k)+\mathscr{O}(\|w_k\|_2^2)+\mathscr{O}(\|d_k\|_2^2)+\mathscr{O}(\varepsilon_k)\\
&=-\Delta\check{p}_{\varepsilon_k}(w_k;H_k(\mu_k),x_k,\mu_k)+\beta_k v_k^T\nabla_x p_{\varepsilon_k}(x_k;\mu_k)-\beta_k\sum_{i\in M_1^k}v_k^T\nabla_x c_i(x_k)+\mathscr{O}(\|d_k\|_2^2)+\mathscr{O}(\varepsilon_k),
\end{aligned}\tag{3.11}$$

where the second equality is obtained from the definition of the step direction being $d_k=Z_kw_k+\beta_kv_k$, the definition of $\Delta\check{p}_{\varepsilon_k}(w_k;H_k(\mu_k),x_k,\mu_k)$, and $Z_k^T\nabla_x c_i(x_k)=0$, for all $i\in M_1^k\subseteq \mathrm{AC}(x_k,\varepsilon_k)$. Hence, (3.11) implies that (3.7) is valid when $v_k=\mathbf{0}$. Furthermore, based on (2.14) and since $M_1^k\subseteq \mathrm{AC}(x_k,\varepsilon_k)$, for $v_k=d_k^{\mathrm{v}}$, we have

$$\begin{aligned}
-\beta_k\sum_{i\in M_1^k}v_k^T\nabla_x c_i(x_k)&=\beta_k\sum_{i\in M_1^k}c_i(x_k+Z_kw_k)\\
&=\beta_k\sum_{i\in M_1^k}\left(c_i(x_k)+w_k^TZ_k^T\nabla_x c_i(x_k)+\mathscr{O}(\|w_k\|_2^2)\right)\\
&=\mathscr{O}(\varepsilon_k)+\mathscr{O}(\|w_k\|_2^2).
\end{aligned}\tag{3.12}$$

Therefore, employing (3.12) within (3.11), we obtain (3.7) for $v_k=d_k^{\mathrm{v}}$. □

The following lemma ensures that the predicted reduction is positive. As a result of this lemma, we can deduce that defining the ratio of the actual reduction to the predicted reduction as (2.18) is well-posed.

**Lemma 3.4.** *Let Assumptions 3.1-3.4 hold, $g_k^{\mathrm{p}}(\mu_k) := Z_k^T \nabla_x p_{\varepsilon_k}(x_k;\mu_k) \neq \mathbf{0}$ and the step direction be computed as $d_k = Z_k w_k + \beta_k v_k$ with $v_k \in \{d_k^{\mathrm{v}}, \mathbf{0}\}$, and $w_k \neq \mathbf{0}$ as an approximate solution of (pTR($\mu$)) for $\mu = \mu_k$, satisfying (2.13). Assume that $\beta_k$ is determined as (2.16) if $v_k = d_k^{\mathrm{v}}$, and $\beta_k = 1$, if $v_k = \mathbf{0}$. Then, for a predetermined constant $\eta \in (0,1)$,*

$$\begin{aligned}\mathrm{Pred}_k \geq & (1-\eta)\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) \\ \geq & \frac{1-\eta}{2}\left\|g_k^{\mathrm{p}}(\mu_k)\right\|_2 \min\left\{\frac{\left\|g_k^{\mathrm{p}}(\mu_k)\right\|_2}{1+\|H_k(\mu_k)\|_2}, \delta_k\right\} > 0, \end{aligned} \tag{3.13}$$

*and $\|d_k\|_2 \leq 2\delta_k$.*

*Proof.* In the case $v_k = \mathbf{0}$, since $\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) \geq 0$ and $\eta \in (0,1)$, we can conclude

$$\mathrm{Pred}_k = \Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) \geq (1-\eta)\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k).$$

If $v_k = d_k^{\mathrm{v}}$ and $\nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}} \leq 0$, then

$$\begin{aligned}\mathrm{Pred}_k &= \Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) - \beta_k \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}} \\ &\geq \Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) \\ &\geq (1-\eta)\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k),\end{aligned}$$

where the first inequality follows from $\beta_k > 0$ and the last inequality stems from the fact that $\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) \geq 0$ and $\eta \in (0,1)$. Otherwise, in the case $\nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}} > 0$, the choice of $\beta_k$ as (2.16) indicates that

$$\begin{aligned}\mathrm{Pred}_k &= \Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) - \beta_k \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}} \\ &\geq \Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) - \eta \frac{\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k)}{\nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}}} \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k) d_k^{\mathrm{v}} \\ &\geq (1-\eta)\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k).\end{aligned}$$

Therefore, in all the above cases, we have

$$\mathrm{Pred}_k \geq (1-\eta)\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k). \tag{3.14}$$

Moreover, from Theorem 6.3.1 of [24], for $w_k \neq \mathbf{0}$ satisfying (2.13), we have

$$\begin{aligned}\Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) &\geq \Delta\check{p}_{\varepsilon_k}(w_k^{\mathrm{PC}}(\mu_k); H_k(\mu_k), x_k, \mu_k) \\ &\geq \frac{1}{2}\left\|g_k^{\mathrm{p}}(\mu_k)\right\|_2 \min\left\{\frac{\|g_k^{\mathrm{p}}(\mu_k)\|_2}{1+\|H_k(\mu_k)\|_2}, \delta_k\right\} > 0,\end{aligned}$$

where the final inequality arises from the assumption that $g_k^{\mathrm{p}}(\mu_k) \neq \mathbf{0}$. This along with the inequality (3.14) leads to the conclusion as stated in (3.13). If $v_k = \mathbf{0}$, then

$$\|d_k\|_2 = \|Z_k w_k\|_2 = \|w_k\|_2 \leq \delta_k \leq 2\delta_k.$$

If $v_k = d_k^{\mathrm{v}}$, we find by considering the definition of $d_k$, the triangle inequality, the orthonormality of the columns of $Z_k$, and the definition of $\beta_k$ as (2.16) that

$$\|d_k\|_2 = \|Z_k w_k + \beta_k d_k^{\mathrm{v}}\|_2 \leq \|Z_k w_k\|_2 + \beta_k \|d_k^{\mathrm{v}}\|_2 \leq \|w_k\|_2 + \frac{\delta_k}{\|d_k^{\mathrm{v}}\|_2}\|d_k^{\mathrm{v}}\|_2 \leq 2\delta_k.$$

□

In the next lemma, we derive an upper bound for the difference between the actual reduction and the predicted reduction in terms of the square of the trust region radius.

**Lemma 3.5.** *Let Assumptions 3.1-3.4 hold. Consider the step direction to be computed as* $d_k = Z_k w_k + \beta_k v_k$, *where* $v_k \in \{d_k^{\mathrm{v}}, \mathbf{0}\}$, *and* $w_k \neq \mathbf{0}$ *serves as an approximate solution of* (pTR($\mu$)) *satisfying* (2.13). *Moreover, assume that* $\varepsilon_k \leq \delta_k^2$. *The value of* $\beta_k$ *is calculated by* (2.16) *when* $v_k = d_k^{\mathrm{v}}$, *and* $\beta_k = 1$ *when* $v_k = \mathbf{0}$. *Then,*

$$|p(x_k;\mu_k) - p(x_k+d_k;\mu_k) - \mathrm{Pred}_k| \leq \frac{\kappa}{2}\delta_k^2, \tag{3.15}$$

*where* $\kappa := (\kappa_1 + 4\kappa_2 + 4m)$.

*Proof.* By utilizing the Taylor's theorem, one sees that there exists $\xi_k \in (0,1)$ such that

$$p_{\varepsilon_k}(x_k+d_k;\mu_k) = p_{\varepsilon_k}(x_k;\mu_k) + \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k)d_k + \frac{1}{2}d_k^T \nabla_{xx}^2 p_{\varepsilon_k}(x_k+\xi_k d_k;\mu_k)d_k. \tag{3.16}$$

iven the step direction $d_k = Z_k w_k + \beta_k v_k$, employing the definition of the predicted reduction as (2.17), along with utilization of (3.16), we have

$$\begin{aligned}
&|p(x_k;\mu_k) - p(x_k+d_k;\mu_k) - \mathrm{Pred}_k| \\
&= |p(x_k;\mu_k) - p(x_k+d_k;\mu_k) - \Delta\check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) + \beta_k \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k)v_k| \\
&= \Big| - p_{\varepsilon_k}(x_k+d_k;\mu_k) + \check{p}_{\varepsilon_k}(w_k; H_k(\mu_k), x_k, \mu_k) \\
&\quad + \beta_k \nabla_x^T p_{\varepsilon_k}(x_k;\mu_k)v_k + \sum_{i\in \mathrm{AC}(x_k,\varepsilon_k)} \max\{-c_i(x_k), 0\} - \sum_{i\in \mathrm{AC}(x_k+d_k,\varepsilon_k)} \max\{-c_i(x_k+d_k), 0\} \Big| \\
&= \Big| - \frac{1}{2}d_k^T \nabla_{xx}^2 p_{\varepsilon_k}(x_k+\xi_k d_k;\mu_k)d_k + \frac{1}{2}w_k^T H_k(\mu_k) w_k \\
&\quad - \sum_{i\in \mathrm{AC}_-(x_k,\varepsilon_k)} c_i(x_k) + \sum_{i\in \mathrm{AC}_-(x_k+d_k,\varepsilon_k)} c_i(x_k+d_k) \Big| \\
&\leq \frac{1}{2}\left|d_k^T \nabla_{xx}^2 p_{\varepsilon_k}(x_k+\xi_k d_k;\mu_k)d_k\right| + \frac{1}{2}\left|w_k^T H_k(\mu_k) w_k\right| \\
&\quad + \sum_{i\in \mathrm{AC}_-(x_k,\varepsilon_k)} |c_i(x_k)| + \sum_{i\in \mathrm{AC}_-(x_k+d_k,\varepsilon_k)} |c_i(x_k+d_k)| \\
&\leq \frac{1}{2}\|\nabla_{xx}^2 p_{\varepsilon_k}(x_k+\xi_k d_k;\mu_k)\|_2 \|d_k\|_2^2 + \frac{1}{2}\|H_k(\mu_k)\|_2 \|w_k\|_2^2 + 2m\varepsilon_k \\
&\leq \frac{\kappa_1 + 4\kappa_2 + 4m}{2}\delta_k^2 = \frac{\kappa}{2}\delta_k^2,
\end{aligned}$$

where to deduce the last inequality we used the assumption $\varepsilon_k \leq \delta_k^2$. □

Next, to establish the global convergence of a trust region algorithm, it is crucial to show that when the trust region radius is sufficiently small, the iterate is very successful.

**Lemma 3.6.** *Let* $\kappa = \kappa_1 + 4\kappa_2 + 4m$, *and assumptions of Lemma 3.5 hold. If*

$$\delta_k \leq \|g_k^{\mathrm{p}}(\mu_k)\|_2 \min\left\{\frac{1}{1+\kappa_1}, \frac{(1-\rho^{\mathrm{s}})(1-\eta)}{\kappa}\right\}, \tag{3.17}$$

*then the iteration k is successful, and hence* $\delta_{k+1} \geq \delta_k$.

*Proof.* From (3.17) and Assumption 3.2, we have

$$\delta_k \leq \frac{\|g_k^{\mathrm{p}}(\mu_k)\|_2}{1+\kappa_1} \leq \frac{\|g_k^{\mathrm{p}}(\mu_k)\|_2}{1+H_k(\mu_k)}.$$

By applying the above inequality in (3.13), we observe that

$$\mathrm{Pred}_k \geq \frac{1-\eta}{2} \left\|g_k^{\mathrm{p}}(\mu_k)\right\|_2 \delta_k. \tag{3.18}$$

Considering the definition of $\rho_k$ in (2.18), and utilizing (3.15), (3.17), and (3.18), we have

$$\begin{aligned} |\rho_k - 1| &= \left| \frac{p(x_k;\mu_k) - p(x_k+d_k;\mu_k) - \mathrm{Pred_k}}{\mathrm{Pred_k}} \right| \\ &\leq \left| \frac{\frac{\kappa}{2}\delta_k^2}{\frac{1-\eta}{2}\|g_k^{\mathrm{p}}(\mu_k)\|_2 \delta_k} \right| \\ &\leq \left| \frac{\kappa \delta_k}{(1-\eta)\|g_k^{\mathrm{p}}(\mu_k)\|_2} \right| \leq 1-\rho^{\mathrm{s}}. \end{aligned} \tag{3.19}$$

Inequality (3.19) indicates that $\rho_k \geq \rho^{\mathrm{s}}$, and that the $k$th iteration is successful. Therefore, according to lines 59-64 of Algorithm 1, the trust region radius is updated as $\delta_{k+1} \leftarrow \delta_k$ or $\delta_{k+1} \leftarrow \gamma_e \delta_k > \delta_k$. □

The lemma presented below states that the trust region radius will not shrink to zero at non-stationary points.

**Lemma 3.7.** *Under the assumptions made in Lemma 3.5, if there exists $\varepsilon > 0$ so that $\|g_k^{\mathrm{p}}(\mu_k)\|_2 \geq \varepsilon$, for all $k$, then*

$$\delta_k \geq \delta_{\min} := \varepsilon \gamma_c \min\left\{ \frac{1}{1+\kappa_1}, \frac{(1-\rho^{\mathrm{s}})(1-\eta)}{\kappa} \right\} > 0, \qquad \forall k. \tag{3.20}$$

*Proof.* First, note that according to the flow of Algorithm 1, the trust region remains unchanged when the dropping step direction is computed. To reach a contradiction, let us assume $k_0$ be the first iteration such that

$$\delta_{k_0} \geq \delta_{\min} > \delta_{k_0+1} = \gamma_c \delta_{k_0}. \tag{3.21}$$

Consequently, it follows from (3.20) and (3.21) that

$$\begin{aligned} \delta_{k_0} = \frac{\delta_{k_0+1}}{\gamma_c} < \frac{\delta_{\min}}{\gamma_c} &= \varepsilon \min\left\{ \frac{1}{1+\kappa_1}, \frac{(1-\rho^{\mathrm{s}})(1-\eta)}{\kappa} \right\} \\ &\leq \|g_{k_0}^{\mathrm{p}}(\mu_{k_0})\|_2 \min\left\{ \frac{1}{1+\kappa_1}, \frac{(1-\rho^{\mathrm{s}})(1-\eta)}{\kappa} \right\}. \end{aligned}$$

Using the aforementioned inequality and the result of Lemma 3.6, we conclude that the $k$th iteration is successful and $\delta_{k_0+1} \geq \delta_{k_0}$, which contradicts (3.21). □

Now, let $\mathscr{I}$, $\mathscr{A}$ and $\mathscr{L}$ denote the index sets of infeasible, almost feasible, and local iterations, respectively. The following lemma states that, for any given stationary tolerance $\tau > 0$, there exists a sufficiently large nonnegative integer $k$ so that $\|g_k^{\mathrm{p}}(\mu_k)\|_2 < \tau$.

**Lemma 3.8.** *Let Assumptions 3.1-3.4 hold. Assume that Algorithm 1 generates an infinite sequence of iterates $\{x_k\}$ starting from an arbitrary initial point $x_0$ to solve (NLP). Then, for any chosen stationary tolerance $\tau > 0$, $\|g_k^{\mathrm{p}}(\mu_k)\|_2 < \tau$, for all $k \in \mathscr{I}$, where $\mathscr{I}$ is an infinite subsequence of $\{0,1,2,\ldots\}$.*

*Proof.* To derive a contradiction, we assume that

$$\|g_k^{\mathrm{p}}(\mu_k)\|_2 \geq \tau, \tag{3.22}$$

for some $\tau > 0$ and all $k$. Consequently, $|\mathscr{L}| = 0$. Since Algorithm 1 produces an infinite sequence of iterates, we conclude that $\|g_k^{\mathrm{h}}\|_2 > 0$ for all $k$. Moreover, it follows that there exists a nonnegative integer $k_0$ such that $\mu_k = \bar{\mu} > 0$ for all $k \geq k_0$. Consider the index set of successful iterations as:

$$\mathscr{S} := \{k \in \mathscr{I} \cup \mathscr{A} \ : \ x_{k+1} \leftarrow x_k + Z_k w_k + \beta_k v_k \text{ with } w_k \neq \mathbf{0}\},$$

where, for all $k$, $v_k \in \{\mathbf{0}, d_k^{\mathrm{v}}\}$, $d_k^{\mathrm{v}}$ is computed by (2.15), and $\beta_k$ is computed by (2.16). Also, let the index set of unsuccessful iterations be $\mathscr{U} := \{k \in \mathscr{I} \cup \mathscr{A} \ : \ x_{k+1} \leftarrow x_k\}$. For all $k \in \mathscr{U}$, we have $\delta_{k+1} = \gamma_c \delta_k < \delta_k$. Therefore, if $|\mathscr{U}| = \infty$, then it follows that $\lim_{k\in\mathscr{U}} \delta_k = 0$, which contradicts Lemma 3.7. Consequently, we must have $|\mathscr{S}| = \infty$. For all $k \in \mathscr{S}$ such that $k \geq k_0$, we have $\rho_k \geq \rho^{\mathrm{s}}$. Thus

$$\begin{aligned} p(x_k;\bar{\mu}) - p(x_{k+1};\bar{\mu}) \geq \rho^{\mathrm{s}}\mathrm{Pred}_k &\geq \frac{1-\eta}{2}\rho^{\mathrm{s}} \left\|g_k^{\mathrm{p}}(\bar{\mu})\right\|_2 \min\left\{\frac{\left\|g_k^{\mathrm{p}}(\bar{\mu})\right\|_2}{1+\|H_k(\bar{\mu})\|_2}, \delta_k\right\} \\ &\geq \frac{1-\eta}{2}\rho^{\mathrm{s}}\tau \min\left\{\frac{\tau}{1+\kappa_1}, \delta_{\min}\right\} \stackrel{\mathrm{def}}{=} \delta_\tau > 0. \end{aligned}$$

Choosing $j \geq k_0$ and summing over all $k \in \mathscr{S}_j := \{k_0, \ldots, j \mid k \in \mathscr{S}\}$, we have

$$\begin{aligned} p(x_{k_0};\bar{\mu}) - p(x_{j+1};\bar{\mu}) &= \sum_{k=k_0}^{j} \left(p(x_k;\bar{\mu}) - p(x_{k+1};\bar{\mu})\right) \\ &= \sum_{k\in\mathscr{S}_j} \left(p(x_k;\bar{\mu}) - p(x_{k+1};\bar{\mu})\right) \geq \sum_{k\in\mathscr{S}_j} \delta_\tau. \end{aligned}$$

As $j$ approaches infinity, we obtain

$$\lim_{j\to\infty} \left(p(x_0;\bar{\mu}) - p(x_{j+1};\bar{\mu})\right) \geq \lim_{j\to\infty} \sum_{k\in\mathscr{S}_j} \delta_\tau = \sum_{k\in\mathscr{S}} \delta_\tau = \infty,$$

which implies that $f$ is unbounded below over $\mathscr{X}$. This contradicts Assumption 3.1. Therefore, we conclude that (3.22) is false. This means that $\|g_k^{\mathrm{p}}(\mu_k)\|_2 < \tau$, for some $k$. □

The following lemma demonstrates that the dropping step direction serves as a descent direction for the penalty function. This lemma has some similarity to part 2 of Theorem 1 of [18]. Here, we adapt it in accordance with our notations and terminologies.

**Lemma 3.9.** *Let Assumptions 3.1-3.4 hold. Assume that the kth iteration is local,* $\mathrm{VC}(x_k,\varepsilon_k) = \mathrm{VC}(x_k,0)$, *and* $\mathrm{AC}(x_k,\varepsilon_k) = \mathrm{AC}(x_k,0)$. *Suppose that there exists $r \in \mathrm{AC}(x_k,\varepsilon_k)$ such that $[\lambda_k]_r \notin [0,1]$, and the dropping step direction is computed as (2.20). Then,*

*(1) For small enough positive values of $\alpha$, we have*

$$p(x_k+\alpha d_k^{\mathrm{d}};\mu_k)=p(x_k;\mu_k)+e_{\lambda_k}\alpha+\mathscr{O}(\alpha^2), \tag{3.23}$$

*where $e_{\lambda_k}<0$ is defined as (2.22). Consequently, the dropping step direction $d_k^{\mathrm{d}}$ constitutes a descent direction for the penalty function at $x_k$ with the penalty parameter $\mu_k$.*

*(2) There exists $\bar{\alpha}>0$ such that the Armijo-like condition (2.21) holds for all $\alpha\in[0,\bar{\alpha}]$.*

*Proof.* The definition of the penalty function, Taylor's theorem, and the requirements for $f$, $c_i$, $i=1,\ldots,m$, being twice continuously differentiable, collectively imply that, for sufficiently small $\alpha>0$,

$$\begin{aligned}
p(x_k+\alpha d_k^{\mathrm{d}};\mu_k)&=\mu_k f(x_k+\alpha d_k^{\mathrm{d}})+\sum_{i=1}^{m}\max\{-c_i(x_k+\alpha d_k^{\mathrm{d}}),0\}\\
&=\mu_k\left(f(x_k)+\nabla_x^T f(x_k)d_k^{\mathrm{d}}\alpha+\mathscr{O}(\alpha^2)\right)\\
&\quad+\sum_{i=1}^{m}\max\{-c_i(x_k)-\nabla_x^T c_i(x_k)d_k^{\mathrm{d}}\alpha+\mathscr{O}(\alpha^2),0\}\\
&=\mu_k\left(f(x_k)+\nabla_x^T f(x_k)d_k^{\mathrm{d}}\alpha\right)-\sum_{i\in\mathrm{VC}(x_k,0)}\left(c_i(x_k)+\nabla_x^T c_i(x_k)d_k^{\mathrm{d}}\alpha\right)\\
&\quad+\sum_{i\in\mathrm{AC}(x_k,0)}\max\{-\nabla_x^T c_i(x_k)d_k^{\mathrm{d}}\alpha+\mathscr{O}(\alpha^2),0\}+\mathscr{O}(\alpha^2)\\
&=p(x_k;\mu_k)+\left(\nabla_x^T p_{\varepsilon_k}(x_k;\mu_k)d_k^{\mathrm{d}}+\max\{\mathrm{sgn}([\lambda_k]_r),0\}\right)\alpha+\mathscr{O}(\alpha^2),
\end{aligned} \tag{3.24}$$

where the last equality is obtained from $\mathrm{VC}(x_k,\varepsilon_k)=\mathrm{VC}(x_k,0)$, and the definition of the dropping step direction $d_k^{\mathrm{d}}$ in (2.20). The definition of the dropping step and (2.7) indicate that $d_k^{\mathrm{d}}\in R(A_k)$. Moreover, since $\lambda_k$ is a solution to linear least squares problem (2.8), we have $\nabla_x p_{\varepsilon_k}(x_k;\mu_k)=A_k\lambda_k+u$ for some $u\in N(A_k^T)$. From orthogonality of $N(A_k^T)$ and $R(A_k)$, we observe

$$\begin{aligned}
&\nabla_x^T p_{\varepsilon_k}(x_k;\mu_k)d_k^{\mathrm{d}}=(A_k\lambda_k+u)^T d_k^{\mathrm{d}}=\lambda_k^T A_k^T d_k^{\mathrm{d}}=-\mathrm{sgn}([\lambda_k]_r)\lambda_k^T e_r\\
\Rightarrow\ &\nabla_x^T p_{\varepsilon_k}(x_k;\mu_k)d_k^{\mathrm{d}}=-\mathrm{sgn}([\lambda_k]_r)[\lambda_k]_r=-\left|[\lambda_k]_r\right|.
\end{aligned} \tag{3.25}$$

By substituting (3.25) into (3.24), we obtain:

$$p(x_k+\alpha d_k^{\mathrm{d}};\mu_k)=p(x_k;\mu_k)+\left(-\left|[\lambda_k]_r\right|+\max\{\mathrm{sgn}([\lambda_k]_r),0\}\right)\alpha+\mathscr{O}(\alpha^2), \tag{3.26}$$

for sufficiently small $\alpha>0$. Moreover, for $[\lambda_k]_r>1$, we have

$$-\left|[\lambda_k]_r\right|+\max\{\mathrm{sgn}([\lambda_k]_r),0\}=1-[\lambda_k]_r<0, \tag{3.27}$$

and for $[\lambda_k]_r<0$, it follows that

$$-\left|[\lambda_k]_r\right|+\max\{\mathrm{sgn}([\lambda_k]_r),0\}=[\lambda_k]_r<0. \tag{3.28}$$

Based on (3.26), (3.27), (3.28), and the definition of $e_{\lambda_k}$ in (2.22), equality (3.23) has been established. Hence, for a predefined constant $c_1\in(0,1)$, we have

$$\psi(\alpha):=p(x_k+\alpha d_k^{\mathrm{d}};\mu_k)-p(x_k;\mu_k)-c_1 e_{\lambda_k}\alpha=(1-c_1)e_{\lambda_k}\alpha+\mathscr{O}(\alpha^2).$$

Clearly, $\psi(0) = 0$ and $\psi'_+(0) = (1-c_1)e_{\lambda_k} < 0$. As a result, there exists $\bar{\alpha} > 0$ such that the Armijo-like condition (2.21) holds for all $\alpha \in [0, \bar{\alpha}]$. □

The next lemma tells us that, under Assumption 3.5, the dropping step direction is computed within Algorithm 1 only finitely many times.

**Lemma 3.10.** *Let Assumptions 3.1-3.5 hold. Let $\bar{x}$ be a stationary point of $p(x;\mu_k)$ which is not a first-order point. There exist two positive scalars $\Delta$ and $\beta$ such that $\|x_k - \bar{x}\|_2 \le \Delta$ and $\mathrm{AC}(x_k, \varepsilon_k) = \mathrm{AC}(\bar{x}, 0)$ imply*

*(1) $[\lambda_k]_r \notin [0,1]$ for some $r \in \mathrm{AC}(\bar{x}, 0)$, and*
*(2) $e_{\lambda_k} < -\beta$.*

*Consequently, for all sufficiently large $k$, the dropping step direction is not taken in Algorithm 1.*

*Proof.* See parts 3 and 4 of [18, Theorem 1]. □

The subsequent lemma is a variant of [18, Lemma 1] and [46, Lemma 3.3].

**Lemma 3.11.** *Let Assumptions 3.1-3.4 hold. Let*

*(1) $\bar{x}$ be a strict scond-order point of $p(x;\bar{\mu})$ in $\mathscr{X}$, for some $\bar{\mu} > 0$,*
*(2) $\mu_k = \bar{\mu}$ and $\mathrm{AC}(x_k, \varepsilon_k) = \mathrm{AC}(\bar{x}, 0)$, for all sufficiently large $k$,*
*(3) there exist positive constants $b_1$ and $b_2$ such that for all nonzero vectors $w$,*

$$b_1\|w\|_2^2 \le w^T H_k(\mu_k) w \le b_2\|w\|_2^2.$$

*Then, there exist positive constants $\Delta_1$, $\Delta_2$, and $\beta > 0$ such that $\|x_k - \bar{x}\|_2 \le \Delta_1$ and*

$$\|(H_k(\mu_k) - G_k)s_k\|_2 \le \Delta_2\|s_k\|_2,$$

*with $s_k = \bar{Z}^T(\bar{x} - x_k)$, implying*

$$p(x_k + Z_k w_k + d_k^{\mathrm{v}}; \mu_k) \le p(x_k;\mu_k) - \beta\left(\|g_k^{\mathrm{p}}(\mu_k)\|_2^2 + \left\|[c(x_k)]_{\mathrm{AC}(x_k,\varepsilon_k)}\right\|_1\right),$$

*where $w_k$ is a solution of (2.23) and $d_k^{\mathrm{v}}$ is computed as (2.15).*

*Proof.* Refer to the proof of [46, Lemma 3.3]. □

Now, we state and prove our main global convergence result using the preceding results.

**Theorem 3.1.** *Suppose that Algorithm 1 is employed to solve problem (NLP) starting from an arbitrary initail point $x_0$. Provided that Assumptions 3.1-3.5 hold, exactly one of the following occurs.*

*(1) Algorithm 1 terminates finitely with either a first-order point of the penalty function $p$ or an infeasible stationary point of the constraint violation function $h$, in lines 42 or 13 or 21, respectively.*
*(2) Algorithm 1 produces infinitely many iterates $\{x_k\}$, and there exists $k_0 \ge 0$ such that all the iterations $k \ge k_0$ are infeasible. In this case, there exists a limit point $x^*$ of $\{x_k\}$ that is an infeasible stationary point of the constraint violation function $h$.*
*(3) Algorithm 1 generates infinitely many iterates $\{x_k\}$, $\mu_k = \bar{\mu} > 0$ for all $k$ sufficiently large, and there exists a limit point $x^*$ of $\{x_k\}$ that is a first-order point of the penalty function.*

*Proof.* We assume that cases 1 and 2 do not occur, and we establish the proof for case 3. Since case 1 does not occur, we can deduce that Algorithm 1 generates an infinite sequence of iterations. Moreover, since case 2 is not present, we can conclude that $\mu_k = \bar{\mu} > 0$ for all $k$ sufficiently large. Therefore, with Assumption 3.5 and the results from lemmas 3.8 and 3.10, the outcome for case 3 is at hand. □

In this section, we showed that the sequence generated by our algorithm starting from an arbitrary initial point converges to a stationary point. Indeed, Theorem 3.1 guarantees that if the algorithm does not identify an infeasible stationary point, then, for any stationary tolerance $\tau > 0$, there exists an index $k_0$ sufficiently large so that all iterations $k \geq k_0$ are local and the Newton step direction is used. In the following section, we establish a two-step superlinear local rate of convergence of the algorithm.

## 4. Local Convergence

To have a superlinearly convergent projected exact penalty algorithm, employment of a quasi-Newton approximation of $G_k$ as defined in (2.24) during local iterations is required. We choose $H_k := H_k(\mu_k)$ as a symmetric and positive-definite approximation of $G_k$ fulfilling the following secant relation:

$$H_k s_{k-1} = y_{k-1}; \qquad \begin{aligned} y_{k-1} &:= Z_k^T \left(\mu_k \left(\nabla_x f(x_k) - \nabla_x f(x_{k-1})\right) + A_{k-1}\lambda_k\right), \\ s_{k-1} &:= Z_k^T (x_k - x_{k-1}). \end{aligned}$$

As stated in Section 2 and similar to the approach of Nocedal and Overton [50], the approximate projected Hessian is updated using either $H_k = H_{k-1} + \mathrm{DFP}(H_{k-1}, s_{k-1}, y_{k-1})$ or $H_k = H_{k-1} + \mathrm{BFGS}(H_{k-1}, s_{k-1}, y_{k-1})$, only if condition (2.32) is met. If (2.32) does not hold, then $H_k$ remains unchanged as $H_{k-1}$.

Here, we present a two-step superlinear local rate of convergence for our algorithm, akin to the analysis of [50], under the following assumptions.

**Assumption 4.1.** Let assumptions 3.1-3.4 be satisfied. Furthermore, let matrix functions $Y(x)$ and $Z(x)$ be derived through a specific implementation of the QR decomposition of the normal matrix $A(x)$, ensuring that $N(A(x)^T) = R(Z(x))$ and $R(A(x)) = R(Y(x))$, are Lipschitz continuous on $\mathscr{X}$. Moreover, assume that $\nabla^2_{xx} f(x)$, $\nabla^2_{xx} c_i(x)$, $i = 1, \ldots, m$, are Lipschitz continuous on $\mathscr{X}$.

**Assumption 4.2.** For $(x^*, \lambda^*, \mu^*)$, the limit point of the generated sequence $\{(x_k, \lambda_k, \mu_k)\}$ produced by Algorithm 1, we have $\mathrm{AC}(x_k, \varepsilon_k) = \mathrm{AC}(x^*, 0)$, for all sufficiently large $k$.

Note that Assumption 4.2 is satisfied for our algorithm due to adjustment of $\varepsilon_k$ in every iteration as specified in line 6 of Algorithm 1. The following lemma due to Nocedal and Overton [50] states a sufficient condition for a two-step superlinear convergence.

**Lemma 4.1.** *Assume that assumptions 4.1 and 4.2 hold. Let*

$$G^* := Z(x^*)^T \left( \mu^* \nabla^2_{xx} f(x^*) - \sum_{i \in \mathrm{AC}(x^*, 0)} [\lambda^*]_i \nabla^2_{xx} c_i(x^*) \right) Z(x^*).$$

*If the computed* $\{H_k\}$ *is such that, for a constant* $\kappa > 0$, $\|H_k^{-1}\| \leq \kappa$, *for all sufficiently large* $k$, *and*

$$\lim_{k\to\infty} \frac{\left\|(H_k - G^*)Z_k^T(x_{k+1} - x_k)\right\|_2}{\|x_{k+1} - x_k\|_2} = 0, \tag{4.1}$$

*then* $x_k \to x^*$ *at a two-step Q-superlinear rate; i.e.,*

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|_2}{\|x_{k-1} - x^*\|_2} = 0.$$

*Proof.* See [50, Theorem 4.1 and Corollary 4.1]. □

Now, according to Lemma 3.10, Lemma 3.11, and Theorem 3.1, if Algorithm 1 does not detect an infeasible stationary point, then all sufficiently large iterations of the algorithm are local and the full Newton step direction is utilized. Therefore, analogous to the analyses of [50], it can be shown that condition (4.1) is satisfied and Algorithm 1 has a two-step Q-superlinear rate of convergence. Therefore, the following result is at hand.

**Theorem 4.1.** *Let assumptions 4.1 and 4.2 hold. Assume that the approximate projected Hessians,* $H_k$, *are updated by (2.30). Let* $\{x_k\}$ *be generated by Algorithm 1. Then, The sequence* $\{x_k\}$ *converges to* $x^*$ *at a two-step Q-superlinear rate.*

*Proof.* See proof of [50, Theorem 4.4]. □

## 5. Experimental Results

We made a MATLAB implementation of Algorithm 1, named as TRLSEP, and tested it on some small, medium and large scale test problems from the CUTEst library [35], as listed in Table 1, Table 2 and Table 3, respectively. In these tables, $n$ and $m$ stand for the number of variables and number of constraints, respectively. Note that the values of the algorithm parameters used in our implementation were set as:

$$\theta = 10^{-6},\ \gamma = 10^{-14},\ \eta = 0.001,\ \eta_1 = 0.1,\ \eta_2 = 0.5,\ \rho^{\mathrm{s}} = 0.25,\ \rho^{\mathrm{vs}} = 0.75,\ c_1 = 10^{-2},$$
$$c_2 = 10^{-5},\ \gamma_c = 0.5,\ \gamma_e = 2,\ \delta_0 = 10,\ \varepsilon_0 = 0.3,\ \tau = 0.3,\ \mu_0 = 1,\ \varphi = 1,\ \upsilon = 0.01.$$

Here we present comparative results of the MATLAB implementation of our algorithm (TRLSEP), KINTRO/Active-Set [8, 11], SNOPT [30], filterSQP [29] and LANCELOT [22] software packages. Numerical results of testing KINTRO/Active-Set, SNOPT, filterSQP and LANCELOT on 282 small scale test problems in Table 1 were taken from [57]. Moreover, Numerical results of testing KINTRO/Active-Set, SNOPT, filterSQP, and LANCELOT on 73 medium scale test problems in Table 2 and 61 large scale test problems in Table 3 were taken from NEOS server [25, 27, 31, 58]. We ran TRLSEP on a laptop with CPU Intel(R) Core(TM) i7-5500U 2.4 GHz×4, 8 GB of RAM memory, double precision format and Linux OS (Ubuntu 18.04.6 LTS).

The KNITRO/Active-Set is a powerful and effective solver within the KNITRO software package [11], designed for solving NLPs using an active-set strategy and the exact penalty function. It identifies the active set and updates the penalty parameter by solving a linear programming problem. Subsequently, using the obtained active set, an equality constrained quadratic programming problem is formed and solved to calculate the step direction. The exact penalty function serves as a merit function for evaluating the computed step direction.

The filterSQP algorithm is a powerful tool for solving nonlinear optimization problems using an integration of SQP with a filter trust region technique [29]. In this algorithm, the search direction is computed by solving a general quadratic program. Then, the computed search direction is evaluated by a filter trust region technique. To approach the iterates towards the feasible region, a restoration phase is utilized. Moreover, to avoid the occurrence of the Maratos phenomenon [48], it employs second-order corrector steps.

SNOPT (Sparse Nonlinear OPTimizer) [30] is a powerfull and robust software package for solving large-scale NLPs utilizing a sparse SQP (Sequential Quadratic Programming) algorithm along with limited-memory quasi-Newton approximation of the Hessian of the Lagrangian. Employing an augmented Lagrangian merit function, convergence is ensured from any starting point while methodically handling infeasible problems through elastic bounds on nonlinear constraints.

LANCELOT [22] is an open source solver in the GALAHAD software package [38] for solving nonlinear optimization problems, which implements a trust-region strategy combined with a spectral projected gradient approach. It utilizes the augmented Lagrangian function to evaluate the step directions.

We used the performance profile of Dolan and Moré [28] to compare the considered programs based on the number of function evaluations and number of gradient evaluations. The performance profile, as explained in [28], aims to compare the results obtained by the solvers in a set $\mathscr{S}$ on a problem set $\mathscr{P}$. Letting $\theta_{p,s}$ stand for number of function evaluations, or number of gradient evaluations for solving problem $p \in \mathscr{P}$ by solver $s \in \mathscr{S}$, the performance ratio is defined as follows:

$$r_{p,s} = \frac{\theta_{p,s}}{\min\{\theta_{p,s} | s \in \mathscr{S}\}}.$$

Moreover, for every $s \in \mathscr{S}$ and $t \geq 1$, the following cumulative distribution function for the performance ratio is defined:

$$P_s(t) = \frac{\textbf{size}\{p \in \mathscr{P} | r_{p,s} \leq t\}}{\textbf{size}(\mathscr{P})}.$$

Obviously, $P_s(1)$ represents the percentage of problems for which solver $s$ performs the best. Also, if $t \to \infty$, $P_s(t)$ indicates the percentage of test problems in $\mathscr{P}$ that are solved successfully by solver $s \in \mathscr{S}$.

Figures 1 and 2 show the performance profiles for number of function evaluations and number of gradient evaluations, respectively, obtained from testing TRLSEP, KNITRO/Active-Set, filterSQP, LANCELOT and SNOPT on the small test problems given in Table 1. The performance profiles for number of function evaluations and number of gradient evaluations on the medium test problems given in Table 2 are shown in figures 3 and 4, respectively. Also, figures 5 and 6 show the performance profiles for the number of function evaluations and number of gradient evaluations, respectively, on the large test problems given in Table 3. The resulting profiles show that TRLSEP is competitive as compared to the four software packages KNITRO/Active-Set, filterSQP, LANCELOT and SNOPT for solving general constrained nonlinear optimization problems.

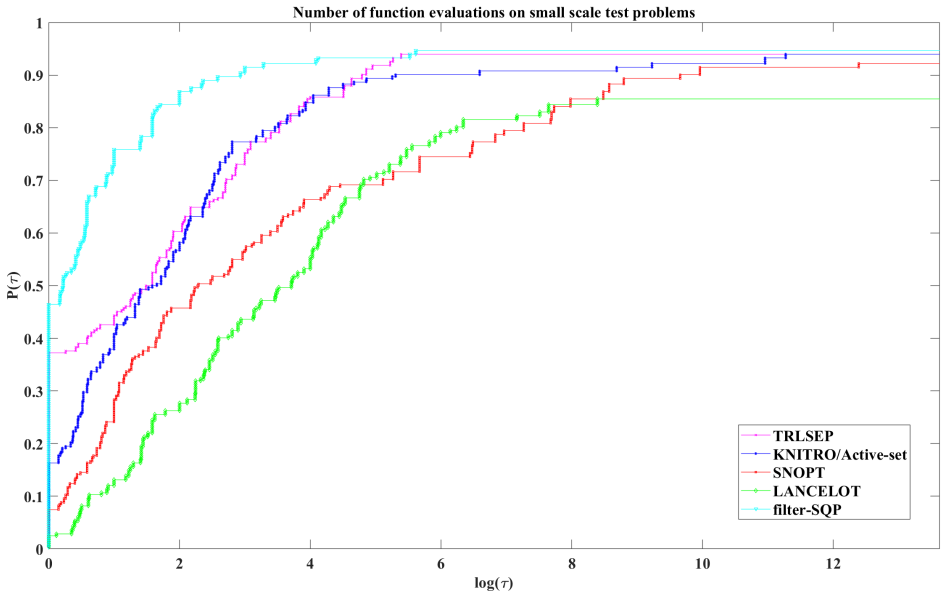


FIGURE 1. Performance profiles for number of function evaluations on small test problems corresponding to TRLSEP, KNITRO/Active-Set, filterSQP, SNOPT and LANCELOT.

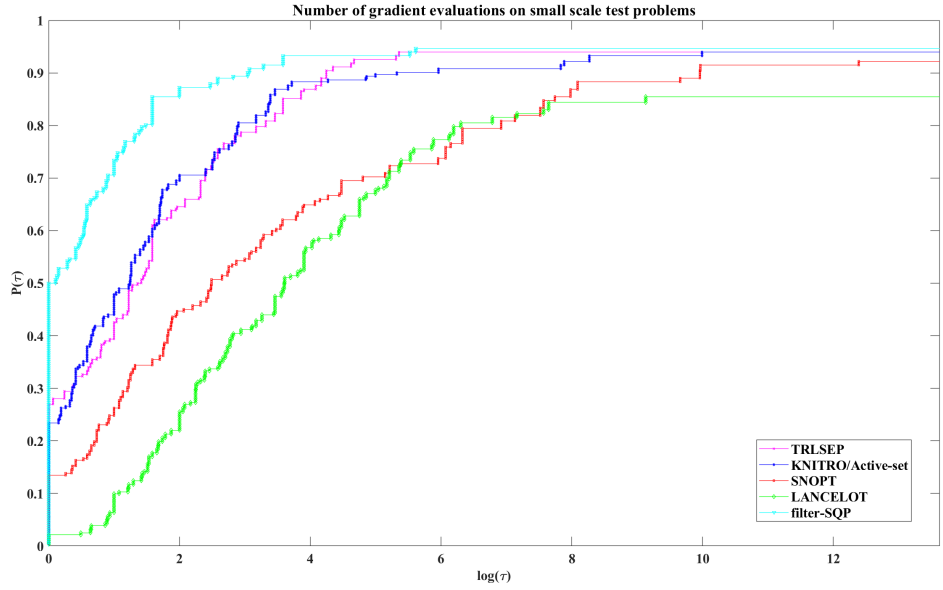


FIGURE 2. Performance profiles for number of gradient evaluations on small test problems corresponding to TRLSEP, KNITRO/Active-Set, filterSQP, SNOPT and LANCELOT.

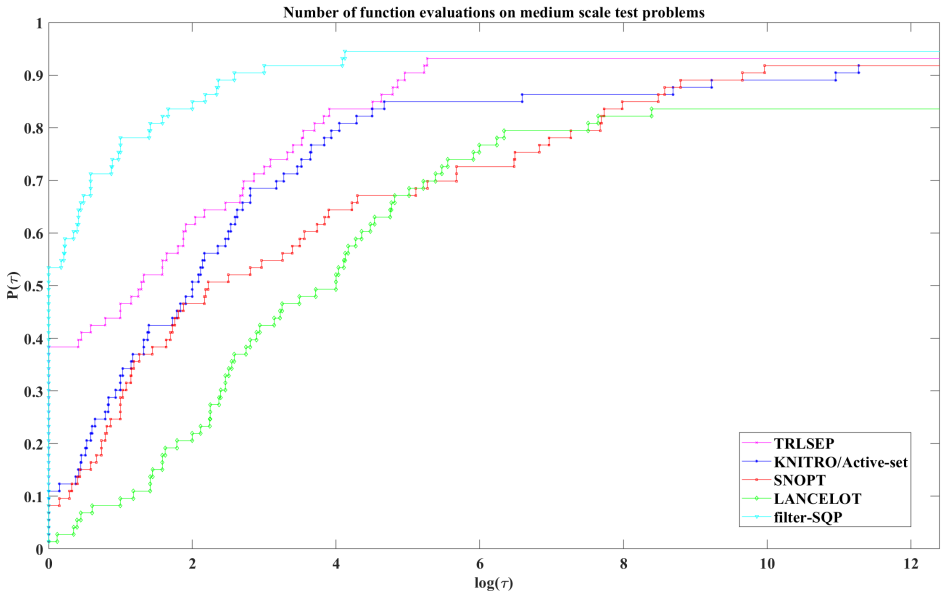


FIGURE 3. Performance profiles for number of function evaluations on medium test problems corresponding to TRLSEP, KNITRO/Active-Set, filter-SQP, SNOPT and LANCELOT.

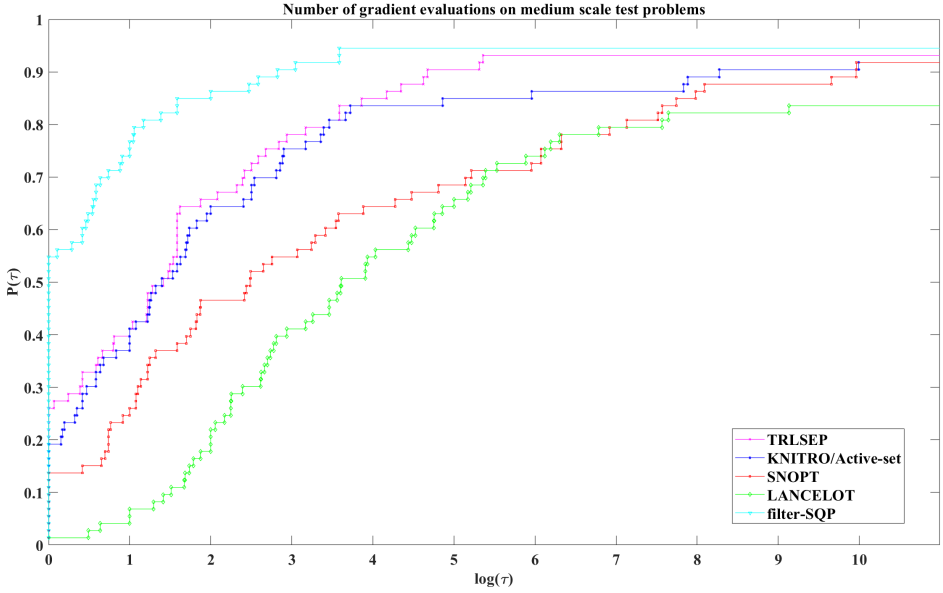


FIGURE 4. Performance profiles for number of gradient evaluations on medium test problems corresponding to TRLSEP, KNITRO/Active-Set, filter-SQP, SNOPT and LANCELOT.

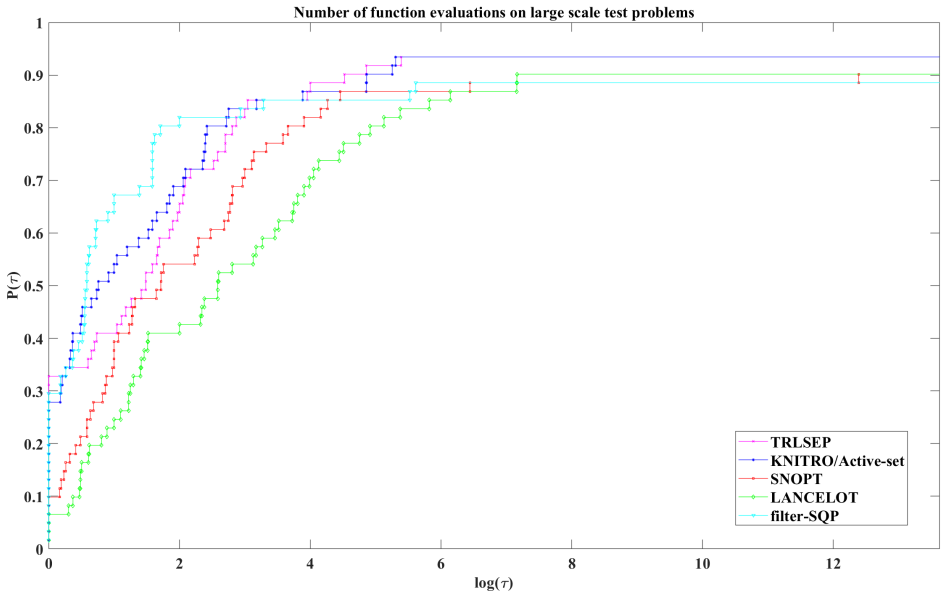


FIGURE 5. Performance profiles for number of function evaluations on large test problems corresponding to TRLSEP, KNITRO/Active-Set, filterSQP, SNOPT and LANCELOT.

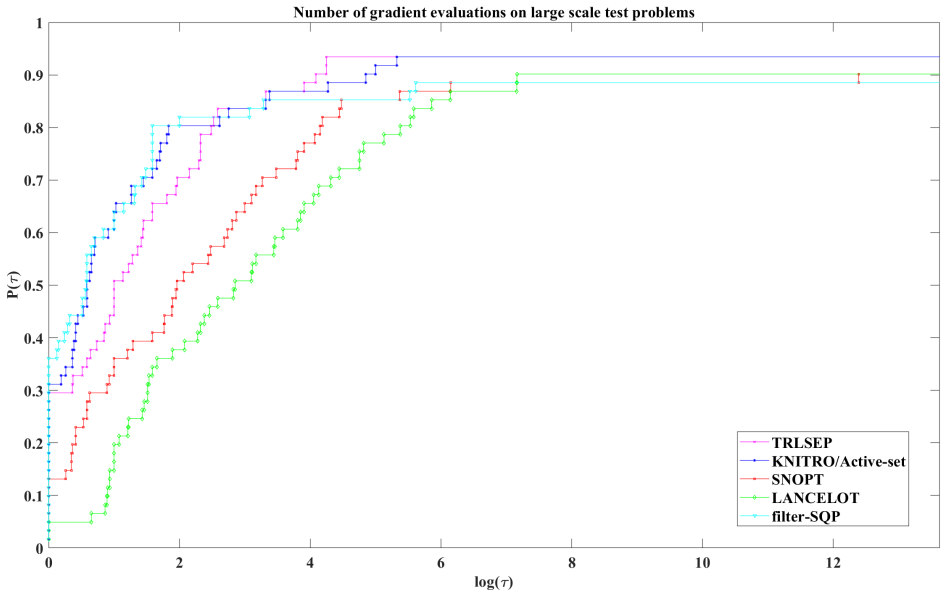


FIGURE 6. Performance profiles for number of gradient evaluations on large test problems corresponding to TRLSEP, KNITRO/Active-Set, filterSQP, SNOPT and LANCELOT.

TABLE 1. CUTEst test problems of size $m \geq 1$ and $\max\{m,n\} < 100$.

| # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | aircrfta | 8 | 5 | 48 | dnieper | 61 | 24 | 95 | hs11 | 2 | 1 |
| 2 | airport | 84 | 42 | 49 | dual1 | 85 | 1 | 96 | hs12 | 2 | 1 |
| 3 | aljazzaf | 3 | 1 | 50 | dual2 | 96 | 1 | 97 | hs13 | 2 | 1 |
| 4 | allinitc | 4 | 1 | 51 | dual4 | 75 | 1 | 98 | hs14 | 2 | 2 |
| 5 | alsotame | 2 | 1 | 52 | eigencco | 30 | 15 | 99 | hs15 | 2 | 3 |
| 6 | argauss | 3 | 15 | 53 | eigmaxc | 22 | 22 | 100 | hs16 | 2 | 4 |
| 7 | avgasa | 8 | 10 | 54 | eigminc | 22 | 22 | 101 | hs17 | 2 | 4 |
| 8 | avgasb | 8 | 10 | 55 | expfita | 5 | 22 | 102 | hs18 | 2 | 4 |
| 9 | avion2 | 49 | 15 | 56 | extrasim | 2 | 1 | 103 | hs19 | 2 | 4 |
| 10 | batch | 48 | 73 | 57 | fccu | 19 | 8 | 104 | hs20 | 2 | 4 |
| 11 | biggsc4 | 4 | 7 | 58 | fletcher | 4 | 4 | 105 | hs21 | 2 | 3 |
| 12 | booth | 2 | 2 | 59 | genhs28 | 10 | 8 | 106 | hs22 | 2 | 2 |
| 13 | bt1 | 2 | 1 | 60 | gigomez1 | 3 | 3 | 107 | hs23 | 2 | 5 |
| 14 | bt10 | 2 | 2 | 61 | goffin | 51 | 50 | 108 | hs24 | 2 | 3 |
| 15 | bt11 | 5 | 3 | 62 | gottfr | 2 | 2 | 109 | hs25 | 3 | 3 |
| 16 | bt12 | 5 | 3 | 63 | gridnetg | 60 | 36 | 110 | hs26 | 3 | 1 |
| 17 | bt13 | 5 | 2 | 64 | gridneth | 60 | 36 | 111 | hs27 | 3 | 1 |
| 18 | bt2 | 3 | 1 | 65 | gridneti | 60 | 36 | 112 | hs28 | 3 | 1 |
| 19 | bt3 | 5 | 3 | 66 | haifas | 13 | 9 | 113 | hs29 | 3 | 1 |
| 20 | bt4 | 3 | 2 | 67 | haldmads | 6 | 42 | 114 | hs30 | 3 | 4 |
| 21 | bt5 | 3 | 2 | 68 | hatfldf | 3 | 3 | 115 | hs31 | 3 | 4 |
| 22 | bt6 | 5 | 2 | 69 | hatfldg | 25 | 25 | 116 | hs32 | 3 | 2 |
| 23 | bt7 | 5 | 3 | 70 | hatfldh | 4 | 7 | 117 | hs33 | 3 | 3 |
| 24 | bt8 | 5 | 2 | 71 | heart6 | 6 | 6 | 118 | hs34 | 3 | 5 |
| 25 | bt9 | 4 | 2 | 72 | heart8 | 8 | 8 | 119 | hs35 | 3 | 1 |
| 26 | byrdsphr | 3 | 2 | 73 | himmelba | 2 | 2 | 120 | hs36 | 3 | 4 |
| 27 | cantilvr | 5 | 1 | 74 | himmelbc | 2 | 2 | 121 | hs37 | 3 | 2 |
| 28 | catena | 33 | 10 | 75 | himmelbd | 2 | 2 | 122 | hs39 | 4 | 2 |
| 29 | cb2 | 3 | 3 | 76 | himmelbe | 3 | 3 | 123 | hs40 | 4 | 3 |
| 30 | cb3 | 3 | 3 | 77 | himmelbj | 45 | 16 | 124 | hs41 | 4 | 5 |
| 31 | chaconn1 | 3 | 3 | 78 | himmelbk | 24 | 14 | 125 | hs42 | 4 | 2 |
| 32 | chaconn2 | 3 | 3 | 79 | himmelp2 | 2 | 1 | 126 | hs43 | 4 | 3 |
| 33 | cluster | 2 | 2 | 80 | himmelp3 | 2 | 2 | 127 | hs44 | 4 | 6 |
| 34 | concon | 15 | 11 | 81 | himmelp4 | 2 | 3 | 128 | hs46 | 5 | 2 |
| 35 | congigmz | 3 | 5 | 82 | himmelp5 | 2 | 3 | 129 | hs47 | 5 | 3 |
| 36 | coshfun | 61 | 20 | 83 | himmelp6 | 2 | 5 | 130 | hs48 | 5 | 2 |
| 37 | cresc4 | 6 | 8 | 84 | hong | 4 | 1 | 131 | hs49 | 5 | 2 |
| 38 | csfi1 | 5 | 4 | 85 | hs1 | 2 | 1 | 132 | hs50 | 5 | 3 |
| 39 | csfi2 | 5 | 4 | 86 | hs2 | 2 | 1 | 133 | hs51 | 5 | 3 |
| 40 | dallass | 46 | 31 | 87 | hs3 | 2 | 1 | 134 | hs52 | 5 | 3 |
| 41 | degenlpa | 20 | 15 | 88 | hs4 | 2 | 2 | 135 | hs53 | 5 | 3 |
| 42 | degenlpb | 20 | 15 | 89 | hs5 | 2 | 2 | 136 | hs54 | 6 | 1 |
| 43 | demymalo | 3 | 3 | 90 | hs6 | 2 | 1 | 137 | hs55 | 6 | 6 |
| 44 | dipigri | 7 | 4 | 91 | hs7 | 2 | 1 | 138 | hs56 | 7 | 4 |
| 45 | disc2 | 29 | 23 | 92 | hs8 | 2 | 2 | 139 | hs57 | 2 | 3 |
| 46 | discs | 36 | 66 | 93 | hs9 | 2 | 1 | 140 | hs59 | 2 | 3 |
| 47 | dixchlng | 10 | 5 | 94 | hs10 | 2 | 1 | 141 | hs60 | 3 | 1 |

**Table 1**: CUTEst test problems of size $m \geq 1$ and $\max\{m,n\} < 100$ (continued).

| # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 142 | hs61 | 3 | 2 | 189 | hs111 | 10 | 3 | 236 | oslbqp | 8 | 8 |
| 143 | hs62 | 3 | 1 | 190 | hs111lnp | 10 | 3 | 237 | pentagon | 6 | 15 |
| 144 | hs63 | 3 | 2 | 191 | hs112 | 10 | 3 | 238 | polak1 | 3 | 2 |
| 145 | hs64 | 3 | 1 | 192 | hs113 | 10 | 8 | 239 | polak2 | 11 | 2 |
| 146 | hs65 | 3 | 4 | 193 | hs114 | 14 | 15 | 240 | polak3 | 12 | 10 |
| 147 | hs66 | 3 | 5 | 194 | hs116 | 13 | 28 | 241 | polak4 | 3 | 3 |
| 148 | hs70 | 24 | 21 | 195 | hs117 | 15 | 5 | 242 | polak5 | 3 | 2 |
| 149 | hs71 | 4 | 2 | 196 | hs118 | 15 | 17 | 243 | polak6 | 5 | 4 |
| 150 | hs72 | 4 | 6 | 197 | hs119 | 16 | 8 | 244 | portfl1 | 12 | 1 |
| 151 | hs73 | 4 | 3 | 198 | hs21mod | 7 | 7 | 245 | portfl2 | 12 | 1 |
| 152 | hs74 | 4 | 4 | 199 | hs268 | 5 | 5 | 246 | portfl3 | 12 | 1 |
| 153 | hs75 | 4 | 4 | 200 | hs35mod | 3 | 2 | 247 | portfl4 | 12 | 1 |
| 154 | hs76 | 4 | 3 | 201 | hs3mod | 2 | 1 | 248 | portfl6 | 12 | 1 |
| 155 | hs77 | 5 | 2 | 202 | hs44new | 4 | 6 | 249 | powellbs | 2 | 2 |
| 156 | hs78 | 5 | 3 | 203 | hs99exp | 31 | 21 | 250 | powellsq | 2 | 2 |
| 157 | hs79 | 5 | 3 | 204 | hubfit | 2 | 1 | 251 | prodpl0 | 60 | 29 |
| 158 | hs80 | 5 | 3 | 205 | hypcir | 2 | 2 | 252 | prodpl1 | 60 | 29 |
| 159 | hs81 | 5 | 3 | 206 | kiwcresc | 3 | 2 | 253 | pspdoc | 4 | 1 |
| 160 | hs83 | 5 | 3 | 207 | lakes | 90 | 78 | 254 | recipe | 3 | 3 |
| 161 | hs84 | 5 | 3 | 208 | launch | 25 | 29 | 255 | res | 18 | 14 |
| 162 | hs85 | 5 | 21 | 209 | lewispol | 6 | 9 | 256 | rk23 | 17 | 11 |
| 163 | hs86 | 5 | 10 | 210 | linspanh | 97 | 33 | 257 | robot | 14 | 9 |
| 164 | hs87 | 6 | 4 | 211 | loadbal | 31 | 31 | 258 | rosenmmx | 5 | 4 |
| 165 | hs88 | 32 | 31 | 212 | lootsma | 3 | 3 | 259 | s365mod | 9 | 11 |
| 166 | hs89 | 33 | 31 | 213 | lotschd | 12 | 7 | 260 | simbqp | 2 | 1 |
| 167 | hs90 | 34 | 31 | 214 | lsnnodoc | 5 | 4 | 261 | simpllpa | 2 | 2 |
| 168 | hs91 | 35 | 31 | 215 | lsqfit | 2 | 1 | 262 | simpllpb | 2 | 3 |
| 169 | hs92 | 36 | 31 | 216 | madsen | 3 | 6 | 263 | snake | 2 | 2 |
| 170 | hs93 | 6 | 2 | 217 | makela1 | 3 | 2 | 264 | spanhyd | 97 | 33 |
| 171 | hs95 | 6 | 4 | 218 | makela2 | 3 | 3 | 265 | spiral | 3 | 2 |
| 172 | hs96 | 6 | 4 | 219 | makela3 | 21 | 20 | 266 | ssnlbeam | 33 | 20 |
| 173 | hs97 | 6 | 4 | 220 | makela4 | 21 | 40 | 267 | stancmin | 3 | 2 |
| 174 | hs98 | 6 | 4 | 221 | maratos | 2 | 1 | 268 | supersim | 2 | 2 |
| 175 | hs99 | 7 | 2 | 222 | matrix2 | 6 | 2 | 269 | swopf | 83 | 92 |
| 176 | hs100 | 7 | 4 | 223 | mconcon | 15 | 16 | 270 | synthes1 | 6 | 6 |
| 177 | hs100lnp | 7 | 2 | 224 | mifflin1 | 3 | 2 | 271 | tame | 2 | 1 |
| 178 | hs100mod | 7 | 4 | 225 | mifflin2 | 3 | 2 | 272 | try-b | 2 | 1 |
| 179 | hs101 | 7 | 6 | 226 | minmaxbd | 5 | 20 | 273 | twobars | 2 | 2 |
| 180 | hs102 | 7 | 6 | 227 | minmaxrb | 3 | 4 | 274 | vanderm4 | 9 | 17 |
| 181 | hs103 | 7 | 6 | 228 | mistake | 9 | 13 | 275 | weeds | 15 | 12 |
| 182 | hs104 | 8 | 6 | 229 | mwright | 5 | 3 | 276 | womflet | 3 | 3 |
| 183 | hs105 | 2 | 1 | 230 | odfits | 10 | 6 | 277 | zangwil3 | 3 | 3 |
| 184 | hs106 | 8 | 14 | 231 | optcntrl | 32 | 21 | 278 | zecevic2 | 2 | 4 |
| 185 | hs107 | 15 | 20 | 232 | optmass | 70 | 55 | 279 | zecevic3 | 2 | 4 |
| 186 | hs108 | 9 | 14 | 233 | optprloc | 30 | 30 | 280 | zecevic4 | 2 | 4 |
| 187 | hs109 | 9 | 10 | 234 | orthregb | 27 | 6 | 281 | zigzag | 64 | 50 |
| 188 | hs110 | 10 | 1 | 235 | orthrege | 36 | 20 | 282 | zy2 | 3 | 2 |

TABLE 2. CUTEst test problems of size $m \geq 1$ and $100 \leq \max\{m,n\} < 1000$.

| # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | argtrig | 100 | 100 | 26 | eigmina | 101 | 101 | 51 | qpcstair | 467 | 356 |
| 2 | britgas | 450 | 360 | 27 | eigminb | 101 | 101 | 52 | qpnboei1 | 384 | 351 |
| 3 | catenary | 501 | 166 | 28 | expfitb | 5 | 102 | 53 | qpnboei2 | 143 | 166 |
| 4 | chandheq | 100 | 100 | 29 | expfitc | 5 | 502 | 54 | qpnstair | 467 | 356 |
| 5 | core2 | 157 | 134 | 30 | gouldqp3 | 699 | 349 | 55 | reading3 | 202 | 102 |
| 6 | cresc100 | 6 | 200 | 31 | gpp | 250 | 498 | 56 | sawpath | 583 | 774 |
| 7 | cresc50 | 6 | 100 | 32 | grouping | 100 | 125 | 57 | smbank | 117 | 64 |
| 8 | dallasl | 906 | 667 | 33 | hadamard | 648 | 257 | 58 | smmpsf | 720 | 263 |
| 9 | dallasm | 196 | 151 | 34 | haifam | 99 | 150 | 59 | sseblin | 194 | 72 |
| 10 | dittert | 327 | 264 | 35 | hanging | 300 | 180 | 60 | ssebnln | 194 | 96 |
| 11 | dixchlnv | 100 | 50 | 36 | himmelbi | 100 | 12 | 61 | static3 | 434 | 96 |
| 12 | dtoc1nd | 735 | 490 | 37 | hvycrash | 204 | 150 | 62 | steenbra | 432 | 108 |
| 13 | dual3 | 111 | 1 | 38 | integreq | 102 | 100 | 63 | steenbrb | 468 | 108 |
| 14 | dualc1 | 9 | 215 | 39 | kissing | 127 | 903 | 64 | steenbrc | 540 | 126 |
| 15 | dualc2 | 7 | 229 | 40 | lch | 600 | 1 | 65 | steenbrd | 468 | 108 |
| 16 | dualc5 | 8 | 278 | 41 | madsschj | 81 | 158 | 66 | steenbre | 540 | 126 |
| 17 | dualc8 | 22 | 15 | 42 | minc44 | 311 | 262 | 67 | steenbrf | 468 | 108 |
| 18 | eg3 | 101 | 200 | 43 | mosarqp2 | 900 | 600 | 68 | steenbrg | 540 | 126 |
| 19 | eigena2 | 110 | 55 | 44 | optctrl3 | 122 | 81 | 69 | trimloss | 142 | 75 |
| 20 | eigenaco | 110 | 55 | 45 | optctrl6 | 122 | 81 | 70 | twirism1 | 343 | 313 |
| 21 | eigenb2 | 110 | 55 | 46 | orthrds2 | 203 | 100 | 71 | vanderm1 | 100 | 199 |
| 22 | eigenbco | 110 | 55 | 47 | orthrega | 517 | 256 | 72 | vanderm2 | 100 | 199 |
| 23 | eigenc2 | 462 | 231 | 48 | pt | 2 | 501 | 73 | vanderm3 | 100 | 199 |
| 24 | eigmaxa | 101 | 101 | 49 | qpcboei1 | 384 | 351 | | | | |
| 25 | eigmaxb | 101 | 101 | 50 | qpcboei2 | 143 | 166 | | | | |

## 6. CONCLUDING REMARKS

A trust region-line search projected exact penalty algorithm was proposed. The iterations of the algorithm were categorized into three types: infeasible, almost feasible, and local. Necessary and sufficient conditions for identifying an infeasible stationary point were established, along with a novel approach for updating the penalty parameter. In infeasible iterations, the step direction was determined using an approximate solution of a trust region subproblem. Almost feasible iterations involved a combined step direction: a horizontal step for minimizing the penalty function and a vertical step for preserving the active set to enhance feasibility. Local iterations entailed computing Lagrange multipliers through solving a linear least squares problem. If the computed Lagrange multipliers exceeded an acceptable range, a dropping step was initiated; otherwise, either a stationary point was detected or a Newton step direction was calculated. A computed step direction underwent evaluation, and the iterate was updated by a new strategy. Moreover, the approximate projected Hessian was updated by the BFGS updating formula in local iterations. The global convergence of the proposed algorithm was demonstrated through a distinct approach from the global convergence analysis of Coleman and Conn, while

TABLE 3. CUTEst test problems of size $m \geq 1$ and $\max\{m,n\} \geq 1000$.

| # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ | # | Problem | $n$ | $m$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | artif | 5000 | 5000 | 22 | bratu2dt | 5184 | 4900 | 43 | minperm | 1113 | 1033 |
| 2 | aug3d | 3873 | 1000 | 23 | bratu3d | 4913 | 3375 | 44 | model | 1542 | 38 |
| 3 | aug3dc | 3873 | 1000 | 24 | broydnbd | 5000 | 5000 | 45 | msqrta | 1024 | 1024 |
| 4 | aug3dcqp | 3873 | 1000 | 25 | cbratu2d | 1058 | 882 | 46 | msqrtb | 1024 | 1024 |
| 5 | aug3dqp | 3873 | 1000 | 26 | cbratu3d | 2000 | 1024 | 47 | ngone | 100 | 1273 |
| 6 | bdvalue | 5002 | 5000 | 27 | clnlbeam | 1503 | 1000 | 48 | oet1 | 3 | 1002 |
| 7 | bigbank | 2230 | 1112 | 28 | corkscrw | 9006 | 7000 | 49 | oet2 | 3 | 1002 |
| 8 | bloweya | 2002 | 1002 | 29 | cresc132 | 6 | 2654 | 50 | oet3 | 4 | 1002 |
| 9 | bloweyb | 2002 | 1002 | 30 | dtoc1na | 1495 | 990 | 51 | oet7 | 7 | 1002 |
| 10 | bloweyc | 2002 | 1002 | 31 | dtoc1nb | 1495 | 990 | 52 | optcdeg2 | 1202 | 800 |
| 11 | brainpc0 | 6907 | 6900 | 32 | dtoc1nc | 1495 | 990 | 53 | optcdeg3 | 1202 | 800 |
| 12 | brainpc1 | 6907 | 6900 | 33 | dtoc2 | 5998 | 3996 | 54 | orthrdm2 | 4003 | 2000 |
| 13 | brainpc2 | 6907 | 6900 | 34 | dtoc5 | 9999 | 4999 | 55 | porous1 | 5184 | 4900 |
| 14 | brainpc3 | 6907 | 6900 | 35 | gausselm | 1496 | 3690 | 56 | porous2 | 5184 | 4900 |
| 15 | brainpc4 | 6907 | 6900 | 36 | gilbert | 1000 | 1 | 57 | semicon1 | 1002 | 1000 |
| 16 | brainpc5 | 6907 | 6900 | 37 | gridnetc | 7564 | 3844 | 58 | semicon2 | 1002 | 1000 |
| 17 | brainpc6 | 6907 | 6900 | 38 | gridnetd | 7565 | 3844 | 59 | sinrosnb | 1000 | 999 |
| 18 | brainpc7 | 6907 | 6900 | 39 | gridnete | 7565 | 3844 | 60 | svanberg | 5000 | 5000 |
| 19 | brainpc8 | 6907 | 6900 | 40 | gridnetf | 7565 | 3844 | 61 | yao | 2002 | 2000 |
| 20 | brainpc9 | 6907 | 6900 | 41 | ksip | 20 | 1001 | | | | |
| 21 | bratu2d | 5184 | 4900 | 42 | manne | 1095 | 730 | | | | |

relying on weaker assumptions. Furthermore, a two-step superlinear local rate of convergence of our algorithm was provided. Drawing from the theoretical findings of our work, we developed an efficient trust region-line search projected exact penalty method to address general nonlinear optimization problems. The obtained comparative experimental results underscored the competitiveness of our proposed algorithm as compared to a number well-developed NLP solvers.

**Acknowledgments**
The authors thank the Research Council of Sharif University of Technology for supporting this work.

## REFERENCES

[1] H. Ahmadzadeh, N. Mahdavi-Amiri, A competitive inexact nonmonotone filter SQP method: convergence analysis and numerical results, Optim. Meth. Softw. 37 (2022), 1310–1343.

[2] H. Ahmadzadeh, N. Mahdavi-Amiri, A new inexact nonmonotone filter sequential quadratic programming algorithm. In: M. Al-Baali, A. Purnama, L. Grandinetti, (eds.) Numerical Analysis and Optimization, pp. 1–24. Springer, Cham, 2021.

[3] M.R. Ansari, N. Mahdavi-Amiri, A robust combined trust region–line search exact penalty projected structured scheme for constrained nonlinear least squares, Optim. Meth. Softw. 30 (2015), 162–190.

[4] D.P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, 1st ed. Athena Scientific, Belmont, MA, 1996.

[5] N. Bidabadi, N. Mahdavi-Amiri, Superlinearly convergent exact penalty methods with projected structured secant updates for constrained nonlinear least squares, J. Optim. Theory Appl. 162 (2014), 154–190.
[6] N. Bidabadi, N. Mahdavi-Amiri, Superlinearly convergent exact penalty methods with projected structured secant updates for constrained nonlinear least squares, J. Optim. Theory Appl. 162 (2013), 154–190.
[7] N. Bidabadi, N. Mahdavi-Amiri, A two-step superlinearly convergent projected structured BFGS method for constrained nonlinear least squares, Optimization 62 (2013), 797–815.
[8] R.H. Byrd, N.I.M. Gould, J. Nocedal, R.A. Waltz, An algorithm for nonlinear optimization using linear programming and equality constrained subproblems, Math. Program. 100 (2003), 27–48.
[9] R.H. Byrd, G. Lopez-Calva, J. Nocedal, A line search exact penalty method using steering rules, Math. Program. 133 (2012), 39–73.
[10] R.H. Byrd, J. Nocedal, R.A Waltz, Steering exact penalty methods for nonlinear programming, Optim. Meth. Softw. 23 (2008), 197–213.
[11] R.H. Byrd, J. Nocedal, R.A. Waltz, Knitro: An integrated package for nonlinear optimization, In: G. Di Pillo, M. Roma, M. (eds.) , pp. 35–59, Springer, Boston, 2006.
[12] R.H. Byrd, N.I.M. Gould, J. Nocedal, R.A. Waltz, On the convergence of successive linear-quadratic programming algorithms, SIAM J. Optim. 16 (2005), 471–489.
[13] R.H. Byrd, R.B. Schnabel, G.A. Shultz, Approximate solution of the trust region problem by minimization over two-dimensional subspaces, Math. Program. 40 (1980), 247–263.
[14] C. Charalambous, A lower bound for the controlling parameters of the exact penalty functions, Math. Program. 15 (1978), 278–290.
[15] C. Charalambous, On conditions for optimality of the nonlinear $l_1$ problem, Math. Program. 17 (1979), 123–135.
[16] T.F. Coleman, A.R. Conn, Second-order conditions for an exact penalty function, Math. Program. 19 (1980), 178–185.
[17] T.F. Coleman, A.R. Conn, Nonlinear programming via an exact penalty function: Global analysis, Math. Program. 24 (1982), 137–161.
[18] T.F. Coleman, A.R. Conn, A.R.: On the local convergence of a quasi-newton method for the nonlinear programming problem, SIAM J. Numer. Anal. 21 (1984), 755–769.
[19] T.F. Coleman, A.R. Conn, Nonlinear programming via an exact penalty function: Asymptotic analysis, Math. Program. 24 (1982), 123–136.
[20] A.R. Conn, T. Pietrzykowski, A penalty function method converging directly to a constrained optimum, SIAM J. Numer. Anal. 14 (1977), 348–375.
[21] A.R. Conn, N.I.M. Gould, P. Toint, A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, SIAM J. Numer. Anal. 28 (1991), 545–572.
[22] A.R. Conn, N.I.M. Gould, P. Toint, Lancelot: a Fortran Package for Large-Scale Nonlinear Optimization (Release A). Springer, Berlin, Heidelberg, 1992.
[23] A.R. Conn, Constrained optimization using a nondifferentiable penalty function, SIAM J. Numer. Anal. 10 (1973), 760–784.
[24] A.R. Conn, N.I.M. Gould, P.L. Toint, Trust Region Methods, 1st edn. Society for Industrial and Applied Mathematics, Philadelphia, 2000.
[25] J. Czyzyk, M.P. Mesnier, J.J. More, The neos server, IEEE Comput. Sci. Eng. 5 (1988), 68–75.
[26] J.E. Dennis, Jr., J.J. Moré, Quasi-newton methods, motivation and theory, SIAM Rev. 19 (1977), 46–89.
[27] E.D. Dolan, The neos server 4.0 administrative guide. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.
[28] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002), 201–213.
[29] R. Fletcher, S. Leyffer, Nonlinear programming without a penalty function, Math. Program. 91 (2002), 239–269.
[30] P.E. Gill, W. Murray, M.A. Saunders, Snopt: An SQP algorithm for large-scale constrained optimization, SIAM Rev. 47 (2005), 99–131.

[31] W. Gropp, J.J. Moré, Optimization environments and the neos server. In: M.D. Buhman, A. Iserles, (eds.) Approximation Theory and Optimization: Tributes to M.J.D. Powell, pp. 167–182. Cambridge University Press, Cambridge, 1997.
[32] N.I.M. Gould, D.P. Robinson, A second derivative SQP method: Global convergence, SIAM J. Optim. 20 (2010), 2023–2048.
[33] N.I.M. Gould, Y. Loh, D. Robinson, A filter method with unified step computation for nonlinear optimization, SIAM J. Optim. 24 (2014), 175–209.
[34] N.I.M. Gould, Y. Loh, D. Robinson, A nonmonotone filter SQP method: Local convergence and numerical results, SIAM J. Optim. 25 (2015), 1885-1911.
[35] N.I.M. Gould, D. Orban, P.L. Toint Cutest: a constrained and unconstrained testing environment with safe threads for mathematical optimization, Comput. Optim. Appl. 60 (2015), 545–557.
[36] N.I.M. Gould, D.P. Robinson, A second derivative SQP method: Local convergence and practical issues, SIAM J. Optim. 20 (2010), 2049–2079.
[37] N.I.M. Gould, S. Lucidi, M. Roma, P.L. Toint, Solving the trust-region sub-problem using the lanczos method, SIAM J. Optim. 9 (1999), 504–525.
[38] N.I.M. Gould, D. Orban, P.L. Toint, Galahad, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization, ACM Trans. Math. Softw. 29 (2003), 353–372.
[39] S.-P. Han, O.L. Mangasarian, Exact penalty functions in nonlinear programming, Math. Program. 17 (1979), 251–269.
[40] S.P. Han, A globally convergent method for nonlinear programming, J. Optim. Theory Appl. 22 (1977), 297–309.
[41] M.R. Hestenes, Multiplier and gradient methods, J. Optim. Theory Appl. 4 (1969), 303–320.
[42] D. Luenberger, Control problems with kinks, IEEE Trans. Auto. Control 15 (1970), 570–575.
[43] N. Mahdavi-Amiri, R.H. Bartels, Constrained nonlinear least squares: An exact penalty approach with projected structured quasi-newton updates, ACM Trans. Math. Softw. 15 (1989), 220–242.
[44] N. Mahdavi-Amiri, M.R. Ansari, Superlinearly convergent exact penalty projected structured Hessian updating schemes for constrained nonlinear least squares: asymptotic analysis, Bull. Iranian Math. Soc. 38 (2012), 767–786.
[45] N. Mahdavi-Amiri, M.R. Ansari, A superlinearly convergent penalty method with nonsmooth line search for constrained nonlinear least squares, Sultan Qaboos Univ. J. Sco. 17 (2012), 103–124.
[46] N. Mahdavi-Amiri, M.R. Ansari, A superlinearly convergent exact penalty method for constrained nonlinear least squares: global analysis, Optimization, 62 (2013), 675–691,
[47] N. Mahdavi-Amiri, N. Bidabadi, Constrained nonlinear least squares: A super-linearly convergent projected structured secant method, Int. J. Electrical Comput. Eng. 1 (2012), 1-8.
[48] N. Maratos, Exact penalty function algorithms for finite dimensional and control optimization problems, Imperial College London, 1978.
[49] J. Nocedal, S.J. Wright, Numerical Optimization, 2nd ed. Springer, New York, 2006.
[50] J. Nocedal, M.L. Overton, Projected Hessian updating algorithms for non- linearly constrained optimization, SIAM J. Numer. Anal. 22 (1985), 821–850.
[51] T. Pietrzykowski, An exact potential method for constrained maxima, SIAM J. Numer. Anal. 6 (1969), 299–304.
[52] M.J.D. Powell, A Method for Nonlinear Constraints in Minimization Problems, In: R. Fletcher, (ed.), Optimization, pp. 283-298, Academic Press, New York, 1969.
[53] M.J.D. Powell, A fast algorithm for nonlinearly constrained optimization calculations, In: Watson, G.A. (ed.) Numerical Analysis, pp. 144–157. Springer, Berlin, Heidelberg, 1978.
[54] M.J.D. Powell, A new algorithm for unconstrained optimization. In: J.B. Rosen, O.L. Mangasarian, K. Ritter, (eds.) Nonlinear Programming, pp. 31–65. Academic Press, New York, 1970.
[55] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, SIAM J. Numer. Anal. 20 (1983), 626–637.
[56] P. Toint, Towards an efficient sparsity exploiting Newton method for minimization, In: I. Duff, (ed.), Sparse Matrices and Their Uses, pp. 57–88, Academic press, London 1981.

[57] C. Vanaret, Comparison of nonlinear optimization solvers, 2023. https://github.com/cvanaret/NLP_comparison/tree/main Accessed 2023-09-19
[58] NEOS Server for Optimization. https://neos-server.org/neos/ Accessed 2023-09-19