HYBRID FILTER METHODS

FOR NONLINEAR OPTIMIZATION

by

Yueling Loh

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

May, 2016

Abstract

Globalization strategies used by algorithms to solve nonlinear constrained optimization problems must balance the oftentimes conflicting goals of reducing the objective function and satisfying the constraints. The use of merit functions and filters are two such popular strategies, both of which have their strengths and weaknesses. In particular, traditional filter methods require the use of a restoration phase that is designed to reduce infeasibility while ignoring the objective function. For this reason, there is often a significant decrease in performance when restoration is triggered.

In Chapter 3, we present a new filter method that addresses this main weakness of traditional filter methods. Specifically, we present a hybrid filter method that avoids a traditional restoration phase and instead employs a penalty mode that is built upon the ℓ_1 penalty function; the penalty mode is entered when an iterate decreases both the penalty function and the constraint violation. Moreover, the algorithm uses the same search direction computation procedure during every iteration and uses local feasibility estimates that

emerge during this procedure to define a new, improved, and adaptive margin (envelope) of the filter. Since we use the penalty function (a combination of the objective function and constraint violation) to define the search direction, our algorithm never ignores the objective function, a property that is not shared by traditional filter methods. Our algorithm thusly draws upon the strengths of both filter and penalty methods to form a novel hybrid approach that is robust and efficient. In particular, under common assumptions, we prove global convergence of our algorithm.

In Chapter 4, we present a nonmonotonic variant of the algorithm in Chapter 3. For this version of our method, we prove that it generates iterates that converge to a first-order solution from an arbitrary starting point, with a superlinear rate of convergence. We also present numerical results that validate the efficiency of our method.

Finally, in Chapter 5, we present a numerical study on the application of a recently developed bound-constrained quadratic optimization algorithm on the dual formulation of sparse large-scale strictly convex quadratic problems. Such problems are of particular interest since they arise as subproblems during every iteration of our new filter methods.

Primary Reader and Advisor: Daniel P. Robinson

Secondary Reader: Amitabh Basu

For my parents.

Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisor Daniel Robinson for his fundamental role in the completion of my thesis. All these years, he has provided me with invaluable guidance, unwavering financial assistance, and great opportunities through his extensive professional network. Many thanks for his endless support and patience; he has made my stay at Hopkins an enriching, productive, and enjoyable experience.

I would also like to give special thanks to Nick Gould for his guidance and encouragement in the course of our research together. This work would not have been possible if not for his intellectual prowess and expertise. I am grateful for the keen insight he has given and his boundless enthusiasm for his craft. It is an honor to collaborate with him.

I would also like to thank Amitabh Basu for agreeing to be a reader of my thesis. I greatly appreciate his time and effort, and the insightful comments to my thesis. I have enjoyed our conversations.

I am indebted to Sauleh Siddiqui for his help, advice, and encouragement.

ACKNOWLEDGMENTS

He has greatly expanded my exposure to the Operations Research field and the complexities of applying mathematical knowledge to practical problems. It is a pleasure to work with him; I always walk away with a new perspective to think about. I thank him for the opportunities he has given me.

And it goes without saying, but I would like to thank my family. Without their infinite love and support, I would not be where I am today.

Contents

List of tables

List of figures

Chapter 1

Introduction

1.1 Overview

Constrained optimization problems involve the minimization of a function over a set of variables that are subject to satisfying a set of constraints. Since most problems do not have a closed form solution, most algorithms for solving these problems generate a sequence of iterates that hopefully converge to a solution of the constrained optimization problem. These iterates are obtained from trial steps that are solutions of subproblems that are constructed using local information, such as first and second derivatives of the problem functions. Once a trial step has been calculated, these algorithms must determine if the step has made sufficient progress towards convergence to an optimal solution. Thus, to achieve convergence guarantees, these algorithms must include a

mechanism for determining when one point is "better" than another. Since the optimization problem has constraints, this mechanism must balance the conflicting aims of reducing the objective function and satisfying the constraints. Two of the most commonly used tools for this purpose are merit (penalty) functions and filters.

A merit function combines the objective function and a measure of constraint violation into a single function, whereby their individual perceived importance is determined by a weighting parameter. The quality of competing points is then measured by comparing their respective merit function values. A potential weakness is that the quality of iterates depends on the value of the weighting parameter, which can make step acceptance sensitive to its value.

In part, filter methods surfaced to mitigate this parameter dependence. A filter views the constrained optimization problem as a multi-criterion optimization problem consisting of minimizing the objective function and minimizing some measure of the constraint violation, with certain preference given to the latter. Roughly, a trial iterate is considered acceptable (better) if it has a smaller value of *either* the objective function or the constraint violation compared to the previously encountered points. Consequently, it is often the case that filter methods accept more iterates and perform better.

A weakness of filter methods is that they (traditionally) require the use of a restoration phase. A restoration phase is (typically) entered when the subprob-

lem used to compute trial steps is infeasible; some algorithms, e.g., [1], enter a restoration phase for additional reasons. In any case, once a restoration phase is triggered, a sequence of iterates focused on reducing the constraint-violation is computed until the desired subproblem becomes feasible. During this phase, the objective function is essentially *ignored*, which is highly undesirable from a practical and a computational perspective.

In this thesis, we present two related methods that address the main weakness of traditional filter methods. Drawing upon the strengths of both filter and penalty methods, our algorithms represent novel and robust algorithms for solving nonlinear and nonconvex constrained optimization problems.

1.2 Contributions

The following summarizes the main contributions and results of this thesis. We remark that the bulk of this work has been published in [2] and [3].

- **Globally convergent filter method without a restoration phase**

  In Chapter 3, we present the *first* known algorithm that uses a filter as a step acceptance mechanism to guarantee convergence, but which never needs to enter a traditional restoration phase. This is accomplished by using subproblems based on an exact penalty function that are always feasible and formed from models of the objective function and constraint

violation. The algorithm computes a search direction using the same procedure during *every* iteration and uses local feasibility estimates that emerge during this procedure to define a new, improved, and adaptive margin (envelope) of the filter. In certain instances, the algorithm may accept an iterate that decrease both the exact penalty function and the constraint violation, even if the iterate is not acceptable to the filter. In essence, we replace an undesirable traditional restoration phase with an attractive penalty phase, in a manner that uses the same subproblems during every iteration, i.e., we never change the goal of the subproblem or its formulation because of restoration considerations. We show that this algorithm is well-posed and, under common assumptions, prove convergence results that are on par with other state-of-the-art filter algorithms.

- **Superlinearly convergent filter method without a restoration phase**

  In Chapter 4, we present a *nonmonotone* variant of our new filter method that inherits the global convergence property. Moreover, we use the nonmonotonicity of the method to establish that no additional mechanism, e.g., a shadow/nonmonotone filter [4–6], is needed to establish local superlinear convergence of the iterates. The numerical results provided in Section 4.7 on problems from the CUTEst [7] test set validate the efficiency of our method.

- **Solving dual formulations of strictly convex quadratic problems**

  In Chapter 5 we present a numerical study on finding the solutions of strictly convex generally constrained quadratic subproblems by solving instead their dual formulation. This is motivated by our need for an efficient iterative method for solving the sparse large-scale strictly convex quadratic subproblems that arise during each iteration of our filter methods. Since the dual problem of a strictly convex constrained quadratic program is a convex (though usually not strictly convex) bound-constrained quadratic problem, we evaluate the application of a recently developed algorithm for nonconvex bound-constrained quadratic optimization problems by Mohy-ud-Din and Robinson [8] that is an extension of an algorithm by Dostál and Schöberl [9].

1.3 Outline of the thesis

This thesis is organized as follows. In Chapter 2, we present background information on the general theory of constrained nonlinear optimization, including the penalty function, traditional filters, and the Maratos effect. We then present our new restoration-free filter method and prove global convergence in Chapter 3. In Chapter 4, we present the nonmonotone variant of our filter algorithm, prove local convergence for this variant, and provide numerical results

from the CUTEst test set. Finally, in Chapter 5, we introduce the nonconvex bound-constrained quadratic optimization algorithm and provide results from a numerical study of its application to the dual formulation of the subproblems that arise in our filter methods.

Chapter 2

Background

2.1 Preliminaries

This thesis considers the general nonlinear optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) \;\; \text{subject to} \;\; c(x) \geq 0, \tag{2.1}$$

where both the objective function $f : \mathbb{R}^n \to \mathbb{R}$ and the constraint function $c : \mathbb{R}^n \to \mathbb{R}^m$ are assumed to be twice continuously differentiable. Problems of this type arise naturally in many areas including optimal control [10–14], resource allocation [15, 16], solution of equilibrium models [17, 18], and structural engineering [19, 20], among others. Here we assume that all constraints are inequalities, but we remark that our algorithms can easily handle equality

constraints directly, i.e., without resorting to converting an equality constraint to a pair of inequality constraints.

The primary goal of most optimization algorithms is to find a local solution for problem (2.1). Ideally, we would like to find a global minimizer, but doing so is very challenging, especially since the objective function and constraints can be nonconvex. Modern methods for global optimization often use heuristics and/or are stochastic. This thesis will focus on finding a local first-order solution for problem (2.1), namely a Karush-Kuhn-Tucker (KKT) point [21, (12.34)].

Definition 1 (KKT-point)**.** We say that x is a first-order KKT point for problem (2.1) with associated Lagrange multiplier vector y if and only if

$$F_{\text{KKT}}(x, y) := \begin{pmatrix} g(x) - J(x)^T y \\ \\ \min\left[c(x), y\right] \end{pmatrix} = \begin{pmatrix} 0 \\ \\ 0 \end{pmatrix}, \tag{2.2}$$

where $g(x) := \nabla f(x) \in \mathbb{R}^n$ is the gradient of the objective function, $J(x) := \nabla c(x) \in \mathbb{R}^{m \times n}$ is the Jacobian of the constraint function, and the minimum is taken component-wise. We also call (x, y) a KKT pair.

Popular methods for solving nonlinear constrained optimization problems can broadly be characterized as interior-point or active-set methods. Each class has its advantages and disadvantages. Interior-point algorithms [1, 22–25] offer polynomial-time complexity bounds in many cases and readily scale-up to problems involving millions of variables, since the predominate cost per iter-

ation is a single symmetric indefinite matrix factorization, for which state-of-the-art algorithms are readily available. Their main disadvantage is their inability to effectively use a good initial estimate of a solution. In fact, interior-point methods move any initial guess of a solution well into the strict interior of the feasible region. It is from this interior location that future iterates are forced to remain and justifies the name "interior-point" methods; more modern "infeasible" interior-point methods avoid this weakness to some degree.

Active-set methods [26–28] complement interior-point methods because they naturally utilize information derived from a good estimate of a solution. In fact, if the optimal active set (the set containing those constraints satisfied as equalities at a solution) was known in advance, then problem (2.1) could be solved as an equality constrained problem and its combinatorial nature would be eliminated. It is precisely this property that makes active-set methods widely used to solve optimal control [10–14], resource allocation [15, 16], equilibrium models [17, 18], and structural engineering [19, 20] problems, among others, despite the fact that there are no known active-set methods that offer polynomial bounds on the number of iterations. The main weakness of active-set algorithms is that each subproblem typically requires the solution of a linear or quadratic program (this involves solving multiple linear systems of equations), which is often expensive when compared to interior-point methods, which require a single linear system solve per iteration.

The broad class of active-set methods includes augmented Lagrangian [29–33], primal-dual penalty [34–36], and sequential quadratic optimization (SQO) methods [37–48] (commonly called sequential quadratic programming (SQP) methods). In addition to its ability to be warm-started, augmented Lagrangian methods may be implemented matrix-free, and thus may be applied to extreme-scale problems. Unfortunately, they can be ineffective/inefficient at identifying those inequality constraints satisfied as equalities at a local solution, i.e., an optimal active-set. SQO methods are celebrated for their optimal active-set identification and warm-start abilities, but are practical only on medium- to large-scale problems, as mentioned earlier.

Each class of methods contains a variety of algorithms that may be distinguished by their details. For example, one distinction is whether a line search or trust region is used, while another is whether globalization is attained via the use of a merit function, a filter, or a step classification scheme [38, 49]. It is no surprise that each variant has advantages and disadvantages, and serves a distinct and vital role in solving real-life optimization problems.

This thesis addresses some of the weaknesses of previous filter methods, which will be presented in the context of a new filter line search SQO algorithm. We note, however, that the ideas and philosophies presented in this thesis may be used (to various degrees) by future filter-based algorithms.

2.2 Notation

We use $\mathbb{R}^+$ to denote the set of nonnegative real numbers. Given vectors a and b with the same dimension, the vector with ith component $a_i b_i$ is denoted by $a \cdot b$. Similarly, $\min(a, b)$ is a vector with components $\min(a_i, b_i)$, and $[a]^-$ is a vector with components $\max(-a, 0)$ with the maximum taken component-wise.

The ith component of a vector labeled with a subscript will be denoted by $[\ \cdot\ ]_i$, e.g., $[\, v\,]_i$ is the ith component of the vector v. The subvector of components with indices in the index set $\mathcal{S}$ is denoted by $[\ \cdot\ ]_{\mathcal{S}}$, e.g., $[\, v\,]_{\mathcal{S}}$ is the vector with components v_i for $i \in \mathcal{S}$.

The vector $g(x)$ is used to denote $\nabla f(x)$, the gradient of $f(x)$. The matrix $J(x)$ denotes the $m \times n$ constraint Jacobian, which has ith row $\nabla c_i(x)^T$, the gradient of the ith constraint function $c_i(x)$. The Lagrangian function associated with problem (2.1) is $L(x, y) := f(x) - c(x)^T y$, where y is an m-vector of Lagrange multipliers (or dual variables) associated with the inequality constraints. The Hessian of the Lagrangian with respect to x is denoted by $H(x, y) := \nabla^2_{xx} f(x) - \sum_{i=1}^m y_i \nabla^2_{xx} c_i(x)$.

The vector pair (x_k, y_k) denotes the kth primal-dual estimate of a solution to (2.1). We use $f_k := f(x_k)$, $g_k := g(x_k)$, $c_k := c(x_k)$, and $J_k := J(x_k)$.

Finally, for any $\epsilon > 0$ and $v \in \mathbb{R}^n$, we let $\mathcal{B}_\epsilon(v) := \{x \in \mathbb{R}^n : \|x - v\|_2 < \epsilon\}$ denote the open ball of radius ϵ centered at v.

2.3 The ℓ_1-penalty function

Our trial step computation is based on the ℓ_1-penalty function

$$\phi(x;\sigma) := f(x) + \sigma v(x), \tag{2.3}$$

where $\sigma > 0$ is a weighting parameter and the constraint violation at x is

$$v(x) := \left\| [c(x)]^- \right\|_1 \tag{2.4}$$

with $[z]^- := \max(-z, 0)$ (the maximum is taken component-wise). This function has been frequently used in optimization since it can be shown [50], under common assumptions, that solutions to problem (2.1) correspond to *unconstrained* minimizers of $\phi(x;\sigma)$ for all sufficiently large $\sigma > 0$. For this reason, the ℓ_1-penalty function is said to be exact.

Our method makes extensive use of models of the objective function and constraint function. In particular, at the point x and along a step s, we use linear and quadratic model approximations of the objective function f given by

$$\ell^f(s;x) := f(x) + g(x)^T s \tag{2.5}$$

and

$$\begin{aligned} q^f(s;x,M) &:= \ell^f(s;x) + \tfrac{1}{2}s^T M s \\ &= f(x) + g(x)^T s + \tfrac{1}{2}s^T M s \end{aligned} \tag{2.6}$$

for a given symmetric matrix $M \in \mathbb{R}^{n\times n}$, and a piecewise-linear approximation to the constraint violation function v given by

$$\ell^v(s;x) := \left\| [c(x) + J(x)s]^- \right\|_1$$

to form the following linear and quadratic models of ϕ:

$$\begin{aligned} \ell^\phi(s;x,\sigma) &:= \ell^f(s;x) + \sigma\ell^v(s;x) \\ &= f(x) + g(x)^T s + \sigma \left\| [c(x) + J(x)s]^- \right\|_1 \end{aligned} \tag{2.7}$$

$$\begin{aligned} q^\phi(s;x,M,\sigma) &:= q^f(s;x,M) + \sigma\ell^v(s;x) \\ &= f(x) + g(x)^T s + \tfrac{1}{2}s^T M s + \sigma \left\| [c(x) + J(x)s]^- \right\|_1 . \end{aligned} \tag{2.8}$$

Using these models, we may predict the *change* in v with the function

$$\begin{aligned} \Delta\ell^v(s;x) &:= \ell^v(0;x) - \ell^v(s;x) \\ &= \left\| [c(x)]^- \right\|_1 - \left\| [c(x) + J(x)s]^- \right\|_1 , \end{aligned} \tag{2.9}$$

the change in f with the functions

$$\begin{aligned}\Delta\ell^f(s;x) &:= \ell^f(0;x) - \ell^f(s;x) \\ &= -g(x)^T s \end{aligned} \tag{2.10}$$

and

$$\begin{aligned}\Delta q^f(s;x,M) &:= q^f(0;x,M) - q^f(s;x,M) \\ &= \Delta\ell^f(s;x) - \tfrac{1}{2}s^T M s \\ &= -g(x)^T s - \tfrac{1}{2}s^T M s, \end{aligned} \tag{2.11}$$

and the change in the penalty function ϕ with the functions

$$\begin{aligned}\Delta\ell^\phi(s;x,\sigma) &:= \ell^\phi(0;x,\sigma) - \ell^\phi(s;x,\sigma) \\ &= \Delta\ell^f(s;x) + \sigma\Delta\ell^v(s;x) \\ &= -g(x)^T s + \sigma\left(\left\|[c(x)]^-\right\|_1 - \left\|[c(x)+J(x)s]^-\right\|_1\right) \end{aligned} \tag{2.12}$$

and

$$\begin{aligned}\Delta q^{\phi}(s;x,M,\sigma) &:= q^{\phi}(0;x,M,\sigma) - q^{\phi}(s;x,M,\sigma) \\ &= \Delta \ell^{\phi}(s;x,\sigma) - \tfrac{1}{2}s^{T}Ms \\ &= -g(x)^{T}s - \tfrac{1}{2}s^{T}Ms + \sigma\left(\left\|[c(x)]^{-}\right\|_{1} - \left\|[c(x)+J(x)s]^{-}\right\|_{1}\right).\end{aligned} \tag{2.13}$$

2.4 Traditional filter SQO methods

Filter methods were introduced to the nonlinear optimization community by Fletcher, Leyffer, and Toint [41, 51] and have since been very popular [24, 42, 52–57]. This thesis makes further advances in the use and understanding of how a filter may be used as the globalization strategy in the framework of SQO methods.

Methods for nonlinear optimization, such as SQO methods, compute a sequence of iterates $\{x_k\}$. Given the kth estimate of a solution x_k, traditional SQO methods compute a trial direction from the following subproblem:

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; f_k + g_k^{T}s + \tfrac{1}{2}s^{T}H_k s \quad \text{subject to} \;\; c_k + J_k s \geq 0, \tag{2.14}$$

where H_k is a symmetric matrix such that $H_k \approx \nabla^2_{xx}L(x_k, y_k)$ with y_k being the kth estimate of a Lagrange multiplier vector. The exact form of the subprob-

lem varies with the algorithm, especially with the choice of H_k whose choice is greatly influenced by whether a line search or a trust-region framework is adopted. Line search methods usually require H_k to be positive definite, which is usually achieved by explicitly modifying the matrix as necessary. Once the solution, say s_k is obtained, line search methods seek an improved new point by searching along the direction s_k, where the quality of prospective new points is typically measured with either a merit function or a filter. Trust-region methods, on the other hand, do not require the matrix H_k to be positive definite, but compensate for this fact by including a trust-region constraint of the form $\|s\|_2 \leq \delta_k$ for some k-th trust-region radius $\delta_k > 0$. If the solution s_k does not yield an improved point $x_k + s_k$ when compared to x_k, then the trust-region radius δ_k is reduced, and a new subproblem is solved.

The traditional SQO search direction computed from (2.14) can be viewed as an application of Newton's method for finding a first-order KKT pair (see Definition 1), i.e., applying Newton's method for finding a zero of F_{KKT}. Since it is well-known that Newton's method does not necessarily converge from an arbitrary starting point, it is no surprise that modern methods combine the search direction with a globalization mechanism (merit function or filter) to ensure convergence from remote starting points. In the next section, we describe the details of a filter approach since this strategy is the focus of this thesis.

2.4.1 The filter

A filter provides a mechanism for combining the often competing aims associated with (2.1), namely obtaining feasibility and reducing the objective function. Before giving a precise definition of the filter, we need a definition.

Definition 2 (domination)**.** We say that the point x_i *dominates* x_j if their associated ordered pairs (v_i, f_i) and (v_j, f_j) satisfy

$$v_i \leq v_j \text{ and } f_i \leq f_j$$

with $f_i = f(x_i)$ and $v_i = v(x_i)$. In other words, the point x_i has constraint violation and objective function values at least as small as the values associated with the point x_j.

We may now give a general definition of the filter.

Definition 3 (a general filter)**.** A filter $\mathcal{F}$ is a set of ordered pairs $\{(v_i, f_i)\}$ associated with vectors $\{x_i\}$ such that no pair dominates another.

Since the ordered pair $(v(x), f(x))$ is clearly functions of the primal variable x, it is convenient to introduce the following definition.

Definition 4 (being in the filter)**.** We say that "x_i is in the filter $\mathcal{F}$" if its associated ordered pair satisfies $(v_i, f_i) \in \mathcal{F}$. To avoid confusion, we will always

refer to x_i as being *in* the filter and refer to its associated ordered pair (v_i, f_i) as being an *element* of the filter.

We note that the above definitions imply that

$$x_i \text{ in } \mathcal{F} \qquad \Longrightarrow \qquad v_i < v_j \text{ or } f_i < f_j \text{ for all } x_j \text{ in } \mathcal{F}.$$

A *necessary* condition for accepting a new trial iterate is that it should not be dominated by any point in the filter.

Most filter methods use a "slanted" filter envelop, which is encapsulated in the following definition of being acceptable to the filter. We remark that the parenthetical "traditional method" is used in the definition since we use a new and improved definition in our newly proposed methods in Chapters 3 and 4.

Definition 5 (acceptable to $\mathcal{F}_k$ (traditional method))**.** Let $\beta \in (0, 1)$. The point x is acceptable to $\mathcal{F}_k$ (here $\mathcal{F}_k$ represents the filter during iteration k) if its associated ordered pair $\big(v(x), f(x)\big)$ satisfies

$$v(x) \leq (1 - \beta)v_i \tag{2.15a}$$

or

$$f(x) \leq f_i - \beta v_i \tag{2.15b}$$

for all $0 \leq i < k$ such that $(v_i, f_i) \in \mathcal{F}_k$.

From Definition 5, we see that for traditional methods, a point x is acceptable to the k-th filter $\mathcal{F}_k$ if it has a "substantially" smaller constraint violation or lower objective value compared to all previous iterates. The amount that they must be better is controlled by the parameter $\beta \in (0, 1)$, which is typically chosen close to zero, e.g., $\beta = 0.05$. The envelope is the area in (v, f) space determined by these conditions, as depicted in Figure 2.1.

It should be mentioned that all known provably convergent filter methods are based on filter acceptance criteria that have a weak dependence between the constraint violation and the objective value (cf. (2.15)). In fact, this observation partly motivated the work on *flexible* penalty methods by Curtis and Nocedal [58]. They describe how a *single* element filter is essentially equivalent to the union of points acceptable to the ℓ_1-penalty function over a range of values for the weighting parameter σ.

A filter with four elements is illustrated by Figure 2.1. We note that Definition 5 does not require nor imply that the current vector x_k has to be in the filter when determining whether to accept a prospective trial point. However, when discussing the inclusion of a new iterate into the filter, it does not make sense to accept a new point unless it is acceptable to the current filter *and* is better than the current point x_k. This leads to the following definition of an augmented filter.

Definition 6 (augmented filter)**.** The vector y is acceptable to the filter $\mathcal{F}$ aug-

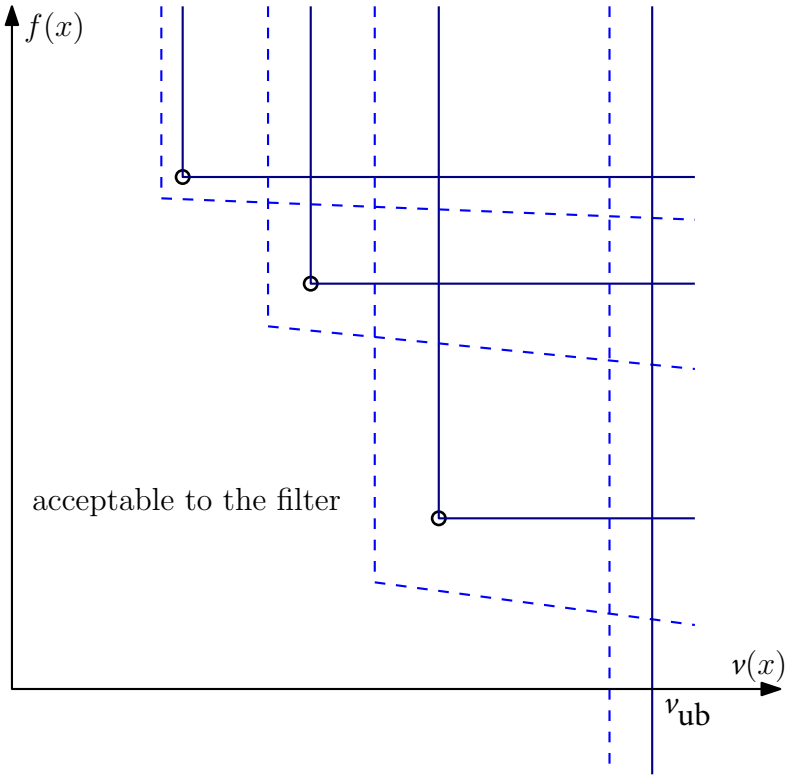

Figure 2.1: Illustration of a filter with four elements, one of which is the fictitious element x_{F} that has the associated ordered pair $(v_{\text{ub}}, -\infty)$. The dashed lines represent the inequalities defined by (2.15). The points that are acceptable to the filter are those vectors x with ordered pairs $(v(x), f(x))$ that lie below and to the left of the dashed lines.

mented by x if y is acceptable to the *augmented filter* $\mathcal{F} \cup \big(v(x), f(x)\big)$.

Finally, we require two definitions that deal with adding and removing (trimming) ordered pairs from the filter. The need to remove (trim) filter entries is primarily motivated by efficiency and computer storage concerns, while the addition of entries is a vital feature needed to establish convergence of filter methods. Roughly, if infinitely many iterates are added to the filter, then the envelope of the filter will push the iterates toward the y-axis, i.e., towards fea-

sibility. This is a key feature in establishing that limit points of the sequence of iterates added to the filter are optimal for problem (2.1).

Definition 7 (adding to the filter)**.** We say that x is added to the filter $\mathcal{F}$ if the ordered pair $\big(v(x), f(x)\big)$ is added to $\mathcal{F}$.

Definition 8 (trim the filter)**.** To trim the filter $\mathcal{F}$ means to remove all elements of $\mathcal{F}$ that are dominated by any element of $\mathcal{F}$.

2.4.2 Switching conditions

Acceptability to the filter alone is insufficient to guarantee convergence to an optimal solution. This can be seen by observing that any sequence of iterates $\{x_k\}$ that converges to any feasible point while increasing the objective function during every iteration, will be acceptable to the filter, but can easily not converge to an optimal solution. To combat this possibility, modern filter methods use a "switching condition" whereby if an iterate is *predicted* to sufficiently decrease the objective function according to some local model, then the iterate is accepted only when it also satisfies an Armijo-like sufficient decrease condition on the *actual* objective function f.

While the specific form of the switching condition varies among algorithms, the following is one such example [55]:

$$\Delta q^f(s_k; x_k, H_k) \geq \gamma_v\big(v(x_k)\big)^\psi \tag{2.16}$$

where $\gamma_v \in (0,1)$ and $\psi > 1/(1+\mu)$, with $\mu \in (0,1)$. If the switching condition (2.16) holds, the trial iterate $x_k + s_k$ is accepted when it is acceptable to the filter and

$$f(x_k + s_k) \leq f(x_k) - \gamma_f \Delta q^f(s_k; x_k, H_k) \tag{2.17}$$

where γ_v and ψ are the same constants used in (2.16) and $\gamma_f \in (0,1)$. If (2.16) does not hold, then (2.17) does not play a role in the step acceptance criterion.

2.4.3 Restoration

One of the main contributions of this thesis is to overcome the undesirable fact that previous filter methods require a special restoration phase to handle various scenarios that would otherwise lead to failure of the method. These scenarios occur in different forms in both line search and trust-region methods. For an example in the context of line search methods, it is possible that the only points along the direction s_k that are acceptable to the filter (thus, has made sufficient progress) are unacceptably small in norm. (Here, by unacceptably small in norm, we mean that we can not prove convergence of the iterates.) Alternatively, in trust-region methods, it is possible that the search for the next point requires the algorithm to reduce the trust-region radius to a point at which the SQO subproblem becomes infeasible.

When these situations arise, the restoration phase is triggered, and conse-

quently a search for a point that is acceptable to the filter is initiated. During this restoration phase, iterates are obtained from subproblems aimed at approximately solving the nonsmooth problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; v(x) \tag{2.18}$$

starting at the current iterate x_k. Essentially, the restoration phase temporarily ignores the objective function and iterates toward the feasible region until the issue that triggered the restoration phase is resolved. This may fail if the critical point of (2.18) is infeasible, which would result in the algorithm exiting with a first-order solution to problem (2.18) that is infeasible; such a point, called an infeasible station point in the literature, is always a possible outcome when solving optimization problems with nonlinear and nonconvex constraints.

Solving the nonsmooth restoration problem (2.18) can be as difficult as solving the original optimization problem. Different filter methods [1, 41, 42, 56, 57] tackle this problem in various ways and often times utilize sophisticated heuristics to enhance performance. However, since the objective function is ignored during these restoration iterations, a significant decrease in performance is often observed on problems for which restoration plays a notable role. Thus, the restoration phase can be a major weakness for filter SQO methods, which was the impetus for the restoration-free filter methods presented in this thesis.

Our work is not the only one designed to resolve this weakness of traditional filter methods. Chen and Goldfarb [59] presented an *interior-point* method that uses two penalty functions to determine step acceptance: a piecewise linear penalty function whose break points are essentially elements in the filter, and the ℓ_2-penalty function. Under this scheme, a trial step is accepted if it provides sufficient reduction for either penalty function.

2.5 The Maratos effect and its influence on local convergence

The Maratos effect (e.g., see [60], [4], [6], [5]) describes the situation where, near a local solution, good quality steps (such as full Newton steps for finding a zero of the gradient of the Lagrangian function) actually increase both the objective function and the constraint violation. (An example can be seen in Figure 2.2.) If accepted, the resulting iterate would lead to superlinear convergence to a solution, but because both the objective and the constraint violation increase, the step is rejected by algorithms that are globalized by filters and certain merit functions (e.g., the ℓ_1-penalty function). As a result, these algorithms can suffer from poor local convergence.

In Chapter 3, we introduce a restoration-free filter SQO method that has global convergence guarantees. Although this method has certain practical

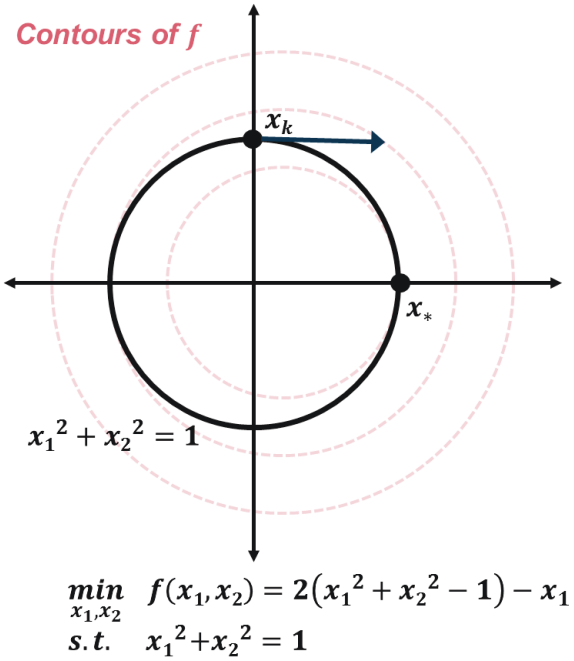

Figure 2.2: An example of the Maratos effect [21, Example 15.4], [61]. At $x_k = (0, 1)$, $f(x_k) = 0$ and $v(x_k) = 0$. If we take the pure SQO direction, then $x_{k+1} = (1, 1)$, and the objective and constraint violation increase to $f(x_{k+1}) = 1$ and $v(x_{k+1}) = 1$, respectively. Note that the optimal solution occurs at $x_* = (1, 0)$.

benefits compared to traditional SQO methods, it also may suffer from the Maratos effect so that "good" steps are rejected near a local solution. Modifications must therefore be made to obtain an algorithm with superlinear convergence. Such a modified algorithm is the topic of Chapter 4.

Popular mechanisms to address the Maratos effect include nonmonotone (sometimes called watchdog) approaches, the use of second-order correction steps, and the use of certain merit functions that do not suffer from the Maratos effect, e.g., the augmented Lagrangian penalty function. Thus, algorithms that are based on the augmented Lagrangian function are often able to avoid the

Maratos effect. Along the same lines, Ulbrich [62] uses the the value of the Lagrangian function in place of the objective function when defining the filter structure, and is able to establish local superlinear convergence.

In a nonmonotone approach for penalty SQO methods, "sufficient progress" is not required during every iteration; instead, the algorithm is allowed to use the SQO trial steps for several iterations in a "nonmonotone" phase regardless of whether the merit function decreases. If sufficient progress has not been made after a predetermined fixed number of steps, the algorithm resets the iterate to that which was obtained just prior to the nonmonotone phase, and then proceeds with either a line search along the search direction or a reduction in the trust-region radius. For the filter SQO method by Shen, Leyffer, and Fletcher [5], a nonmonotone filter is used to establish local superlinear convergence, while a standard filter is used to prove global convergence. In particular, they show that a point acceptable to the local filter is obtained after a constant number of nonmonotone steps.

Finally, the filter SQO method by Biegler and Wächter [56] uses a second-order correction step when a trial step is rejected by the filter mechanism. Various second-order correction steps are possible, but they are all designed with the same purpose, namely to avoid the Maratos effect by ensuring that points acceptable to the algorithm will be computed in the neighborhood of a minimizer. Importantly, we note that all of these previous filter methods

still require a traditional restoration phase, and therefore can suffer from its associated weaknesses.

In Chapter 4, we present a nonmonotone filter SQO algorithm that does not require additional mechanisms like a shadow/nonmonotone filter [4, 6]. In fact, it only uses a single filter that has been slightly modified to be based on weaker (better) conditions. Interestingly, the second of two consecutive steps in a nonmonotone SQO approach is an example of a second-order correction step. Consequently, the sum of two consecutive trial steps in our nonmonotone approach represents a valid second-order correction step. We are able to leverage this fact to establish that the sequence of iterates generated by our nonmonotone approach exhibits local superlinear convergence while still avoiding the use of a traditional restoration phase.

Chapter 3

A filter SQO method

3.1 Description of the method

In this section we describe our new filter SQO method, called FiSQO. The algorithm is iterative and relies on computing trial steps from carefully constructed subproblems defined by local models of the nonlinear problem functions. The subproblems are always feasible since they are based on an exact penalty function. To ensure that these models result in productive steps, we use steering techniques [63] to adaptively adjust the weighting (penalty) parameter. In contrast to original steering methods, we use a step convexification procedure similar to [64, 65] to avoid solving multiple quadratic programs during each iteration. These subproblems and the resulting trial steps are explained in Sections 3.1.1–3.1.6. In Section 3.1.7 we introduce our new filter

construct and related terminology; we emphasize that acceptability to the filter is only a necessary condition for accepting a trial iterate. A full statement and description of the algorithm is given in Section 3.1.8.

Before proceeding, we remind the reader that (x_k, y_k) is always used to denote the kth estimate of a solution to problem (2.1). Also, we extensively use the host of liner and quadratic movies of f, v, and the ℓ_1 penalty function as provided in Section 2.3.

3.1.1 The steering step s_k^s

In order to strike a proper balance between reducing the objective function and the constraint violation, we compute a *steering* step s_k^s as a solution to the linear program

$$\underset{(s,r)\in\mathbb{R}^{n+m}}{\text{minimize}} \; e^T r \quad \text{subject to} \quad c_k + J_k s + r \geq 0, \quad r \geq 0, \quad \|s\|_\infty \leq \delta_k, \tag{3.1}$$

where $c_k = c(x_k)$, $J_k = J(x_k)$, $\delta_k \in [\delta_{\min}, \delta_{\max}]$, and $0 < \delta_{\min} \leq \delta_{\max} < \infty$. Problem (3.1) is equivalent to the nonsmooth problem

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \; \ell^v(s; x_k) \quad \text{subject to} \quad \|s\|_\infty \leq \delta_k \tag{3.2}$$

since s solves (3.2) if and only if (s, r) solves (3.1), where $r = \max(-(c_k + J_k s), 0)$. Since $\ell^v(0; x_k) = v(x_k)$, ℓ^v is a convex function, and $s = 0$ is feasible for (3.2), it follows from (2.9) that $\Delta\ell^v(s_k^s; x_k) \geq 0$. The quantity $\Delta\ell^v(s_k^s; x_k)$ is the best local improvement in linearized constraint feasibility for steps of size δ_k.

All methods for nonconvex optimization may converge to an infeasible point that is a local minimizer of the constraint violation as measured by v. Points of this type are known as infeasible stationary points, which we now define by utilizing the steering subproblem.

Definition 9 (Infeasible stationary point)**.** The vector x^{I} is an infeasible stationary point if and only if

$$v(x^{\mathrm{I}}) > 0 \;\text{ and }\; \Delta\ell^v(s^{\mathrm{I}}; x^{\mathrm{I}}) = 0, \tag{3.3}$$

where $s^{\mathrm{I}} = \operatorname{argmin}_{s \in \mathbb{R}^n} \ell^v(s; x^{\mathrm{I}}) \;\text{ subject to }\; \|s\|_\infty \leq \delta$ for some $\delta > 0$.

3.1.2 The predictor step s_k^p

The *predictor* step is computed as the *unique* solution to one of the following strictly convex minimization problems:

$$s_k^p = \begin{cases} \underset{s\in\mathbb{R}^n}{\operatorname{argmin}} \;\; f_k + g_k^T s + \frac{1}{2} s^T B_k s \;\; \text{subject to} \;\; c_k + J_k s \geq 0, \\ \text{if } \Delta\ell^v(s_k^s; x_k) = v(x_k), & \text{(3.4a)} \\ \\ \underset{s\in\mathbb{R}^n}{\operatorname{argmin}} \;\; q^\phi(s; x_k, B_k, \sigma_k), \\ \text{otherwise,} & \text{(3.4b)} \end{cases}$$

where $\sigma_k > 0$ is the kth value of the penalty parameter, $f_k = f(x_k)$, $g_k = \nabla f(x_k)$, $c_k = c(x_k)$, $J_k = \nabla c(x_k)$, B_k is a positive-definite matrix that we are free to choose such that $B_k \approx \nabla^2_{xx} L(x_k, y_k)$, and the Lagrangian L is defined by $L(x, y) = f(x) - c(x)^T y$. Analogous to the steering subproblem, the nonsmooth minimization problem (3.4b) is equivalent to the smooth problem

$$\underset{s\in\mathbb{R}^n, r\in\mathbb{R}^n}{\text{minimize}} \;\; f_k + g_k^T s + \tfrac{1}{2} s^T B_k s + \sigma_k e^T r \;\; \text{subject to} \;\; c_k + J_k s + r \geq 0, \;\; r \geq 0, \quad \text{(3.5)}$$

which is the problem solved in practice. We use y_k^p to denote the Lagrange multiplier vector for the constraint $c_k + J_k s \geq 0$ in (3.4a) and $c_k + J_k s + r \geq 0$ in (3.5) (equivalently (3.4b)). Possible choices for the positive-define matrix in-

clude (i) the trivial choice $B_k = I$, (ii) a scaled diagonal matrix based on the Barzilai-Borwein [66] method, (iii) quasi-Newton updates such as BFGS [21] and L-BFGS [67], and (iv) modified Cholesky factorizations [68, 69] are possible. Moreover, relaxing the positive-definition assumption on B_k may also be possible provided conditions such as those used in [65] are enforced to ensure that sufficient descent direction are computed. Since this relaxation would introduce unnecessary complications into our algorithm and require the use of an indefinite quadratic problem (QP) solver, we will assume throughout that B_k is a positive-definite matrix.

The next result shows how convergence to KKT points may be deduced from the predictor problem.

Lemma 1. *Suppose that x_* satisfies*

$$\begin{aligned} &0 = v(x_*) \quad \textit{and} \\ &0 = \underset{s\in\mathbb{R}^n}{\operatorname{argmin}}\, f(x_*) + g(x_*)^T s + \tfrac{1}{2} s^T B s \;\; \text{subject to} \;\; c(x_*) + J(x_*)s \geq 0 \end{aligned} \tag{3.6}$$

for some positive definite matrix B, and let y_* denote the associated Lagrange multiplier vector. Then, it follows that (x_*, y_*) is a KKT point for problem (2.1) *as defined by* (2.2)*.*

Proof. Since B is positive definite, $s = 0$ is the unique solution to the optimization problem in (3.6). It then follows from the first-order necessary optimality

conditions at $s = 0$ that

$$g(x_*) = J(x_*)^T y_*, \quad \text{and} \quad \min\big(c(x_*), y_*\big) = 0,$$

where y_* is the Lagrange multiplier for the constraint $c(x_*) + J(x_*)s \geq 0$. It now follows from Definition 2.2 that (x_*, y_*) is a KKT point for problem (2.1). □

3.1.3 The search direction s_k

The steering direction s_k^s provides a measure of local progress in infeasibility. Since we desire a search direction s_k that makes progress towards feasibility, we define

$$s_k := (1 - \tau_k)s_k^s + \tau_k s_k^p \tag{3.7}$$

where τ_k is the largest number on $[0, 1]$ such that

$$\Delta\ell^v(s_k; x_k) \geq \eta_v \Delta\ell^v(s_k^s; x_k) \geq 0 \ \text{ for some } \eta_v \in (0, 1). \tag{3.8}$$

The next lemma shows that $\tau_k > 0$ when x_k is not an infeasible stationary point. This is important since the step s_k then has a significant contribution from s_k^p, which was computed from a subproblem that modeled *both* the objective and constraint functions; this contrasts traditional filter methods when restoration is entered since the subproblem formulations then focus solely on

the constraint violation.

Lemma 2. *If x_k is not an infeasible stationary point as given by Definition 9, then $\tau_k > 0$.*

Proof. If $v(x_k) = 0$, then $\Delta\ell^v(s_k^s; x_k) = 0$. It then follows from (3.4a) that $c_k + J_k s_k^p \geq 0$, which in turn implies that $\Delta\ell^v(s_k^p; x_k) = 0$. Thus, the choice $\tau_k = 1$ satisfies (3.7) and (3.8).

Now suppose that $v(x_k) > 0$ and define

$$s(\tau) = (1 - \tau)s_k^s + \tau s_k^p$$

so that $\lim_{\tau \downarrow 0} s(\tau) = s_k^s$. It then follows from continuity of $\Delta\ell^v(\cdot\,; x_k)$ and the fact that $\Delta\ell^v(s_k^s; x_k) > 0$ (this holds since x_k is not an infeasible stationary point by assumption), that

$$\lim_{\tau \downarrow 0} \Delta\ell^v\big(s(\tau); x_k\big) = \Delta\ell^v(s_k^s; x_k) > 0.$$

Therefore, there exists $\tau' > 0$ such that

$$\big|\Delta\ell^v\big(s(\tau); x_k\big) - \Delta\ell^v(s_k^s; x_k)\big| < (1 - \eta_v)\Delta\ell^v(s_k^s; x_k) \quad \text{for all } \tau \in [0, \tau']$$

since $\eta_v \in (0, 1)$ in (3.8) and $\Delta\ell^v(s_k^s; x_k) > 0$. However, this implies that

$$\Delta\ell^v\big(s(\tau); x_k\big) \geq \eta_v \Delta\ell^v(s_k^s; x_k) \quad \text{for all } \tau \in [0, \tau'],$$

which guarantees that $t_k \geq \tau' > 0$. □

We now proceed to show that if $\Delta\ell^v(s_k^s; x_k) > 0$, then s_k is a descent direction for $v(\cdot)$. We require the definition of the directional derivative of a function.

Definition 10 (directional derivative)**.** The directional derivative of a function $h(\cdot)$ in the direction d and at the point x is defined (when it exists) as

$$[D_d h](x) := \lim_{t \downarrow 0} \frac{h(x + td) - h(x)}{t}.$$

We now show that the directional derivative is bounded by the negative of the change in its model.

Lemma 3. *At any point x and for any direction d, it follows that*

$$[D_d v](x) \leq -\Delta\ell^v(d; x),$$

where the function $D_d v$ is the directional derivative of v in the direction d.

Proof. Since ℓ^v is a convex function and $\ell^v(0; x)$ is finite, it follows from [70, Theorem 23.1] that

$$\frac{\ell^v(td; x) - \ell^v(0; x)}{t}$$

is monotonically non-decreasing with t, $[D_d \ell^v](0; x)$ exists, and

$$[D_d \ell^v](0; x) = \inf_{t > 0} \frac{\ell^v(td; x) - \ell^v(0; x)}{t}. \tag{3.9}$$

It then follows from [71, Lemma 3.1], (3.9) and the definition of $\Delta\ell^v$ that

$$[D_d v](x) = [D_d \ell^v](0; x) \leq \ell^v(d; x) - \ell^v(0; x) = -\Delta\ell^v(d; x),$$

which is the desired result. □

It follows from Lemma 3 that the search direction s_k is a descent direction for v when our infeasibility measure is positive.

Lemma 4. *If $\Delta\ell^v(s_k^s; x_k) > 0$, the direction s_k is a descent direction for v at the point x_k, i.e.,*

$$[D_{s_k} v](x_k) \leq -\Delta\ell^v(s_k; x_k) \leq -\eta_v \Delta\ell^v(s_k^s; x_k) < 0, \quad \textit{where } \eta_v \textit{ is defined in } (3.8).$$

Proof. It follows directly from Lemma 3, (3.8), and $\Delta\ell^v(s_k^s; x_k) > 0$ that

$$[D_{s_k} v](x_k) \leq -\Delta\ell^v(s_k; x_k) \leq -\eta_v \Delta\ell^v(s_k^s; x_k) < 0,$$

which implies that s_k is a descent direction for v at the point x_k. □

We now consider the case when our infeasibility measure is zero.

Lemma 5. *Suppose $\Delta\ell^v(s_k^s; x_k) = 0$, then one of the following must occur:*

(i) $v(x_k) > 0$ *and* x_k *is an infeasible stationary point; or*

(ii) $v(x_k) = 0$ *and* $\Delta\ell^\phi(s_k; x_k, \sigma) \geq \frac{1}{2} s_k^{pT} B_k s_k^p$ *for all* $0 < \sigma < \infty$.

Proof. If $v(x_k) > 0$, then by Definition 9, x_k is an infeasible stationary point which is part (i). Now, suppose that $v(x_k) = 0$. As in the proof of Lemma 2, it follows that

$$\Delta\ell^v(s_k^p; x_k) = 0, \quad \tau_k = 1, \quad \text{and} \quad s_k = s_k^p. \tag{3.10}$$

We may then use the definition of s_k^p in (3.4a), (3.10), and (2.11) to conclude that

$$0 \leq \Delta q^\phi(s_k^p; x_k, B_k, \sigma_k) = \Delta q^f(s_k^p; x_k, B_k) = \Delta\ell^f(s_k^p; x_k) - \tfrac{1}{2} s_k^{pT} B_k s_k^p,$$

which yields $\Delta\ell^f(s_k^p; x_k) \geq \frac{1}{2} s_k^{pT} B_k s_k^p$. Combining this with (2.12) and (3.10), we have that

$$\begin{aligned}\Delta\ell^\phi(s_k; x_k, \sigma) &= \Delta\ell^f(s_k; x_k) + \sigma\Delta\ell^v(s_k; x_k) \\ &= \Delta\ell^f(s_k; x_k) = \Delta\ell^f(s_k^p; x_k) \geq \tfrac{1}{2} s_k^{pT} B_k s_k^p \text{ for all finite } \sigma,\end{aligned}$$

which completes the proof. □

3.1.4 Updating the weighting parameter

By design, the trial step s_k is a descent direction for v when local improvement in feasibility is possible. Since the weighting parameter provides a balance between reducing the objective function and the constraint violation, it

makes sense to adjust the weighting parameter so that s_k is also a descent direction for ϕ. If we denote $\Delta\ell_k^\phi = \Delta\ell^\phi(s_k; x_k, \sigma_k)$ and $\Delta\ell_k^v = \Delta\ell^v(s_k^s; x_k)$, then we can accomplish this by defining

$$\sigma_{k+1} = \begin{cases} \sigma_k & \text{if } \Delta\ell_k^\phi \geq \sigma_k \eta_\sigma \Delta\ell_k^v, \\ \max\left\{\sigma_k + \sigma_{\text{inc}},\ \frac{-\Delta\ell^f(s_k;x_k)}{\Delta\ell^v(s_k;x_k) - \eta_\sigma \Delta\ell^v(s_k^s;x_k)}\right\} & \text{otherwise,} \end{cases} \tag{3.11}$$

for some $\sigma_{\text{inc}} > 0$ and η_σ satisfying $0 < \eta_\sigma < \eta_v < 1$, where η_v is defined in (3.8).

Lemma 6. *If x_k is not an infeasible stationary point, then the parameter update* (3.11) *is well defined and ensures that*

$$\Delta\ell^\phi(s_k; x_k, \sigma_{k+1}) \geq \sigma_{k+1}\eta_\sigma \Delta\ell^v(s_k^s; x_k) \geq 0 \quad \textit{for all } k \geq 0. \tag{3.12}$$

Proof. If $\Delta\ell^\phi(s_k; x_k, \sigma_k) \geq \sigma_k \eta_\sigma \Delta\ell^v(s_k^s; x_k)$, then the desired result immediately follows from the update $\sigma_{k+1} = \sigma_k$. Thus, for the remainder of the proof we assume that

$$\Delta\ell^\phi(s_k; x_k, \sigma_k) < \sigma_k \eta_\sigma \Delta\ell^v(s_k^s; x_k). \tag{3.13}$$

Suppose, for a contradiction, that $\Delta\ell^v(s_k^s; x_k) = 0$. Since x_k is not an infeasible stationary point by assumption, it follows that $v(x_k) = 0$. Then, it follows from Lemma 5 and the fact that B_k is positive definite by assumption that $\Delta\ell^\phi(s_k; x_k, \sigma_k) \geq \frac{1}{2} s_k^{pT} B_k s_k^p \geq 0$, which contradicts (3.13) since $\Delta\ell^v(s_k^s; x_k) = 0$.

Thus, we conclude that $\Delta\ell^v(s_k^s; x_k) > 0$. Combining this with the choice $0 < \eta_\sigma < \eta_v < 1$ in (3.11) and (3.8) we conclude that $\Delta\ell^v(s_k; x_k) \geq \eta_v \Delta\ell^v(s_k^s; x_k) > \eta_\sigma \Delta\ell^v(s_k^s; x_k) > 0$, and thus

$$\eta_\sigma \Delta\ell^v(s_k^s; x_k) - \Delta\ell^v(s_k; x_k) < 0. \tag{3.14}$$

It then follows from (2.12), (3.13), (3.14), and the fact that $\sigma_k > 0$ that

$$\Delta\ell^f(s_k; x_k) = \Delta\ell^\phi(s_k; x_k, \sigma_k) - \sigma_k \Delta\ell^v(s_k; x_k) < \sigma_k \big[\eta_\sigma \Delta\ell^v(s_k^s; x_k) - \Delta\ell^v(s_k; x_k)\big] < 0. \tag{3.15}$$

Inequalities (3.14) and (3.15) imply that the penalty parameter update (3.11) is well-defined and positive.

It now follows from (3.11) that

$$\sigma_{k+1} \geq \frac{-\Delta\ell^f(s_k; x_k)}{\Delta\ell^v(s_k; x_k) - \eta_\sigma \Delta\ell^v(s_k^s; x_k)},$$

which may then be combined with (3.14) to yield

$$\sigma_{k+1}\eta_\sigma \Delta\ell^v(s_k^s; x_k) \leq \Delta\ell^f(s_k; x_k) + \sigma_{k+1}\Delta\ell^v(s_k; x_k) = \Delta\ell^\phi(s_k; x_k, \sigma_{k+1}),$$

which is the desired result (3.12). □

The next result will allow us to show that s_k is a descent direction for ϕ

under certain assumptions.

Lemma 7. *For any given value of the penalty parameter σ, point x, direction d, and positive-definite matrix B, it follows that*

$$[D_d\phi](x;\sigma) \leq -\Delta\ell^{\phi}(d;x,\sigma) \leq -\Delta q^{\phi}(d;x,B,\sigma).$$

Proof. Linearity of the directional derivative, (2.10), Lemma 3, (2.12), (2.13), and the fact that B_k is positive definite by choice, imply that

$$\begin{aligned}
[D_d\phi](x;\sigma) &= [D_d f](x) + \sigma[D_d v](x) \\
&= -g(x)^T d + \sigma[D_d v](x) \\
&\leq -\Delta\ell^f(d;x) - \sigma\Delta\ell^v(d;x) \\
&= -\Delta\ell^{\phi}(d;x,\sigma) \\
&= -\Delta q^{\phi}(d;x,B,\sigma) - \tfrac{1}{2}d^T B d \\
&\leq -\Delta q^{\phi}(d;x,B,\sigma),
\end{aligned}$$

which is the desired result. □

In most situations, we may now show that s_k is a descent direction for the penalty function.

Lemma 8. *If x_k is neither an infeasible stationary point nor a KKT point for problem* (2.1)*, then the direction s_k is a descent direction for $\phi(x;\sigma_{k+1})$ at the*

point x_k, i.e.,

$$[D_{s_k}\phi](x_k;\sigma_{k+1}) \leq -\Delta\ell^{\phi}(s_k;x_k,\sigma_{k+1}) < 0.$$

Proof. If $\Delta\ell^{v}(s_k^s;x_k) > 0$, then x_k cannot be an infeasible stationary point, and it follows from Lemma 7, Lemma 6, and (3.12) that $[D_{s_k}\phi](x_k;\sigma_{k+1}) \leq -\Delta\ell^{\phi}(s_k;x_k,\sigma_{k+1}) < 0$, which is the desired result. Conversely, if $\Delta\ell^{v}(s_k^s;x_k) = 0$, then $v(x_k) = 0$ since x_k is not an infeasible stationary point by assumption. It now follows from Lemma 7, $v(x_k) = 0$, Lemma 5, the fact that B_k is positive definite, and $s_k^p \neq 0$ since x_k is not a KKT point for problem (2.1) by assumption (see Lemma 1), that $[D_{s_k}\phi](x_k;\sigma_{k+1}) \leq -\Delta\ell^{\phi}(s_k;x_k,\sigma_{k+1}) \leq -\frac{1}{2}s_k^{pT}B_k s_k^p < 0$, which completes the proof. □

3.1.5 The accelerator step s_k^a

To improve performance, we compute an additional "acceleration" step; here we consider a single (simple) possibility, but other variants may be used [44].

Under common assumptions, the predictor step s_k^p will ultimately correctly identify those constraints that are active at a local solution of (2.1) [72]. A prediction based on s_k^p is formulated by

$$\mathcal{A}_k := \mathcal{A}(s_k^p) := \{i : [c_k + J_k s_k^p]_i = 0\}. \tag{3.16}$$

It is then natural to compute an *accelerator* step s_k^a as

$$s_k^a := s_k^p + s_k^{a'}, \tag{3.17}$$

where $s_k^{a'}$ is the solution to

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; q^f(s_k^p + s; x_k, H_k) \;\; \text{subject to} \;\; [J_k s]_{\mathcal{A}_k} = 0, \;\; \|s\|_2 \leq \delta_k^a, \tag{3.18}$$

where $\delta_k^a > 0$ is the trust-region radius, H_k is the exact second-derivative of the Lagrangian $\nabla^2_{xx} L(x_k, y_k)$, and y_k is a suitable Lagrange multiplier vector such as those from the predictor subproblem. (In fact, our global convergence analysis allows for any symmetric bounded sequence $\{H_k\}$, but here for concreteness we simply use $H_k = \nabla^2_{xx} L(x_k, y_k)$.) We note that subproblem (3.18) may be solved, for example, with the projected GLTR algorithm (see [73, Section 7.5.4] and the notes at the end that describe how to cope with the affine constraints $[J_k s]_{\mathcal{A}_k} = 0$). It can be shown that if $c_k + J_k s \geq 0$ is feasible, σ_k is sufficiently large, and x_k is "close enough" to a solution of (2.1) that satisfies certain second-order sufficient optimality conditions, then s_k^a is the solution to

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; q^f(s; x_k, H_k) \;\; \text{subject to} \;\; c_k + J_k s \geq 0, \tag{3.19}$$

which is the traditional SQO subproblem. However, our method of step compu-

tation is robust whereas the generally nonconvex subproblem (3.19) introduces many points of contention such as multiple solutions, unboundedness, and inconsistent constraints.

3.1.6 The Cauchy steps s_k^{cf} and $s_k^{c\phi}$

Since the matrix B_k is positive definite by construction and the exact second-derivative matrix H_k is generally an indefinite matrix, they may differ dramatically. To account for this when assessing overall step acceptance, we define and use a *Cauchy-f* step s_k^{cf} and *Cauchy-ϕ* step $s_k^{c\phi}$ as follows.

Given the search direction s_k, we define the Cauchy-f step as

$$s_k^{cf} := \alpha_k^f s_k, \quad \text{where} \quad \alpha_k^f := \underset{0\leq\alpha\leq 1}{\operatorname{argmin}} \; q^f(\alpha s_k; x_k, H_k). \tag{3.20}$$

Similarly, we define the Cauchy-ϕ step as

$$s_k^{c\phi} := \alpha_k^\phi s_k, \quad \text{where} \quad \alpha_k^\phi := \underset{0\leq\alpha\leq 1}{\operatorname{argmin}} \; q^\phi(\alpha s_k; x_k, H_k, \sigma_{k+1}). \tag{3.21}$$

We remark that the step size α_k^ϕ may be efficiently computed by examining the piecewise quadratic function $q^\phi(\alpha s_k; x_k, H_k, \sigma_{k+1})$ segment-by-segment between its derivative discontinuities.

3.1.7 The filter

The global convergence proof for our method is driven by maintaining and updating a filter $\mathcal{F}_k$ during each iteration. A filter is defined as follows, where $\mathbb{R}^+$ denotes the positive real numbers.

Definition 11 (filter)**.** A *filter* is any *finite* set of points in $\mathbb{R}^+ \times \mathbb{R}$.

The initial filter is defined to be $\mathcal{F}_0 = \emptyset$ and then sequentially updated in a manner that guarantees that $\mathcal{F}_k \subseteq \{(v_j, f_j) : 0 \leq j < k\}$. The decision to add certain ordered pairs to the filter depends on the concept of trial points being acceptable to the filter, which we now define.

Definition 12 (acceptable to $\mathcal{F}_k$)**.** We say that the point x is acceptable to $\mathcal{F}_k$ if its associated ordered pair $\big(v(x), f(x)\big)$ satisfies

$$v(x) \leq \max\left\{v_i - \alpha_i \eta_v \Delta\ell^v(s_i^s; x_i),\ \beta v_i\right\} \tag{3.22a}$$

or

$$f(x) \leq f_i - \gamma \min\left\{v_i - \alpha_i \eta_v \Delta\ell^v(s_i^s; x_i),\ \beta v_i\right\} \tag{3.22b}$$

for all $0 \leq i < k$ such that $(v_i, f_i) \in \mathcal{F}_k$ and some constants $\{\eta_v, \beta, \gamma\} \subset (0, 1)$.

The first inequality in (3.22) ensures that the constraint violation has been sufficiently reduced. We note that previous filter methods have not used the first quantity in the max on the right-hand side. Our improved condition takes

advantage of the information supplied by the steering steps s_k^s. Previous filter methods may easily have requested a decrease in the constraint violation that was unreasonable. In these circumstances, the trust-region radius would be decreased until the subproblem became infeasible and then a feasibility restoration phase would be entered. Our modified definition provides a practical target constraint violation based on local information derived from the steering step s_k^s. The second inequality in (3.22) guarantees that the objective function is sufficiently smaller at the point x than at points x_i whose ordered pair is in the current filter $\mathcal{F}_k$. These two conditions provide a so-called *margin* around the elements of the filter.

Note that Definition 12 does not require and does not imply that the current vector x_k is in $\mathcal{F}_k$ when determining acceptability. During our search for an improved estimate of a solution to (2.1), it often does not make sense to accept a new point unless it is acceptable to the current filter *and* better than the current point x_k. This leads to the following definition.

Definition 13 (acceptable to $\mathcal{F}_k$ augmented by x_k)**.** We say that x is acceptable to $\mathcal{F}_k$ augmented by x_k if x is acceptable to $\mathcal{F}_k$ as given by Definition 12 and (3.22) holds with $i = k$.

In the next section we present our main filter SQO method. Each iteration requires the search for a new point that must satisfy a subset of specified conditions. We stress that the updated point x_{k+1} is not necessarily acceptable to $\mathcal{F}_k$.

Moreover, the vector x_{k+1} being acceptable to $\mathcal{F}_k$ (possibly augmented by x_k) is a necessary, but not sufficient condition, for adding the ordered pair (v_{k+1}, f_{k+1}) to the filter $\mathcal{F}_k$. Details of how we update $\mathcal{F}_k$ are described in the next section.

3.1.8 The complete algorithm

Our method is formally stated as Algorithm 1. Every iteration begins by computing the set of trial steps $\{s_k^a, s_k\}$ as described in Sections 3.1.5 and 3.1.3. Once these trial steps are computed, we seek a step length α_k such that for some $\hat{s}_k \in \{s_k^a, s_k\}$ the step $x_k + \alpha_k \hat{s}_k$ satisfies one of four possible sets of conditions. Which sets of conditions we seek to satisfy depends on whether the algorithm is in filter mode (roughly a traditional filter strategy) or penalty mode (our alternative to a traditional restoration phase). We now discuss these two modes in detail.

In filter mode, we perform a backtracking line search until we find a pair $(\alpha_k, \hat{s}_k)$ with $\hat{s}_k \in \{s_k, s_k^a\}$ that forms a v-pair or o-pair, or a pair (α_k, s_k) that forms a b-pair. We discuss these in turn.

A v-pair is defined as follows.

Definition 14 (v-pair)**.** The pair (α, s) constitutes a v-pair if $x_k + \alpha s$ is acceptable to $\mathcal{F}_k$ augmented by x_k and

$$\Delta\ell^f(s_k; x_k) < \gamma_v \Delta\ell^v(s_k; x_k) \text{ for some } \gamma_v \in (0, 1). \tag{3.23}$$

Algorithm 1 Filter sequential quadratic programming algorithm.

Input an initial primal-dual pair (x_0, y_0).
Choose $\{\eta_v, \eta_\sigma, \eta_\phi, \sigma_{\text{inc}}, \beta, \gamma, \gamma_v, \gamma_f, \gamma_\phi, \xi\} \subset (0,1)$ and $0 < \delta_{\min} \leq \delta_{\max} < \infty$.
Set $k \leftarrow 0$, $\mathcal{F}_0 \leftarrow \emptyset$, $\mathcal{P}$-$mode \leftarrow$ false, and choose $\sigma_0 > 0$ and $\delta_0 \in [\delta_{\min}, \delta_{\max}]$.
loop
    Compute s_k^s as a solution of (3.1).
    Calculate $\Delta\ell^v(s_k^s; x_k)$ from (2.9).
    if $\Delta\ell^v(s_k^s; x_k) = 0$ and $v(x_k) > 0$, **then**
        return with the infeasible stationary point x_k for problem (2.1).
    Choose $B_k \succ 0$. Compute s_k^p as the solution of (3.4) with multiplier y_k^p.
    if $\Delta q^\phi(s_k^p; x_k, \sigma_k) = v(x_k) = 0$, **then**
        return with the KKT point (x_k, y_k^p) for problem (2.1).
    Compute $s_k = (1 - \tau_k)s_k^s + \tau_k s_k^p$ from (3.7) such that (3.8) is satisfied.
    Compute the new weight σ_{k+1} from (3.11).
    Choose $\delta_k^a > 0$. Solve (3.17) and (3.18) to get s_k^a and y_k^a..
    Compute $s_k^{c\phi}$ from (3.21). Calculate $\Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1})$ from (2.13).

16: **if** $\mathcal{P}$-$mode$ **then**
17: **for** $j = 0, 1, 2, \ldots$ **do**
18: Set $\alpha_k \leftarrow \xi^j$.
19: **for** $\hat{s}_k \in \{s_k^a, s_k\}$ **do**
20: **if** $(\alpha_k, \hat{s}_k)$ is a p-pair **then**
21: Set $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$ and go to Line 22. ▷ p-iterate
22: **if** $x_k + \alpha_k \hat{s}_k$ is acceptable to $\mathcal{F}_k$ **then**
23: Set $\mathcal{P}$-$mode \leftarrow$ false.
24: **else**
25: Compute s_k^{cf} from (3.20). Calculate $\Delta q^f(s_k^{cf}; x_k, H_k)$ from (2.11).
26: **for** $j = 0, 1, 2, \ldots$ **do**
27: Set $\alpha_k \leftarrow \xi^j$.
28: **for** $\hat{s}_k \in \{s_k^a, s_k\}$ **do**
29: **if** $(\alpha_k, \hat{s}_k)$ is a v-pair **then**
30: Set $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_k, f_k)\}$ and go to Line 36. ▷ v-iterate
31: **if** $(\alpha_k, \hat{s}_k)$ is an o-pair **then**
32: Set $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$ and go to Line 36. ▷ o-iterate
33: **if** (α_k, s_k) is a b-pair **then**
34: Set $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_k, f_k)\}$, and $\mathcal{P}$-$mode \leftarrow$ true. ▷ b-iterate
35: Go to Line 36.
36: **if** (3.29) is satisfied **then**
37: Set $\sigma_{k+1} \leftarrow \sigma_{k+1} + \sigma_{\text{inc}}$.
38: Set $x_{k+1} \leftarrow x_k + \alpha_k \hat{s}_k$, $y_{k+1} \leftarrow y_k^p$, $\delta_{k+1} \in [\delta_{\min}, \delta_{\max}]$, and $k \leftarrow k + 1$.

A v-pair $(\alpha_k, \hat{s}_k)$ earns its name since the step $x_k + \alpha_k \hat{s}_k$ is acceptable to the current filter augmented by x_k, but the step s_k did not predict sufficient decrease in f as measured by (3.23); we say that k is a v-iterate since the focus of the iteration is on reducing the constraint violation v. In this case, we choose to add the pair (v_k, f_k) to the filter $\mathcal{F}_k$.

An o-pair is characterized as follows.

Definition 15 (o-pair)**.** The pair (α, s) constitutes an o-pair if $x_k + \alpha s$ is acceptable to $\mathcal{F}_k$,

$$\Delta \ell^f(s_k; x_k) \geq \gamma_v \Delta \ell^v(s_k; x_k), \tag{3.24a}$$

and

$$f(x_k + \alpha s) \leq f(x_k) - \gamma_f \alpha \rho_k^f, \tag{3.24b}$$

where $\gamma_v \in (0, 1)$ is the same constant used to define a v-pair, $\gamma_f \in (0, 1)$, and

$$\rho_k^f := \min \left[\Delta \ell^f(s_k; x_k),\ \Delta q^f(s_k^{cf}; x_k, H_k) \right]. \tag{3.25}$$

An o-pair $(\alpha_k, \hat{s}_k)$ is so designated since $x_k + \alpha_k \hat{s}_k$ is acceptable to the filter, s_k predicts decrease in the objective function as measured by (3.24a), and a sufficient decrease in the objective is realized as given by (3.24b); we say that k is an o-iterate since progress has been made on decreasing the objective function.

In this case we do not add any new entries to the filter.

The definitions of v- and o-pairs are natural in light of the mechanism of the filter and are similar in spirit to conditions used by previous methods [1, 41, 42, 52, 54, 55]. As for these methods, these two sets of conditions are not sufficient for ensuring convergence since previously added filter entries may prevent (block) additional progress. In this situation, other filter algorithms typically decrease the trust-region radius or perform backtracking until a restoration phase is triggered. To prevent this undesirable situation we introduce the following definition of a b-pair.

Definition 16 (b-pair)**.** The pair (α, s) constitutes a b-pair if

$$v(x_k + \alpha s) < v(x_k) \tag{3.26}$$

and

$$\phi(x_k + \alpha s; \sigma_{k+1}) \leq \phi(x_k; \sigma_{k+1}) - \gamma_\phi \alpha \rho_k^\phi \quad \text{for some } \gamma_\phi \in (0, 1), \tag{3.27}$$

where

$$\rho_k^\phi := \min \left[\Delta \ell^\phi(s_k; x_k, \sigma_{k+1}),\ \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1}) \right]. \tag{3.28}$$

An iterate $x_k + \alpha_k s_k$ associated with a b-pair (α_k, s_k) decreases both the constraint violation and penalty function; we say that k is a b-iterate since the

conditions that define a b-pair suggest that one or more filter entries is blocking a productive step. In this case, we choose to accept the trial point, add (v_k, f_k) to the filter, and enter what we will refer to as a penalty mode. We view penalty mode as an alternative to a traditional restoration phase. Moreover, since steps are always tested for acceptability based on the filter criteria, i.e., o- and v-pairs, before checking for decrease in the constraint violation and penalty function as stipulated by b-pairs, we give clear preference to staying in filter mode.

In penalty mode, we calculate a new iterate by perform a backtracking line search until we find a pair $(\alpha_k, \hat{s}_k)$ for some $\hat{s}_k \in \{s_k^a, s_k\}$ that satisfies the following conditions that define a p-pair.

Definition 17 (p-pair)**.** The pair (α, s) is a p-pair if (3.27) is satisfied.

If $(\alpha_k, \hat{s}_k)$ is a p-pair, then $\phi(x_k + \alpha_k \hat{s}_k; \sigma_{k+1})$ is sufficiently smaller than $\phi(x_k; \sigma_{k+1})$; we say that k is a p-iterate since the penalty function has been decreased. In addition, if $x_k + \alpha_k \hat{s}_k$ is acceptable to the current filter, we return to filter mode; otherwise, we remain in penalty mode.

Finally, after a new trial step is computed, we choose to increase the penalty parameter if

$$\Delta q^{\phi}(s_k; x_k, B_k, \sigma_{k+1}) < \eta_\phi \Delta q^{\phi}(s_k^p; x_k, B_k, \sigma_{k+1}) \quad \text{for some } \eta_\phi \in (0, 1), \tag{3.29}$$

since this indicates that τ_k is very small and the search direction s_k does not adequately reflect the decrease predicted by s_k^p in the penalty function.

For future reference we define the following index sets based on the different types of pairs:

$$\mathcal{S}_v = \{k : \ k \text{ is a v-iterate } \}, \qquad \mathcal{S}_o = \{k : \ k \text{ is an o-iterate } \},$$

$$\mathcal{S}_p = \{k : \ k \text{ is a p-iterate } \}, \qquad \mathcal{S}_b = \{k : \ k \text{ is an b-iterate } \}.$$

We complete this section by summarizing the computational complexity of each iteration of Algorithm 1, which requires the calculation of multiple directions. Specifically, each iteration requires the solution of a linear program to obtain the steering step (3.1), a strictly convex quadratic program to get the predictor step (3.4), a single matrix-vector multiplication to solve the one-dimensional optimization problem for the Cauchy-f step (3.20), a one-dimensional search along a piecewise linear path to obtain the Cauchy-ϕ step (see (3.21)), and an approximate solution to an equality-constrained quadratic problem for the accelerator step (3.18). Thus, the predominant computational cost for each iteration is the calculation of the steering and predictor step.

3.2 Well-posedness

In this section, we prove that the method is well-posed, that is, at each iteration k, either the method exits with x_k as an infeasible stationary point (Line 8) or as a KKT point (Line 11), or it successfully obtains the next iterate x_{k+1} and penalty parameter σ_{k+1}. To do this, we will need to show that every subproblem and step computation is well-defined and that the backtracking line search procedure terminates finitely.

We will now make the following assumption, which we do not explicitly state for each result.

Assumption 3.2.1. The functions f and c are both differentiable with Lipschitz continuous derivatives in the neighborhood of the point x_k.

We begin by observing that the steering problem (3.2) is convex, always feasible, and the objective function is bounded below by zero, i.e., it is well-defined. If $v(x_k) > 0$ and $\Delta\ell^v(s_k^s; x_k) = 0$, then x_k is an infeasible stationary point and we exit in line 8 of Algorithm 1. Otherwise, x_k is not an infeasible stationary point and we proceed to compute a predictor step from problem (3.4), which we now argue is well-defined. This is obvious when $\Delta\ell^v(s_k^s; x_k) \neq v(x_k)$ since then the strictly convex problem (3.4b) is always feasible. On the other hand, if $\Delta\ell^v(s_k^s; x_k) = v(x_k)$, then it follows that $||[c(x_k) + J(x_k)s_k^s]^-||_1 = 0$, which implies that $c_k + J_k s_k^s \geq 0$. Thus, $s = s_k^s$ is feasible for (3.4a), and the predictor

problem is well-defined. Lemma 2 shows that $\tau_k > 0$ and Lemma 6 shows that the update to the weighting parameter is well-defined. The accelerator problem (3.18) does not cause difficulties since by construction it is feasible, has bounded solutions, and may be solved (approximately) as noted in Section 3.1.5. It is also easy to see that both Cauchy step problems (3.20) and (3.21) are well-defined.

We now proceed to show that the line search terminates finitely. To this end, we first show that feasible iterates are never added to the filter.

Lemma 9. *Algorithm 1 ensures that if* (v_k, f_k) *is added to the filter, then* $v_k > 0$.

Proof. For a proof by contradiction, suppose that $v(x_k) = 0$. It follows from $v(x_k) = 0$ and convexity of ℓ^v that $\Delta\ell^v(s_k^s, x_k) = 0$, and we may then use (3.4a), (3.7), (3.8), and the fact that x_k is not a KKT point for (2.1) (otherwise we would already have exited on Line 11 of Algorithm 1) to show that

$$\tau_k = 1, \quad s_k = s_k^p \neq 0, \quad \text{and} \quad \Delta\ell^v(s_k; x_k) = \Delta\ell^v(s_k^p; x_k) = 0. \tag{3.30}$$

It then follows from (3.30), (2.12), Lemma 5, $v(x_k) = 0$, and $B_k \succ 0$ that

$$\Delta\ell^f(s_k; x_k) = \Delta\ell^f(s_k^p; x_k) = \Delta\ell^\phi(s_k^p; x_k, \sigma_{k+1}) = \Delta\ell^\phi(s_k; x_k, \sigma_{k+1}) \geq \tfrac{1}{2} s_k^{pT} B_k s_k^p > 0. \tag{3.31}$$

Since (v_k, f_k) was added to the filter, it follows from the construction of Algo-

rithm 1 that either $(\alpha_k, \hat{s}_k)$ is a v-iterate or (α_k, s_k) is a b-pair, which implies that at least one of $v(x_k + \alpha_k s_k) < v(x_k)$ or $\Delta\ell^f(s_k; x_k) < \gamma_v \Delta\ell^v(s_k; x_k)$ holds, amongst other requirements. However, since $v(x_k + \alpha_k s_k) < v(x_k) = 0$ is not possible, we conclude that $\Delta\ell^f(s_k; x_k) < \gamma_v \Delta\ell^v(s_k; x_k) = 0$, where we have also used (3.30); this contradicts (3.31) and proves the result. □

The next two results show that our line search procedure terminates anytime $\mathcal{P}\text{-}mode$ has the value false at the beginning of the kth iteration. We first consider the case when x_k is feasible.

Lemma 10. *If* $\mathcal{P}\text{-}mode =$ false *at the beginning of the* k*th iteration,* $v(x_k) = 0$*, and* x_k *is not a first-order solution to problem* (2.1)*, then the pair* (α, s_k) *is an o-pair for all* $\alpha > 0$ *sufficiently small. Moreover,* $k \in \mathcal{S}_o$*.*

Proof. As in the proof of Lemma 9, it follows that $v(x_k) = \Delta\ell^v(s_k^s; x_k) = 0$. This may be combined with the fact that x_k is assumed to not be a first-order solution to (2.1), (3.4a), (3.7), (3.8), Lemma 5, the fact that B_k is positive definite, and the definition of $\Delta\ell^\phi$ to conclude that

$$s_k = s_k^p \neq 0, \tag{3.32a}$$

$$c_k + J_k s_k \geq 0, \tag{3.32b}$$

and

$$\Delta\ell^f(s_k; x_k) = \Delta\ell^\phi(s_k; x_k, \sigma_{k+1}) > 0 = \gamma_v \Delta\ell^v(s_k; x_k). \tag{3.32c}$$

Next, $v(x_k) = 0$ and (3.32) imply that $c_k + \alpha J_k s_k \geq 0$ for all $\alpha \in [0, 1]$. Combining this fact with Taylor's Theorem, Assumption 3.2.1, and (3.32) yields

$$v(x_k + \alpha s_k) = \left\| [c(x_k + \alpha s_k)]^- \right\|_1 = \left\| [c_k + \alpha J_k s_k + O(\alpha^2)]^- \right\|_1 \leq O(\alpha^2) \qquad (3.33)$$

for $\alpha \in [0, 1]$. Since Lemma 9 implies that $v_i > 0$ for all $(v_i, f_i) \in \mathcal{F}_k$, we may conclude from (3.33) that

$$v(x_k + \alpha s_k) \leq \min_{(v_i, f_i) \in \mathcal{F}_k} \beta v_i \quad \text{for all } \alpha > 0 \text{ sufficiently small,}$$

where $\beta \in (0, 1)$ is defined in (3.22), so that

$$x_k + \alpha s_k \text{ is acceptable to the filter for all } \alpha > 0 \text{ sufficiently small.} \qquad (3.34)$$

Next, Taylor's Theorem, Assumption 3.2.1, the definition of $\Delta \ell^f$, and (3.32) imply that

$$\begin{aligned} f(x_k + \alpha s_k) &= f_k + \alpha g_k^T s_k + O(\alpha^2) = f_k - \alpha \Delta \ell^f(s_k; x_k) + O(\alpha^2) \\ &\leq f_k - \gamma_f \alpha \Delta \ell^f(s_k; x_k) \text{ for all } \alpha > 0 \text{ sufficiently small,} \end{aligned} \qquad (3.35)$$

where $\gamma_f \in (0, 1)$ is defined in (3.24b). It follows from (3.32), (3.34), and (3.35) that (α, s_k) is an o-pair for all $\alpha > 0$ sufficiently small, which proves the first

result of this lemma.

We just proved that the **for** loop on line 26 in Algorithm 1 always terminates. Moreover, it cannot terminate on line 29 since (3.32) holds. Also, it can never terminate as a result of the **if** on line 33 since $v(x_k + \alpha s_k) < v(x_k) = 0$ is impossible for all α. Therefore, the line search must terminate with an o-pair $(\alpha_k, \hat{s}_k)$, which implies that $k \in \mathcal{S}_o$. □

We now consider the case when x_k is infeasible.

Lemma 11. *If $\mathcal{P}$-mode $=$* false *at the beginning of iteration k, $v(x_k) > 0$, and x_k is not an infeasible stationary point, then (α, s_k) is a b-pair for all $\alpha > 0$ sufficiently small.*

Proof. It follows from the assumptions of this lemma and Lemma 8 that

$$[D_{s_k}\phi](x_k; \sigma_{k+1}) \leq -\Delta\ell^{\phi}(s_k; x_k, \sigma_{k+1}) < 0 \tag{3.36}$$

so that the direction s_k is a strict descent direction for ϕ at x_k with penalty parameter σ_{k+1}. Using the definition of the directional derivative, (3.36), $\gamma_\phi \in (0, 1)$ defined in (3.27), and (3.28) we conclude that

$$\begin{aligned}
\phi(x_k + \alpha s_k; \sigma_{k+1}) &\leq \phi(x_k; \sigma_{k+1}) + \alpha\gamma_\phi[D_{s_k}\phi](x_k; \sigma_{k+1}) \\
&\leq \phi(x_k; \sigma_{k+1}) - \alpha\gamma_\phi\Delta\ell^{\phi}(s_k; x_k, \sigma_{k+1}) \qquad (3.37) \\
&\leq \phi(x_k; \sigma_{k+1}) - \alpha\gamma_\phi\rho_k^{\phi} \quad \text{for all } \alpha > 0 \text{ sufficiently small.}
\end{aligned}$$

Since $v(x_k) \neq 0$ and x_k is not an infeasible stationary point, we know that $\Delta\ell^v(s_k^s; x_k) > 0$. Lemma 4 then implies that

$$[D_{s_k} v](x_k) \leq -\Delta\ell^v(s_k; x_k) \leq -\eta_v \Delta\ell^v(s_k^s; x_k) < 0$$

so that s_k is a descent direction for v at x_k. A similar argument as the one that lead to (3.37), yields

$$v(x_k + \alpha s_k) < v(x_k) \quad \text{for all } \alpha > 0 \text{ sufficiently small.} \tag{3.38}$$

It follows from (3.37) and (3.38) that (α, s_k) is a b-pair for all $\alpha > 0$ sufficiently small, as claimed. □

The next lemma considers the case when $\mathcal{P}\text{-}mode$ is true at the beginning of the kth iteration, and shows that successful trial iterates may be obtained through backtracking as performed in Algorithm 1.

Lemma 12. *If* $\mathcal{P}\text{-}mode =$ true *at the beginning of the* k*th iteration and* x_k *is neither an infeasible stationary point nor a first-order solution to problem* (2.1)*, then* (α, s_k) *is a p-pair for all* $\alpha > 0$ *sufficiently small.*

Proof. The proof follows exactly as in the first part of Lemma 11. □

We now combine these results to prove that Algorithm 1 is well-posed.

Theorem 1. *Algorithm 1 is well-posed.*

Proof. As described in the first paragraph of Section 3.2, every subproblem and step computation is well defined, and Lemma 6 ensures that the update to the weighting parameter is well defined.

All that remains is to prove that the line search terminates. First, if $\mathcal{P}$-$mode$ has the value false at the beginning of iteration x_k and $v(x_k) = 0$, then by Lemma 10, we are guaranteed that finite termination and that $k \in \mathcal{S}_o$. Second, if $\mathcal{P}$-$mode$ has the value false and $v(x_k) > 0$, then Lemma 11 ensures that the backtracking line search will terminate finitely. Finally, suppose that $\mathcal{P}$-$mode$ has the value true at the beginning of iteration k. It then follows from Lemma 12 that the backtracking terminates finitely. □

3.3 Global convergence

We prove that limit points of the iterates generated by Algorithm 1 have desirable properties. To this end, we use the following common assumptions.

Assumption 3.3.1. The iterates $\{x_k\}$ lie in an open, bounded, convex set $\mathcal{X}$.

Assumption 3.3.2. The problem functions $f(x)$ and $c(x)$ are twice continuously differentiable on $\mathcal{X}$.

Assumption 3.3.3. The matrices B_k are uniformly positive definite and bounded, i.e., there exists values $0 < \lambda_{\min} < \lambda_{\max} < \infty$ such that $\lambda_{\min} \|s\|_2^2 \leq s^T B_k s \leq \lambda_{\max} \|s\|_2^2$ for all $s \in \mathbb{R}^n$ and all B_k.

Assumption 3.3.4. The matrices H_k are uniformly bounded, i.e., $\|H_k\|_2 \leq \mu_{\max}$ for some $\mu_{\max} \geq 1$.

For clarity and motivational purposes, we immediately state our main convergence theorem that makes use of the Mangasarian-Fromovitz constraint qualification (MFCQ) [74].

Theorem 2. *If Assumptions 3.3.1–3.3.4 hold, then one of the following holds:*

(i) *Algorithm 1 terminates finitely with either a first-order KKT point or an infeasible stationary point in lines 11 or 8, respectively, for problem* (2.1).

(ii) *Algorithm 1 generates infinitely many iterations* $\{x_k\}$, $\sigma_k = \bar{\sigma} < \infty$ *for all* k *sufficiently large, and there exists a limit point* x_* *of* $\{x_k\}$ *that is either a first-order KKT point or an infeasible stationary point for problem* (2.1).

(iii) *Algorithm 1 generates infinitely many iterations* $\{x_k\}$, $\lim_{k\to\infty} \sigma_k = \infty$, *and there exists a limit point* x_* *of* $\{x_k\}$ *that is either an infeasible stationary point or a feasible point at which the MFCQ fails.*

Proof. The result follows from the following analysis that considers the various cases that can occur. In particular, it follows from Theorems 3, 4, 5, 6, and the construction of Algorithm 1. □

We now present a sequence of lemmas that will be useful in the convergence analysis. The first result is adapted from [71, Theorem 3.6] and provides a bound on the trial step s_k.

Lemma 13. *If Assumptions 3.3.1–3.3.3 hold and x_k and s_k are generated by Algorithm 1, then*

$$\|s_k\|_2 \leq \max\left\{1, \frac{2}{\lambda_{min}}\big[\,\|g_k\|_2 + \sigma_k v(x_k)\big], \sqrt{n}\delta_{max}\right\}. \tag{3.39}$$

Furthermore, if $\{\sigma_k\}$ is bounded, then there exists a constant $M_s > 0$ such that $\|s_k\|_2 \leq M_s$ for all k.

Proof. First, we claim that the predictor step s_k^p must satisfy

$$\|s_k^p\|_2 \leq \max\left\{1, \frac{2}{\lambda_{\min}}\big[\,\|g_k\|_2 + \sigma_k v(x_k)\big]\right\}, \tag{3.40}$$

which can be seen as follows. Suppose that (3.40) is not satisfied so that

$$\|s_k^p\|_2 > 1 \quad \text{and} \quad \tfrac{1}{2}\lambda_{\min}\|s_k^p\|_2 > \|g_k\|_2 + \sigma_k v(x_k). \tag{3.41}$$

It then follows from the definitions of Δq^ϕ and ℓ^v, the Cauchy-Schwarz inequality, Assumption 3.3.3, and (3.41) that

$$\begin{aligned}
\Delta q^\phi(s_k^p; x_k, B_k, \sigma_k) &= -g_k^T s_k^p - \tfrac{1}{2} s_k^{pT} B_k s_k^p + \sigma_k\big(\ell^v(0; x_k) - \ell^v(s_k^p; x_k)\big) \\
&\leq \|g_k\|_2 \|s_k^p\|_2 - \tfrac{1}{2}\lambda_{\min}\|s_k^p\|_2^2 + \sigma_k v(x_k) \\
&\leq \|g_k\|_2 \|s_k^p\|_2 - \tfrac{1}{2}\lambda_{\min}\|s_k^p\|_2^2 + \|s_k^p\|_2 \sigma_k v(x_k) \\
&= \|s_k^p\|_2 \Big(\,\|g_k\|_2 - \tfrac{1}{2}\lambda_{\min}\|s_k^p\|_2 + \sigma_k v(x_k)\Big) < 0,
\end{aligned}$$

which contradicts the fact that s_k^p is the unique global minimizer to the strictly convex predictor problem. Thus, (3.40) must hold and when combined with (3.7), the use of the triangle-inequality, the use of the trust-region radius $\delta_k \in [\delta_{\min}, \delta_{\max}]$ in the steering problem, implies that

$$\|s_k\|_2 \leq \max\left\{1, \frac{2}{\lambda_{\min}}\left[\,\|g_k\|_2 + \sigma_k v(x_k)\right], \sqrt{n}\delta_{\max}\right\} \tag{3.42}$$

which proves (3.39). Since g_k and $v(x_k)$ are uniformly bounded as a result of Assumptions 3.3.1 and 3.3.2, it is clear that if $\{\sigma_k\}$ is bounded, then there exists $M_s < \infty$ such that $\|s_k\|_2 \leq M_s$ for all k. □

The following result provides a relationship between the predicted change in the linear model and the change achieved in the line search process for both the objective function and the constraint violation.

Lemma 14. *(Equivalent to [57, Lemma 3]) Suppose that Assumptions 3.3.1 and 3.3.2 hold. Then, there exist constants $\{C_f, C_v\} > 0$ such that for all k and $\alpha \in (0, 1]$, we have*

$$f(x_k + \alpha s) \leq f(x_k) - \alpha\Delta\ell^f(s; x_k) + \alpha^2 C_f \|s\|_2^2 \tag{3.43}$$

and

$$v(x_k + \alpha s) \leq v(x_k) - \alpha\Delta\ell^v(s; x_k) + \alpha^2 C_v \|s\|_2^2 . \tag{3.44}$$

Proof. The inequality in (3.43) is a direct result of applying Taylor's theorem and Assumption 3.3.2.

For (3.44), it follows from the integral mean-value theorem, Assumptions 3.3.1 and 3.3.2 and the implied Lipschitz continuity of $J(x)$, the triangle inequality, and the convexity of ℓ^v, that for some constant Lipschitz constant C,

$$\begin{aligned} v(x_k + \alpha s) &= \left\| [c(x_k + \alpha s)]^- \right\|_1 \\ &= \left\| \left[c(x_k) + \alpha J_k s + \alpha \int_0^1 [J(x_k + \theta \alpha s) - J(x_k)] s d\theta \right]^- \right\|_1 \\ &\leq \left\| [c(x_k) + \alpha J_k s]^- \right\|_1 + \alpha^2 \sqrt{n} C \left\| s \right\|_2^2 \\ &\leq (1 - \alpha) \left\| [c(x_k)]^- \right\|_1 + \alpha \left\| [c(x_k) + J_k s]^- \right\|_1 + \alpha^2 \sqrt{n} C \left\| s \right\|_2^2 \\ &= v(x_k) - \alpha \Delta \ell^v(s; x_k) + \alpha^2 \sqrt{n} C \left\| s \right\|_2^2 \quad \text{for all } \alpha \in (0, 1]. \end{aligned}$$

This proves (3.44) by defining $C_v := \sqrt{n} C$. □

The next two lemmas provide a relationship between the predicted linear decrease in the objective function and the quantity ρ_k^f defined by (3.25).

Lemma 15. *If Assumption 3.3.4 holds and $\Delta \ell^f(s_k; x_k) \geq 0$, then*

$$\Delta q^f(s_k^{cf}; x_k, H_k) \geq \tfrac{1}{2} \Delta \ell^f(s_k; x_k) \min \left\{ \frac{\Delta \ell^f(s_k; x_k)}{\mu_{max} \left\| s_k \right\|_2^2}, 1 \right\}. \tag{3.45}$$

Proof. If $\Delta \ell^f(s_k; x_k) = 0$, then the result follows immediately from the definition of s_k^{cf} in (3.20).

Now, suppose that $\Delta\ell^f(s_k; x_k) > 0$. It follows from (3.20) and the definition of Δq^f that

$$\Delta q^f(s_k^{cf}; x_k, H_k) \geq \Delta q^f(\alpha s_k; x_k, H_k) = -\alpha g_k^T s_k - \tfrac{1}{2}\alpha^2 s_k^T H_k s_k \text{ for all } 0 \leq \alpha \leq 1.$$

The right hand side of the previous equation may be written as

$$q(\alpha) = a\alpha^2 + b\alpha, \text{ where } a = -\tfrac{1}{2}s_k^T H_k s_k \text{ and } b = \Delta\ell^f(s_k; x_k) = -g_k^T s_k > 0.$$

We wish to maximize q on the interval $[0, 1]$ so we differentiate $q(\alpha)$ with respect to α and set the result to zero to obtain a stationary point at $-\frac{b}{2a}$. Now, consider three cases.

Case 1: ($a < 0$ and $-\frac{b}{2a} \leq 1$) The maximum of $q(\alpha)$ on the interval $[0, 1]$ is achieved at $\alpha = -\frac{b}{2a}$. Note that $\alpha > 0$, since $b = \Delta\ell^f(s_k; x_k) > 0$ by assumption. Then, we have

$$q\left(-\frac{b}{2a}\right) = a\frac{b^2}{4a^2} - b\frac{b}{2a} = -\frac{b^2}{4a}.$$

It follows from the definition of a and b, the Cauchy-Schwarz inequality, and Assumption 3.3.4 that

$$q\left(-\frac{b}{2a}\right) = \frac{\Delta\ell^f(s_k; x_k)^2}{2s_k^T H_k s_k} \geq \frac{\Delta\ell^f(s_k; x_k)^2}{2\left\|H_k\right\|_2 \left\|s_k\right\|_2^2} \geq \frac{\Delta\ell^f(s_k; x_k)^2}{2\mu_{\max}\left\|s_k\right\|_2^2}.$$

Case 2: ($a < 0$ and $-\frac{b}{2a} > 1$) The maximum of $q(\alpha)$ on the interval $[0, 1]$ is

achieved at $\alpha = 1$, where

$$q(1) = a + b > -\tfrac{1}{2}b + b = \tfrac{1}{2}b = \tfrac{1}{2}\Delta\ell^f(s_k; x_k).$$

Case 3: ($a \geq 0$) The maximum of $q(\alpha)$ on the interval $[0, 1]$ is achieved at $\alpha = 1$ so that

$$q(1) = a + b > b > \tfrac{1}{2}b = \tfrac{1}{2}\Delta\ell^f(s_k; x_k).$$

Finally, combining all three cases and defining $\alpha' = \arg\max_{\alpha\in[0,1]} q(\alpha)$, it follows that

$$\begin{aligned}\Delta q^f(s_k^{cf}; x_k, H_k) &= q(\alpha') \\ &\geq \min\left\{\frac{\Delta\ell^f(s_k; x_k)^2}{2\mu_{\max}\|s_k\|_2^2},\ \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right\} \\ &= \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\min\left\{\frac{\Delta\ell^f(s_k; x_k)}{\mu_{\max}\|s_k\|_2^2},\ 1\right\}\end{aligned}$$

as desired. □

Lemma 16. *Suppose that the Assumptions 3.3.1–3.3.4 are satisfied and that* $\{\sigma_k\}$ *is bounded. Then, there exists a constant* $C_\rho > 0$ *such that whenever* $\Delta\ell^f(s_k; x_k) \geq 0$*, it follows that*

$$\rho_k^f \geq \min\left[C_\rho\Delta\ell^f(s_k; x_k)^2,\ \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right]. \tag{3.46}$$

Proof. It follows from (3.25), Lemma 15, Lemma 13 and the assumption that $\Delta\ell^f(s_k; x_k) \geq 0$ that

$$\begin{aligned}
\rho_k^f &= \min\left[\Delta\ell^f(s_k; x_k),\, \Delta q^f(s_k^{cf}; x_k, H_k)\right] \\
&\geq \min\left[\Delta\ell^f(s_k; x_k),\, \min\left\{\frac{\Delta\ell^f(s_k; x_k)^2}{2\mu_{\max}\left\|s_k\right\|_2^2},\, \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right\}\right] \\
&\geq \min\left[\frac{\Delta\ell^f(s_k; x_k)^2}{2\mu_{\max}\left\|s_k\right\|_2^2},\, \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right] \\
&\geq \min\left[\frac{\Delta\ell^f(s_k; x_k)^2}{2\mu_{\max}M_s^2},\, \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right],
\end{aligned}$$

where $\{M_s, \mu_{\max}\} \subset (0, \infty)$ are defined in (3.39) and Assumption 3.3.4, respectively. The result now follows by defining $C_\rho := 1/(2\mu_{\max}M_s^2)$. □

The next two results provide a relationship between the predicted linear change in the penalty function and the quantity ρ_k^ϕ defined by (3.28).

Lemma 17. *If Assumption 3.3.4 holds and x_k is not an infeasible stationary point, then*

$$\Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1}) \geq \tfrac{1}{2}\Delta\ell^\phi(s_k; x_k, \sigma_{k+1}) \min\left\{\frac{\Delta\ell^\phi(s_k; x_k, \sigma_{k+1})}{\mu_{max}\left\|s_k\right\|_2^2},\, 1\right\}. \tag{3.47}$$

Proof. Since x_k is not an infeasible stationary point by assumption, it follows from Lemma 6 that $\Delta\ell^\phi(s_k; x_k, \sigma_{k+1}) \geq 0$. If $\Delta\ell^\phi(s_k; x_k, \sigma_{k+1}) = 0$, then the result follows immediately. Thus, for the remainder we assume $\Delta\ell^\phi(s_k; x_k, \sigma_{k+1}) > 0$.

It follows from (3.21), the convexity of $\ell^v(\cdot)$, and simple algebra that

$$
\begin{aligned}
&\Delta q^{\phi}(s_k^{c\phi}; x_k, H_k, \sigma_{k+1}) \\
&\quad \geq \Delta q^{\phi}(\alpha s_k; x_k, H_k, \sigma_{k+1}) \\
&\quad = -\alpha g_k^T s_k - \tfrac{1}{2}\alpha^2 s_k^T H_k s_k + \sigma_{k+1}\left(\left\|[c_k]^-\right\|_1 - \left\|[c_k + \alpha J_k s_k]^-\right\|_1\right) \\
&\quad \geq -\alpha g_k^T s_k - \tfrac{1}{2}\alpha^2 s_k^T H_k s_k + \sigma_{k+1}\left(\left\|[c_k]^-\right\|_1 - \alpha\left\|[c_k + J_k s_k]^-\right\|_1 - (1-\alpha)\left\|[c_k]^-\right\|_1\right) \\
&\quad = -\alpha g_k^T s_k - \tfrac{1}{2}\alpha^2 s_k^T H_k s_k + \alpha\sigma_{k+1}\left(\left\|[c_k]^-\right\|_1 - \left\|[c_k + J_k s_k]^-\right\|_1\right) \\
&\quad = \alpha\Delta\ell^{\phi}(s_k; x_k, \sigma_{k+1}) - \tfrac{1}{2}\alpha^2 s_k^T H_k s_k \ \text{ for all } \alpha \in [0,1].
\end{aligned}
$$

The right hand side of the equation is a quadratic function of α:

$$
q(\alpha) = a\alpha^2 + b\alpha, \ \text{ where } \ a = -\tfrac{1}{2}s_k^T H_k s_k \ \text{ and } \ b = \Delta\ell^{\phi}(s_k; x_k, \sigma_{k+1}) > 0.
$$

Analysis similar to that used in the proof of Lemma 15 yields

$$
\Delta q^{\phi}(s_k^{c\phi}; x_k, H_k, \sigma_{k+1}) \geq \min\left\{\frac{\Delta\ell^{\phi}(s_k; x_k, \sigma_{k+1})^2}{2\mu_{\max}\left\|s_k\right\|_2^2}, \ \tfrac{1}{2}\Delta\ell^{\phi}(s_k; x_k, \sigma_{k+1})\right\}, \tag{3.48}
$$

where $\mu_{\max}$ is from Assumption 3.3.4, as desired. □

Lemma 18. *Suppose that Assumptions 3.3.1–3.3.4 are satisfied, that Algorithm 1 never encounters an infeasible stationary point, and $\{\sigma_k\}$ is bounded.*

Then, there exists a constant $C_\rho \in (0, \infty)$ *such that*

$$\rho_k^\phi \geq \min\left[C_\rho \Delta\ell^\phi(s_k; x_k, \sigma_{k+1})^2,\ \tfrac{1}{2}\Delta\ell^\phi(s_k; x_k, \sigma_{k+1})\right] \quad \textit{for all } k \geq 0.$$

Proof. The proof follows exactly as in Lemma 16. □

3.3.1 Analysis: bounded weighting parameter

In this section we study Algorithm 1 under the assumption that the weighting parameter stays bounded. It follows from this assumption and Lemma 13 that there exists some k' and $\bar{\sigma} < \infty$ such that

$$\|s_k\|_2 \leq M_s < \infty \quad \text{and} \quad \sigma_k = \bar{\sigma} < \infty \quad \text{for all } k \geq k'. \tag{3.49}$$

Part (iii) of Theorem 2 implies that this scenario is guaranteed to occur, for example, when all limit points are neither infeasible stationary points nor feasible points at which the MFCQ fails.

We begin by showing that the line search step length is bounded away from zero in certain situations.

Lemma 19. *If Assumptions 3.3.1–3.3.3 and* (3.49) *hold and* $\epsilon > 0$*, then the following hold:*

(i) *There exists a constant* $\alpha_P > 0$ *such that* $\alpha_k \geq \alpha_P > 0$ *for all* $k \in \mathcal{K}_P$*, where*

$$\mathcal{K}_P = \{k \in \mathcal{S}_p : k \geq k' \text{ and } \Delta\ell^{\phi}(s_k; x_k, \bar{\sigma}) \geq \epsilon\}.$$

(ii) *There exists a constant* $\alpha_F > 0$ *such that* $\alpha_k \geq \alpha_F > 0$ *for all* $k \in \mathcal{K}_F$*, where*

$$\mathcal{K}_F = \{k \in \mathcal{S}_v \cup \mathcal{S}_o \cup \mathcal{S}_b : k \geq k' \text{ and } \Delta\ell^{v}(s_k^s; x_k) \geq \epsilon\}.$$

(iii) *There exists a constant* $\alpha_f > 0$ *such that* $(\alpha, s) = (\alpha, s_k)$ *satisfies* (3.24b) *for all* $0 < \alpha \leq \alpha_f$ *and all* $k \in \mathcal{K}_f$*, where*

$$\mathcal{K}_f = \{k \geq k' : \Delta\ell^{f}(s_k; x_k) \geq \epsilon\}.$$

Proof. From [75, Lemma 3.4], there exists some positive constant C_ϕ such that

$$\left|\phi(x_k + \alpha s_k; \bar{\sigma}) - \ell^{\phi}(\alpha s_k; x_k, \bar{\sigma})\right| \leq C_\phi \left\|\alpha s_k\right\|_2^2 \quad \text{for all } k \geq k' \text{ and } \alpha \in [0, 1]. \tag{3.50}$$

We first prove part (i). Suppose that α satisfies

$$0 \leq \alpha \leq \frac{(1 - \gamma_\phi)\epsilon}{C_\phi M_s^2}, \tag{3.51}$$

where $\gamma_\phi \in (0, 1)$ is set in Algorithm 1 and M_s is defined in (3.49). We then use

$\phi(x_k;\bar{\sigma}) = \ell^\phi(0;x_k,\bar{\sigma})$, the convexity of $\ell^\phi(\,\cdot\,;x_k,\bar{\sigma})$, (3.50), $\Delta\ell^\phi(s_k;x_k,\bar{\sigma}) \geq \epsilon$ for $k \in \mathcal{K}_P$, (3.49), (3.51), and (3.28) to conclude that

$$
\begin{aligned}
&\phi(x_k;\bar{\sigma}) - \phi(x_k+\alpha s_k;\bar{\sigma}) \\
&\quad = \big[\ell^\phi(0;x_k,\bar{\sigma}) - \ell^\phi(\alpha s_k;x_k,\bar{\sigma})\big] - \big[\phi(x_k+\alpha s_k;\bar{\sigma}) - \ell^\phi(\alpha s_k;x_k,\bar{\sigma})\big] \\
&\quad \geq \big[\ell^\phi(0;x_k,\bar{\sigma}) - \alpha\ell^\phi(s_k;x_k,\bar{\sigma}) - (1-\alpha)\ell^\phi(0;x_k,\bar{\sigma})\big] - C_\phi\alpha^2\left\|s_k\right\|_2^2 \\
&\quad = \alpha\big[\ell^\phi(0;x_k,\bar{\sigma}) - \ell^\phi(s_k;x_k,\bar{\sigma})\big] - C_\phi\alpha^2\left\|s_k\right\|_2^2 \\
&\quad = \gamma_\phi\alpha\Delta\ell^\phi(s_k;x_k,\bar{\sigma}) + (1-\gamma_\phi)\alpha\Delta\ell^\phi(s_k;x_k,\bar{\sigma}) - C_\phi\alpha^2\left\|s_k\right\|_2^2 \\
&\quad \geq \gamma_\phi\alpha\Delta\ell^\phi(s_k;x_k,\bar{\sigma}) + (1-\gamma_\phi)\alpha\epsilon - C_\phi\alpha^2\left\|s_k\right\|_2^2 \\
&\quad \geq \gamma_\phi\alpha\Delta\ell^\phi(s_k;x_k,\bar{\sigma}) \\
&\quad \geq \gamma_\phi\alpha\rho_k^\phi \ \text{ for all } k \in \mathcal{K}_P,
\end{aligned}
$$

which with (3.27) implies that (α, s_k) is a p-pair. Thus, Algorithm 1 must select an α_k that satisfies

$$
\alpha_k \geq \min\left\{\frac{\xi(1-\gamma_\phi)\epsilon}{C_\phi M_s^2}, 1\right\} =: \alpha_{\mathrm{P}}, \tag{3.52}
$$

where $\xi \in (0,1)$ is the backtracking parameter in Algorithm 1, which completes the proof of part (i).

We now prove part (ii). It follows from (3.12) that

$$
\Delta\ell^\phi(s_k;x_k,\bar{\sigma}) \geq \bar{\sigma}\eta_\sigma\Delta\ell^v(s_k^s;x_k) \geq \bar{\sigma}\eta_\sigma\epsilon \ \text{ for } k \in \mathcal{K}_F. \tag{3.53}
$$

If α satisfies

$$\alpha < \min\left[\frac{(1-\gamma_\phi)\bar{\sigma}\eta_\sigma\epsilon}{C_\phi M_s^2}, \frac{\eta_v\epsilon}{C_v M_s^2}\right], \tag{3.54}$$

where C_v is defined in (3.44) and η_v is defined in (3.8), then we may use (3.53) and proceed as in the proof of part (i) to conclude that (3.27) holds. Moreover, we have from Lemma 14, (3.8), (3.54), $\Delta\ell^v(s_k^s; x_k) \geq \epsilon$ for $k \in \mathcal{K}_F$, and (3.49) that

$$\begin{aligned} v(x_k + \alpha s_k) - v(x_k) &\leq -\alpha\Delta\ell^v(s_k; x_k) + \alpha^2 C_v \|s_k\|_2^2 \\ &< -\alpha\eta_v\Delta\ell^v(s_k^s; x_k) + \alpha\frac{\eta_v\epsilon}{C_v M_s^2} C_v \|s_k\|_2^2 \\ &\leq -\alpha\eta_v\epsilon + \alpha\eta_v\epsilon = 0 \ \text{ for all } k \in \mathcal{K}_F, \end{aligned} \tag{3.55}$$

where the strict inequality holds since $s_k \neq 0$ as a result of (3.53). Combining (3.55) with (3.27) implies that (α, s_k) is a b-pair. Thus, we conclude from the structure of Algorithm 1 that

$$\alpha_k \geq \min\left\{\frac{\xi(1-\gamma_\phi)\bar{\sigma}\eta_\sigma\epsilon}{C_\phi M_s^2}, \frac{\xi\eta_v\epsilon}{C_v M_s^2}, 1\right\} =: \alpha_{\mathrm{F}} > 0 \ \text{ for all } k \in \mathcal{K}_F, \tag{3.56}$$

where $\xi \in (0, 1)$ is the backtracking parameter used in Algorithm 1.

Part (iii) is a standard result used in continuous unconstrained optimization that follows since $\Delta\ell^f(s_k; x_k) \geq \epsilon$ is equivalent to $g(x_k)^T s_k \leq -\epsilon < 0$ and s_k is uniformly bounded by (3.49). □

The next lemma justifies the three cases that we consider when analyzing

Algorithm 1.

Lemma 20. *If Algorithm 1 does not terminate finitely, then one of the following scenarios occurs:*

Case 1: $k \in \mathcal{S}_p$ *for all* k *sufficiently large;*

Case 2: $k \in \mathcal{S}_o$ *for all* k *sufficiently large; or*

Case 3: $|\mathcal{S}_v \cup \mathcal{S}_b| = \infty$.

Proof. We proceed by contradiction and assume that none of the cases occur. In particular, since Case 3 does not hold it follows that $k \in \mathcal{S}_p \cup \mathcal{S}_o$ for all k sufficiently large. Combining this with the fact that Cases 1 and 2 do not hold implies that the iterates must oscillate between p- and o-iterates. However, this is not possible since there is no mechanism in Algorithm 1 that allows for iterate $k+1$ to be a p-iterate if iterate k is an o-iterate. □

We now analyze Algorithm 1 for each of the three possible scenarios stated in the previous result.

Case 1: $k \in \mathcal{S}_p$ for all k sufficiently large

In this case, there exists k'' such that

$$k \in \mathcal{S}_p \quad \text{for all } k \geq k'' \geq k', \tag{3.57}$$

where k' is defined in (3.49). We first show that our measure of feasibility converges to zero.

Lemma 21. *If Assumptions 3.3.1—3.3.4,* (3.49)*, and* (3.57) *hold, then*

$$\lim_{k\to\infty} \Delta\ell^v(s_k^s; x_k) = 0.$$

Proof. For a proof by contradiction, we suppose that there exists an infinite subsequence

$$\mathcal{S}'' := \{k \geq k'' : \Delta\ell^v(s_k^s; x_k) \geq \epsilon''\}$$

for some constant $\epsilon'' > 0$. It follows from (3.57), (3.12), (3.49), and the definition of $\mathcal{S}''$ that

$$\Delta\ell^\phi(s_k; x_k, \bar{\sigma}) \geq \bar{\sigma}\eta_\sigma \Delta\ell^v(s_k^s; x_k) \geq \bar{\sigma}\eta_\sigma\epsilon'' =: \epsilon > 0 \text{ for all } k \in \mathcal{S}'', \tag{3.58}$$

which implies with (3.57) that

$$\mathcal{S}'' \subseteq \mathcal{K}_P := \{k \in \mathcal{S}_p : k \geq k' \text{ and } \Delta\ell^\phi(s_k; x_k, \bar{\sigma}) \geq \epsilon > 0\}.$$

Combining $\mathcal{K}_P$ with Lemma 19 implies the existence of a positive α_{P} such that $\alpha_k \geq \alpha_{\text{P}} > 0$ for all $k \in \mathcal{S}''$, which used with the definitions of $\mathcal{S}''$ and $\mathcal{S}_p$, (3.57),

Lemma 18, and (3.58) yield

$$\begin{aligned}\phi(x_k;\bar{\sigma}) - \phi(x_k + \alpha_k \hat{s}_k;\bar{\sigma}) &\geq \gamma_\phi \alpha_k \rho_k^\phi \\ &\geq \gamma_\phi \alpha_k \min\left[C_\rho \Delta\ell^\phi(s_k;x_k,\bar{\sigma})^2,\ \tfrac{1}{2}\Delta\ell^\phi(s_k;x_k,\bar{\sigma})\right] \\ &\geq \gamma_\phi \alpha_{\mathrm{P}} \min\left[C_\rho \epsilon^2,\ \tfrac{1}{2}\epsilon\right] \\ &> 0 \ \text{ for all } k \in \mathcal{S}''. \end{aligned} \tag{3.59}$$

Now, for $k'' \leq k \in \mathcal{S}_p \setminus \mathcal{S}''$, it follows from (3.27), (3.12), and (3.21) that

$$\begin{aligned}\phi(x_k;\bar{\sigma}) - \phi(x_k + \alpha_k \hat{s}_k;\bar{\sigma}) &\geq \gamma_\phi \alpha_k \min\left[\Delta\ell^\phi(s_k;x_k,\bar{\sigma}),\ \Delta q^\phi(s_k^{c\phi};x_k,H_k,\bar{\sigma})\right] \\ &\geq 0. \end{aligned} \tag{3.60}$$

It now follows from (3.59), (3.60), and (3.57) that $\phi(x_k;\bar{\sigma}) \to -\infty$, which contradicts Assumptions 3.3.1 and 3.3.2. We conclude that $\lim_{k\to\infty} \Delta\ell^v(s_k^s;x_k) = 0$. □

The next result shows that all limit points are infeasible stationary points for problem (2.1).

Theorem 3. *Suppose that Assumptions 3.3.1—3.3.4,* (3.49)*, and* (3.57) *hold. If* x_* *is any limit point of the sequence* $\{x_k\}$ *generated by Algorithm 1, then* x_* *is an infeasible stationary point for problem* (2.1)*.*

Proof. Let $v_{\min} := \min\{v_j : (v_j, f_j) \in \mathcal{F}_{k''}\} \equiv \min\{v_j : (v_j, f_j) \in \mathcal{F}_k \ \text{ and } \ k \geq k''\}$, where the second equality holds since by assumption $k \in \mathcal{S}_p$ for all $k \geq k''$ and

the filter is never expanded when $k \in \mathcal{S}_p$. It follows from Lemma 9 that $v_{\min} > 0$. But then if there was a feasible limit point x_*, there must be iterates x_k, $k > k''$ that are arbitrarily close to feasibility, and thus ultimately one such that x_k is acceptable to $\mathcal{F}_k$. Thus line 22 of Algorithm 1 implies that there will be an iterate $k > k''$ for which $k \notin \mathcal{S}_p$ which contradicts (3.57). Thus, all limit points are infeasible. It follows from this fact, Lemma 21, and Lemma 9 that all limit points are infeasible stationary points. □

Importantly, the previous result shows that our algorithm will remain in penalty mode for all k sufficiently large only when all limit points are infeasible stationary points.

Case 2: $k \in \mathcal{S}_o$ for all k sufficiently large

In this case, there exists k'' such that

$$k \in \mathcal{S}_o \quad \text{for all } k \geq k'' \geq k', \tag{3.61}$$

where k' is defined in (3.49). We begin by showing that our feasibility measure converges to zero.

Lemma 22. *If Assumptions 3.3.1–3.3.4,* (3.49)*, and* (3.61) *hold, then*

$$\lim_{k \to \infty} \Delta \ell^v(s_k^s; x_k) = 0.$$

Proof. For a proof by contradiction, suppose that there exists $\epsilon'' > 0$ and an infinite subsequence

$$\mathcal{S}'' := \{k \geq k'' : \Delta\ell^v(s_k^s; x_k) \geq \epsilon''\} \subseteq \mathcal{S}_o,$$

where we have used k'' defined in (3.61). It then follows from the definition of $\mathcal{S}_o$, the o-pair $(\alpha_k, \hat{s}_k)$ selected in Algorithm 1, (3.24b), (3.49), Lemma 16, (3.24a), (3.8), and part (ii) of Lemma 19 that

$$\begin{aligned}
f(x_k) - f(x_k + \alpha_k \hat{s}_k) &\geq \gamma_f \alpha_k \rho_k^f \\
&\geq \gamma_f \alpha_k \min\left\{C_\rho \Delta\ell^f(s_k; x_k)^2,\ \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right\} \\
&\geq \gamma_f \alpha_k \min\left\{C_\rho \left[\gamma_v \Delta\ell^v(s_k; x_k)\right]^2,\ \tfrac{1}{2}\gamma_v \Delta\ell^v(s_k; x_k)\right\} \\
&\geq \gamma_f \alpha_k \min\left\{C_\rho \left[\gamma_v \eta_v \Delta\ell^v(s_k^s; x_k)\right]^2,\ \tfrac{1}{2}\gamma_v \eta_v \Delta\ell^v(s_k^s; x_k)\right\} \\
&\geq \gamma_f \alpha_{\mathrm{F}} \min\left\{C_\rho \left[\gamma_v \eta_v \epsilon''\right]^2,\ \tfrac{1}{2}\gamma_v \eta_v \epsilon''\right\} \quad \text{for all } k \in \mathcal{S}'',
\end{aligned}$$

for some $\alpha_F > 0$. Similarly, for $k'' \leq k \in \mathcal{S}_o \setminus \mathcal{S}''$, it follows from (3.24), (3.8), and (3.20) that

$$f(x_k) - f(x_k + \alpha_k \hat{s}_k) \geq \gamma_f \alpha_k \rho_k^f \geq \gamma_f \alpha_k \min\left\{\gamma_v \Delta\ell^v(s_k; x_k),\ \Delta q^f(s_k^{cf}; x_k, H_k)\right\} \geq 0.$$

Combining the two previous inequalities with the definition of k'' yields the limit $f(x_k) \to -\infty$, which contradicts the fact that f is bounded as a conse-

quence of Assumptions 3.3.1 and 3.3.2. This proves the result. □

We now show that feasible limit points are also first-order solutions of the penalty function.

Lemma 23. *Suppose that Assumptions 3.3.1–3.3.4,* (3.49)*, and* (3.61) *hold. If* $x_* = \lim_{k \in \mathcal{S}} x_k$ *for some subsequence* $\mathcal{S}$ *and* $v(x_*) = 0$*, then*

$$\lim_{k \in \mathcal{S}} \Delta q^{\phi}(s_k^p; x_k, B_k, \bar{\sigma}) = 0.$$

Proof. Suppose that there exists a constant $\epsilon'' > 0$ and an infinite subsequence

$$\mathcal{S}'' := \{k \in \mathcal{S} : k \geq k'' : \Delta q^{\phi}(s_k^p; x_k, B_k, \bar{\sigma}) \geq \epsilon''\},$$

where k'' is defined in (3.61). It follows from line 36 of Algorithm 1, (3.49), and (3.61) that

$$\Delta q^{\phi}(s_k; x_k, B_k, \bar{\sigma}) \geq \eta_{\phi} \Delta q^{\phi}(s_k^p; x_k, B_k, \bar{\sigma}) \geq \eta_{\phi} \epsilon'' \quad \text{for } k \in \mathcal{S}''. \tag{3.62}$$

From (2.9) and (3.8), we know that $v(x_k) \geq \Delta \ell^v(s_k; x_k) \geq \eta_v \Delta \ell^v(s_k^s; x_k) \geq 0$ for all k, which may be combined with $\lim_{k \in \mathcal{S}} v(x_k) = v(x_*) = 0$ (holds by assumption) to conclude that

$$\Delta \ell^v(s_k; x_k) \leq \frac{\eta_{\phi} \epsilon''}{\bar{\sigma} + \gamma_v} \quad \text{for } k \in \mathcal{S} \text{ sufficiently large,} \tag{3.63}$$

where $\gamma_v \in (0,1)$ is defined in (3.24a). It follows from (2.12), (2.13), (3.62), $B_k \succ 0$, and (3.63) that

$$\begin{aligned}\Delta\ell^f(s_k;x_k) &\geq \tfrac{1}{2}s_k^T B_k s_k - \bar{\sigma}\Delta\ell^v(s_k;x_k) + \eta_\phi\epsilon'' > \eta_\phi\epsilon'' - \bar{\sigma}\Delta\ell^v(s_k;x_k) \\ &\geq \eta_\phi\epsilon'' - \bar{\sigma}\frac{\eta_\phi\epsilon''}{\bar{\sigma}+\gamma_v} = \frac{\gamma_v\eta_\phi\epsilon''}{\bar{\sigma}+\gamma_v} =: \epsilon^f > 0 \quad \text{for } k \in \mathcal{S}'' \text{ sufficiently large.}\end{aligned} \tag{3.64}$$

Combining this with part (iii) of Lemma 19, we know that there exists some $\alpha_f > 0$ such that (α, s_k) satisfies (3.24b) for all $k \in \mathcal{S}''$ sufficiently large and $\alpha \in (0, \alpha_f]$, since by assumption $\mathcal{S}_o = \mathcal{S}_v \cup \mathcal{S}_o \cup \mathcal{S}_b$ for $k \geq k''$.

Next, we define

$$\Phi_k := \min_{(v_i,f_i)\in\mathcal{F}_k} \Big\{ \max\big[\, v_i - \alpha_i\eta_v\Delta\ell^v(s_i^s;x_i),\ \beta v_i \,\big] \Big\} > 0, \tag{3.65}$$

where $\mathcal{F}_k$ is the kth filter. The fact that $\Phi_k > 0$ follows since $v_i > 0$ for all $(v_i, f_i) \in \mathcal{F}_k$ as a consequence of Lemma 9. Moreover, it follows from (3.61) that $\mathcal{F}_k \equiv \mathcal{F}_{k''}$ for all $k \in \mathcal{S}''$ so that $\Phi_k \equiv \Phi_{k''} > 0$ for all $k \in \mathcal{S}''$. Now, pick $\epsilon^v > 0$ such that $\Phi_{k''} - C_v M_s^2 \leq \epsilon^v < \Phi_{k''}$ and consider α such that

$$0 < \alpha \leq \frac{\Phi_{k''} - \epsilon^v}{C_v M_s^2} \leq 1. \tag{3.66}$$

It then follows from Lemma 14, $\lim_{x\in\mathcal{S}} v(x_k) = 0$, (3.8), (3.61), and (3.66), that

$$\begin{aligned} v(x_k + \alpha s_k) &\leq v(x_k) - \alpha\Delta\ell^v(s_k; x_k) + \alpha^2 C_v \left\|s_k\right\|_2^2 \\ &\leq \epsilon^v + \alpha^2 C_v M_s^2 \\ &\leq \epsilon^v + \alpha C_v M_s^2 \\ &\leq \epsilon^v + \frac{\Phi_{k''} - \epsilon^v}{C_v M_s^2} C_v M_s^2 \\ &= \Phi_{k''} \quad \text{for all } k \in \mathcal{S}'' \text{ sufficiently large.} \end{aligned}$$

Thus, $x_k + \alpha s_k$ is acceptable to $\mathcal{F}_k \equiv \mathcal{F}_{k''}$ for all α satisfying (3.66) and $k \in \mathcal{S}''$ sufficiently large.

The above, (3.61), and the structure of Algorithm 1, shows that

$$\alpha_k \geq \min\left\{\xi\frac{\Phi_{k''} - \epsilon^v}{C_v M_s^2}, \xi\alpha_f, 1\right\} =: \alpha_{\min} > 0 \text{ for all } k \in \mathcal{S}'' \text{ sufficiently large,} \tag{3.67}$$

where $\xi \in (0, 1)$ is the backtracking parameter used in Algorithm 1. It then follows from (3.61), (3.24b), (3.8), Lemma 16, (3.24a), (3.67), and (3.64) that

$$\begin{aligned} f(x_k) - f(x_k + \alpha_k \hat{s}_k) &\geq \gamma_f \alpha_k \rho_k^f \\ &\geq \gamma_f \alpha_k \min\left[C_\rho \Delta\ell^f(s_k; x_k)^2,\ \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right] \\ &\geq \gamma_f \alpha_{\min} \min\left[C_\rho\left(\epsilon^f\right)^2,\ \tfrac{1}{2}\epsilon^f\right] \\ &> 0 \text{ for all } k \in \mathcal{S}'' \text{ sufficiently large.} \end{aligned} \tag{3.68}$$

However, for *all* $k \in \mathcal{S}_o$, it follows from (3.24b), (3.24a), (3.8), and (3.20) that $f(x_k) - f(x_k + \alpha_k \hat{s}_k) \geq 0$. This observation combined with (3.68) implies that $\lim_{k \to \infty} f(x_k) = -\infty$, which contradicts the fact that f is bounded as a consequence of Assumptions 3.3.1 and 3.3.2. This completes the proof. □

We now show that limit points are either infeasible stationary points or KKT points for problem (2.1).

Theorem 4. *Suppose that Assumptions 3.3.1–3.3.4,* (3.49)*, and* (3.61) *hold. If* x_* *is a limit point of* $\{x_k\}$*, then either*

(i) x_* *is an infeasible stationary point; or*

(ii) x_* *is a KKT point for problem* (2.1)*.*

Proof. Suppose that $\lim_{k \in \mathcal{S}} x_k = x_*$ for some subsequence $\mathcal{S}$. It follows from Lemma 22 that $\lim_{k \to \infty} \Delta \ell^v(s_k^s; x_k) = 0$ so that if $v(x_*) > 0$, then x_* is an infeasible stationary point (see Definition 9). Otherwise, we have that $v(x_*) = 0$. In this case, it follows from Lemma 23 and (3.49) that $\lim_{k \in \mathcal{S}} \Delta q^\phi(s_k^p; x_k, B_k, \bar{\sigma}) = 0$. It follows from this fact, $v(x_*) = 0$, and Lemma 1 that x_* is a KKT point for problem (2.1). □

Case 3: $|\mathcal{S}_v \cup \mathcal{S}_b| = \infty$

The next result shows that if $\mathcal{P}\text{-}mode$ = false at the beginning of the kth iteration, then x_k is acceptable to the filter $\mathcal{F}_k$.

Lemma 24. *If $\mathcal{P}$-$mode =$* false *at the beginning of iteration k, then x_k is acceptable to $\mathcal{F}_k$.*

Proof. The result immediately follows from the construction of Algorithm 1 and consideration of the possible outcomes associated with iteration $k-1$. □

We first show that the feasibility measure converges to zero along $\mathcal{S}_v \cup \mathcal{S}_b$.

Lemma 25. *If Assumptions 3.3.1–3.3.3 hold and $|\mathcal{S}_v \cup \mathcal{S}_b| = \infty$, then*

$$\lim_{k \in \mathcal{S}_v \cup \mathcal{S}_b} \Delta\ell^v(s_k^s; x_k) = 0.$$

Proof. To reach a contradiction, suppose that we have the infinite subsequence

$$\mathcal{S} := \{k \in \mathcal{S}_v \cup \mathcal{S}_b : \Delta\ell^v(s_k^s; x_k) \geq \epsilon\}$$

for some constant $\epsilon > 0$. It follows from the definition of $\mathcal{S}$, Lemma 24 and (3.22) that

$$v_k \leq \max\left\{v_j - \alpha_j \eta_v \Delta\ell^v(s_j^s; x_j),\ \beta v_j\right\}$$

or (3.69)

$$f_k \leq f_j - \gamma \min\left\{v_j - \alpha_j \eta_v \Delta\ell^v(s_j^s; x_j),\ \beta v_j\right\}$$

for $k \in \mathcal{S}$ and $(v_j, f_j) \in \mathcal{F}_k$; note that by construction $(v_k, f_k) \in \mathcal{F}_{k+1}$ for all $k \in \mathcal{S}$.

Moreover, it follows from the definitions of $\Delta\ell^v$ and $\mathcal{S}$ that $v_k \geq \Delta\ell^v(s_k^s; x_k) \geq \epsilon$ for $k \in \mathcal{S}$. Using Assumptions 3.3.1 and 3.3.2 we have a subsequence $\mathcal{S}' \subseteq \mathcal{S}$ satisfying the following limits:

$$\lim_{k\in\mathcal{S}'} \Delta\ell^v(s_k^s; x_k) = \theta_\ell \quad \text{and} \quad \lim_{k\in\mathcal{S}'} v_k = \theta_v \quad \text{for constants } \theta_v \geq \theta_\ell \geq \epsilon > 0.$$

For *any* $\epsilon_\ell \in (0, \theta_\ell)$ and $\epsilon_v \in (0, \theta_v)$, it follows that

$$|\Delta\ell^v(s_k^s; x_k) - \theta_\ell| < \epsilon_\ell \quad \text{and} \quad |v_k - \theta_v| < \epsilon_v \text{ for all } k \in \mathcal{S}' \subseteq \mathcal{S} \text{ sufficiently large.} \tag{3.70}$$

Using (3.70), the definitions of ϵ_ℓ, η_v, $\Delta\ell^v$ and $\mathcal{S}$, $\alpha_k \in (0, 1]$, $\mathcal{S}' \subseteq \mathcal{S}$, and part (ii) of Lemma 19 gives

$$0 \leq v_k - \alpha_k\eta_v\Delta\ell^v(s_k^s; x_k) < v_k - \alpha_{\text{F}}\eta_v(\theta_\ell - \epsilon_\ell) \leq \beta_2 v_k \text{ for all } k \in \mathcal{S}' \text{ sufficiently large} \tag{3.71}$$

and some $\alpha_{\text{F}} > 0$, where

$$\beta_2 := \frac{(\theta_v + \epsilon_v) - \alpha_{\text{F}}\eta_v(\theta_\ell - \epsilon_\ell)}{(\theta_v + \epsilon_v)} \in (0, 1)$$

and β_2 may be forced to lie in $(0, 1)$ by choosing ϵ_v sufficiently close to zero and

ϵ_ℓ sufficiently close to θ_ℓ. Now define $\beta^* := \max\{\beta_2, \beta\} \in (0,1)$,

$$\epsilon^* = \min\left\{\frac{1-\beta^*}{1+\beta^*}\theta_v,\ \epsilon_v\right\} > 0,$$

and the subsequence $\mathcal{S}'' = \{k \in \mathcal{S}' : |v_k - \theta_v| < \epsilon^*\}$ so that

$$\frac{2\beta^*}{1+\beta^*}\theta_v < v_k < \frac{2}{1+\beta^*}\theta_v \ \text{ for all } k \in \mathcal{S}'' \subseteq \mathcal{S}' \text{ sufficiently large.} \tag{3.72}$$

Given $k \in \mathcal{S}''$, define $k^+ \in \mathcal{S}''$ to be the successor to k in $\mathcal{S}''$. It then follows from (3.72), the definition of β^*, and (3.71) that

$$\begin{aligned} v_{k^+} &> \frac{2\beta^*}{1+\beta^*}\theta_v \\ &> \beta^* v_k \\ &= \max\{\beta_2, \beta\} v_k \\ &\geq \max\left\{v_k - \alpha_k \eta_v \Delta\ell^v(s_k^s; x_k),\ \beta v_k\right\} \ \text{ for all } k \in \mathcal{S}''. \end{aligned}$$

Since $\mathcal{S}'' \subseteq \mathcal{S}' \subseteq \mathcal{S}$, it follows from the previous inequality, the definition of k^+, the fact that $(v_k, f_k) \in \mathcal{F}_{k^+}$, (3.69), the definition of $\Delta\ell^v(s_k^s; x_k)$, $\alpha_k \in (0,1]$,

$\eta_v \in (0,1)$, $\beta \in (0,1)$, $\gamma \in (0,1)$, $\theta_v > \epsilon_v \geq \epsilon^*$ and the definition of $\mathcal{S}''$ that

$$\begin{aligned} f_k - f_{k+} &\geq \gamma \min\left\{v_k - \alpha_k \eta_v \Delta \ell^v(s_k^s; x_k),\ \beta v_k\right\} \\ &= \gamma \min\left\{(1-\alpha_k\eta_v)v_k + \alpha_k\eta_v \left\|[c(x_k) + J(x_k)s_k^s]^-\right\|_1,\ \beta v_k\right\} \\ &\geq \gamma \min\{1-\alpha_k\eta_v, \beta\} v_k \geq \gamma \min\{1-\eta_v, \beta\}(\theta_v - \epsilon^*) > 0 \ \text{ for all } k \text{ in } \mathcal{S}''. \end{aligned}$$

Summing over $k \in \mathcal{S}''$, we deduce that $\{f_k\}_{k\in\mathcal{S}''} \to -\infty$, which contradicts Assumptions 3.3.1 and 3.3.2. □

We now prove that our optimality measure for ϕ converges to zero along a certain subsequence.

Lemma 26. *Suppose that Assumptions 3.3.1–3.3.4 and* (3.49) *hold, and that* $|\mathcal{S}_v \cup \mathcal{S}_b| = \infty$. *The following then hold.*

(i) *If* $|\mathcal{S}_v| = \infty$ *and* $\lim_{k\in\mathcal{S}_v} x_k = x_*$ *for some* x_* *satisfying* $v(x_*) = 0$, *then*

$$\lim_{k\in\mathcal{S}_v} \Delta q^\phi(s_k^p; x_k, B_k, \bar{\sigma}) = 0.$$

(ii) *If* $|\mathcal{S}_v| < \infty$ *and* $\lim_{k\in\mathcal{S}_b} x_k = x_*$ *for some* x_* *satisfying* $v(x_*) = 0$, *then*

$$\liminf_{k\in\mathcal{S}_b} \Delta q^\phi(s_k^p; x_k, B_k, \bar{\sigma}) = 0.$$

Proof. We first prove part (i). To obtain a contradiction, suppose that there

exists the subsequence

$$\mathcal{S}' := \{k \in \mathcal{S}_v : k \geq k' \text{ and } \Delta q^{\phi}(s_k^p; x_k, B_k, \bar{\sigma}) \geq \epsilon'\}$$

for some constant $\epsilon' > 0$ and k' defined in (3.49). It then follows from line 36 of Algorithm 1 that

$$\Delta q^{\phi}(s_k; x_k, B_k, \bar{\sigma}) \geq \eta_{\phi} \Delta q^{\phi}(s_k^p; x_k, B_k, \bar{\sigma}) \geq \eta_{\phi} \epsilon' \text{ for } k \in \mathcal{S}'. \tag{3.73}$$

Then, since $v(x_*) = 0$ by assumption, we may use (3.73) (analogous to (3.62)) and follow the same steps that led to (3.64) to show that

$$\Delta \ell^f(s_k; x_k) \geq \epsilon^f \geq \gamma_v \Delta \ell^v(s_k; x_k) \text{ for } k \in \mathcal{S}' \text{ sufficiently large and some } \epsilon^f > 0,$$

where the second inequality follows from $\lim_{k \in \mathcal{S}_v} x_k = x_*$, $v(x_*) = 0$, and the definition of $\Delta \ell^v$. Thus, (3.23) does not hold and implies that $k \notin \mathcal{S}_v$. This is a contradiction and proves part (i).

We now prove part (ii), where $|\mathcal{S}_v| < \infty = |\mathcal{S}_b|$. To obtain a contradiction, suppose that

$$\Delta q^{\phi}(s_k^p; x_k, B_k, \bar{\sigma}) \geq \epsilon' \text{ for } k \in \mathcal{S}_b \text{ sufficiently large}$$

and some constant $\epsilon' > 0$. It then follows from line 36 of Algorithm 1 that

$$\Delta q^{\phi}(s_k; x_k, B_k, \bar{\sigma}) \geq \eta_{\phi} \Delta q^{\phi}(s_k^p; x_k, B_k, \bar{\sigma}) \geq \eta_{\phi} \epsilon' \text{ for } k \in \mathcal{S}_b \text{ sufficiently large.} \tag{3.74}$$

Since (3.74) is analogous to (3.73), we may again conclude as above that

$$\Delta \ell^f(s_k; x_k) \geq \epsilon^f \geq \gamma_v \Delta \ell^v(s_k; x_k) \text{ for } k \in \mathcal{S}_b \text{ sufficiently large and some } \epsilon^f > 0. \tag{3.75}$$

Using (3.75), $\lim_{k \in \mathcal{S}_b} v(x_k) = v(x_*) = 0$, and part (iii) of Lemma 19, we may conclude that there exists $\alpha_f > 0$ such that (α, s_k) satisfies (3.24b) for all $\alpha \in (0, \alpha_f]$ and $k \in \mathcal{S}_b$ sufficiently large. Now, if $\alpha_k \to 0$ along some subsequence $\mathcal{S}_b' \subseteq \mathcal{S}_b$, then it follows from the previous sentence and (3.75) that (α_k, s_k) satisfies (3.24a) and (3.24b) for all $k \in \mathcal{S}_b$ sufficiently large. We now show that $x_k + \alpha_k s_k$ is also acceptable to the filter $\mathcal{F}_k$ for all $k \in \mathcal{S}_b'$ sufficiently large.

To this end, let $(v_i, f_i) \in \mathcal{F}_k$ for some $k \in \mathcal{S}_b'$. It then follows from Lemma 24 that either $v_k \leq \max\{v_i - \alpha_i \eta_v \Delta \ell^v(s_i^s; x_i),\ \beta v_i\}$ or $f_k \leq f_i - \gamma \min\{v_i - \alpha_i \eta_v \Delta \ell^v(s_i^s; x_i),\ \beta v_i\}$. In this first case, it follows from the definition of a b-pair that $v(x_k + \alpha_k s_k) \leq v_k \leq \max\{v_i - \alpha_i \eta_v \Delta \ell^v(s_i^s; x_i), \beta v_i\}$ for all $k \in \mathcal{S}_b$. In the second case, we have from the fact that (3.24b) holds for $k \in \mathcal{S}_b'$ sufficiently large (recall that $\alpha_k \to 0$ on $\mathcal{S}_b'$), (3.75), and Lemma 16 that $f(x_k + \alpha_k s_k) \leq f_k \leq f_i - \gamma \min\left\{v_i - \alpha_i \eta_v \Delta \ell^v(s_i^s; x_i),\ \beta v_i\right\}$. Thus, in either case we have that $\big(v(x_k + \alpha_k s_k), f(x_k + \alpha_k s_k)\big)$

is acceptable to the single element filter $\{(v_i, f_i)\}$ for all $k \in \mathcal{S}_b'$ sufficiently large. Since this element (v_i, f_i) of the filter $\mathcal{F}_k$ was arbitrary, we may conclude that $\big(v(x_k + \alpha_k s_k), f(x_k + \alpha_k s_k)\big)$ is, in fact, acceptable to the filter $\mathcal{F}_k$ for all $k \in \mathcal{S}_b'$ sufficiently large.

To summarize, we have shown that (α_k, s_k) is an o-pair for $k \in \mathcal{S}_b'$ sufficiently large. This is a contradiction since Algorithm 1 would have labeled such an iterate as an o-iterate, not a b-iterate. Thus, there exists α_b such that $\alpha_k \geq \alpha_b > 0$ for all $k \in \mathcal{S}_b$ sufficiently large. Combining this with (2.3), (3.27), $v(\cdot) \geq 0$, Lemma 18, (2.12), (3.8), and (3.75) gives

$$
\begin{aligned}
&f(x_k) - f(x_k + \alpha_k \hat{s}_k) \\
&\quad = \phi(x_k; \bar{\sigma}) - \phi(x_k + \alpha_k \hat{s}_k; \bar{\sigma}) - \bar{\sigma}\big(v(x_k) - v(x_k + \alpha_k \hat{s})\big) \\
&\quad \geq \gamma_\phi \alpha_k \rho_k^\phi - \bar{\sigma} v(x_k) \\
&\quad \geq \gamma_\phi \alpha_b \min\left[C_\rho \Delta\ell^\phi(s_k; x_k, \bar{\sigma})^2,\ \tfrac{1}{2}\Delta\ell^\phi(s_k; x_k, \bar{\sigma})\right] - \bar{\sigma} v(x_k) \\
&\quad = \gamma_\phi \alpha_b \min\Big[C_\rho\big(\Delta\ell^f(s_k; x_k)^2 + 2\bar{\sigma}\Delta\ell^f(s_k; x_k)\Delta\ell^v(s_k; x_k) + \bar{\sigma}^2\Delta\ell^v(s_k; x_k)^2\big), \\
&\qquad\qquad \tfrac{1}{2}\big(\Delta\ell^f(s_k; x_k) + \bar{\sigma}\Delta\ell^v(s_k; x_k)\big)\Big] - \bar{\sigma} v(x_k) \\
&\quad \geq \gamma_\phi \alpha_b \min\left[C_\rho \Delta\ell^f(s_k; x_k)^2,\ \tfrac{1}{2}\Delta\ell^f(s_k; x_k)\right] - \bar{\sigma} v(x_k) \\
&\quad \geq \gamma_\phi \alpha_b \min\left[C_\rho\big(\epsilon^f\big)^2,\ \tfrac{1}{2}\epsilon^f\right] - \bar{\sigma} v(x_k) \ \text{ for } k \in \mathcal{S}_b \text{ sufficiently large.} \qquad (3.76)
\end{aligned}
$$

Since $|\mathcal{S}_b| = \infty > |\mathcal{S}_v|$, we may define k^+ as the first iteration greater than k such that $k^+ \in \mathcal{S}_b \cup \mathcal{S}_o$. It then follows from the construction of Algorithm 1 and

$|\mathcal{S}_v| < \infty$ that the following is a valid statement:

if $k \in \mathcal{S}_b$ is sufficiently large, then $k^+ \in \mathcal{S}_b \cup \mathcal{S}_o$ and $l \in \mathcal{S}_p$ for all $k < l < k^+$.

Using (3.27), $\alpha_i \geq 0$, (3.12), and (3.21) we conclude that

$$\begin{aligned}\phi(x_{k+1};\bar{\sigma}) - \phi(x_{k^+};\bar{\sigma}) &= \sum_{i=k+1}^{k^+-1} \phi(x_i;\bar{\sigma}) - \phi(x_i + \alpha_i \hat{s}_i;\bar{\sigma}) \geq \sum_{i=k+1}^{k^+-1} \gamma_\phi \alpha_i \rho_i^\phi \\ &= \sum_{i=k+1}^{k^+-1} \gamma_\phi \alpha_i \min\Big[\, \Delta\ell^\phi(s_i; x_i, \bar{\sigma}),\, \Delta q^\phi(s_i^{c\phi}; x_i, H_i, \bar{\sigma}) \,\Big] \\ &\geq 0 \;\; \text{for } k \in \mathcal{S}_b \text{ sufficiently large,}\end{aligned}$$

which may be combined with (2.3), $v(\cdot) \geq 0$, and (3.26) to conclude that

$$f(x_{k+1}) - f(x_{k^+}) \geq \bar{\sigma}\big(v(x_{k^+}) - v(x_{k+1})\big) \geq -\bar{\sigma} v(x_{k+1}) > -\bar{\sigma} v(x_k) \tag{3.77}$$

for $k \in \mathcal{S}_b$ sufficiently large. It then follows from (3.76) and (3.77) that

$$\begin{aligned}f(x_k) - f(x_{k^+}) &= \big(f(x_k) - f(x_k + \alpha_k \hat{s}_k)\big) + \big(f(x_{k+1}) - f(x_{k^+})\big) \\ &> \gamma_\phi \alpha_b \min\Big[C_\rho\big(\epsilon^f\big)^2,\, \tfrac{1}{2}\epsilon^f\Big] - 2\bar{\sigma} v(x_k) \;\; \text{for } k \in \mathcal{S}_b \text{ sufficiently large.}\end{aligned} \tag{3.78}$$

Next, since $\lim_{k\in\mathcal{S}_b} v(x_k) = 0$ we know that

$$v(x_k) \leq \frac{1}{4\bar{\sigma}}\gamma_\phi\alpha_b \min\left[C_\rho\big(\epsilon^f\big)^2,\ \tfrac{1}{2}\epsilon^f\right] \quad \text{for } k \in \mathcal{S}_b \text{ sufficiently large,}$$

which may be combined with (3.78) to deduce that

$$f(x_k) - f(x_{k^+}) > \tfrac{1}{2}\gamma_\phi\alpha_b \min\left[C_\rho\big(\epsilon^f\big)^2,\ \tfrac{1}{2}\epsilon^f\right] =: \epsilon^v > 0 \quad \text{for } k \in \mathcal{S}_b \text{ sufficiently large.} \tag{3.79}$$

If we define $\hat{k}^+$ to be the first b-iteration greater than k (thus, $\hat{k}^+ \geq k^+$), it follows from (3.79), the fact that Algorithm 1 does not allow further p-iterations until it has its next b-iteration, and the fact that the objective f is decreased during o-iterations that $f(x_k) - f(x_{\hat{k}^+}) > \epsilon^v$ for $k \in \mathcal{S}_b$ sufficiently large. Since $|\mathcal{S}_b| = \infty$, this implies that $f(x_k) \to -\infty$, which contradicts the fact that f is bounded as a consequence of Assumptions 3.3.1 and 3.3.2. □

We now show that limit points of $\{x_k\}_{\mathcal{S}_v\cup\mathcal{S}_b}$ are infeasible stationary or KKT point for problem (2.1).

Theorem 5. *Suppose that the Assumptions 3.3.1–3.3.4,* (3.49)*, and* $|\mathcal{S}_v\cup\mathcal{S}_b| = \infty$ *hold. Then, there exists a limit point* x_* *of* $\{x_k\}_{\mathcal{S}_v\cup\mathcal{S}_b}$ *such that either*

(i) x_* *is a KKT point of problem* (2.1) *or*

(ii) x_* *is an infeasible stationary point.*

Proof. From Assumptions 3.3.1 and 3.3.2 we know that there exists a limit point x_* of $\{x_k\}_{\mathcal{S}_v \cup \mathcal{S}_b}$. First, if $v(x_*) > 0$, then it follows from Lemma 25 and Lemma 9 that x_* is an infeasible stationary point, which is part (ii) of this theorem. Second, if $v(x_*) = 0$ and $|\mathcal{S}_v| = \infty$, then it follows from part (i) of Lemma 26 and Lemma 1 that x_* is a KKT point of problem (2.1). This is case (i) of this theorem. Finally, if $v(x_*) = 0$ and $|\mathcal{S}_v| < \infty$ (so that $|\mathcal{S}_b| = \infty$), then it follows from part (ii) of Lemma 26 and Lemma 1 that x_* is a KKT point of problem (2.1), which once again is case (i) of the theorem. □

3.3.2 Analysis: unbounded weighting parameter

We now consider the situation when the weighting parameter increases without bound, i.e, that

$$\lim_{k \to \infty} \sigma_k = \infty. \tag{3.80}$$

We begin with the following lemma, which is similar to [71, Lemma 3.8].

Lemma 27. *Suppose that Assumptions 3.3.1–3.3.4 are satisfied,* (3.80) *holds, x_* is a limit point of $\{x_k\}$ satisfying $v(x_*) > 0$, and $\Delta \ell^v(s_*^s; x_*) > 0$, where s_*^s is the solution to*

$$\underset{(s,r) \in \mathbb{R}^{n+m}}{\text{minimize}} \; e^T r \quad \text{subject to} \quad c(x_*) + J(x_*)s + r \geq 0, \;\; r \geq 0, \;\; \|s\|_\infty \leq \delta,$$

for some $\delta \in [\delta_{min}, \delta_{max}]$. *Then, along any subsequence* $\{x_k\}_{k\in\mathcal{K}}$ *that converges to* x_*, *the weighting parameter is updated only a finite number of times.*

Proof. We begin by defining

$$s_k^\phi(\sigma) := \operatorname*{argmin}_{s\in\mathbb{R}^n} \; q^\phi(s; x_k, B_k, \sigma) \tag{3.81}$$

and

$$\mu := \mu(\sigma) := \left(1 - \frac{\eta_\sigma}{\eta_v}\right)\sigma < \sigma, \tag{3.82}$$

where we used the fact that $0 < 1 - \eta_\sigma/\eta_v < 1$ holds since $0 < \eta_\sigma < \eta_v < 1$ is defined in (3.11).

Using the fact that $\Delta q^\phi(s_k^\phi(\mu); x_k, B_k, \mu) \geq 0$, (2.13), and the definition of $\mu = \mu(\sigma)$, we can see that

$$\begin{aligned}
\Delta q^\phi(s_k^\phi(\mu); x_k, B_k, \mu) &= \Delta q^f(s_k^\phi(\mu); x_k, B_k) + \left(1 - \frac{\eta_\sigma}{\eta_v}\right)\sigma\Delta\ell^v(s_k^\phi(\mu); x_k) \\
&= \Delta q^\phi(s_k^\phi(\mu); x_k, B_k, \sigma) - \frac{\eta_\sigma}{\eta_v}\sigma\Delta\ell^v(s_k^\phi(\mu); x_k) \\
&\geq 0 \;\text{ for } \mu = \mu(\sigma) \text{ and all } \sigma > 0,
\end{aligned}$$

which implies that

$$\Delta q^\phi(s_k^\phi(\mu); x_k, B_k, \sigma) \geq \frac{\eta_\sigma}{\eta_v}\sigma\Delta\ell^v(s_k^\phi(\mu); x_k) \;\text{ for } \mu = \mu(\sigma) \text{ and all } \sigma > 0. \tag{3.83}$$

Since $\Delta\ell^v(s_*^s; x_*) > 0$ and $\lim_{k\in\mathcal{K}} x_k = x_*$ by assumption, it follows from [73, Theorem 3.2.8] that there exists $\epsilon \in (0,1)$ and k' such that

$$\Delta\ell^v(s_k^s; x_k) > \epsilon \text{ for all } k' \leq k \in \mathcal{K}. \tag{3.84}$$

Moreover, since the Newton step $-B_k^{-1}g_k$ minimizes $q^f(s; x_k, B_k)$, it follows from Assumption 3.3.3 that

$$\begin{aligned} q^f(s_k^\phi(\sigma); x_k, B_k) &\geq q^f(-B_k^{-1}g_k; x_k, B_k) \\ &= f_k - \tfrac{1}{2}{g_k}^T B_k^{-1} g_k \\ &\geq f_k - \frac{\|g_k\|_2^2}{2\lambda_{\min}} \text{ for all } \sigma > 0. \end{aligned} \tag{3.85}$$

Next, it follows from (3.1), the choice $\delta_k \in [\delta_{\min}, \delta_{\max}]$, norm inequalities, and Assumption 3.3.3 that

$$q^f(s_k^s; x_k, B_k) \leq f_k + \|g_k\|_2 \delta_{\max} + \tfrac{1}{2}\lambda_{\max}\delta_{\max}^2. \tag{3.86}$$

Then, (3.85), (3.86), and Assumptions 3.3.1 and 3.3.2 imply the existence of a constant $C_{\text{qf}} > 0$ such that

$$q^f(s_k^s; x_k, B_k) - q^f(s_k^\phi(\sigma); x_k, B_k) \leq C_{\text{qf}} \text{ for all } \sigma > 0. \tag{3.87}$$

We now define

$$\sigma_{\text{crit}} := \frac{C_{\text{qf}}}{\epsilon(1-\eta_v)\left(1-\frac{\eta_\sigma}{\eta_v}\right)} > \mu\left(\sigma_{\text{crit}}\right) = \frac{C_{\text{qf}}}{\epsilon(1-\eta_v)} > 0, \tag{3.88}$$

and the associated infinite subsequence

$$\mathcal{S}' = \{k \in \mathcal{K} : k \geq k' \text{ and } \sigma_k \geq \sigma_{\text{crit}}\}. \tag{3.89}$$

It follows from the fact that $\Delta q^\phi(s_k^\phi(\sigma); x_k, B_k, \sigma) \geq \Delta q^\phi(s_k^s; x_k, B_k, \sigma)$ (by the definition of $s_k^\phi(\sigma)$), (2.13), (3.87), (3.84), and (3.88) that

$$\begin{aligned}
\Delta\ell^v(s_k^\phi(\sigma); x_k) &\geq \Delta\ell^v(s_k^s; x_k) - \frac{1}{\sigma}\left(q^f(s_k^s; x_k, B_k) - q^f(s_k^\phi(\sigma); x_k, B_k)\right) \\
&\geq \Delta\ell^v(s_k^s; x_k) - \frac{1}{\sigma}C_{\text{qf}} = \Delta\ell^v(s_k^s; x_k)\left(1 - \frac{C_{\text{qf}}}{\sigma\Delta\ell^v(s_k^s; x_k)}\right) \\
&\geq \eta_v\Delta\ell^v(s_k^s; x_k) \text{ for } \sigma \geq \mu(\sigma_{\text{crit}}) \text{ and } k' \leq k \in \mathcal{K}.
\end{aligned} \tag{3.90}$$

We may now use the definition of $\mathcal{S}'$, (3.90), the fact that $s_k^\phi(\sigma_k) \equiv s_k^p$, (3.7), and (3.8) to show that

$$\begin{gathered}
\tau_k = 1, \quad s_k = s_k^p, \\
\text{and} \\
\Delta q^\phi(s_k; x_k, B_k, \sigma_{k+1}) \geq \eta_\phi\Delta q^\phi(s_k^p; x_k, B_k, \sigma_{k+1}) \text{ for } k \in \mathcal{S}'
\end{gathered} \tag{3.91}$$

since $\eta_\phi \in (0,1)$ in Algorithm 1. Next, it follows from (3.91), (2.13), $B_k \succ 0$, the fact that $s_k^p \equiv s_k^\phi(\sigma_k)$ and s_k^p minimizes $q^\phi(s; x_k, B_k, \sigma_k)$, (3.83), the fact that $\mu(\sigma_k) \geq \mu(\sigma_{\text{crit}})$ for $k \in \mathcal{S}'$, and (3.90) that

$$\begin{aligned}
\Delta\ell^\phi(s_k; x_k, \sigma_k) &\geq \Delta q^\phi(s_k^p; x_k, B_k, \sigma_k) \\
&\geq \Delta q^\phi(s_k^\phi\big(\mu(\sigma_k)\big); x_k, B_k, \sigma_k) \\
&\geq \frac{\eta_\sigma}{\eta_v}\sigma_k \Delta\ell^v(s_k^\phi\big(\mu(\sigma_k)\big); x_k) \\
&\geq \frac{\eta_\sigma}{\eta_v}\sigma_k\big(\eta_v \Delta\ell^v(s_k^s; x_k)\big) \\
&= \sigma_k \eta_\sigma \Delta\ell^v(s_k^s; x_k) \;\; \text{for } k \in \mathcal{S}'. \qquad (3.92)
\end{aligned}$$

We now conclude from (3.91), (3.92), (3.11), and the fact that the weighting parameter is only increased in lines 13 and 36 of Algorithm 1, that σ_k is increased a finite number of times on $\mathcal{K}$. □

We now consider feasible limit points at which the MFCQ [74] holds.

Lemma 28. *Suppose that Assumptions 3.3.1–3.3.4 are satisfied,* (3.80) *holds, x_* is a limit point of $\{x_k\}$ at which $v(x_*) = 0$ and the MFCQ holds. Then, the following hold for all x_k sufficiently close to x_* and σ_k sufficiently large:* (i) $\Delta\ell^v(s_k^p; x_k) = v(x_k)$*;* (ii) $s_k = s_k^p$*; and* (iii) σ_k *is not increased during iteration k.*

Proof. We may use [71, Lemmas 3.12 and 3.13] since the proofs only used the properties of the MFCQ, the continuity of the problem functions f and g, and

the convexity of their penalty and steering subproblems. Their subproblem [71, Equations 2.7(a–d)] is equivalent to our predictor subproblem (3.4) and both methods minimize the same quadratic model of the penalty function. A small difference is that our predictor subproblem is designed so that if $\ell^v(s_k^s; x_k) = 0$, then $\ell^v(s_k^p; x_k) = 0$ as well; they satisfy this requirement by increasing their penalty parameter in Step 4a [71, Eqn 2.11] and re-solving for a new step. Their steering subproblem [71, Equations 2.9(a–e)] is equivalent to (3.1).

The assumptions of this lemma and [71, Lemma 3.12] imply the existence of $r > 0$ and $k' \geq 0$ so that

$$\ell^v(s_k^p, x_k) = v(x_k) \text{ for all } k \in \mathcal{S}', \tag{3.93}$$

where $\mathcal{S}' := \{k : \|x_k - x_*\| \leq r \quad \text{and} \quad k \geq k'\}$, which proves part (i). The inequality $\Delta\ell^v(s_k^s; x_k) \geq 0$, (3.93), and the definition of $\Delta\ell^v$ imply

$$\Delta\ell^v(s_k^p; x_k) \geq v(x_k) - \ell^v(s_k^s; x_k) = \Delta\ell^v(s_k^s; x_k) \geq \eta_v \Delta\ell^v(s_k^s; x_k) \text{ for } k \in \mathcal{S}',$$

where $\eta_v \in (0, 1)$ is defined in (3.8). Thus, we conclude from (3.8) that $\tau_k = 1$ and $s_k = s_k^p$ for $k \in \mathcal{S}'$, which proves part (ii). Finally, it follows from [71, Lemma 3.13] and the assumptions of this lemma, that

$$\Delta q^\phi(s_k^p; x_k, B_k, \sigma_k) \geq \sigma_k \eta_\sigma v(x_k) \geq \sigma_k \eta_\sigma \Delta\ell^v(s_k^s; x_k) \text{ for } k \in \mathcal{S}', \tag{3.94}$$

where the last inequality follows from the definition of $\Delta\ell^v$. It then follows from part (ii) of this lemma, (2.13), $B_k \succ 0$, and (3.94) that for $k \in \mathcal{S}'$,

$$\Delta\ell^\phi(s_k; x_k, \sigma_k) = \Delta\ell^\phi(s_k^p; x_k, \sigma_k) \geq \Delta q^\phi(s_k^p; x_k, B_k, \sigma_k) \geq \sigma_k \eta_\sigma \Delta\ell^v(s_k^s; x_k)$$

We may conclude from this inequality, (3.11), and the fact that σ_k will not be increased on Line 36 as a result of part (ii) of this lemma, that $\sigma_{k+1} = \sigma_k$ for $k \in \mathcal{S}'$, which proves part (iii). □

Theorem 6. *If Assumptions 3.3.1–3.3.4 and* (3.80) *hold, there is a limit point* x_* *such that either*

(i) x_* *is an infeasible stationary point; or*

(ii) x_* *is feasible, but the MFCQ does not hold.*

Proof. Let $\mathcal{D}$ to be the infinite index set consisting of the iterations for which the weighting parameter is increased. Then, let x_* be a limit point of $\{x_k\}_{k\in\mathcal{D}}$, which must exist as a consequence of Assumptions 3.3.1 and 3.3.2. First, suppose that $v(x_*) > 0$. It then follows from Lemma 27 that if $\Delta\ell^v(s_*^s; x_*) > 0$ (s_*^s is defined in Lemma 27), then the weighting parameter is updated only a finite number of times along $\mathcal{D}$, which is a contradiction. Therefore, we deduce that $\Delta\ell^v(s_*^s; x_*) = 0$ and consequently that x_* is an infeasible stationary point. Second, suppose that $v(x_*) = 0$. It then follows from Lemma 28 that if the MFCQ

holds at x_*, then σ_k only be increased a finite number of times along $\mathcal{D}$. This is a contradiction and, therefore, the MFCQ does not hold at x_*. □

3.4 Conclusions and discussion

In this chapter, we presented a new filter line search method that replaced the traditional restoration phase with a penalty mode that systematically decreased an exact penalty function. Importantly, we solved a *single* strictly convex quadratic program subproblem during each iteration that was *always* feasible. Each search direction was defined as a convex combination of a steering step (a solution of a linear program) that represented the best local improvement in constraint violation and a predictor step that reduced our strictly convex quadratic model of the exact penalty function. We also allowed for the computation of an accelerator step defined as a solution to a simple equality constrained quadratic program (plus trust-region constraint) to promote fast local convergence. In this manner, the trial step always incorporated information from both the objective function and constraint violation. To further promote step acceptance, we utilized second-order information in the computation of Cauchy steps that provided realistic measurements of the decrease one might expect from the nonlinear problem functions. By using local feasibility estimates that emerged during the steering process, we defined a new and

improved margin (envelope) of the filter. This new definition encouraged the acceptance of steps that made reasonable progress, but might be considered inadmissible by a traditional filter. Under standard assumptions, we proved global convergence of our algorithm.

The fact that every subproblem of our method is feasible has an interesting (favorable) consequence when compared to previous SQO filter methods. Those methods trigger a restoration phase in multiple situations, the most common being when the traditional SQO subproblem is infeasible. In this case, the primary role of the restoration phase is to obtain a new feasible subproblem. This undesirable situation is not encountered in our method since all subproblems are feasible. Our method still may enter a penalty phase, but only when overwhelming evidence indicates that previously added filter entries are blocking progress. We believe this feature of our method is far more attractive and practical in comparison to previous filter methods.

Local convergence issues have not been considered here. It is evident that our method—like other filter SQO methods based on exact penalty functions—may experience the Maratos effect [60], i.e., reject the unit step (the traditional SQO step) when the current iterate is arbitrarily close to a minimizer. This potential issue, and more generally the task of establishing local superlinear convergence of a nonmonotone variant of FiSQO, is considered in Chapter 4. Also, we remark that numerical results are presented in Section 4.7 after we

have described the details of our nonmonotone version of FiSQO.

Straightforward modifications to our method allow for the solution of problems defined by a mixture of inequality and equality constraints. For instance, the definition of the constraint violation would be augmented to represent an ℓ_1 measure of infeasibility for both the inequality and equality constraints. Each key subproblem must also be modified. For example, (3.5) would additionally include the linearized equality constraints augmented by a pair of nonnegative elastic variables. Otherwise, the algorithm remains unchanged.

Chapter 4

A nonmonotone filter SQO method

4.1 Algorithm overview

The method presented in this section is a *nonmonotone* variant of the filter SQO method presented in Chapter 3. Most aspects of the two methods are identical, which include the trial step computation and the general framework that allows for filter and penalty modes when determining step acceptance. The key differences are twofold: (i) allowing for nonmonotonicity in the sense that steps may be temporarily used that are not acceptable to the filter and (ii) altering the step acceptance conditions to account for the nonmonotonicity but that still allow us to provide global convergence guarantees and, as we will

show, a superlinear rate of convergence result under standard assumptions.

To obtain local superlinear convergence, it is typical to incorporate either a second-order correction step [73, Section 10.4.2] or a nonmonotone strategy [73, Section 10.1.1]. As mentioned above, here we choose to use a nonmonotone strategy in which we temporarily accept steps even when they do not satisfy the conditions required to terminate the line search as used in the monotone variant presented in Chapter 3. In fact, we allow this to continue for a pre-specified number of iterations in what is called a nonmonotone phase. If appropriate conditions (to be described shortly) are not satisfied within the pre-specified number of iterations, we return to the first iterate of the nonmonotone phase and perform a backtracking line search as outlined earlier. Technically, we are not using a nonmonotone strategy since our step acceptance is driven by a filter in which monotonicity is not typical, so perhaps it is more accurately categorized as a watchdog strategy [76, 77].

In order to make this chapter nearly self contained, we give a short overview of the step computation—which is the same as for the monotone variant presented in Chapter 3—in Section 4.2. The new step acceptance criteria that account for nonmonotonicity in the algorithm are described in Section 4.3. Global and local analysis are presented in Sections 4.5 and 4.6, while our numerical results are presented in Section 4.7.

4.2 Search direction computation

In this section we provide a compact description of the calculations needed to compute the search directions s_k and s_k^a, which we remind the reader are the same as those for the monotone algorithm presented in Chapter 3; see Section 3.1 for additional details.

Let x_k be the current iterate. The search direction s_k is defined as a convex combination of a steering step s_k^s and a predictor step s_k^p. The steering step s_k^s is defined as a solution (not necessarily unique) to the convex piecewise linear steering subproblem (3.2) or the equivalent linear program (3.1).

Once the steering step s_k^s has been computed, we calculate the change in the linearized constraint violation given by (2.9) which provides a prediction of the decrease in infeasibility that one might expect from the step s_k^s. Moreover, this quantity allows us to determine whether x_k is an infeasible stationary point, i.e., an infeasible first-order minimizer of v, as given by Definition 9.

The computation of the predictor step s_k^p involves the quadratic model of the objective function $q^f(s; x, M)$ as defined by (2.6) for any symmetric matrix M, and the piecewise quadratic model of ϕ given by (2.8). The predictor step is then the unique solution to the strictly convex predictor subproblem (3.4).

The search direction s_k is defined as the convex combination of the steering step s_k^s and the predictor step s_k^p in (3.7), which makes s_k a descent direction for

v when $\Delta\ell^v(s_k^s; x_k) > 0$ as shown in Lemma 4. Next, the penalty parameter σ_{k+1} is updated so that s_k is also a decent direction for the penalty function ϕ at x_k as shown in Lemma 8.

Given the search direction s_k, the penalty parameter σ_{k+1}, the kth Lagrange multiplier estimate y_k^p from the predictor step problem (3.4), and the matrix $H_k := \nabla^2_{xx} L(x_k, y_k^p)$, we compute the Cauchy-$f$ step, denoted s_k^{cf}, from (3.20) and the Cauchy-ϕ step, denoted $s_k^{c\phi}$, from (3.21). These Cauchy steps are used to measure the predicted decrease in the objective function f and penalty function ϕ, respectively, along the search direction s_k using the quadratic models q^f and q^ϕ defined with the exact Hessian matrix H_k.

Finally, to accelerate convergence, we compute an accelerator step s_k^a from (3.17), which involves the solution of an equality-constrained quadratic program (3.18).

4.3 Step acceptance

The iterations of our algorithm consist of the disjoint union of two types of iterations. The first type, denoted by $\mathcal{S}$ and called the set of successful iterations, are those iterations for which at least one of four sets of conditions are satisfied (we also always include iteration zero). These sets of conditions are described later in this section.

The second type, denoted by $\mathcal{U}$ and called the set of unsuccessful iterations, is the complementary set consisting of the nonmonotone iterations, i.e, those iterations during which the full trial step is accepted even though none of the sets of conditions described below are satisfied. Note that $\mathcal{S} \cap \mathcal{U} = \emptyset$ and that every iteration belongs in either $\mathcal{S}$ or $\mathcal{U}$.

To handle the nonmonotone nature of our algorithm, it is convenient to define $R(k)$ as the last successful iteration (which may in fact be k), i.e.,

$$R(k) := \max\{i : k \geq i \in \mathcal{S}\}.$$

Consequently, if $R(k) < j \leq k$, then $j \in \mathcal{U}$ and iteration j is part of a nonmonotone sequence of iterations.

We now begin to describe the sets of conditions that determine when an iteration is included in the set of successful iterations $\mathcal{S}$. Central to this task is the concept of a filter as used in Chapter 3 and repeated here for convenience. Specifically, a filter is formally defined as any finite set of points in $\mathbb{R}^+ \times \mathbb{R}$. In our case, we initialize the filter as $\mathcal{F}_0 = \emptyset$ and then update it so that at each iteration k the filter satisfies $\mathcal{F}_k \subseteq \{(v_j, f_j) : 0 \leq j < k\}$. Whether an ordered pair is added to the filter at the end of each iteration depends in part on whether the iterate is acceptable to the current filter as defined next.

Definition 18 (acceptable to $\mathcal{F}_k$)**.** The point x is acceptable to $\mathcal{F}_k$ if its associ-

ated ordered pair $\big(v(x), f(x)\big)$ satisfies

$$\begin{aligned} v(x) &\leq \max\left\{v_i - \alpha_i \eta_v \Delta\ell^v(s_i^s; x_i),\, \beta v_i\right\} \quad \text{or} \\ f(x) &\leq f_i - \gamma \min\left\{v_i - \alpha_i \eta_v \Delta\ell^v(s_i^s; x_i),\, \beta v_i\right\} \end{aligned} \tag{4.1}$$

for all $0 \leq i < k$ satisfying $(v_i, f_i) \in \mathcal{F}_k$, where $\alpha_i \in (0, 1]$ is the ith step length, and $\{\eta_v, \beta, \gamma\} \subset (0, 1)$ are some constants.

Note that the two inequalities in (4.1) provide a margin around the elements of the filter in (v, f)-space, ensuring that the constraint violation or the objective function at x is sufficiently smaller than at points x_i whose ordered pair is in the current filter $\mathcal{F}_k$.

In certain situations, we need to know that a trial iterate is acceptable to the filter defined by the union of $\mathcal{F}_k$ with an ordered pair (v_j, f_j) associated with some x_j that is not in the filter. This leads to the definition of being acceptable to the augmented filter.

Definition 19 (acceptable to $\mathcal{F}_k$ augmented by x_j)**.** The point x is acceptable to $\mathcal{F}_k$ *augmented* by x_j if x is acceptable to $\mathcal{F}_k$ as given by Definition 12 and (4.1) holds with $i = j$.

Acceptability to the filter is only one aspect used to define the four sets of conditions that are checked during each iteration. Which sets of conditions are checked depends on the current mode, i.e., filter or penalty mode. At this point,

the reader only needs to know that step acceptance in filter mode (Section 4.3.1) is driven by acceptability to the filter, whereas step acceptance in penalty mode (Section 4.3.2) is driven by reducing the penalty function.

In the next two sections, the names used to denote certain pairs of the form (α, s) for some step length α and search direction s (e.g., see Definition 20) are the same as in Chapter 3. This highlights that the definitions in this chapter are generalizations of those in Chapter 3 that account for nonmonotonicity.

4.3.1 Step acceptance in filter mode

In filter mode, we seek to obtain a v-(violation)-pair, an o-(objective)-pair, or a b-(blocking)-pair. The pair $(\alpha_k, \hat{s}_k)$ for some $\hat{s}_k \in \{s_k^a, s_k\}$ is deemed to be a v-pair based on the following.

Definition 20 (v-pair)**.** The pair (α, s) constitutes a v-pair if $x_k + \alpha s$ is acceptable to $\mathcal{F}_{R(k)}$ augmented by $x_{R(k)}$ and

$$\Delta\ell^f(s_{R(k)}; x_{R(k)}) < \gamma_v \Delta\ell^v(s_{R(k)}; x_{R(k)}) \text{ for some } \gamma_v \in (0, 1). \tag{4.2}$$

If $(\alpha_k, \hat{s}_k)$ is a v-pair, we say that $x_{R(k)}$ is a v-iterate and set $x_{k+1} \leftarrow x_k + \alpha_k \hat{s}_k$. In this case, we add $k + 1$ to the set of successful iterates $\mathcal{S}$ and $(v_{R(k)}, f_{R(k)})$ to the filter $\mathcal{F}_{R(k)}$. We remain in filter mode.

The pair $(\alpha_k, \hat{s}_k)$ for some $\hat{s}_k \in \{s_k^a, s_k\}$ is determined to be an o-pair based

on the following definition.

Definition 21 (o-pair)**.** The pair (α, s) constitutes an o-pair if $x_k + \alpha s$ is acceptable to $\mathcal{F}_{R(k)}$,

$$\Delta\ell^f(s_{R(k)}; x_{R(k)}) \geq \gamma_v \Delta\ell^v(s_{R(k)}; x_{R(k)}), \quad \text{and} \tag{4.3a}$$

$$f(x_k + \alpha s) \leq f(x_{R(k)}) - \gamma_f \alpha \rho^f_{R(k)}, \tag{4.3b}$$

where $\gamma_v \in (0, 1)$ is the same constant used to define a v-pair, $\gamma_f \in (0, 1)$, and

$$\rho^f_{R(k)} := \min\left\{\Delta\ell^f(s_{R(k)}; x_{R(k)}),\ \Delta q^f(s^{cf}_{R(k)}; x_{R(k)}, H_{R(k)})\right\}. \tag{4.4}$$

If $(\alpha_k, \hat{s}_k)$ is an o-pair, we say that $x_{R(k)}$ is an o-iterate and set $x_{k+1} \leftarrow x_k + \alpha_k \hat{s}_k$. In this case, we add $k + 1$ to the set of successful iterates $\mathcal{S}$, but do not modify the filter. We remain in filter mode. For these types of pairs, the value of the objective function at x_{k+1} is significantly smaller than the value at $x_{R(k)}$.

Finally, the following is used to determine whether (α_k, s_k) is a b-pair.

Definition 22 (b-pair)**.** The pair (α, s) constitutes a b-pair if

$$v(x_k + \alpha s) < v(x_{R(k)}) \tag{4.5}$$

and

$$\phi(x_k + \alpha s; \sigma_{k+1}) \leq \phi(x_{R(k)}; \sigma_{\mathrm{R(k)+1}}) - \gamma_\phi \alpha \rho^\phi_{R(k)} \quad \text{for some } \gamma_\phi \in (0,1), \tag{4.6}$$

where

$$\rho^\phi_{R(k)} := \min \left\{ \Delta \ell^\phi(s_{R(k)}; x_{R(k)}, \sigma_{\mathrm{R(k)+1}}),\ \Delta q^\phi(s^{c\phi}_{R(k)}; x_{R(k)}, H_{\mathrm{R(k)}}, \sigma_{\mathrm{R(k)+1}}) \right\}. \tag{4.7}$$

If (α_k, s_k) is a b-pair, we say that $x_{R(k)}$ is a b-iterate and set $x_{k+1} \leftarrow x_k + \alpha_k s_k$. In this case, we add $k+1$ to the set of successful iterates $\mathcal{S}$, add $(v_{R(k)}, f_{R(k)})$ to the filter $\mathcal{F}_k$, and then enter penalty mode. For these pairs, the constraint violation and penalty function at x_{k+1} are smaller than at $x_{R(k)}$. Since b-pairs will only be checked for after the conditions of a v- and an o-pair are checked, it indicates that the current filter entries may be blocking productive steps. Therefore, we respond by accepting the step x_{k+1} and entering penalty mode. We note that this is the only scenario in which we enter penalty mode.

4.3.2 Step acceptance in penalty mode

If penalty mode is entered, we have reason to believe that the current filter entries are blocking productive steps. Thus, in penalty mode we seek steps that decrease the penalty function, but return to filter mode as soon as is deemed

appropriate. The following definition is used to determine when the pair $(\alpha_k, \hat{s}_k)$ for some $\hat{s}_k \in \{s_k^a, s_k\}$ is a p-(penalty)-pair.

Definition 23 (p-pair)**.** The pair (α, s) is a p-pair if (4.6) is satisfied.

If $(\alpha_k, \hat{s}_k)$ is a p-pair, we say that $x_{R(k)}$ is a p-iterate, set $x_{k+1} \leftarrow x_k + \alpha_k \hat{s}_k$, and add $k+1$ to the set of successful iterations $\mathcal{S}$. Also, if $x_k + \alpha_k \hat{s}_k$ is acceptable to the filter $\mathcal{F}_{R(k)}$, we return to filter mode, but otherwise remain in penalty mode. It is clear that in this case the value of the penalty function at iterate x_{k+1} is significantly less than the value at $x_{R(k)}$.

4.4 The complete algorithm

Our method is stated as Algorithm 2. The logical flow during each iteration depends on the value of several parameters: *fails* holds the number of consecutive unsuccessful iterations that have been performed, max_fails holds the value of the maximum allowed consecutive unsuccessful iterations, and $\mathcal{P}$-$mode$ is a flag that indicates whether the current mode is penalty or filter mode. Although the parameter max_fails is only required to be nonnegative, for the rest of this section we assume that $max\_fails > 0$ so that the algorithm is nonmonotone.

To explain the flow of logic, let us first examine the algorithm when *fails* $\leq$ max_fails. In this case, the condition in Step 6 tests false so that the search

directions s_k and s_k^a are computed in Lines 10–18 as described in Section 4.2 (with full detail found in Sections 3.1.1–3.1.6). We now consider two possible scenarios. First, suppose that $\mathcal{P}$-$mode$ has the value false in Line 20, i.e., the algorithm is in filter mode (the default mode). Then, since ***fails*** $\leq max\_fails$ and $max\_fails > 0$, we only check whether the pair $(1, s_k^a)$ is a v-pair, an o-pair, or a b-pair, i.e., we only consider the full accelerator step. If $(1, s_k^a)$ does satisfy the conditions that define the various pairs, we set $x_{k+1} \leftarrow x_k + s_k^a$ and add iteration $k+1$ to the set of successful iterations $\mathcal{S}$. Otherwise, our nonmonotone strategy still chooses to set $x_{k+1} \leftarrow x_k + s_k^a$, to stay in filter mode, and to increase the *fails* counter. Second, suppose that $\mathcal{P}$-$mode$ has the value true in Line 20. Then, since ***fails*** $\leq max\_fails$ and $max\_fails > 0$, we only check whether the pair $(1, s_k^a)$ is a p-pair. If $(1, s_k^a)$ is a valid p-pair, we set $x_{k+1} \leftarrow x_k + s_k^a$ and add iteration $k+1$ to the set of successful iterations $\mathcal{S}$. Otherwise, our nonmonotone strategy still chooses to set $x_{k+1} \leftarrow x_k + s_k^a$, to stay in penalty mode and to increase the *fails* counter.

If the counter *fails* is ever incremented to a value larger than max_fails, then the flow of logic changes. In short, we return to the last successful iterate (see Line 7) and then perform a backtracking line search. To give more details, first suppose that $\mathcal{P}$-$mode$ has the value false in Line 20. Then, the backtracking loop starts in Line 32 and proceeds until either a valid v-, o-, or b-pair is found. Note that in Line 34, the phase $\hat{s}_k \in \{s_k^a, s_k\}$ should be interpreted as

first setting $\hat{s}_k$ to the value s_k^a and second setting it to the value s_k. Also note that we check whether (α_k, s_k^a) or (α_k, s_k) are acceptable as v- or o-pairs before checking if (α_k, s_k) is a valid b-pair; this gives preference to filter mode since b-pairs trigger entrance into penalty mode. Second, suppose that $\mathcal{P}\text{-}mode$ has the value true in Line 20. Then, the backtracking loop starts in Line 21 and proceeds until a valid p-pair is found. Once a valid p-pair $(\alpha_k, \hat{s}_k)$ is obtained, we immediately go to Line 28 to test whether the next iterate $x_k + \alpha_k \hat{s}_k$ is acceptable to the filter $\mathcal{F}_k$; if it is acceptable, we return to filter mode by setting $\mathcal{P}\text{-}mode$ to false.

Finally, at the end of every iteration and regardless of the current mode, we choose to increase the penalty parameter if (3.29) holds, as restated below:

$$\Delta q^{\phi}(s_k; x_k, B_k, \sigma_{k+1}) < \eta_{\phi} \Delta q^{\phi}(s_k^p; x_k, B_k, \sigma_{k+1}) \quad \text{for some } \eta_{\phi} \in (0, 1).$$

The satisfaction of (3.29) indicates that the contribution of s_k^p to the definition of s_k in (3.7) is dwarfed by the contribution of the steering step s_k^s. This typically results when the value of τ_k used in the definition of s_k is very small, which is a sign (see (3.7) and (3.8)) that the predictor step s_k^p did not make significant progress toward linearized feasibility. Thus, the natural course of action is to increase the penalty parameter (see Line 50) to promote linearized feasibility of the predictor step during the next iteration.

Algorithm 2 A nonmonotone filter SQO algorithm.

Input an initial primal-dual pair (x_0, y_0).
Choose $\{\eta_v, \eta_\sigma, \eta_\phi, \sigma_{\text{inc}}, \beta, \gamma, \gamma_v, \gamma_f, \gamma_\phi, \xi\} \subset (0,1)$, $0 < \delta_{\min} \leq \delta_{\max} \leq \delta_a < \infty$.
Choose $0 \leq max\_fails \in \mathbb{N}$, and set *fails* $\leftarrow 0$, $\mathcal{S} \leftarrow \{0\}$, $\mathcal{U} \leftarrow \emptyset$.
Set $k \leftarrow 0$, $\mathcal{F}_0 \leftarrow \emptyset$, and $\mathcal{P}$-$mode \leftarrow$ false, and choose $\sigma_0 > 0$ and $\delta_0 \in [\delta_{\min}, \delta_{\max}]$.
loop
    if *fails* $> max\_fails$ **then**
        $x_k \leftarrow x_{R(k)}$, $s_k \leftarrow s_{R(k)}$, $s_k^a \leftarrow s_{R(k)}^a$, $y_k^p \leftarrow y_{R(k)}^p$, $\sigma_{k+1} \leftarrow \sigma_{R(k)+1}$, $H_k \leftarrow H_{R(k)}$.
    else
        Compute s_k^s as a solution of (3.1).
        Calculate $\Delta \ell^v(s_k^s; x_k)$ from (2.9).
        if $\Delta \ell^v(s_k^s; x_k) = 0$ and $v(x_k) > 0$, **then**
            return with the infeasible stationary point x_k for problem (2.1).
        Choose $B_k \succ 0$. Compute s_k^p as the solution of (3.4) with multiplier y_k^p.
        if $\Delta q^\phi(s_k^p; x_k, \sigma_k) = v(x_k) = 0$, **then**
            return with the KKT point (x_k, y_k^p) for problem (2.1).
        Compute $s_k = (1 - \tau_k) s_k^s + \tau_k s_k^p$ from (3.7) such that (3.8) is satisfied.
        Compute the new weight σ_{k+1} from (3.11).
        Evaluate $H_k = \nabla_{xx}^2 L(x_k, y_k^p)$. Solve (3.17) and (3.18) to get s_k^a and y_k^a.
    Compute $s_k^{c\phi}$ from (3.21). Calculate $\Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1})$ from (2.13).

20: **if** $\mathcal{P}$-$mode$ **then**
21: **for** $j = 0, 1, 2, \ldots$ **do**
22: $\alpha_k \leftarrow \xi^j$.
23: **for** $\hat{s}_k \in \{s_k^a, s_k\}$ **do**
24: **if** $(\alpha_k, \hat{s}_k)$ is a p-pair **then**
25: $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$ and go to Line 28. ▷ $k+1 \in \mathcal{S}$
26: **if** *fails* $\leq max\_fails$ and $max\_fails > 0$ **then**
27: *fails* $\leftarrow$ *fails* $+ 1$, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, go to Line 49. ▷ $k+1 \in \mathcal{U}$
28: **if** $x_k + \alpha_k \hat{s}_k$ is acceptable to $\mathcal{F}_k$ **then**
29: $\mathcal{P}$-$mode \leftarrow$ false.
30: **else**
31: Compute s_k^{cf} from (3.20). Calculate $\Delta q^f(s_k^{cf}; x_k, H_k)$ from (2.11).
32: **for** $j = 0, 1, 2, \ldots$ **do**
33: $\alpha_k \leftarrow \xi^j$.
34: **for** $\hat{s}_k \in \{s_k^a, s_k\}$ **do**
35: **if** $(\alpha_k, \hat{s}_k)$ is a v-pair **then**
36: $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_{R(k)}, f_{R(k)})\}$, go to Line 48. ▷ $k+1 \in \mathcal{S}$
37: **if** $(\alpha_k, \hat{s}_k)$ is an o-pair **then**
38: $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, go to Line 48. ▷ $k+1 \in \mathcal{S}$
39: **if** *fails* $\leq max\_fails$ and $max\_fails > 0$ **then**
40: **if** $(\alpha_k, \hat{s}_k)$ is a b-pair **then**
41: $\mathcal{P}$-$mode \leftarrow$ **true**. ▷ $k+1 \in \mathcal{S}$
42: $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_{R(k)}, f_{R(k)})\}$, go to Line 48.
43: **else**
44: *fails* $\leftarrow$ *fails* $+ 1$, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, go to Line 49. ▷ $k+1 \in \mathcal{U}$
45: **if** (α_k, s_k) is a b-pair **then**
46: $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_{R(k)}, f_{R(k)})\}$, and $\mathcal{P}$-$mode \leftarrow$ **true**. ▷ $k+1 \in \mathcal{S}$
47: Go to Line 48.
48: *fails* $\leftarrow 0$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{k+1\}$.
49: **if** (3.29) is satisfied **then**
50: $\sigma_{k+1} \leftarrow \sigma_{k+1} + \sigma_{\text{inc}}$.
51: $x_{k+1} \leftarrow x_k + \alpha_k \hat{s}_k$, $y_{k+1} \leftarrow y_k^a$, $\delta_{k+1} \in [\delta_{\min}, \delta_{\max}]$, $k \leftarrow k+1$.

4.5 Global convergence

In this section, we establish the same global convergence result as in Chapter 3 under the same assumptions (Assumptions 3.3.1–3.3.4). Using these assumptions we may state our global convergence result, which is identical to Theorem 2 and uses the Mangasarian-Fromovitz constraint qualification (MFCQ) [74]. Since the proof is essentially the same, here we only describe the differences that result from the nonmonotonicity of Algorithm 2.

Theorem 7. *If Assumptions* 3.3.1–3.3.4 *hold, then one of the following holds:*

(i) *Algorithm* 2 *terminates finitely with either a first-order KKT point or an infeasible stationary point in Lines* 15 *or* 12*, respectively, for problem* (2.1)*.*

(ii) *Algorithm* 2 *generates infinitely many iterations* $\{x_k\}$*,* $\sigma_k = \bar{\sigma} < \infty$ *for all* k *sufficiently large, and there exists a limit point* x_* *of* $\{x_k\}$ *that is either a first-order KKT point or an infeasible stationary point for problem* (2.1)*.*

(iii) *Algorithm* 2 *generates infinitely many iterations* $\{x_k\}$*,* $\lim_{k\to\infty} \sigma_k = \infty$*, and there exists a limit point* x_* *of* $\{x_k\}$ *that is either an infeasible stationary point or a feasible point at which the MFCQ fails.*

Proof. The proof for the monotone variant Algorithm 1 hinged on guaranteeing sufficient progress during every iteration as measured by conditions placed on the (α, s) pairs. In this chapter, we have formulated conditions on (α, s) pairs

(see Section 4.3) that generalize the conditions used in Chapter 3 (see Section 3.1.8). The key difference is that the conditions in this chapter are defined with respect to the last successful iteration $R(k)$, as opposed to the current iterate k. In this way, the sequence of successful iterates inherits the properties of the sequence of iterates generated by the monotone algorithm. Thus, from a theoretical perspective, we can essentially ignore the unsuccessful iterations and focus our attention on the successful ones. We also note that if we set max_fails to the value zero in Algorithm 2, our method reduces to the monotone variant analyzed in Chapter 3.

We will now establish global convergence of Algorithm 2 by walking the reader through the global convergence analysis in Chapter 3 (Section 3.3) and highlighting the differences that surface.

First, note that outcome (i) can occur since it is possible to locate either an infeasible stationary point (see (9)) or a KKT point in a finite number of iterations (see Lines 12 and 15 of Algorithm 2).

If outcome (i) does not happen, then it is possible that outcome (iii) occurs so that the penalty parameter converges to infinity. For this case we can follow the proofs in Section 3.3.2 since they only depend on Assumptions 3.3.1–3.3.4, the manner in which the trial steps are computed and the properties of their associated subproblems. Since these aspects have not changed from the monotone algorithm in Chapter 3, we may again deduce that Lemma 27, Lemma 28,

and Theorem 6 still hold, which proves outcome (iii).

Finally, suppose that outcomes (i) and (iii) do not occur so that infinitely many iterations are performed and the penalty parameter is fixed for all k sufficiently large. We may then establish that outcome (ii) holds by using the proofs in Section 3.3.1 with minor modifications that we now describe.

The first difference is related to the definition of v-iterates and the associated v-pairs. The conditions that define them in Chapter 3 are stated in Definition 14, but are restated here: $x_k + \alpha s$ is acceptable to the filter $\mathcal{F}_k$ augmented by x_k, and $\Delta\ell^f(s_k; x_k) < \gamma_v \Delta\ell^v(s_k; x_k)$. However, in our nonmonotone Algorithm 2, there is no guarantee that x_k is acceptable to the filter, let alone that it satisfies any additional conditions. Thus, to handle the nonmonotonicity, we use in place of x_k the last successful iterate $x_{R(k)}$ since we know that it satisfies the same conditions required in the monotone algorithm. It is then natural to use Definition 20 to define a v-pair in the nonmonotone setting. The v-iterates are also used to update the filter. In the monotone algorithm, after a v-pair was found, the pair (v_k, f_k) was added to the filter. In our nonmonotone Algorithm 2, we add the pair $(v_{R(k)}, f_{R(k)})$, which maintains the same properties of the filter. For instance, using the filter inequalities in (4.1), it can be shown that if infinitely many entries are added to the filter, then some subsequence of the iterates converges to a first-order minimizer of the constraint violation as shown in Lemma 25.

The second difference arises in the definition of o-iterates and the associated o-pairs. In the monotone Algorithm 1, a pair (α, s) constituted an o-pair at iteration k if the following conditions (see Definition 15) were satisfied: $x_k + \alpha s$ is acceptable to the filter $\mathcal{F}_k$; $\Delta\ell^f(s_k; x_k) \geq \gamma_v \Delta\ell^v(s_k; x_k)$; and $f(x_k + \alpha s) \leq f(x_k) - \gamma_f \alpha \rho_k^f$, where $\{\gamma_v, \gamma_f\} \subset (0, 1)$ and $\rho_k^f = \min\left[\Delta\ell^f(s_k; x_k),\ \Delta q^f(s_k^{cf}; x_k, H_k)\right]$. In this case for the monotone algorithm, the trial point $x_k + \alpha s$ has sufficiently reduced the objective function from the current point x_k. For the nonmonotone Algorithm 2 we again use the last successful iterate $x_{R(k)}$, which leads to Definition 21. Now, the trial point $x_k + \alpha s$ associated with an o-pair (α, s) sufficiently reduces the objective function when compared to the iterate $x_{R(k)}$. This is the key property used to show that if all sufficiently large successful iterates are o-iterates, then the sequence of successful iterates converges to a first-order minimizer of the constraint violation, as shown in Lemma 22, and the penalty function, as shown in Lemma 23.

The third difference is the definition of b-pairs. In the monotone algorithm, the conditions for that define a b-pair (see Definition 16) are that $v(x_k + \alpha s) < v(x_k)$ and $\phi(x_k + \alpha s; \sigma_{k+1}) \leq \phi(x_k; \sigma_{k+1}) - \gamma_\phi \alpha \rho_k^\phi$, where $\gamma_\phi \in (0, 1)$ and $\rho_k^\phi = \min\left[\ \Delta\ell^\phi(s_k; x_k, \sigma_{k+1}),\ \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1})\ \right]$. A b-pair (α, s) therefore defined an iterate $x_k + \alpha s$ that reduced the constraint violation and sufficiently reduced the penalty function. Again, we adjust our conditions to be based on the last successful iterate $R(k)$ as given by Definition 22. When a b-pair is found, we

add the entry $(v_{R(k)}, f_{R(k)})$ to the filter to preserve the required relationships between the filter entries as used in the monotone algorithm. In particular, if infinitely many b-iterates are found so that infinitely many entries are added to the filter, then a subsequence of the iterates converges to a first-order solution of the penalty function (see Lemma 26(ii)).

The fourth difference is the definition of a p-pair. In the monotone algorithm, the condition that defines a p-pair (see Definition 17) is that $\phi(x_k + \alpha s; \sigma_{k+1}) \leq \phi(x_k; \sigma_{k+1}) - \gamma_\phi \alpha \rho_k^\phi$, where $\gamma_\phi \in (0, 1)$ is a chosen fixed constant and we defined $\rho_k^\phi := \min\big[\, \Delta\ell^\phi(s_k; x_k, \sigma_{k+1}),\, \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1}) \,\big]$. This condition ensured that the penalty function was sufficiently reduced. In the nonmonotone Algorithm 2 we have again adjusted our conditions to be based on the last successful iterate $R(k)$ as given by Definition 23. This condition maintains the important property that the penalty function is sufficiently reduced, but this time between consecutive successful iterations. Using this property, it now follows as in Lemma 21 and Theorem 3 that if all sufficiently large successful iterates are p-iterates, then there exists a limit point of the sequence of iterates that is an infeasible stationary point (see Definition 9).

A fifth difference is that the monotone algorithm only searches along the direction s_k for a b-pair, whereas our nonmonotone Algorithm 2 additionally checks whether $(1, s_k^a)$ is a b-pair in Line 40. This impacts the proof of Lemma 19(iii), which is described only for the step s_k. In turn, Lemma 19(iii) is used in

the proofs of Lemmas 23 and 26(ii) to show that α_k is uniformly bounded away from zero along a certain subsequence. Since $\alpha_k = 1$ when a b-pair is found in Line 40 of Algorithm 2, the lower bound on α_k remains intact.

The final difference also involves the step length calculation. Specifically, in our nonmonotone algorithm, α_k is either equal to one (the unit step) during a nonmonotone phase or obtained through a line search procedure, the latter of which is the computation used during every iteration of the monotone algorithm. It is clear, however, that this difference has no effect on the lower bounds derived for the step lengths (e.g., Lemma 19). □

4.6 Local convergence

We show that Algorithm 2 is Q-quadratically convergent by making use of the following additional assumption.

Assumption 4.6.1. Algorithm 2 generates an infinite sequence of iterates $\{x_k\}$ that converges to a KKT-point x_* for problem (2.1) with an associated Lagrange multiplier vector y_* such that (x_*, y_*) satisfies the following strong second-order sufficient optimality conditions:

(i) there exists $\lambda_{\min} > 0$ such that $s^T H_* s \geq \lambda_{\min} \|s\|_2^2$ for all s satisfying $J_{\mathcal{A}_*} s = 0$, where $H_* := H(x_*, y_*)$, $\mathcal{A}_* := \{i : c(x_*) = 0\}$, and $J_{\mathcal{A}_*} := [J(x_*)]_{\mathcal{A}_*}$ denotes the active rows of the Jacobian;

(ii) strict complementarity holds, i.e., $[y_*]_{\mathcal{A}_*} > 0$; and

(iii) the linear independent constraint qualification (LICQ) holds, i.e., $J_{\mathcal{A}*}$ has full row rank.

Note that $\lambda_{\min}$ is, without loss of generality, the same value in Assumption 3.3.3.

To show that the iterates $\{x_k\}$ converge to x_* at a Q-superlinear rate, we first show that under the above assumptions and for penalty parameter sufficiently large, the accelerator step s_k^a is equivalent to the traditional SQO step. We then show that for all sufficiently large $k \in \mathcal{S}$, either $x_{k+1} = x_k + s_k^a$ or $x_{k+2} = x_k + s_k^a + s_{k+1}^a$ is accepted by Algorithm 2, by considering the filter and penalty mode separately. In particular, if $\mathcal{P}\text{-}mode =$ **false** during iteration k, we show that at least one of the above two points is acceptable to the augmented filter. We then show that it must also satisfy conditions that make it either a v-iterate, an o-iterate, or a b-iterate. On the other hand, if $\mathcal{P}\text{-}mode =$ **true** during iteration k, we show that one of the two points must be a p-iterate. Theorem 8 ties all of these facts together.

We begin by first showing that under Assumption 4.6.1, the penalty parameter is bounded and that infinitely many iterations occur in filter mode.

Lemma 29. *If Assumption* 4.6.1 *holds, then* (i) *the penalty parameter* $\sigma_k = \bar{\sigma} < \infty$ *for all* k *sufficiently large, and* (ii) $\mathcal{P}\text{-}mode =$ ***false*** *along an infinite subsequence of iterates.*

Proof. Since (x_*, y_*) is a KKT pair and the LICQ holds at x_* (which implies that the MFCQ holds), it follows from Theorem 6 that $\sigma_k = \bar{\sigma} < \infty$ for all k sufficiently large, which proves (i). Moreover, since (i) has been established, it follows as in Lemma 21 and Theorem 3 that if all sufficiently large successful iterations are p-iterates, then x_* is an infeasible stationary point (Definition 9). This contradicts that x_* is a KKT point (in particular that it is feasible), and thus we must conclude that there exists an infinite number of successful v - , o-, or b-iterates. This completes the proof of part (ii) since v-, o-, and b-iterates only occur in filter mode, i.e., when $\mathcal{P}$-$mode$ has the value false. □

We next show that the penalty parameter is eventually at least as large as the infinity norm of the Lagrange multiplier vector y_*. To understand the relevance of this result, see [73, Theorem 14.5.1].

Lemma 30. *If Assumptions* 3.3.3 *and* 4.6.1 *hold, then* $\sigma_k \equiv \bar{\sigma} \geq \|y_*\|_\infty$ *for all sufficiently large* k*.*

Proof. By Lemma 29, we know that $\sigma_k \equiv \bar{\sigma}$ for all sufficiently large k. To reach a contradiction, let us suppose that $\bar{\sigma} < \|y_*\|_\infty$. It then follows from [73, Theorem 14.5.2] and the fact that (x_*, y_*) is a first-order KKT pair for problem (2.1) that x_* is not a local minimizer of $\phi(x; \bar{\sigma})$.

Since x_* is a KKT-point, we know from Theorem 3 that not all iterations are p-iterates for sufficiently large k. It then follows from Lemma 20 that either

all iterates are o-iterates for sufficiently large k, there are infinitely many v-iterates, or there are infinitely many b-iterates. In the first case, let $\mathcal{K}_1$ be the subsequence of o-iterates; in the second case, let $\mathcal{K}_1$ be the subsequence of v-iterates; and in the third case, let $\mathcal{K}_1$ be the subsequence of b-iterates. We may now use Assumption 3.3.3 to declare the existence of a subsequence $\mathcal{K} \subseteq \mathcal{K}_1 \subseteq \mathbb{N}$ and positive-definite matrix B_* such that $\lim_{k\in\mathcal{K}} B_k = B_*$.

We next establish that the predictor step is computed from (3.4a) for all k sufficiently large. Since $\lim_{k\to\infty} x_k = x_*$ and x_* is a KKT-point, we know that there exists a constant $\varepsilon > 0$ such that $[c_k]_i \geq \varepsilon$ for all $i \notin \mathcal{A}_*$ and sufficiently large k. On the other hand, we have from Assumption 4.6.1(iii) that $[c_k + J_k s]_{\mathcal{A}_*} = 0$ is feasible for all k sufficiently large, and that the least-length solution converges to zero since $\lim_{k\to\infty}[c_k]_{\mathcal{A}_*} = 0$. Combining these observations shows that the linear inequality $c_k + J_k s \geq 0$ will have a solution with $\|s\|_\infty \leq \delta_{\min} \leq \delta_k$ (see (3.2)) for all k sufficiently large; therefore $\Delta\ell^v(s_k^s; x_k) = v_k$ and the predictor step is computed from (3.4a) as claimed.

Let s_*^p be the unique minimizer of $q^f(s; x_*, B_*)$ subject to $c_* + J_* s \geq 0$. Since x_* is not a local minimizer of $\phi(x; \bar{\sigma})$, we know from [71, Theorem 3.2(a)] that $s_*^p \neq 0$. Using this fact, that s_k^p is computed from (3.4a) for all sufficiently large k, and Assumptions 3.3.2 and 4.6.1, we have $\lim_{k\in\mathcal{K}} s_k^p = s_*^p \neq 0$, and therefore $\lim_{k\in\mathcal{K}} \Delta q^\phi(s_k^p; x_k, B_k, \bar{\sigma}) \neq 0$. In the first case above, i.e., iterates are o-iterates for sufficiently large k, this contradicts Lemma 23; in the second, this

contradicts Lemma 26(i); and in the third, this contradicts Lemma 26(ii). □

We have shown that $\sigma_k \equiv \bar{\sigma} \geq \|y_*\|_\infty$ for all sufficiently large k, but our local analysis requires a strict inequality. This is stated as an assumption.

Assumption 4.6.2. For all sufficiently large k we have $\sigma_k \equiv \bar{\sigma} > \|y_*\|_\infty$

Next, we present some key results from [45].

Lemma 31. *Let $w_* = (x_*, y_*)$ be the minimizer for problem* (2.1) *that satisfies Assumption* 4.6.1*, let Assumptions* 3.3.3 *and* 4.6.2 *hold, and finally let $\gamma_{c\phi} \in \big(\max\{\gamma_f, \gamma_\phi\}, 1\big)$ with γ_f and γ_ϕ defined in Definitions* 21 *and* 22*. Then, there exists positive number $\delta > 0$ such that if $k \in \mathcal{S}$ and $w_k = (x_k, y_k) \in \mathcal{B}_\delta(w_*)$, then*

(i) $\mathcal{A}_k(s_k^p) = \mathcal{A}_*$ (see (3.16) *and Assumption* 4.6.1*);*

(ii) s_k^a is the minimum norm solution to the traditional SQO subproblem (3.19)*, restated below:*

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \;\; g_k^T s + \tfrac{1}{2} s^T H_k s \;\; \text{subject to} \;\; c_k + J_k s \geq 0;$$

and

(iii) $\Delta q^\phi(s_k^a; x_k, H_k, \bar{\sigma}) \geq \gamma_{c\phi} \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \bar{\sigma})$.

Proof. Many results from [45] directly apply here since the step computation is the same. In particular, part (i) is equivalent to [45, Lemma 3.7(ii)], part (ii) is

established by [45, proof of Theorem 3.12], and part (iii) follows from [45, proof of Theorem 3.12, equation (2.7), and equation (2.9)]. □

We now give an asymptotic property of the accelerator steps associated with successful iterates.

Lemma 32. *Let Assumptions* 3.3.3*,* 4.6.1*, and* 4.6.2 *hold and define*

$$\mathcal{S}_2 = \{k \in \mathcal{S} : x_{k+1} = x_k + s_k^a \ \textit{and} \ \ k+1 \notin \mathcal{S}\}.$$

Then, either the set $\mathcal{S}_2$ is finite or

$$\lim_{k \in \mathcal{S}_2 \to \infty} \frac{\phi(x_k; \bar{\sigma}) - \phi(x_k + s_k^a + s_{k+1}^a; \bar{\sigma})}{\Delta q^{\phi}(s_k^a; x_k, H_k, \bar{\sigma})} = 1, \tag{4.8}$$

with $\Delta q^{\phi}(s_k^a; x_k, H_k, \bar{\sigma}) > 0$ for all $k \in \mathcal{S}$ sufficiently large.

Proof. The limit in (4.8) follows from [73, Theorem 15.3.7], and from the fact $\Delta q^{\phi}(s_k^a; x_k, H_k, \bar{\sigma}) > 0$ for all $k \in \mathcal{S}$ sufficiently large can be found in the first line of the proof of [73, Theorem 15.3.7]. □

The next result gives some properties of the iteration following a specific feasible successful iteration.

Lemma 33. *Let $w_* = (x_*, y_*)$ be the minimizer for problem* (2.1) *that satisfies Assumption* 4.6.1*. Also, let Assumptions* 3.3.3 *and* 4.6.2 *hold, and $\gamma_{c\phi}$ and $\delta > 0$*

be as defined in Lemma 31. *It follows that if* $max\_fails > 0$, $\mathcal{P}$-$mode =$ **false** *at the beginning of iteration* k, $k \in \mathcal{S}$ *is sufficiently large,* $k+1 \notin \mathcal{S}$, $w_k = (x_k, y_k) \in \mathcal{B}_\delta(w_*)$, *and* $v(x_k) = 0$, *then* $x_{k+1} = x_k + s_k^a$ *and both* (4.3a) *and* (4.3b) *are satisfied, with* k *replaced by* $k+1$, *by the pair* $(1, s_{k+1}^a)$.

Proof. Since $k \in \mathcal{S}$, Algorithm 2 starts iteration $k+1$ with *fails* $= 0$, and thus $x_{k+1} = x_k + s_k^a$ as $max\_fails > 0$, which is the first result. We also note that since $k+1 \notin \mathcal{S}$ that $R(k+1) = k$.

It follows from (3.8), $v(x_k) = 0$, and the definition of s_k^s that $0 = v(x_k) = \Delta\ell^v(s_k^s; x_k) = \Delta\ell^v(s_k; x_k)$. We also have, from Lemma 5(ii), the inequalities $\Delta\ell^\phi(s_k; x_k, \bar{\sigma}) \geq \frac{1}{2} s_k^{pT} B_k s_k^p \geq 0$, which combined show that

$$\Delta\ell^f(s_k; x_k) = \Delta\ell^\phi(s_k; x_k, \bar{\sigma}) \geq 0 = \gamma_v \Delta\ell^v(s_k; x_k),$$

where we used $\Delta\ell^v(s_k; x_k) = 0$ to obtain the first equation. This shows that (4.3a) is satisfied with k replaced by $k+1$ since $R(k+1) = k$.

Next, note that $v(x_k) = \Delta\ell^v(s_k^s; x_k) = 0$ implies that problem (3.4a) is solved during iteration k and therefore $c_k + J_k s_k^p \geq 0$. Using this and $c_k \geq 0$ allows us to conclude that $c_k + \alpha J_k s_k^p \geq 0$ for all $\alpha \in [0, 1]$. Combining this fact with $s_k = s_k^p$ (since $\tau_k = 1$ in (3.7)) shows that s_k^{cf} and $s_k^{c\phi}$ are also linearly feasible, i.e., $c_k + J_k s_k^{cf} \geq 0$ and $c_k + J_k s_k^{c\phi} \geq 0$.

Pick $\kappa \in (\gamma_f/\gamma_{c\phi}, 1)$, which is possible since $0 < \gamma_f < \gamma_{c\phi} < 1$ by definition

of $\gamma_{c\phi}$ (see Lemma 31). Then, it follows from definition of ϕ, $x_{k+1} = x_k + s_k^a$, $v(x_k) = 0$, $v(x_{k+1} + s_{k+1}^a) \geq 0$, Lemma 32, Lemma 31(iii), the definitions of $s_k^{c\phi}$ and s_k^{cf}, the definition of Δq^ϕ, the fact that s_k^{cf} is linearly feasible, and our selection of κ that

$$\begin{aligned}
f(x_k) - f(x_{k+1} + s_{k+1}^a) &= \phi(x_k; \bar{\sigma}) - \phi(x_{k+1} + s_{k+1}^a; \bar{\sigma}) - \bar{\sigma}\big(v(x_k) - v(x_{k+1} + s_{k+1}^a)\big) \\
&\geq \phi(x_k; \bar{\sigma}) - \phi(x_{k+1} + s_{k+1}^a; \bar{\sigma}) \\
&\geq \kappa \Delta q^\phi(s_k^a; x_k, H_k, \bar{\sigma}) \\
&\geq \kappa \gamma_{c\phi} \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \bar{\sigma}) \geq \kappa \gamma_{c\phi} \Delta q^\phi(s_k^{cf}; x_k, H_k, \bar{\sigma}) \\
&= \kappa \gamma_{c\phi} \Delta q^f(s_k^{cf}; x_k, H_k) + \kappa \gamma_{c\phi} \bar{\sigma}\big(v(x_k) - \big\| \big[c_k + J_k s_k^{cf}\big]^- \big\|_1\big) \\
&\geq \gamma_f \Delta q^f(s_k^{cf}; x_k, H_k) \\
&\geq \gamma_f \min\big(\Delta \ell^f(s_k, x_k), \Delta q^f(s_k^{cf}; x_k, H_k)\big)
\end{aligned}$$

for k sufficiently large. This is equivalent to (4.3b), with k replaced by $k+1$, since $R(k+1) = k$. □

By contrast, we now consider properties of the pair of iterations following a specific infeasible successful iteration.

Lemma 34. *Let $w_* = (x_*, y_*)$ be the minimizer for problem* (2.1) *that satisfies Assumption* 4.6.1. *Also, let Assumptions* 3.3.3 *and* 4.6.2 *hold, and $\gamma_{c\phi}$ and $\delta > 0$ be defined as in Lemma* 33. *Furthermore, suppose that $max\_fails > 0$, $\mathcal{P}$-$mode =$*

false *at the beginning of iteration* k, $k \in \mathcal{S}$ *is sufficiently large,* $k+1 \notin \mathcal{S}$, $w_k = (x_k, y_k) \in \mathcal{B}_\delta(w_*)$, *and* $v(x_k) > 0$. *It follow that if* (4.3a) *is satisfied and* (4.3b) *is violated (both with* k *replaced by* $k+1$*) by the pair* $(1, s^a_{k+1})$, *then* $x_{k+1} = x_k + s^a_k$, $(1, s^a_{k+1})$ *is a b-pair during iteration* $k+1$, $k+2 \in \mathcal{S}$, *and* $x_{k+2} = x_k + s^a_k + s^a_{k+1}$.

Proof. The first result follows (as in the previous proof) as Algorithm 2 sets $x_{k+1} = x_k + s^a_k$ because $k \in \mathcal{S}$ and $max\_fails > 0$, and $R(k+1) = k$ since $k+1 \notin \mathcal{S}$.

We show that (4.6) is satisfied, with k replaced by $k+1$, by the pair $(1, s^a_{k+1})$. It follows from Lemma 32, Lemma 31(iii), the choice of κ, and (4.7) that

$$\begin{aligned} \phi(x_k; \bar{\sigma}) - \phi(x_{k+1} + s^a_{k+1}; \bar{\sigma}) &\geq \kappa \Delta q^\phi(s^a_k; x_k, H_k, \bar{\sigma}) \\ &\geq \kappa \gamma_{c\phi} \Delta q^\phi(s^{c\phi}_k; x_k, H_k, \bar{\sigma}) \geq \gamma_\phi \rho^\phi_k \end{aligned} \tag{4.9}$$

for sufficiently large k, which shows that (4.6), with k replaced by $k+1$, is satisfied by $(1, s^a_{k+1})$.

We next show that (4.5) is satisfied, with k replaced by $k+1$, by the pair $(1, s^a_{k+1})$. Define $\mathcal{G} = \{i : c_i(x_k) < 0\}$ and $\mathcal{H} = \{i : c_i(x_k) \geq 0\}$ and observe that Lemma 31(i)–(ii) and the definition of s^a_k (see (3.17) and (3.18)) imply that $c_k + J_k s^p_k \geq 0$. We also know that $s_k = s^p_k$ since $\tau_k = 1$ (see (3.7)) so that $c_k + J_k s_k \geq 0$. It then follows that $c_i(x_k) + \alpha \nabla c_i(x_k)^T s_k \geq 0$ for all $i \in \mathcal{H}$ and $\alpha \in [0,1]$, and that $\nabla c_i(x_k)^T s_k \geq -c_i(x_k) > 0$ for all $i \in \mathcal{G}$. Combining these conditions together shows that $[c_i(x_k)]^- - \left[c_i(x_k) + \alpha \nabla c_i(x_k)^T s_k\right]^- \geq 0$ for all i and $\alpha \in [0,1]$, and after

summing over all the constraints and using the definition of s_k^{cf}, leads to

$$\Delta \ell^v(s_k^{cf}; x_k) = \left\| [c_k]^- \right\|_1 - \left\| \left[c_k + J_k s_k^{cf}\right]^- \right\|_1 \geq 0. \tag{4.10}$$

Now, choose any $\kappa \in (\max\{\gamma_f, \gamma_\phi\}/\gamma_{c\phi}, 1)$. This is possible since $\max\{\gamma_f, \gamma_\phi\} < \gamma_{c\phi} < 1$ by the definition of $\gamma_{c\phi}$ in Lemma 31. Note that $\Delta q^f(s_k^{cf}; x_k, H_k) \geq 0$ by construction, We consider two cases.

Case 1: $\Delta q^f(s_k^{cf}; x_k, H_k) > 0$**.** We may use the definition of ϕ, Lemma 32, the supposition that (4.3b) is violated (with k replaced by $k+1$) by the pair $(1, s_{k+1}^a)$, $R(k+1) = k$, Lemma 31(iii), the definitions of ρ_k^f, $s_k^{c\phi}$, s_k^{cf}, and Δq^ϕ, (4.10), the selection of κ, and $\Delta q^f(s_k^{cf}; x_k, H_k) > 0$ to conclude that

$$\begin{aligned}
&\bar{\sigma}\big(v(x_k) - v(x_{k+1} + s_{k+1}^a)\big) \\
&\qquad = \phi(x_k; \bar{\sigma}) - \phi(x_{k+1} + s_{k+1}^a; \bar{\sigma}) - \big(f(x_k) - f(x_{k+1} + s_{k+1}^a)\big) \\
&\qquad \geq \kappa \Delta q^\phi(s_k^a; x_k, H_k, \bar{\sigma}) - \gamma_f \rho_k^f \\
&\qquad \geq \kappa\gamma_{c\phi} \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \bar{\sigma}) - \gamma_f \Delta q^f(s_k^{cf}; x_k, H_k) \\
&\qquad \geq \kappa\gamma_{c\phi} \Delta q^\phi(s_k^{cf}; x_k, H_k, \bar{\sigma}) - \gamma_f \Delta q^f(s_k^{cf}; x_k, H_k) \\
&\qquad \geq \kappa\gamma_{c\phi} \Big(\Delta q^f(s_k^{cf}; x_k, H_k) + \bar{\sigma} \Delta \ell^v(s_k^{cf}; x_k)\Big) - \gamma_f \Delta q^f(s_k^{cf}; x_k, H_k) \\
&\qquad \geq (\kappa\gamma_{c\phi} - \gamma_f)\, \Delta q^f(s_k^{cf}; x_k, H_k) > 0 \ \text{ for all sufficiently large } k,
\end{aligned}$$

so that (4.5) is satisfied, with k replaced by $k+1$, by the pair $(1, s_{k+1}^a)$.

Case 2: $\Delta q^f(s_k^{cf}; x_k, H_k) = 0$**.** Since $v(x_k) > 0$ by assumption, we have $\Delta \ell^v(s_k^s; x_k) > 0$ because otherwise Algorithm 2 would have exited in Line 12. It then follows from the definition of ϕ, the fact that we have already shown that (4.6) (with k replaced by $k+1$) is satisfied by $(1, s_{k+1}^a)$, $R(k+1) = k$, the assumption that (4.3b) is violated (with k replaced by $k+1$) by the pair $(1, s_{k+1}^a)$, $\Delta q^f(s_k^{cf}; x_k, H_k) = 0$, the definition of ρ_k^ϕ, Lemma 6, $\Delta \ell^v(s_k^s; x_k) > 0$, and (3.21) that

$$\begin{aligned}
&\bar{\sigma}\big(v(x_k) - v(x_{k+1} + s_{k+1}^a)\big) \\
&\qquad = \phi(x_k; \bar{\sigma}) - \phi(x_{k+1} + s_{k+1}^a; \bar{\sigma}) - \big(f(x_k) - f(x_{k+1} + s_{k+1}^a)\big) \\
&\qquad \geq \gamma_\phi \rho_k^\phi - \gamma_f \rho_k^f \\
&\qquad \geq \gamma_\phi \rho_k^\phi = \gamma_\phi \min\big\{ \Delta \ell^\phi(s_k; x_k, \bar{\sigma}),\, \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \bar{\sigma}) \big\} > 0
\end{aligned}$$

so that (4.5) is again satisfied, with k replaced by $k+1$, by the pair $(1, s_{k+1}^a)$.

Since we have shown that (4.5) and (4.6) (with k replaced by $k+1$) are satisfied by $(1, s_{k+1}^a)$, and we know from assumption that (4.3a) is satisfied and (4.3b) is violated (both with k replaced by $k+1$) by the pair $(1, s_{k+1}^a)$, we may conclude that $(1, s_{k+1}^a)$ is a b-pair during iteration $k+1$, as claimed. It then follows immediately from the construction of Algorithm 2 that $k+2 \in \mathcal{S}$ and that $x_{k+2} = x_{k+1} + s_{k+1}^a$, which completes the proof since $x_{k+1} = x_k + s_k^a$. □

We may now state our local convergence result.

Theorem 8. *Let* $w_* = (x_*, y_*)$ *be the minimizer for problem* (2.1) *that satisfies*

Assumption 4.6.1. *Furthermore, let Assumptions* 3.3.3 *and* 4.6.2 *hold,* $\delta > 0$ *be given as in Lemma* 33*, and* $max\_fails > 0$*. Then, the iterates* $\{x_k\}$ *and* $\{y_k\}$ *converge to* x_* *and* y_* *at a Q-superlinear and R-superlinear rate, respectively. Moreover, if* $\nabla^2_{xx} L(x, y)$ *is Lipschitz continuous in a neighborhood of* (x_*, y_*)*, then they converge at a Q-quadratic and R-quadratic rate, respectively.*

Proof. We first show that for all sufficiently large $k \in \mathcal{S}$ such that $\mathcal{P}\text{-}mode =$ **false** during iteration k, we have $x_{k+1} = x_k + s^a_k$ and either (i) $k + 1 \in \mathcal{S}$ or (ii) $k + 2 \in \mathcal{S}$ and $x_{k+2} = x_k + s^a_k + s^a_{k+1}$. The fact that $x_{k+1} = x_k + s^a_k$ follows from $k \in \mathcal{S}$, *fails* $= 0$, $max\_fails > 0$, and the structure of Algorithm 2. To prove the rest, we suppose that (i) does not hold and proceed to prove that (ii) holds.

So, suppose that (i) does not hold, i.e., that $k + 1 \notin \mathcal{S}$. Our first goal is to use [56, Lemmas 4.5 and 4.6, Theorem 4.7] to establish that $x_k + s^a_k + s^a_{k+1}$ is acceptable to the augmented filter. We may use these results since the conditions that define our filter are weaker in comparison to the conditions that define the filter in [56]. Specifically, if a point is acceptable to the filter given by [56, equation (10)], then it is also acceptable to our filter given by Definition 12. This is easy to see since the inequalities in Definition 12 use max/min terms based on quantities derived from the steering subproblem to formulate weaker, and more practical, conditions that define the filter. With this observation, one may now follow the proofs of [56, Lemmas 4.5 and 4.6, Theorem 4.7] to show that $x_k + s^a_k + s^a_{k+1}$ is acceptable to the augmented filter under the

current assumptions.

We now consider three cases.

Case 1: condition (4.3a) is not satisfied at iteration $k+1$. Since (4.2) holds and we already proved that $x_k + s_k^a + s_{k+1}^a$ is acceptable to the augmented filter, we may conclude that $(1, s_{k+1}^a)$ is a v-pair during iteration $k+1$, $k+2 \in \mathcal{S}$, and $x_{k+2} = x_{k+1} + s_k^a + s_{k+1}^a$, as claimed.

Case 2: conditions (4.3a) and (4.3b) are both satisfied at iteration $k+1$. Combining this with the fact that we already proved that $x_k + s_k^a + s_{k+1}^a$ is acceptable to the augmented filter, we may conclude that $(1, s_{k+1}^a)$ is an o-pair during iteration $k+1$, $k+2 \in \mathcal{S}$, and $x_{k+2} = x_{k+1} + s_k^a + s_{k+1}^a$, as claimed.

Case 3: condition (4.3a) holds but condition (4.3b) is violated at iteration $k+1$. Under the current assumptions, it follows from Lemma 33 that $v(x_k) > 0$, or else there would be a contradiction. We may now use Lemma 34 to conclude that $(1, s_{k+1}^a)$ is a b-pair during iteration $k+1$, $k+2 \in \mathcal{S}$, and $x_{k+2} = x_{k+1} + s_k^a + s_{k+1}^a$, as claimed.

Since one of the above three cases must occur, we have established that part (ii) holds. To summarize, we have shown that for all sufficiently large $k \in \mathcal{S}$ such that $\mathcal{P}\text{-}mode =$ **false** at the beginning of iteration k, we have $x_{k+1} = x_k + s_k^a$ and either (i) $k+1 \in \mathcal{S}$ or (ii) $k+2 \in \mathcal{S}$ and $x_{k+2} = x_k + s_k^a + s_{k+1}^a$.

Next, we show a similar result holds when $\mathcal{P}\text{-}mode =$ **true** at the beginning of iteration k. Specifically, we show that for all sufficiently large $k \in \mathcal{S}$ such

that $\mathcal{P}$-$mode$ = **true** at the beginning of iteration k, we have $x_{k+1} = x_k + s_k^a$ and either (i) $k+1 \in \mathcal{S}$ or (ii)'$k+2 \in \mathcal{S}$ and $x_{k+2} = x_k + s_k^a + s_{k+1}^a$. The fact that $x_{k+1} = x_k + s_k^a$ follows from $k \in \mathcal{S}$, $max\_fails > 0$, and the structure of Algorithm 2. To prove the rest, we suppose that (i) does not hold and proceed to prove that (ii) holds. When (i) does not held, then the same argument that lead to (4.9) may again be used to show that $(1, s_{k+1}^a)$ is a p-pair during iteration $k+1$ and therefore $k+2 \in \mathcal{S}$ and $x_{k+2} = x_{k+1} + s_k^a + s_{k+1}^a$, as claimed.

We have shown that $x_{k+1} = x_k + s_k^a$ for all k sufficiently large. In light of (3.19), this means that Algorithm 2 accepts the traditional SQO step at (x_k, y_k) for all k sufficiently large. Since this was the precise condition required to establish [45, Theorem 3.12], we have the same conclusions as in that theorem, which completes the proof of Theorem 8. □

4.7 Numerical results

We present numerical experiments performed with a basic MATLAB implementation of Algorithm 2 (which will henceforth be called FiSQO) on the set of low-dimensional CUTEst [7] problems. We comment up front that we do not compare FiSQO to methods that use a feasibility restoration phase. This decision was made for a variety of reasons. First, different filter methods use different formulations of the restoration phase, which makes it difficult, if not

impossible, to make any general statements about them. Second, the implementation of a restoration phase often, if not always, includes heuristics that are designed to improve the general performance. Finally, a key motivation for our work is the design of a filter method that does not use a restoration phase since it is generally accepted that it is the most dissatisfying aspect of such filter-based methods.

With the previous remarks in mind, we now mention that the purpose of our numerical experiments is to validate the general effectiveness of FiSQO and to investigate any numerical anomalies associated with b-pairs. We focus on such pairs since, roughly, they serve as our alternative to feasibility restoration. With respect to both goals, we find it instructive to compare FiSQO to our own implementation of a penalty SQO line search method (henceforth referred to as PenSQO). Since we have complete control over both algorithms, we are able to isolate any aspect of interest (e.g., the influence of b-pairs), design and perform revealing numerical tests, and confidently present the numerical results. As described in the next section, the only difference between the two methods is in the step acceptance criteria.

4.7.1 Implementation details

During each iteration, FiSQO requires the solution of a linear program and a quadratic program in order to obtain the steering step s_k^s and the predic-

tor step s_k^p in lines 10 and 13, respectively. In our implementation, they were obtained by using the primal simplex active-set solver in Cplex [78], which we generally found to be reliable. The formulation of the predictor subproblem (3.4) required a positive-definite matrix B_k, which we obtained by a modified Newton strategy as follows. First, we computed the spectral decomposition of H_k, i.e., $H_k = V_k D_k V_k^T$, where V_k is an orthogonal set of eigenvectors and D_k is a diagonal matrix of eigenvalues for H_k. Second, we set $\varepsilon = 1$ if $H_k = 0$, and $\varepsilon = \|H_k\|_2 / 10^8$ otherwise. We then obtained the desired positive-definite matrix as $B_k = V_k \widehat{D}_k V_k^T$, where the diagonal entries of the diagonal matrix $\widehat{D}_k$ were given as

$$[\widehat{D}_k]_{ii} = \begin{cases} [D_k]_{ii} & \text{if } [D_k]_{ii} \geq \varepsilon, \\ -[D_k]_{ii} & \text{if } [D_k]_{ii} \leq -\varepsilon, \\ \varepsilon & \text{otherwise.} \end{cases}$$

It is not difficult to see that the matrix B_k is positive definite with a condition number bounded by 10^8. We note that this strategy was chosen for simplicity and that it is not suitable for large-scale problems. In the large-scale setting, choosing B_k based on limited-memory quasi-Newton updates, e.g., L-BFGS [67], would be appropriate. Nonetheless, we remain satisfied with this simple choice since it was used by both FiSQO and PenSQO in our experiments.

The value of τ_k needed in line 16 was obtained by performing backtracking (starting with an initial guess of $\tau_k = 1$) until condition (3.8) was satisfied. We

note that although Algorithm 2 states that τ_k should be computed as the *largest* value on $[0, 1]$ that satisfies (3.8), this is not necessary. The simple backtracking procedure that we implemented ensures that the sequence $\{\tau_k\}$ possesses the properties required to obtain the global and local convergence results established in this chapter (e.g., using an initial guess of $\tau_k = 1$). The final aspect of the search direction was the computation of an accelerator step s_k^a in line 18. We defined s_k^a via (3.17), where s_k^a was computed from subproblem (3.18) in the following way. We first used the backslash operator in MATLAB in an attempt to solve the linear system

$$\begin{pmatrix} H_k & [J_k]_{\mathcal{A}_k}^T \\ [J_k]_{\mathcal{A}_k} & 0 \end{pmatrix} \begin{pmatrix} s_k^a \\ -[y_k]_{\mathcal{A}_k} \end{pmatrix} = - \begin{pmatrix} g_k + H(x_k, y_k^p) s_k^p \\ 0 \end{pmatrix}, \tag{4.11}$$

where $\mathcal{A}_k$ is defined by (3.16). The motivation for considering this particular linear system is that if $[J_k]_{\mathcal{A}_k}$ has full row rank, H_k is positive definite when restricted to the null space of $[J_k]_{\mathcal{A}_k}$, and δ_a is sufficiently large, then s_k^a will, in fact, be the unique minimizer to (3.18). Of course, a solution to (4.11) may not exist, and even when it does exist, it is a solution to (3.18) only when H_k is positive definite when restricted to the null space of $[J_k]_{\mathcal{A}_k}$ and δ_a is sufficiently large. Therefore, if MATLAB returned a "NaN" in any component of s_k^a or $[y_k]_{\mathcal{A}_k}$, we reset both of them to zero, and continued with the iteration. Otherwise, we proceeded to perform a scaling of s_k^a to make it have norm bounded by δ_a, i.e.,

to make it satisfy the trust-region constraint in (3.18), but we did not scale the associated Lagrange multiplier estimate $[y_k]_{\mathcal{A}_k}$. This procedure may result in a step s_k^a that does not solve (3.18), but nonetheless is a reasonable strategy for calculating an approximate solution in a cost-efficient manner. We also comment that, since the purpose of the accelerator step is to accelerate local convergence, this change has no effect on the global convergence properties of FiSQO. Moreover, since the predictor step ultimately predicts via $\mathcal{A}_k$ the constraints that are active at a local minimizer (under Assumption 4.6.1), our procedure for computing s_k^a will asymptotically give the unique minimizer to problem (3.18). In particular, this means that our local convergence theory remains valid.

The computations just described, as well as most of the other steps taken in FiSQO, use control parameters and require the choice of initial values; we used the values in Table 4.1. These choices were made based on our experience of developing other nonlinear optimization algorithms, and no fine-tuning of our choices was attempted for this basic implementation. Although not stated in Algorithm 2, in our implementation we imposed an iteration limit of 10000 iterations and a CPU time limit of 10 minutes.

Finally, we discuss the termination tests used by both FiSQO and PenSQO. First, we declared x_k to be an infeasible stationary point, as predicated by

Table 4.1: Control parameters and initial values for FiSQO and PenSQO.

Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
η_v	10^{-3}	η_σ	10^{-6}	η_ϕ	10^{-3}	$\sigma_{\texttt{inc}}$	5
γ	10^{-3}	γ_v	10^{-3}	γ_f	10^{-4}	γ_ϕ	10^{-4}
β	0.99	ξ	0.5	$\delta_{\texttt{min}}$	1	$\delta_{\texttt{max}}$	10^{+4}
δ_a	10^{+2}	σ_0	10	δ_0	10^{+2}	$\tau_{\texttt{stop}}$	10^{-5}

line 12 of Algorithm 2, if it satisfied

$$v_k \geq 100\,\tau_{\text{stop}} \quad \text{and} \quad \Delta\ell^v(s_k^s; x_k) \leq 10^{-12}, \tag{4.12}$$

where the value of the termination tolerance τ_{stop} is given in Table 4.1. It is clear that these conditions were motivated by (3.3), but designed to account for numerical error. Finally, we concluded that x_k is a solution to (2.1) in line 15 if

$$v_k \leq \tau_{\text{stop}} \;\text{ and }\; \Delta q^\phi(s_k^p; x_k, \sigma_k) \leq 10^{-12} \tag{4.13}$$

were satisfied, or if the primal-dual pair (x_k, y_k) satisfied the approximate KKT condition

$$\|F_{\text{KKT}}(x_k, y_k)\|_\infty \leq \tau_{\text{stop}}. \tag{4.14}$$

In our implementation, we set $y_k \leftarrow y_k^p$ if $\|F_{\text{KKT}}(x_k, y_k^p)\|_\infty < \|F_{\text{KKT}}(x_k, y_k^a)\|_\infty$, and $y_k \leftarrow y_k^a$ otherwise. Also, although we did not explicitly check for unboundedness, it did not seem to affect our numerical results. Nonetheless, production

quality software should include such a check since it is a possible outcome.

Our penalty-SQO algorithm PenSQO was obtained by making two simple modifications to FiSQO. First, $\mathcal{P}$-$mode$ was initialized to the value true. Second, anytime the condition in line 28 tested true, $\mathcal{P}$-$mode$ was not set to the value false. In short, our modification forced $\mathcal{P}$-$mode$ to always have the value true. Consequently, the only difference between FiSQO and PenSQO is in the step acceptance criteria.

4.7.2 Collection of CUTEst test problems

We tested our MATLAB implementations of FiSQO and PenSQO on two subsets of problems from the CUTEst [7] collection. The first subset was obtained by first identifying those CUTEst problems with at least one general constraint (i.e., $m \geq 1$) and at most 100 variables and constraints (i.e., $\max\{m, n\} \leq 100$). From this set, we removed problems *deconvc*, *discs*, *hs99exp*, *lakes*, *tr04x4*, *tr06x2*, *truspyr1*, and *truspyr2* since the Cplex solver "hung" and prevented the algorithms from continuing, which left us with a total of 301 test problems. A detailed presentation of the results on a problem-by-problem basis may be found in the Appendix (see Appendix Tables A.1 and A.2).

Here, to illustrate the performance of our software, we use performance profiles as introduced by Dolan and Moré [79] to give visual comparisons of numerical performance. Consider a performance profile that measures perfor-

mance in terms of the number of iterations until successful termination. In this case, if the graph associated with an algorithm passes through the point $(\alpha, 0.\beta)$, then it means that on $\beta\%$ of the problems, the number of iterations required by the algorithm was less than α times the number of iterations required by the algorithm that required the fewest. Therefore, an algorithm with a higher value on the vertical axis may be considered more efficient, whereas an algorithm on top at the far right may be considered more reliable. We note that for every profile, a problem was considered to be successfully solve if an approximate infeasible stationary point satisfying (4.12) or an approximate KKT pair (x_k, y_k) satisfying either (4.13) or (4.14) was found.

Figure 4.1 shows the results for the two line search algorithms FiSQO and PenSQO. We note, however, that these profiles were created after we removed additional problems from the test set. Specifically, we removed all (two in this case) problems for which at least one of FiSQO or PenSQO returned a value indicating a failure in the subproblem solver, or a value signaling a function evaluation error since they did not necessarily give any useful information about the algorithms. This left us with a total of 299 test problems used in the performance profiles. By inspecting the right-hand-side of the graphs, we can see that FiSQO and PenSQO are similar in terms of robustness, with a slight edge going to FiSQO. Taken in tandem, the two graphs indicate that the number of function evaluations are significantly less for FiSQO than for PenSQO, while

the difference in the number of iterations (equivalently, the number of gradient evaluations) is less significant. This phenomenon makes sense because acceptance based on the combination of v-, o-, b-, and p-pairs is more relaxed when compared to acceptance based on the penalty function alone, which translates into substantially fewer (overall) function evaluations during the line search. However, since the difference in the number of iterations is less significant, we may conclude that, although more trial steps are accepted by FiSQO, many of them make less progress toward a solution when compared to PenSQO.

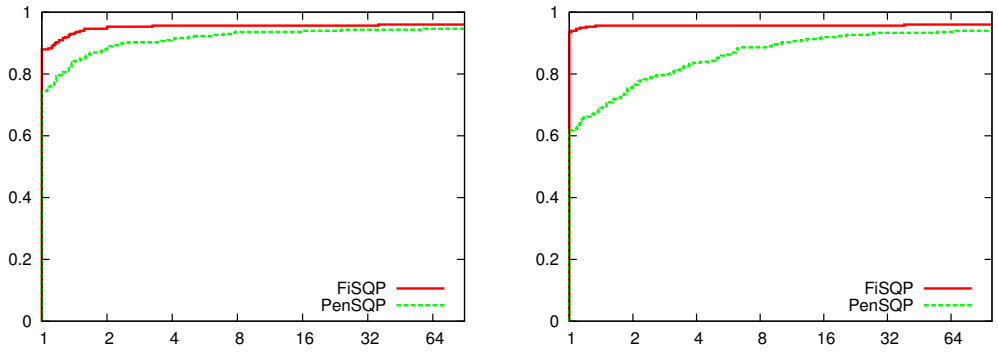

Figure 4.1: Performance profiles on the CUTEst problems with $m \geq 1$ and $\max\{m, n\} \leq 100$ for number of iterations (left) and function evaluations (right).

A few additional comments concerning these results are in order. First, we believe it is interesting to see the numerical trade-off between accepting more (on average lower quality) steps versus fewer (on average higher quality) steps; perhaps, this should not have been a surprise. Second, these results indicate that our implementation of FiSQO appears to be fairly robust and at least as efficient as PenSQO. Third, we could probably further improve the

FiSQO results by using (when $\mathcal{P}$-$mode$ has the value of true) a flexible penalty approach [58, 80–82], which would require a modification to the definition of a p-pair. It is important to emphasize that we are not claiming that FiSQO is better than a *flexible* penalty-SQO approach, but rather that FiSQO appears to be better than a *standard* penalty-SQO approach.

The second subset of test problems was the CUTEst problems with $m \geq 1$ and $100 < \max\{m, n\} \leq 1000$. As for the previous test set, we removed problems for which Cplex "hung", which included *a4x12*, *mss2*, *steenbrd tr011x3 tr05x5*, and *yorknet*. This resulted in a subset of 68 CUTEst test problems. (See Appendix Tables A.3 and A.4 for a detailed presentation of the results on a problem-by-problem basis.)

Figure 4.2 shows the results for algorithms FiSQO and PenSQO. As for the previous profiles, they were created after removing all (in this case 11) problems for which at least one of FiSQO or PenSQO returned a value indicating a failure in solving the subproblem, which left us with a total of 57 problems. Viewed together, they lead us to the same conclusion as for the previous test set, which was comprised of smaller problems: FiSQO needed significantly fewer function evaluations compared to PenSQO, but the difference in the number of iterations (equivalently, the number of gradient evaluations) was less significant.

To summarize, we believe that the numerical results presented in this section validate the effectiveness of FiSQO. Its ability to accept more steps signif-

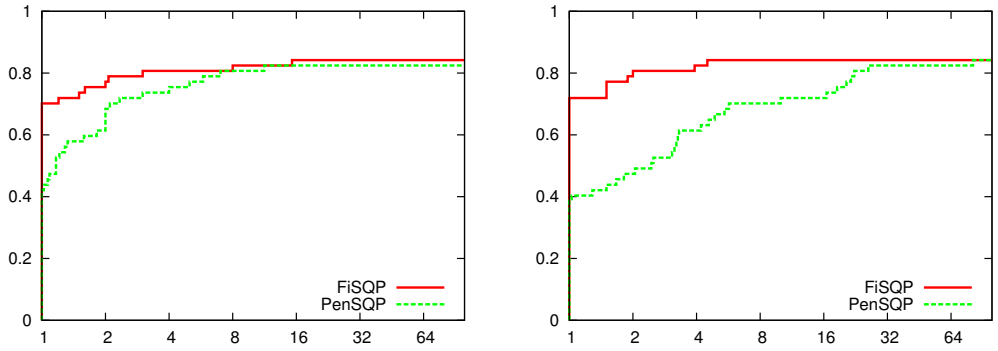

Figure 4.2: Performance profiles on the CUTEst problems with $m \geq 1$ and $100 < \max\{m, n\} \leq 1000$ for the number of iterations (left) and function evaluations (right).

icantly reduces the number of function evaluations needed by the line search procedure, while the decrease in the number of iterations/gradient evaluations is mild.

4.7.3 Gauging the influence of b- and p-pairs

Most filter algorithms have multiple sets of conditions that trigger feasibility restoration. For example, most (if not all) filter trust-region methods enter feasibility restoration if the trust-region subproblem is infeasible. Since our research has focused on avoiding such traditional restoration phases, we were careful about how our trial steps were calculated (e.g., the subproblems used in FiSQO are always feasible). Consequently, one might expect that if our particular trial step computation was used within a traditional filter method, then the frequency with which feasibility restoration would be needed would

be reduced. (For this discussion, we are ignoring the fact that it is unclear how global convergence of this fictitious algorithm would be established.) Thus, it is natural to wonder how important b- and p-pairs (essentially, our replacement for feasibility restoration) are to the overall success of FiSQO; that is the topic of this section.

We first consider the frequency with which b- and p-pairs occur. We may observe from Appendix Tables A.1 and A.4 that for $(32 + 11)/(301 + 66) \approx 12\%$ of the problems, b-pairs (consequently, also p-pairs) were computed. For these problems (i.e., those for which at least one b-pair was computed), it is also clear from Appendix Tables A.1 and A.4 that the number of p-pairs is typically very small; notable exceptions are problems *allinita*, *hatfldf*, *mss1*, and *table7*, none of which were successfully solved.

We have now seen that the number of problems for which FiSQO used at least one b-pair is significant. To investigate their importance, we first identified the problems for which FiSQO required at least one b-pair during the solution process. We then modified our MATLAB implementation of FiSQO so that it never allowed the acceptance of a b-pair, which, as a consequence, meant that p-pairs were never accepted (i.e., we only accepted v- and o-pairs). We stress that this modified algorithm (henceforth referred to as modFiSQO) does not enjoy the global convergence results established for FiSQO. Nonetheless, we are interested in the outcome of this experiment as it gives us additional

insight into the potential importance of b-pairs.

We ran modFiSQO on the problems identified in the previous paragraph and the results are presented in Table 4.2. (We included only those problems for which a solution was successfully obtained by at least one of FiSQO or modFiSQO, which left us with a total of 32 out of the 43 originally identified problems.) The values under "status" (a value of 0 indicates that optimality was achieved, while a value of 1 means that the maximum number of iterations was reached), "iters" (number of iterations), "fevals" (number of function evaluations), and "σ" (final penalty parameter) are given in the format a/b with a the value for FiSQO and b the value for modFiSQO. The column "status" shows that the modified algorithm modFiSQO solved the problems with the exception of *haldmads*. The two measures of efficiency (number of iterations and function evaluations), however, tell a different story. FiSQO required more iterations on only $4/32$ of the problems, three of which (*himmelp2*, *hs111lnp*, and *hs27*) required a single extra iteration and one (*hs92*) required 5 additional iterations. We also note that for those 4 problems, FiSQO did not require more function evaluations compared to modFiSQO. Over the entire set of problems, FiSQO required more function evaluations on $5/32$ instances; *acopp14* ($42/11$), *bt7* ($106/77$), *qpcboei1* ($17/16$), *qpcstair* ($28/12$), and *qpnboei2* ($45/39$). (Interestingly, for all 5 problems, FiSQO still required fewer iterations.) So, although FiSQO is occasionally less efficient in terms of the number of function evalua-

tions, the difference was not dramatic. In contrast, among the $27/32$ problems for which FiSQO was at least as efficient as modFiSQO in terms of the number of function evaluations, the difference was sometimes dramatic, e.g., *hs101* ($72/1879$), *tenbars1* ($76/525$), and *acopr57* ($16/101$), to name a few.

Table 4.2: Results for FiSQO/modFiSQO on the CUTEst problems of size $1 \leq m \leq \max\{m,n\} \leq 1000$ for which at least one b-pair was needed.

prob	m	n	status	iters	feval	σ
ACOPP14	68	38	0/0	5/6	42/11	1.2e+03/7.3e+02
ACOPR14	82	38	0/0	7/16	45/57	1.4e+02/2.0e+01
BT11	3	5	0/0	7/8	9/21	1.0e+01/1.0e+01
BT12	3	5	0/0	3/4	4/6	1.0e+01/1.0e+01
BT2	1	3	0/0	11/11	13/16	1.0e+01/1.0e+01
BT7	3	5	0/0	26/32	106/77	4.7e+02/2.1e+03
HALDMADS	42	6	0/1	215/6001	423/490438	1.0e+01/1.0e+01
HIMMELP2	1	2	0/0	15/14	23/23	1.0e+01/1.0e+01
HS101	5	7	0/0	27/149	72/1879	5.1e+03/5.3e+11
HS102	5	7	0/0	26/39	38/200	1.3e+04/2.9e+03
HS103	5	7	0/0	18/40	24/92	1.3e+03/5.3e+04
HS111LNP	3	10	0/0	18/17	21/21	2.0e+01/2.0e+01
HS27	1	3	0/0	22/21	150/155	1.0e+01/1.0e+01
HS29	1	3	0/0	6/7	11/13	1.0e+01/1.0e+01
HS61	2	3	0/0	6/7	8/18	1.0e+01/1.0e+01
HS77	2	5	0/0	9/9	13/14	1.0e+01/1.0e+01
HS88	1	2	0/0	16/19	24/69	1.1e+03/1.3e+03
HS89	1	3	0/0	16/25	29/54	1.9e+03/2.1e+03
HS92	1	6	0/0	21/16	56/62	1.3e+03/1.1e+03

PFIT2	3	3	0/0	131/132	1720/1730	1.7e+39/3.4e+39
SYNTHES1	6	6	0/0	4/5	6/14	1.0e+01/1.0e+01
TENBARS1	9	18	0/0	45/64	76/525	4.0e+01/6.5e+01
TENBARS2	8	18	0/0	49/89	98/945	2.0e+01/7.1e+01
TENBARS3	8	18	0/0	46/82	104/967	1.0e+01/1.2e+02
WATER	10	31	0/0	10/18	11/86	3.5e+02/3.5e+02
ACOPR57	331	128	0/0	7/22	16/101	3.8e+03/3.8e+03
GMNCASE1	300	175	0/0	1/2	3/5	1.0e+01/1.0e+01
LEUVEN7	946	360	0/0	2/3	3/23	1.0e+01/1.0e+01
QPCBOEI1	351	384	0/0	12/13	17/16	9.5e+05/9.5e+05
QPCSTAIR	356	467	0/0	9/10	28/12	6.5e+04/6.5e+04
QPNBOEI2	166	143	0/0	23/25	45/39	1.1e+05/1.1e+05
ZAMB2-8	48	138	0/0	11/23	19/43	1.0e+01/1.0e+01

Overall, we believe that these results show the practical importance of b-pairs. On the other hand, these results do not provide any clear evidence of their theoretical significance. With that said, it is difficult to imagine how any global convergence theory for modFiSQO could be established, unless additional modifications were introduced.

4.8 Conclusions and discussion

This chapter considered the local convergence properties and numerical performance of FiSQO: a nonmonotone variant of the filter line search algorithm proposed in Chapter 3 for which (in contrast to most filter methods) every subproblem is feasible. We proved, under standard assumptions, that the iterates computed by FiSQO converge superlinearly to a local minimizer. To accompany the theoretical results, we presented numerical results on subsets of the CUTEst problems. These results showed that FiSQO was more efficient than a penalty-SQO algorithm that used exactly the same step calculation procedure. In this manner, we were able to isolate the influence that our new step acceptance criteria had on numerical performance. The results were quite clear. First, our acceptance criteria based on o-, v-, b-, and p-pairs typically accepted more trial steps, which had the effect of significantly reducing (overall) the number of required function evaluations. Second, the number of iterations (equivalently, the number of gradient evaluations) was also reduced, but the difference was not as dramatic. We found it interesting to see the numerical trade-off between accepting more (on average lower quality) steps versus fewer (on average higher quality) steps. To further understand the importance of b- and p-pairs (essentially, our substitute for feasibility restoration), we performed the following experiment. We first identified the test problems

for which FiSQO used at least one b-pair during the solution process. We solved those problems with a modified variant of FiSQO, called modFiSQO, that differed by not allowing b-pairs to be accepted during the line search. The results showed that modFiSQO performed substantially worse (in general), which validated the numerical importance of b-pairs. Since modFiSQO solved all except one of the problems, there was no clear numerical evidence to suggest a theoretical advantage (in terms of convergence guarantees) for FiSQO.

We did not compare FiSQO to methods that use a feasibility restoration phase. This decision was made for a variety of reasons. First, different filter methods use different formulations of the restoration phase, which makes it difficult to make any general statements about them. Second, the implementation of a restoration phase often includes heuristics that are designed to improve the general performance. Third, a key motivation for our work is the design of a filter method that does not use a restoration phase since it is generally considered to be the most dissatisfying aspect of such filter-based methods.

It is interesting to note that there was little numerical difference between our monotone and nonmonotone algorithms. This contrasts the typical difference between monotone and nonmonotone penalty function methods, for which nonmonotone variants routinely outperform their monotone counterparts. Our monotone method appears to be less susceptible to the Maratos effect because of our carefully integrated filter and penalty function acceptance tests. Of

course, the Maratos effect can still affect our monotone variant, but in this chapter we have established that this is not a concern for our nonmonotone algorithm under common assumptions.

This work showed that, by pairing carefully constructed trial steps with b- and p-pairs, it is possible to define a convergent filter method that does not require feasibility restoration. We suspect that a disadvantage of our approach is that methods, such as filter-SQP [41], would typically perform better on infeasible problems, a feature directly attributed to feasibility restoration. We believe, however, that recent advances in feasibility detection [83] could be used within our framework and perhaps reduce, if not entirely mitigate, this disadvantage.

Chapter 5

A solver for the SQO subproblem

5.1 Motivation

A key challenge for SQO algorithms is the need to solve its eponymous quadratic problem (QP) during each iteration. These QPs are the source of much of the computation burden that SQO algorithms typically face, and they limit the ability of the algorithms to tackle huge-scale problems. Thus, the efficiency and scalability of SQO algorithms are highly dependent on both the characteristics and formulation of the QP, and the QP solver that is utilized.

The restoration-free filter SQO algorithms described in Chapters 3 and 4 require the solution of a linear program (the steering subproblem (3.1)), the solution of a strictly convex QP (the predictor subproblem (3.4)), and the approximate solution to an equality-constrained QP (the accelerator subprob-

lem (3.18)) during each iteration. Current linear optimization solvers (such as [78, 84–88]) can tackle problems with perhaps millions of variables and constraints, and equality constrained QPs can be solved by applying matrix-free projection iterative methods on (some form of) their KKT system of equations. However, while there are many very good convex QP solvers that are currently available (e.g. [78, 84, 87, 88]), they often do not scale as well as the linear optimization solvers and are usually not matrix-free (though recent developments by Gondzio [89] are very promising).

Thus, we want to explore an alternate approach to solving the predictor subproblem that hopefully is scalable and efficient. Specifically, we want to solve the dual of the predictor subproblem, taking advantage of the fact that the dual of a strictly convex constrained QP can be formulated as a convex (though usually not strictly convex) bound-constrained QP. We can then apply a recently developed algorithm proposed by Mohy-ud-Din and Robinson [8] for nonconvex bound-constrained quadratic optimization. As an extension of a method by Dostál and Schöberl [9], this algorithm is desirable because it is an iterative method that relies only on matrix-vector products.

We want to explore the efficiency of this dual approach and the situations in which it is competitive compared to the application of cutting edge convex quadratic optimization algorithms directly to the primal predictor problem.

5.2 The dual formulation

Recall that the predictor subproblem (3.4) takes different forms depending on the predicted decrease in the linear model of the constraint violation calculated from the steering subproblem. Specifically, if $\Delta\ell^v(s_k^s; x_k) = v(x_k)$, the subproblem we want to solve is (3.4a):

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; f_k + g_k^T s + \tfrac{1}{2} s^T B_k s \;\; \text{subject to} \;\; c_k + J_k s \geq 0,$$

where B_k is a positive-definite matrix. Since B_k is positive definite, we may write the dual formulation of (3.4a) as

$$\underset{y\in\mathbb{R}^m}{\text{minimize}} \;\; \hat{g}_k^T y + \tfrac{1}{2} y^T \hat{B}_k y \;\; \text{subject to} \;\; y \geq 0, \tag{5.1}$$

where

$$\hat{B}_k := J_k {B_k}^{-1} {J_k}^T \;\text{ and }\; \hat{g}_k := c_k - J_k {B_k}^{-1} g_k. \tag{5.2}$$

On the other hand, if $\Delta\ell^v(s_k^s; x_k) < v(x_k)$, then the predictor subproblem becomes (3.5):

$$\underset{s\in\mathbb{R}^n, r\in\mathbb{R}^n}{\text{minimize}} \;\; f_k + g_k^T s + \tfrac{1}{2} s^T B_k s + \sigma_k e^T r \;\; \text{subject to} \;\; c_k + J_k s + r \geq 0, \;\; r \geq 0.$$

The dual formulation of (3.5) is then

$$\underset{y\in\mathbb{R}^m}{\text{minimize}} \;\; \hat{g}_k^T y + \tfrac{1}{2} y^T \hat{B}_k y \;\; \text{subject to} \;\; \sigma e \geq y \geq 0, \tag{5.3}$$

where $\hat{B}_k$ and $\hat{g}_k$ are defined as above in (5.2) and $e \in \mathbb{R}^m$ is a vector of ones. The above problem differs from (5.1) only in that each coordinate of the dual solution is now bounded above by the penalty parameter.

As stated, these dual formulations require the computation of the inverse of B_k, which is undesirable. However, in the context of the predictor subproblem, there will be no such computational burden. Theoretically, the only restriction on $\{B_k\}$ is that they must be positive-definite with eigenvalues bounded uniformly away from zero and infinity; practically, we typically choose B_k to be either a scaled identity matrix [66] (for which the computation of the inverse becomes trivial) or a quasi-Newton update such as L-BFGS [67]. In the latter case, as part of the L-BFGS method, we can calculate matrix-vector products involving ${B_k}^{-1}$. Thus, we neither compute ${B_k}^{-1}$ nor form $\hat{B}_k$.

We note that $\hat{B}_k$ is positive-semidefinite and is singular when J does not have full row rank. However, since the primal predictor subproblems are constructed to always be feasible, the dual problems (5.1) and (5.3) will always have finite solutions. Once this solution (which we will call y_k^p) is obtained, we

can then calculate the solution to the primal predictor problem as follows:

$$s_k^p = -{B_k}^{-1} \left(g_k - J_k^T y_k^p \right). \tag{5.4}$$

Thus, we have exchanged a strictly convex QP with general linear constraints for a convex (often singular) QP with simple bound constraints. We can now utilize the nonconvex bound-constrained quadratic optimization solver developed by Mohy-ud-Din and Robinson [8].

5.3 Overview of the bound-constrained QP algorithm

We will now give an overview of the algorithm developed by Mohy-ud-Din and Robinson, but will omit the technical details. A full description of this algorithm can be found in [8].

Algorithms for bound-constrained quadratic problems (BCQP) generally try to predict the active set of variables at the optimal solution (i.e. the variables that are equal to their upper or lower bounds) and then, using these predictions, perform some sort of acceleration procedure within a reduced subspace. To identify the active variables, algorithms such as [90,91] compute cheap gradient projection steps. They then try to minimize the objective function in the

reduced space spanned by the complementary set of free variables through the use of conjugate gradient (CG) iterations.

In the process of iteratively calculating these projected gradient steps and then performing the subspace minimization, a key element is the criteria to terminate optimization in the reduced subspace. Many algorithms only provide heuristics, but in the context of strictly convex QPs with lower bounds on optimization variables, Dostál and Schöberl [9] present an adaptive condition, based on KKT conditions, that determines when the CG iterations should be terminated. Mohy-ud-Din and Robinson [8] extend this method to the nonconvex case that also allows for upper and lower bounds on the decision variables.

The adaptive condition used to determine the termination of CG iterations is based on the concept of proportioning. At each iteration k, given some current active set $\mathcal{A}_k$, the algorithm calculates a measure of optimality in the space of active variables and a measure of optimality in the space of free variables, and compares them. For a more precise definition of "measure of optimality," we need to repeat some definitions found in [8, Section 1.2]. We will use notation consistent with [8] for the remainder of this section (and only this section). The problem of interest [8, Problem (1.1)] now becomes

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; q(x) := \tfrac{1}{2}x^T H x - c^T x \;\; \text{subject to} \;\; l \leq x \leq u,$$

where H is a symmetric matrix (the Hessian matrix of $q(x)$) and the ith component of l and u satisfy $l_i \in \mathbb{R} \cup \{-\infty\}$ and $u_i \in \mathbb{R} \cup \{\infty\}$ respectively, with $l_i < u_i$.

We now define the active set and free set of variables [8, (1.3)] as

$$\mathcal{A}(x) := \{i : [x]_i \in \{l_i, u_i\}\} \quad \text{and} \quad \mathcal{F}(x) := \{1, 2, \ldots, n\} \setminus \mathcal{A}(x),$$

with the lower-active set and upper-active sets defined by

$$\mathcal{A}_l(x) := \{i \in \mathcal{A}(x) : [x]_i = l_i\} \quad \text{and} \quad \mathcal{A}_u(x) := \{i \in \mathcal{A}(x) : [x]_i = u_i\}.$$

At a point x, the free gradient $\varphi(x)$ and chopped gradient $\beta(x)$ [8, (1.4)] are defined component-wise as

$$[\varphi(x)]_i := \begin{cases} [g(x)]_i & \text{if } i \in \mathcal{F}(x); \\ 0 & \text{if } i \in \mathcal{A}(x); \end{cases} \quad \text{and} \quad [\beta(x)]_i := \begin{cases} 0 & \text{if } i \in \mathcal{F}(x); \\ [g(x)]_i^- & \text{if } i \in \mathcal{A}_l(x); \\ [g(x)]_i^+ & \text{if } i \in \mathcal{A}_u(x). \end{cases}$$

For any $\alpha > 0$, the reduced free gradient $\varphi(x, \alpha)$ [8, (1.5)] is defined as

$$\varphi(x, \alpha) := \begin{cases} \min\left\{\frac{x-l}{\alpha}, \varphi(x)\right\} & \text{if } \varphi(x) \geq 0; \\ \max\left\{\frac{x-u}{\alpha}, \varphi(x)\right\} & \text{if } \varphi(x) < 0; \end{cases}$$

and the reduced chopped gradient $\beta(x, \alpha)$ [8, (1.6)] is defined as

$$\beta(x,\alpha) := \begin{cases} \min\left\{\frac{x-l}{\alpha}, \beta(x)\right\} & \text{if } \beta(x) \geq 0; \\ \max\left\{\frac{x-u}{\alpha}, \beta(x)\right\} & \text{if } \beta(x) < 0. \end{cases}$$

If we let P_Ω be the projection operator onto the feasible region $\Omega := \{x \in \mathbb{R}^n : l \leq x \leq u\}$, then for $\alpha > 0$, we have from [8, (1.7)] that

$$P_\Omega\left(x - \alpha g(x)\right) = x - \alpha\left(\varphi(x,\alpha) + \beta(x,\alpha)\right).$$

We say $\varphi(x, \alpha)$ and $\beta(x, \alpha)$ are measures of optimality since if we define

$$\nu(x) := x - P_\Omega\big(x - g(x)\big) = \varphi(x,1) + \beta(x,1),$$

then [73, Section 12.1] shows that $\nu(x)$ is a first-order criticality measure.

Central to this algorithm is the desire to make the optimality measures in the free and active spaces *proportional*. If the measure of optimality in the free space (i.e., a measure of the reduced free gradient $\varphi(x, \alpha)$) is big compared to that in the active space (i.e., a measure of the reduced chopped gradient $\beta(x, \alpha)$), this implies that there is greater potential to make significant progress if the algorithm were to optimize the objective function in the reduced space of

free variables. The precise condition we check is [8, (2.2)], which is

$$\beta(x_k, \bar{\alpha})^T \beta(x_k) \leq \Gamma \varphi(x_k, \bar{\alpha})^T \varphi(x_k) \tag{5.5}$$

for some $\Gamma > 0$ and $\bar{\alpha} \in (0, 2\|H\|^{-1}]$. Whether this condition holds or not, determines the steps taken by the QP solver. We outline these steps next.

If (5.5) holds so that the measure of optimality in the active space is small relative to that in the free space, the algorithm aims to reduce the objective function in the reduced free space by computing a CG step.

1. **CG iteration:** If the CG step is feasible, the algorithm uses the CG step and performs standard CG updates. (See [8, Algorithm 2, Lines 29-30].)

2. **Expansion iteration:** If the CG step is infeasible, then the largest feasible step allowed along the CG step is taken, and the variables that become active are added to the active set. The algorithm then takes a projected gradient step in the reduced space of free variables (as opposed to the full space), which has the potential of making large changes to the active set. (See [8, Algorithm 2, Lines 32-34].)

3. **Negative curvature CG iteration:** If the CG step is a direction of negative or zero curvature, the algorithm takes the largest feasible step to the boundary. At the boundary, the variables that become active are added to the active set, and a projected gradient step in the space of free variables

is taken. The algorithm has now obtained a new set of active variables. (See [8, Algorithm 2, Lines 20-25].)

On the other hand, if (5.5) does not hold and the measure of optimality in the active space is big relative to that in the free space, the algorithm aims to reduce the objective function in the active space through the computation of a proportioning iteration.

4. **Proportioning iteration:** The algorithm removes variables from the active set by stepping off of all bounds whose multipliers are of the wrong sign. Specifically, the algorithm minimizes the objective along the reduced chopped gradient. If the minimizer along this direction is infeasible, the largest feasible step is taken. (See [8, Algorithm 2, Lines 36-44].)

We summarize this iterative approach below in Algorithm 3, which is a condensed summary of [8, Algorithm 2]. Note that we exclude "saddle point iterations" [8, Algorithm 2, Lines 9-16] because the dual BCQP we want to solve is convex; it may have zero curvature but will not have negative curvature, and hence convergence to a first-order point is sufficient for our purposes.

Algorithm 3 Framework from [8] for solving nonconvex BCQPs.

while $\max\{\|\varphi(x_k)\|_\infty, \|\beta(x_k)\|_\infty\} > \tau_{\text{stop}}$ **do**
  if $\beta(x_k, \bar{\alpha})^T \beta(x_k) \leq \Gamma \varphi(x_k, \bar{\alpha})^T \varphi(x_k)$ **then**
    if $s_k^T H s_k \leq 0$ **then**
      Compute a **negative curvature CG step** .
    else
      Set $\alpha_{cg} \leftarrow \big(g_k^T \varphi(x_k)\big)/(s_k^T H s_k)$ and $\alpha_{\text{feas}} \leftarrow \max\{\alpha : x_k - \alpha s_k \in \Omega\}$.
      if $\alpha_{cg} \leq \alpha_{\text{feas}}$ **then**
        Accept the **CG step**.
      else
        Compute an **expansion step**.
  else
    Compute a **proportioning step**.

Note that Algorithm 3 requires only matrix-vector products, and additionally that it has nice convergence results as shown by Dostál and Schöberl [9]. Numerical studies performed in [8] give promising results, especially compared to standard gradient projection algorithms that utilize subspace minimization. Moreover, the proportioning of the two measures of optimality seems to be an effective mechanism in determining when to terminate subspace minimization.

What is not obvious is whether applying this algorithm to the dual problem will be efficient, as opposed to simply solving the primal problem directly using a standard convex quadratic optimization solver. We explore this topic in the following section.

5.4 Numerical results

In order to analyze the performance of Algorithm 3, we performed two types of numerical experiments.

In the first set of numerical experiments, we applied a basic MATLAB implementation of Algorithm 3 on the dual formulations of a set of randomly generated strictly convex QPs. The purpose of these experiments is to examine how properties of the SQO subproblem (such as the number of constraints and the size of the optimal active set) impact the performance of the algorithm in the dual space, but in an isolated setting. From [8], we know Algorithm 3 performs efficiently on general convex and nonconvex BCQPs and that it is impacted by the condition number of the Hessian matrix, with performance degrading as the condition number increases (see [8, Tables 4.2 and 4.6]). However, we would like insight into how aspects of the primal subproblem can impact overall performance when we are not directly solving it. This funnels into our main goal, which is the inclusion of Algorithm 3 into Algorithm FiSQO as a QP solver for the predictor subproblem. We would like to identify situations in which it would be beneficial to tackle the subproblems in the dual space, as well as situations where a primal approach is more efficient.

This leads us to the second set of numerical experiments. Within the framework of Algorithm FiSQO, instead of using Cplex to solve the predictor sub-

problem (3.4) (see Section 4.7.1), we used the MATLAB implementation of Algorithm 3 as the BCQP solver for the dual formulations (5.1) and (5.3) of the predictor subproblem. We then used this implementation of Algorithm FiSQO to solve select large problems from the CUTEst test problem set; we show that such usage of Algorithm 3 is a valid and efficient choice under certain situations.

5.4.1 Performance on randomly generated QPs

We randomly generated a set of strictly convex QPs that has the structure of our predictor subproblem (3.4a). For each synthetic problem, we formed its dual and applied our implementation of Algorithm 3. We are interested in the impact of the number of variables n, the number of constraints m, and the number of active constraints at the optimal solution m_{act}. Thus, we considered a range of values for n (specifically $n \in \{100, 500, 1000\}$) and then defined m and m_{act} in relation to n. The range of values for m and m_{act} that we considered are $m \in \{0.5n, n, 1.5n, 2n, 5n, 10n\}$ and $m_{\text{act}} \in \{0.01n, 0.10n, 0.25n, 0.50n, 0.75n\}$ respectively.

For each sample problem, we generated J_k with MATLAB's *sprandn* routine (with density of 0.2). We also used MATLAB's *sprandsym* routine to generate B_k with a condition number of 100 and density of 0.05 (percent of nonzero entries). We did not vary the condition number because [8] has demonstrated that in-

creasing the condition number will negatively impact performance. We do note that generating B_k in this manner does differ from the choices of B_k that are typically used in the predictor subproblem. Specifically, B_k is typically a scaled identity or obtained from L-BFGS (as mentioned in Section 5.2), which would allow us to avoid the calculation of ${B_k}^{-1}$ and the formation of $\hat{B}_k$. But because we have randomly generated B_k for this set of experiments, we do not have such an advantage in this set of numerical experiments. Instead we avoided calculating ${B_k}^{-1}$ (and forming $\hat{B}_k$) by performing a single sparse Cholesky factorization on B_k to obtain a nonsingular upper triangular matrix R_k such that $B_k = R_k^T R_k$. Note that for a vector $y \in \mathbb{R}^m$, $\hat{B}_k y = J_k({R_k}^T R_k)^{-1} {J_k}^T y$. We can then calculate matrix-vector products with $\hat{g}_k$ using Algorithm 4, shown below.

Algorithm 4 Compute $\hat{B}_k y$ given a Cholesky factorization of B_k.

1: Input $y \in \mathbb{R}^m$, $R_k \in \mathbb{R}^n$, $J_k \in \mathbb{R}^{m \times n}$.
2: Calculate $v = {J_k}^T y$.
3: Solve ${R_k}^T z = v$ for z.
4: Solve $R_k w = z$ for w.
5: Return $\hat{B}_k y = J_k w$.

Thus, when using Algorithm 3 to solve the dual of the randomly generated QP, we simply passed along the Cholesky factor R_k and used Algorithm 4 to perform the matrix-vector product calculations.

We generated a sample of 50 problems for each problem of a specific size (n, m, m_{act}). We then ran a simple MATLAB implementation of Algorithm 2 from

[8], with control parameters $\eta = 0.5$ and $\Gamma = 1$. We used a stopping tolerance of $\tau_{\text{stop}} = 10^{-7}$. The maximum number of iterations allowed was $10,000$ and the maximum number of matrix-vector products allowed was 10^6.

The algorithm we implemented did vary from the algorithm presented in [8] in one way: the calculation of $\bar{\alpha}$. Since we did not directly calculate $\hat{B}_k$, we had no easy way to estimate its norm to obtain $\bar{\alpha}$. Instead, we started with $\bar{\alpha} = 1$ and then adaptively decreased it to ensure that every iteration monotonically decreased the objective function. In solving the dual formulation, $\bar{\alpha}$ comes to play in the negative curvature CG and expansion iterations (lines 24 and 33 of Algorithm 2 in [8]). Thus, instead of using the fixed $\bar{\alpha}$ calculated at the beginning of the algorithm, we let $\bar{\alpha} = 1$, and then during each negative CG and expansion iteration, we did a line search along the projected free gradient into the feasible region and checked if the current $\bar{\alpha}$ step size resulted in a trial point that reduced the objective function. If it did not, we then lowered $\bar{\alpha}$ until the new point decreases the objective. We note that this places extra computation burden since it requires the evaluation of the objective during the line search. However, we feel this burden is minimal. An alternative to this line search procedure is to estimate the one-norm of $\hat{B}_k$ iteratively using only matrix-vector products [92, 93].

In the first experiment, we fixed the number of primal variables n and allowed m and m_{act} to vary. We did this because we wanted to explore the impact

of J_k on the performance of Algorithm 3, since the matrix $\hat{B}_k$ is depends on both ${B_k}^{-1}$ and the Jacobian J_k (see definition (5.2)). As we increase m, not only is Algorithm 3 is optimizing over a (perhaps much) larger dual space, the degeneracy of the Hessian matrix also increases. We also wanted to explore the impact of the size of the active set at the optimal solution, since that would impact the reduced free subspace over which the CG iterates will run.

Tables 5.1 and 5.2 summarize the results from this experiment, where $n = 100$ and $n = 500$ respectively. These tables report the mean (mean) and standard deviation (s.d.), over a sample of 50 generated problems, for the number of matrix-vector products required for the algorithm to converge to an optimal solution. The algorithm was able to successfully solve all of these problems. Note that the top right hand corner of the tables are blank because $m_{\text{act}} \leq m$. Also note that when $m > n$, the number of dual variables is larger than the number of primal variables, and $\hat{B}_k$ is singular.

If we look down each column, we can see that for a fixed m_{act}, if we increase m, we see a significant increase in the number of matrix-vector products required. [8] observed an modest uptick in the number of matrix-vector products required when increasing the number of variables (and holding the percentage of active variables constant). However, in this experiment, an increase in the number of general constraints m results in an increase in *both* the dimension of the dual problem and the degeneracy of the dual Hessian when there are more

Table 5.1: Results showing the number of matrix-vector products required by Algorithm 3 on the duals of 50 randomly generated strictly convex problems where $n = 100$.

		m_{act}				
m		$0.01n$	$0.10n$	$0.25n$	$0.50n$	$0.75n$
$0.5n$	mean	26.1	61.0	92.8	116.7	
	s.d.	6.6	10.3	10.9	11.7	
n	mean	48.5	106.7	161.4	245.0	380.2
	s.d.	22.2	21.7	21.3	27.6	46.2
$1.5n$	mean	92.6	182.3	270.7	400.9	673.9
	s.d.	36.4	37.5	40.8	47.9	91.7
$2n$	mean	157.0	272.6	402.9	582.7	1000.1
	s.d.	60.6	49.5	65.7	61.8	132.3
$5n$	mean	719.7	1010.2	1347.3	1846.4	2700.1
	s.d.	109.2	114.6	132.1	193.5	319.5
$10n$	mean	1748.1	2205.4	2623.6	3398.9	4568.0
	s.d.	176.5	219.0	231.5	296.9	374.9

constraints than primal variables. Therefore, the not-insignificant degradation of performance is expected, since [8] also show a degradation with the increase in the condition number (with a bigger impact when the number of variables is large).

If we look across each row, we can see that for a fixed number of constraints m, if we increase the size of the optimal active set m_{act}, the number of required matrix-vector products increases somewhat modestly. This shows that the size of the optimal active set does matter and not just simply the number of constraints.

Table 5.2: Results showing the number of matrix-vector products required by Algorithm 3 on the duals of 50 randomly generated strictly convex problems where $n = 500$.

		m_{act}				
m		$0.01n$	$0.10n$	$0.25n$	$0.50n$	$0.75n$
$0.5n$	mean	42.3	114.7	187.3	187.7	
	s.d.	8.7	18.8	15.3	8.6	
n	mean	126.2	266.3	409.6	630.1	898.9
	s.d.	37.0	27.6	34.6	34.8	47.9
$1.5n$	mean	260.1	578.7	899.5	1416.4	2473.3
	s.d.	78.0	118.2	93.6	121.5	194.5
$2n$	mean	548.9	1184.2	1748.1	2663.1	4387.8
	s.d.	118.7	116.4	139.3	197.4	294.9
$5n$	mean	4123.7	5550.7	6894.2	9204.3	12922.1
	s.d.	390.8	364.3	329.1	395.1	794.7
$10n$	mean	8611.8	11265.5	13436.5	17264.5	23891.4
	s.d.	622.3	820.9	899.7	1183.8	1445.6

In the second experiment, we compared the computational time that it takes Algorithm 3 to solve the dual problem versus the computational time it takes Cplex [78] to directly solve the primal problem, on randomly generated problems with with varying n and m, but with a constant $m_{\text{act}} = 0.01n$. Table 5.3 summarizes these results.

We note that we used the Cplex's default parameters to mimic its performance in practice. Cplex's algorithm of choice is the barrier method in the case, which should help its performance since these problems are created independently and their initial points are randomly generated; there are no oppor-

Table 5.3: Comparison of the computational time (seconds) taken by Cplex and Algorithm 3 on the duals of 50 randomly generated strictly convex primal problems, where $m_{\text{act}} = 0.01n$.

		Cplex		Algorithm 3	
n	m	mean	s.d.	mean	s.d.
100	$0.50n$	2.021e-02	4.492e-03	1.147e-02	3.493e-03
100	$1.00n$	2.554e-02	5.096e-03	1.866e-02	8.815e-03
100	$5.00n$	1.575e-01	3.698e-02	3.782e-01	7.164e-02
100	$10.00n$	4.625e-01	1.531e-01	1.206e+00	2.200e-01
500	$0.50n$	1.958e-01	5.076e-02	4.789e-02	1.037e-02
500	$1.00n$	5.747e-01	8.115e-02	1.551e-01	4.790e-02
500	$5.00n$	5.529e+00	2.221e-01	8.963e+00	9.865e-01
500	$10.00n$	4.480e+01	1.247e+02	3.427e+01	3.359e+00
1000	$0.50n$	1.187e+00	1.424e-01	2.473e-01	4.684e-02
1000	$1.00n$	2.717e+00	2.403e-01	8.895e-01	1.772e-01
1000	$5.00n$	4.322e+01	3.637e+00	7.596e+01	8.034e+00
1000	$10.00n$	1.969e+02	6.108e+00	2.379e+02	1.157e+01

tunities for warm-starting. Additionally, Cplex uses multiple threads, which also enhances its performance. On the other hand, we only measured the time Algorithm 3 took to solve the dual problem and did not count the time it would take to retrieve the primal solution. However, this additional time should be minimal and should not make a material difference in the experiment.

But despite the advantages that Cplex enjoys as a commercial solver, we find the results promising. As expected, when the number of constraints is smaller than the number of primal variables, the dual approach is faster, as it is optimizing over a smaller space. Note that we have kept the relative size of

the optimal active set constant. We also note that Cplex implements "presolving", which has the potential to reduce the problem size; our basic implement Algorithm 3 does not do so. Therefore, it is no surprise that as the problems scale up, Algorithm 3 performs worse than Cplex but perhaps not as much as expected.

In the third experiment, we want to compare the performances of Algorithm 3 and Cplex in two cases: when there are fewer constraints than variables ($m = 0.5n$) and when there are more constraints than variables ($m = 5n$). We varied the number of variables and the size of the optimal active set. We then compared the computational times of the two algorithms; We also listed the iterations that Algorithm 3 performed by types. Tables 5.4 and 5.5 summarizes the results when $m = 0.5n$, and Tables 5.6 and 5.7 summarizes the results when $m = 5n$.

We can see that Algorithm 3 is competitive when there are fewer constraints than variables, which is to be expected We do also note that the size of the optimal active set does not impact the computation time required by Cplex, unlike the Algorithm 3; the difference between computation times becomes significant when there are both fewer constraints and a very small optimal active set.

It is also interesting to note that we did not observe any negative (or rather, zero) curvature CG iterations during this experiment, even in the situations

Table 5.4: Comparison of the computational time (seconds) taken by Cplex and Algorithm 3 on the duals of 50 randomly generated strictly convex primal problems, where $m = 0.5n$.

			Cplex		Algorithm 3	
n	m	m_{act}	mean	s.d.	mean	s.d.
100	$0.5n$	$0.01n$	2.021e-02	4.492e-03	1.147e-02	3.493e-03
100	$0.5n$	$0.10n$	2.268e-02	9.446e-03	3.347e-02	1.121e-02
100	$0.5n$	$0.50n$	2.116e-02	7.022e-03	6.648e-02	1.236e-02
500	$0.5n$	$0.01n$	1.958e-01	5.076e-02	4.789e-02	1.037e-02
500	$0.5n$	$0.10n$	1.922e-01	4.401e-02	1.363e-01	2.627e-02
500	$0.5n$	$0.50n$	1.841e-01	5.521e-02	2.620e-01	2.334e-02
1000	$0.5n$	$0.01n$	1.187e+00	1.424e-01	2.473e-01	4.684e-02
1000	$0.5n$	$0.10n$	1.156e+00	9.531e-02	7.196e-01	1.353e-01
1000	$0.5n$	$0.50n$	1.048e+00	9.194e-02	1.014e+00	6.452e-02

Table 5.5: Number of iterations (by type) taken by Algorithm 3 on the duals of 50 randomly generated strictly convex primal problems, where $m = 0.5n$.

			CG		Expansion		Proportioning	
n	m	m_{act}	mean	s.d.	mean	s.d.	mean	s.d.
100	$0.5n$	$0.01n$	2.7	1.5	6.5	2.5	0.9	0.5
100	$0.5n$	$0.10n$	18.2	3.9	15.0	3.8	2.3	0.5
100	$0.5n$	$0.50n$	92.0	10.3	5.2	1.5	3.8	0.7
500	$0.5n$	$0.01n$	5.7	0.9	11.4	4.3	1.4	0.5
500	$0.5n$	$0.10n$	19.7	2.8	39.5	9.7	2.7	0.6
500	$0.5n$	$0.50n$	144.6	7.7	12.5	2.0	4.7	0.6
1000	$0.5n$	$0.01n$	8.0	1.6	14.7	4.6	1.8	0.6
1000	$0.5n$	$0.10n$	16.2	2.6	60.1	15.6	3.1	0.4
1000	$0.5n$	$0.50n$	154.3	6.6	17.7	2.4	5.1	0.5

Table 5.6: Comparison of the computational time (seconds) taken by Cplex and Algorithm 3 on the duals of 50 randomly generated strictly convex primal problems, where $m = 5n$.

			Cplex		Algorithm 3	
n	m	m_{act}	mean	s.d.	mean	s.d.
100	$5n$	$0.01n$	1.575e-01	3.698e-02	3.782e-01	7.164e-02
100	$5n$	$0.10n$	1.685e-01	5.417e-02	5.386e-01	9.085e-02
100	$5n$	$0.50n$	1.677e-01	3.217e-02	1.071e+00	1.454e-01
100	$5n$	$0.75n$	1.820e-01	6.195e-02	1.719e+00	2.389e-01
500	$5n$	$0.01n$	5.529e+00	2.221e-01	8.963e+00	9.865e-01
500	$5n$	$0.10n$	6.301e+00	7.376e-01	1.246e+01	1.464e+00
500	$5n$	$0.50n$	6.430e+00	5.333e-01	2.129e+01	1.192e+00
500	$5n$	$0.75n$	6.645e+00	5.610e-01	3.268e+01	3.696e+00
1000	$5n$	$0.01n$	4.322e+01	3.637e+00	7.596e+01	8.034e+00
1000	$5n$	$0.10n$	4.073e+01	1.894e+00	9.844e+01	6.017e+00
1000	$5n$	$0.50n$	4.611e+01	3.764e+00	1.662e+02	8.566e+00
1000	$5n$	$0.75n$	4.480e+01	1.875e+00	2.350e+02	1.208e+01

Table 5.7: Number of iterations (by type) taken by Algorithm 3 on the duals of 50 randomly generated strictly convex primal problems, where $m = 5n$.

			CG		Expansion		Proportioning	
n	m	m_{act}	mean	s.d.	mean	s.d.	mean	s.d.
100	$5n$	$0.01n$	246.2	46.9	229.8	39.4	0.8	0.5
100	$5n$	$0.10n$	367.9	53.4	313.5	37.2	2.2	0.6
100	$5n$	$0.50n$	978.7	133.7	424.8	40.5	4.9	0.8
100	$5n$	$0.75n$	1772.6	273.4	453.6	38.4	6.9	0.9
500	$5n$	$0.01n$	1267.8	121.6	1419.0	171.4	1.9	0.5
500	$5n$	$0.10n$	1713.4	148.8	1908.6	127.2	4.0	0.6
500	$5n$	$0.50n$	4382.4	252.7	2399.4	106.4	7.2	0.7
500	$5n$	$0.75n$	8076.8	689.9	2409.9	88.4	9.5	0.9
1000	$5n$	$0.01n$	2553.7	162.8	2895.0	322.0	2.5	0.7
1000	$5n$	$0.10n$	3405.5	183.6	3809.7	178.9	4.6	0.7
1000	$5n$	$0.50n$	8396.3	346.7	4796.8	168.9	8.3	0.7
1000	$5n$	$0.75n$	15793.9	748.4	4883.7	147.2	10.7	0.6

where there were more constraints then variables and the Hessian $\hat{B}_k$ in the BCQP was very degenerate. This probably arises from the special structure of $\hat{B}_k$ and requires further study.

5.4.2 Performance in the FiSQO framework

We want to test Algorithm 3 as the QP solver within the FiSQO framework. In particular, we want to see how FiSQO scales when this algorithm is used as the solver for the predictor subproblem, in contrast to when Cplex is used. From the first set of numerical experiments in Section 5.4.1, we can see that Algorithm 3 performs well on QPs that have fewer constraints than decision variables and a small optimal active set. In this experiment, we will only select problems from the CUTEst test set where $m < n$. We would like to avoid solving the predictor subproblem in the dual space when the dual space is bigger than the primal space and the resulting Hessian matrix is (more) degenerate.

Our implementation of FiSQO is as described in Section 4.7.1, save for the following changes implemented to handle large scale problems. Recall that the formulation of the predictor subproblem (3.4) required a positive-definite matrix B_k. Instead of a modified Newton strategy, we used a scaled diagonal

matrix based on the Barzilai-Borwein [66] method, as follows:

$$B_k = \iota_k I, \quad \text{where} \quad \iota_k := \max\left\{\iota_{\min},\ \min\left\{\iota_{\max},\ \frac{(x_k - x_{k-1})^T(g_k - g_{k-1})}{\|x_k - x_{k-1}\|_2^2}\right\}\right\}$$

where I is the $n \times n$ identity matrix, $\iota_{\min} = 10^{-5}$, and $\iota_{\max} = 10^5$.

To obtain the accelerator step s_k^a via (3.17), s_k^a was computed from the subproblem (3.18) using the *minres* function in MATLAB with parameters $tol = 10^{-6}$ and $maxit = 10^5$, instead of the backslash operator. The control parameters remain the same as the values found in Table 4.1, save that $\delta_0 = 1$. The iteration limit remained the same at 10000, but the CPU time limit increased from 10 minutes to 3 hours.

Finally, we modified the termination tests used by FiSQO to account for problem scaling in a fashion similar to [26, 94]. Instead of determining if the primal-dual pair (x_k, y_k) satisfied the approximate KKT conditions by checking (4.14) (namely if $\|F_{\text{KKT}}(x_k, y_k)\|_\infty \leq \tau_{\text{stop}}$), we checked if

$$\left\|g_k - J_k^T y_k\right\|_\infty \leq \tau_1 \cdot \tau_{\text{stop}} \quad \text{and} \quad \left\|\min\left[c_k, y_k\right]\right\|_\infty \leq \tau_2 \cdot \tau_{\text{stop}},$$

where the value of the termination tolerance τ_{stop} remains the same as in Table 4.1, and τ_1 and τ_1 are defined as follows:

$$\tau_1 := \max\left\{1,\ \|g_k\|_\infty,\ \frac{1}{m}\|y_k\|_1\right\} \quad \text{and} \quad \tau_2 := \max\left\{1,\ \|c_k\|_\infty,\ \frac{1}{m}\|y_k\|_1\right\}.$$

We selected several large problems from the CUTEst test set where $m < n$, though we make no claim that this is a complete or comprehensive selection. We first ran the FiSQO algorithm with Cplex as the solver for the predictor subproblem (3.4). Next, we reran the FiSQO algorithm, this time using Algorithm 3 as the solver for the dual formulation of the predictor subproblem ((5.1) or (5.3)). The primal solution was then obtained using (5.4). We then modified the CUTEst *sif* files to increase the problem sizes. Table 5.8 gives a summary of some of the problems that were solved successfully.

Table 5.8: Results for FiSQO on the large CUTEst problems.

prob	m	n	status	f	v	iters	feval	σ	#o	#v	#b	#p	time (s)
Algorithm 3 as the solver for the predictor subproblem													
CATMIX	1600	2403	0	-4.79e-02	2.85e-06	195	1026	6.6e+05	78	86	10	21	3086.6
	3200	4803	0	-4.79e-02	5.60e-06	167	878	2.6e+03	72	54	11	30	9603.3
CHANNEL	9598	9600	0	1.00e+00	8.93e-06	5	16	1.0e+01	0	5	0	0	68.6
DTOC4	2998	4499	0	2.87e+00	9.17e-06	3	4	1.0e+01	1	2	0	0	207.6
	9998	14999	2	2.87e+00	9.86e-06	9	13	1.0e+01	3	6	0	0	11187.0
GILBERT	1	5000	0	2.46e+03	4.35e-08	50	162	4.2e+01	22	24	2	2	9.5
	1	50000	0	2.49e+04	1.49e-09	65	289	2.1e+02	35	26	2	2	181.1
	1	250000	0	1.25e+05	5.99e-06	55	209	4.2e+02	20	29	3	3	3447.3
LUKVLI3	2	100000	0	1.16e+01	0.00e+00	12	16	1.0e+01	12	0	0	0	9.7
	2	500000	0	1.16e+01	1.86e-10	12	17	1.0e+01	12	0	0	0	44.1
	2	750000	2	1.53e+05	0.00e+00	5	10	1.0e+01	5	0	0	0	10872.4
ORTHRDM2	4000	8003	0	3.11e+02	1.83e-07	5	8	1.0e+01	2	3	0	0	8.6
	40000	80003	0	3.11e+03	2.77e-06	5	8	1.0e+01	2	3	0	0	779.8
Cplex as the solver for the predictor subproblem													
CATMIX	1600	2403	0	-4.79e-02	1.11e-06	286	1630	1.0e+01	141	95	17	33	648.2
	3200	4803	0	-4.79e-02	1.64e-06	205	1113	2.0e+01	85	66	15	39	1554.1
CHANNEL	9598	9600	0	1.00e+00	5.05e-09	4	7	1.0e+01	0	4	0	0	385.2
DTOC4	2998	4499	0	2.87e+00	1.06e-06	2	3	1.0e+01	1	1	0	0	10.7
	9998	14999	0	2.87e+00	4.06e-11	10	15	1.0e+01	4	6	0	0	332.7
GILBERT	1	5000	0	2.46e+03	4.34e-08	23	38	4.1e+01	8	9	3	3	28.7
	1	50000	0	2.49e+04	1.49e-09	52	163	1.9e+02	24	26	1	1	8842.9
	1	250000	-6	4.29e+06	1.25e+07	0	1	1.0e+01	0	0	0	0	10832.0
LUKVLI3	2	100000	0	1.16e+01	0.00e+00	12	16	1.0e+01	12	0	0	0	10.9
	2	500000	0	1.16e+01	1.60e-10	12	17	1.0e+01	12	0	0	0	49.8
	2	750000	2	1.53e+05	0.00e+00	5	10	1.0e+01	5	0	0	0	10876.0
ORTHRDM2	4000	8003	0	3.11e+02	8.62e-08	5	8	1.0e+01	2	3	0	0	27.1
	40000	80003	0	3.11e+03	8.78e-08	5	8	1.0e+01	2	3	0	0	5909.8

Even though we ran a simple MATLAB implementation of the FiSQO algorithm, we were able to solve problems that are relatively large. We see that using Algorithm 3 is competitive in some cases, though not all of them. FiSQO with Algorithm 3 performed better on problems *CHANNEL*, *GILBERT*, *ORTHRDM2*, but not on *CATMIX* and *DTOC4*. (Performance was roughly the same on *LUKVLI3*.)

Both *CATMIX* ("Catalyst Mixing" in [95]) or *DTOC4* (Problem 4 in [96]) are optimal control problems. As a result, both had simple upper and lower bounds on the control variables in addition to the equality constraints that modeled the dynamics. Since this implementation of FiSQO does not handle variable bounds separately, the dual space that Algorithm 3 optimized over was very much bigger than the primal space. Consequently, Algorithm 3 performed poorly compared to Cplex. This is evident in the significantly more time required to solve both problems.

On the other hand, problems such as *GILBERT* [97] (which has a single constraint) show that there can be a clear advantage to working in the dual space if it is small relative to the primal space. This advantage is magnified as the number of primal variables increased. We ran a profiler in MATLAB on both implementations for *GILBERT* with $n = 50000$ and set a time limit of an hour. With Cplex, the algorithm timed out after 21 iterations and the predominant computational cost was solving the QP (accounting for over 98% of the time).

With Algorithm 3, the algorithm finished 69 iterations in around 66 seconds. In that implementation solving the accelerator subproblem with *minres* and the linear program with Cplex were the (around 50% and 35% of the time), while Algorithm 3 accounted for about 1% of the time. From Table 5.8, when the problem size increased to $n = 250000$, Cplex was not able to solve the predictor subproblem within the time limit of 3 hours during the first iteration.

The relative performance of the two implementations on problem *LUKVLI3* (Problem 5.3 in [98]) is puzzling, since *LUKVLI3* contains only 2 constraints. We would expect that the implementation that uses Algorithm 3 as the QP solver would significantly outperform the implementation that uses Cplex, as it did for problem *GILBERT*. To explore this situation, we repeated both runs with MATLAB's profiler. We note that while FiSQO with Cplex solves the problem in about 50s, repeating the run with the profiler increases the time to 95s. There was no significant time increase when repeating the run using Algorithm 3 with the profiler. This should be taken into consideration when examining the following results. In the implementation with Cplex, we found that solving the predictor QP accounted for 65% of the CPU time and solving the accelerator subproblem with *minres* accounted for about 12% of the time. On the other hand, in the implementation with Algorithm 3, solving the predictor subproblem using Algorithm 3 accounted only for 2% of the time; the dominant costs arose from *minres* in the accelerator subproblem (34%) and the CUTEst com-

mands to values for the gradients, Jacobians, and Hessians (34%). We conclude that in problem *LUKVLI3*, the gains in time savings from using Algorithm 3 were overshadowed by other aspects of the algorithm.

Because our implementation was in MATLAB, we quickly ran into memory issues when trying to scale up the problem. When we increased the problem size of *LUKVLI3* from $n = 500000$ to $n = 750000$, we were unable to solve the problem within 3 hours. For both implementation, running the profiler for an hour netted two FiSQO iterations and showed that the majority of the time (over 99%) was spent in the formation of the KKT system in the accelerator subproblem (our implementation was perhaps naive). The barrier to scaling was thus memory and not either QP solvers.

5.5 Conclusions and discussion

This chapter presents an alternative approach to solving strictly convex QPs. Instead of directly solving the problem, we can take advantage of that fact that the dual problem is a convex BCQP (a result of the strict convexity of the primal QP) and apply Algorithm 3, which is a new solver for non-convex BCQPs developed by Mohy-ud-Din and Robinson [8]. This approach is of interest because the QPs in SQO methods (like FiSQO) can be a computational bottleneck. Algorithm 3 would then present an alternative to current off-the-

shelf QP solvers, one that requires only matrix-vector products and that has been shown to be efficient in general situations. The hope is that using Algorithm 3 within FiSQO (and SQO methods in general) would allow for these algorithms to scale up and tackle even larger problems.

We find that Algorithm 3 is competitive when there are fewer constraints than variables in the primal problem, a result that is magnified when the size of the optimal active set is also small. This is first shown on randomly generated primal QPs (Section 5.4.1) and then later supported in Section 5.4.2, where an implementation of the FiSQO algorithm (with Algorithm 3 integrated as the QP solver for the predictor subproblem) was tested on several large-scale problems from the CUTEst test set. Table 5.8 summarizes these results and shows that FiSQO with Algorithm 3 can scale well in certain situations, namely when there are few constraints.

We note that the numerical results in the Sections 5.4.1 and 5.4.2 are based on a simple MATLAB implementation of both the FiSQO algorithm and Algorithm 3. As a result, there is much room for improvement, especially if preconditioning is implemented. A typical preconditioning method would be to apply a preconditioner matrix P (which is typically nonsingular) such that $P\hat{B}_k$ has better clustered eigenvalues and a smaller condition number. Since Algorithm 3 is a CG method, applying preconditioning can accelerate convergence during subspace minimization.

Much work has been done on preconditioning procedures for the conjugate gradient methods (such as [99, Section 3.7] and [100–104]), some with a specific focus on indefinite systems [105–107]. Procedures also extend beyond preconditioning the CG iterates and allow for the preservation of the bound constraints [99, Section 3.7]. As for preconditioning in the dual space, work has been done in the context of nonlinear least squares [108–110].

However, applying preconditioning to Algorithm 3 requires careful consideration of several issues. For example, the Hessian matrix $\hat{B}_k$ may not be positive definite, even if we only consider problems with a smaller dual space. Because of different underlying problem structures, the preconditioning methods for indefinite systems mentioned earlier cannot be so straight-forwardly applied to this situation. Moreover, since the dual subproblem is a BCQP, we cannot simply apply a preconditioner matrix to the Hessian $\hat{B}_k$. Changing the Hessian matrix in the quadratic objective function impacts not only the resulting optimal point, but also the optimal Lagrange multipliers. As a result, translating this optimal point back to a solution for the original dual problem is not as easy as it would be if we were minimizing an unconstrained quadratic function.

Preconditioning is an important topic that must be tackled if we want an efficient and scalable implementation that can be integrated into FiSQO and other SQO methods. There is currently much work to be done and many approaches to be explored; we find this to be an exciting direction for future work.

Appendix

A.1 Detailed output from the numerical experiments

Table A.1: Results for FiSQO on the CUTEst problems of size $m \geq 1$ and $\max\{m, n\} \leq 100$.

prob	m	n	status	f	v	iters	feval	σ	#o	#v	#b	#p
ACOPP14	68	38	0	8.08e+03	5.45e-09	5	42	1.2e+03	1	2	1	1
ACOPR14	82	38	0	8.08e+03	4.12e-05	7	45	1.4e+02	1	4	1	1
AIRCRFTA	5	8	0	0.00e+00	6.04e-06	2	3	1.0e+01	0	2	0	0
AIRPORT	42	84	0	4.80e+04	5.35e-13	12	13	1.6e+03	0	12	0	0
ALLINITA	4	4	-9	3.33e+01	1.31e-12	10001	789688	Inf	385	569	133	841
ALLINITC	1	4	0	3.05e+01	5.13e-14	23	23	1.2e+08	1	21	0	0
ALSOTAME	1	2	0	8.21e-02	2.92e-13	4	5	1.0e+01	3	1	0	0
ANTWERP	10	27	1	2.44e+04	7.28e-12	10001	10002	1.0e+01	10001	0	0	0
ARGAUSS	15	3	-8	0.00e+00	3.38e-04	2	3	1.0e+01	0	2	0	0
AVGASA	10	8	0	-4.63e+00	2.83e-15	1	2	1.0e+01	0	1	0	0
AVGASB	10	8	0	-4.48e+00	1.11e-16	1	2	1.0e+01	0	1	0	0
AVION2	15	49	-9	9.47e+07	1.60e-12	10001	319723	1.4e+04	9	1	0	0
BATCH	73	48	0	2.59e+05	1.42e-09	7	8	6.0e+01	0	7	0	0
BIGGSC4	7	4	0	-2.44e+01	0.00e+00	3	5	1.0e+01	2	1	0	0
BOOTH	2	2	0	0.00e+00	0.00e+00	1	2	1.0e+01	0	1	0	0

BT1	1	2	0	-1.00e+00	1.93e-08	130	1655	3.8e+02	129	1	0	0
BT10	2	2	0	-1.00e+00	5.57e-09	6	7	1.0e+01	0	6	0	0
BT11	3	5	0	8.25e-01	2.57e-08	7	9	1.0e+01	2	3	1	1
BT12	3	5	0	6.19e+00	5.15e-06	3	4	1.0e+01	0	1	1	1
BT13	1	5	0	0.00e+00	7.82e-06	15	16	1.0e+01	4	11	0	0
BT2	1	3	0	3.26e-02	9.10e-07	11	13	1.0e+01	9	0	1	1
BT3	3	5	0	4.09e+00	1.07e-14	1	2	1.0e+01	1	0	0	0
BT4	2	3	0	-4.55e+01	7.49e-08	11	18	2.0e+01	6	5	0	0
BT5	2	3	0	9.62e+02	3.20e-06	6	7	1.0e+01	1	5	0	0
BT6	2	5	0	2.77e-01	1.18e-07	8	11	1.0e+01	6	2	0	0
BT7	3	5	0	3.06e+02	6.55e-10	26	106	4.7e+02	5	7	7	7
BT8	2	5	0	1.00e+00	7.63e-06	9	10	1.0e+01	9	0	0	0
BT9	2	4	0	-1.00e+00	6.71e-08	8	9	1.0e+01	1	7	0	0
BURKEHAN	1	1	-1	0.00e+00	1.00e+00	0	1	1.0e+01	0	0	0	0
BYRDSPHR	2	3	0	-4.68e+00	5.06e-09	8	9	2.0e+01	1	7	0	0
CANTILVR	1	5	0	1.34e+00	5.94e-07	10	11	1.0e+01	0	10	0	0
CB2	3	3	0	1.95e+00	9.37e-12	6	7	1.0e+01	1	5	0	0
CB3	3	3	0	2.00e+00	1.43e-12	6	7	1.0e+01	1	5	0	0
CHACONN1	3	3	0	1.95e+00	7.33e-08	4	5	1.0e+01	0	4	0	0
CHACONN2	3	3	0	2.00e+00	3.16e-12	6	7	1.0e+01	0	6	0	0
CLUSTER	2	2	0	0.00e+00	7.22e-06	7	8	1.0e+01	0	7	0	0
CONGIGMZ	5	3	0	2.80e+01	2.95e-10	4	5	1.0e+01	0	4	0	0
COOLHANS	9	9	0	0.00e+00	2.35e-08	199	212	1.0e+01	0	199	0	0
CRESC4	8	6	-1	1.59e-08	7.50e-01	19	26	3.2e+02	5	14	0	0
CRESC50	100	6	0	7.86e-01	2.13e-10	40	70	2.6e+03	4	36	0	0
CSFI1	4	5	0	-4.91e+01	9.61e-11	6	9	1.0e+01	3	3	0	0
CSFI2	4	5	0	5.50e+01	8.81e-13	8	9	1.0e+01	3	5	0	0
CUBENE	2	2	0	0.00e+00	0.00e+00	4	15	1.0e+01	0	4	0	0
DALLASS	31	46	0	-3.24e+04	5.37e-13	13	21	1.0e+01	13	0	0	0
DEGENLPA	15	20	0	3.06e+00	6.55e-07	5	5	3.0e+01	4	0	0	0
DEGENLPB	15	20	0	-3.07e+01	8.34e-07	3	4	3.0e+01	2	1	0	0
DEMBO7	20	16	0	1.75e+02	1.36e-13	1618	1620	2.0e+02	1617	1	0	0
DEMYMALO	3	3	0	-3.00e+00	0.00e+00	8	8	2.0e+01	6	1	0	0
DIPIGRI	4	7	0	6.81e+02	8.71e-10	6	23	1.0e+01	3	3	0	0
DISC2	23	29	0	1.56e+00	3.30e-08	10	16	1.0e+01	3	7	0	0
DIXCHLNG	5	10	0	2.47e+03	1.87e-14	9	10	6.8e+02	6	3	0	0
DNIEPER	24	61	0	1.87e+04	2.11e-08	3	4	1.7e+03	0	3	0	0
DUAL1	1	85	0	3.50e-02	1.08e-16	1	2	1.0e+01	0	1	0	0
DUAL2	1	96	0	3.37e-02	2.93e-16	1	2	1.0e+01	0	1	0	0
DUAL4	1	75	0	7.46e-01	4.36e-16	1	2	1.0e+01	0	1	0	0
EQC	3	9	1	-8.28e+02	0.00e+00	10001	19004	1.0e+01	10001	0	0	0
ERRINBAR	9	18	0	2.80e+01	3.58e-12	519	883	1.0e+01	482	37	0	0
EXPFITA	22	5	0	1.14e-03	4.44e-14	14	16	1.0e+01	14	0	0	0

EXTRASIM	1	2	0	1.00e+00	0.00e+00	1	2	1.0e+01	0	1	0	0
FCCU	8	19	0	1.11e+01	1.78e-14	1	2	1.0e+01	1	0	0	0
FLETCHER	4	4	-1	4.00e+00	1.00e+00	1	2	1.0e+01	0	1	0	0
FLT	2	2	-6	0.00e+00	4.86e-05	8	9	1.0e+01	1	7	0	0
GENHS28	8	10	0	9.27e-01	3.89e-15	1	2	1.0e+01	1	0	0	0
GIGOMEZ1	3	3	0	-3.00e+00	4.58e-08	7	8	1.0e+01	2	5	0	0
GIGOMEZ2	3	3	0	1.95e+00	1.63e-07	5	6	1.0e+01	2	3	0	0
GIGOMEZ3	3	3	0	2.00e+00	9.62e-13	6	7	1.0e+01	1	5	0	0
GOFFIN	50	51	0	7.28e-14	0.00e+00	26	26	2.0e+01	24	1	0	0
GOTTFR	2	2	0	0.00e+00	2.24e-10	5	10	1.0e+01	0	5	0	0
GOULDQP1	17	32	0	-3.49e+03	0.00e+00	1	2	1.0e+01	1	0	0	0
GROWTH	12	3	-1	0.00e+00	2.44e+00	20	75	1.0e+01	0	20	0	0
HAIFAS	9	13	0	-4.50e-01	9.22e-10	14	24	1.0e+01	2	12	0	0
HALDMADS	42	6	0	3.33e-02	3.88e-14	215	423	1.0e+01	204	7	2	2
HATFLDF	3	3	1	0.00e+00	9.50e-03	10001	72682	4.0e+01	0	2479	3572	3950
HATFLDG	25	25	0	0.00e+00	4.26e-05	6	31	1.0e+01	0	6	0	0
HATFLDH	7	4	0	-2.44e+01	8.88e-16	1	2	1.0e+01	1	0	0	0
HEART6	6	6	0	0.00e+00	6.80e-07	33	234	5.4e+09	0	32	0	0
HEART8	8	8	0	0.00e+00	5.60e-08	12	49	5.1e+03	1	11	0	0
HIMMELBA	2	2	0	0.00e+00	0.00e+00	1	2	1.0e+01	0	1	0	0
HIMMELBC	2	2	0	0.00e+00	1.72e-08	5	8	1.0e+01	0	5	0	0
HIMMELBD	2	2	1	0.00e+00	4.67e+01	10001	401002	1.0e+01	0	9977	12	12
HIMMELBE	3	3	0	0.00e+00	2.22e-16	2	3	1.0e+01	0	2	0	0
HIMMELBI	12	100	0	-1.74e+03	8.08e-14	9	10	1.0e+01	9	0	0	0
HIMMELBJ	14	45	-10	-1.91e+03	2.52e-09	154	1566	4.1e+04	142	0	1	0
HIMMELBK	14	24	0	5.18e-02	3.83e-07	5	6	1.0e+01	3	2	0	0
HIMMELP2	1	2	0	-8.20e+00	2.85e-08	15	23	1.0e+01	9	4	1	1
HIMMELP3	2	2	0	-5.90e+01	0.00e+00	3	5	1.0e+01	3	0	0	0
HIMMELP4	3	2	0	-5.90e+01	0.00e+00	3	6	1.0e+01	3	0	0	0
HIMMELP5	3	2	0	-5.90e+01	0.00e+00	7	10	1.0e+01	7	0	0	0
HIMMELP6	5	2	0	-5.90e+01	0.00e+00	6	10	1.0e+01	6	0	0	0
HONG	1	4	0	2.26e+01	1.67e-16	6	7	1.0e+01	6	0	0	0
HS10	1	2	0	-1.00e+00	4.32e-10	9	10	1.0e+01	0	9	0	0
HS100	4	7	0	6.81e+02	8.71e-10	6	23	1.0e+01	3	3	0	0
HS100LNP	2	7	0	6.81e+02	2.31e-06	7	27	1.0e+01	3	4	0	0
HS100MOD	4	7	0	6.79e+02	1.51e-12	7	38	1.0e+01	3	4	0	0
HS101	5	7	0	1.81e+03	1.40e-09	27	72	5.1e+03	8	9	5	5
HS102	5	7	0	9.12e+02	4.90e-13	26	38	1.3e+04	11	7	2	6
HS103	5	7	0	5.44e+02	3.30e-15	18	24	1.3e+03	6	8	2	2
HS104	5	8	0	3.95e+00	3.30e-07	15	17	1.0e+01	4	11	0	0
HS105	1	8	0	1.04e+03	0.00e+00	15	22	1.0e+01	15	0	0	0
HS106	6	8	1	2.12e+03	1.20e+00	10001	349854	3.2e+03	3	9998	0	0
HS107	6	9	0	5.06e+03	8.90e-10	4	5	1.5e+03	1	3	0	0

HS108	13	9	0	-8.66e-01	4.84e-07	7	8	1.0e+01	1	6	0	0
HS109	10	9	0	5.36e+03	7.14e-11	8	9	1.0e+01	2	6	0	0
HS11	1	2	0	-8.50e+00	6.71e-08	5	6	1.0e+01	0	5	0	0
HS111	3	10	0	-4.53e+01	2.86e-06	16	17	2.0e+01	1	15	0	0
HS111LNP	3	10	0	-4.53e+01	1.64e-06	18	21	2.0e+01	1	15	1	1
HS112	3	10	0	-4.78e+01	4.97e-14	11	12	1.0e+01	11	0	0	0
HS113	8	10	0	2.43e+01	2.88e-13	5	6	1.0e+01	1	4	0	0
HS114	11	10	0	-1.77e+03	7.29e-09	11	15	2.0e+02	6	5	0	0
HS116	14	13	0	9.76e+01	1.29e-13	15	19	5.0e+02	7	8	0	0
HS117	5	15	0	3.23e+01	0.00e+00	13	24	1.0e+01	13	0	0	0
HS118	17	15	0	6.65e+02	0.00e+00	1	2	1.0e+01	1	0	0	0
HS119	8	16	0	2.45e+02	6.92e-15	7	8	1.0e+01	7	0	0	0
HS12	1	2	0	-3.00e+01	5.10e-11	8	9	1.0e+01	1	7	0	0
HS13	1	2	0	1.00e+00	0.00e+00	26	26	2.0e+01	25	0	0	0
HS14	2	2	0	1.39e+00	7.67e-13	5	6	1.0e+01	1	4	0	0
HS15	2	2	0	3.06e+02	0.00e+00	5	6	3.3e+03	3	2	0	0
HS16	2	2	0	2.31e+01	0.00e+00	4	5	1.0e+01	4	0	0	0
HS17	2	2	0	1.00e+00	0.00e+00	7	11	1.0e+01	7	0	0	0
HS18	2	2	0	5.00e+00	5.54e-06	5	6	1.0e+01	1	4	0	0
HS19	2	2	0	-6.96e+03	7.26e-09	5	6	2.5e+02	1	4	0	0
HS20	3	2	0	4.02e+01	0.00e+00	3	4	1.0e+01	3	0	0	0
HS21	1	2	0	-1.00e+02	0.00e+00	1	2	1.0e+01	1	0	0	0
HS21MOD	1	7	0	-9.60e+01	0.00e+00	1	2	1.0e+01	1	0	0	0
HS22	2	2	0	1.00e+00	2.10e-09	4	5	1.0e+01	1	3	0	0
HS23	5	2	0	2.00e+00	0.00e+00	5	6	1.0e+01	5	0	0	0
HS24	3	2	0	-1.00e+00	0.00e+00	6	10	2.0e+01	5	0	0	0
HS26	1	3	0	2.14e-10	8.23e-06	15	16	1.0e+01	11	4	0	0
HS268	5	5	0	-2.55e-11	0.00e+00	1	2	1.0e+01	1	0	0	0
HS27	1	3	0	4.00e-02	1.39e-10	22	150	1.0e+01	11	6	3	2
HS28	1	3	0	1.23e-32	4.44e-16	1	2	1.0e+01	1	0	0	0
HS29	1	3	0	-2.26e+01	8.02e-06	6	11	1.0e+01	3	1	1	1
HS30	1	3	0	1.00e+00	0.00e+00	9	10	1.0e+01	9	0	0	0
HS31	1	3	0	6.00e+00	1.64e-08	5	6	1.0e+01	1	4	0	0
HS32	2	3	0	1.00e+00	0.00e+00	1	2	1.0e+01	1	0	0	0
HS33	2	3	0	-4.00e+00	0.00e+00	4	5	1.0e+01	4	0	0	0
HS34	2	3	0	-8.34e-01	2.98e-07	6	7	1.0e+01	3	3	0	0
HS35	1	3	0	1.11e-01	0.00e+00	1	2	1.0e+01	1	0	0	0
HS35I	1	3	0	1.11e-01	0.00e+00	1	2	1.0e+01	1	0	0	0
HS35MOD	1	3	0	2.50e-01	8.88e-16	1	2	1.0e+01	1	0	0	0
HS36	1	3	0	-3.30e+03	0.00e+00	2	4	1.0e+01	2	0	0	0
HS37	2	3	0	-3.46e+03	1.78e-14	4	5	1.0e+01	4	0	0	0
HS39	2	4	0	-1.00e+00	6.71e-08	8	9	1.0e+01	1	7	0	0
HS40	3	4	0	-2.50e-01	2.18e-06	3	4	1.0e+01	0	3	0	0

HS41	1	4	0	1.93e+00	2.22e-16	5	6	1.0e+01	4	1	0	0
HS42	2	4	0	1.39e+01	2.58e-11	5	6	1.0e+01	2	3	0	0
HS43	3	4	0	-4.40e+01	2.05e-06	6	7	1.0e+01	1	5	0	0
HS44	6	4	0	-3.00e+00	0.00e+00	1	2	1.0e+01	1	0	0	0
HS44NEW	6	4	0	-1.50e+01	0.00e+00	5	6	1.0e+01	5	0	0	0
HS46	2	5	0	5.30e-10	5.74e-06	15	16	1.0e+01	13	2	0	0
HS47	3	5	0	3.30e-09	3.26e-06	14	16	1.0e+01	14	0	0	0
HS48	2	5	0	0.00e+00	0.00e+00	1	2	1.0e+01	1	0	0	0
HS49	2	5	0	3.52e-08	1.33e-15	14	15	1.0e+01	14	0	0	0
HS50	3	5	0	6.38e-13	9.33e-15	8	9	1.0e+01	8	0	0	0
HS51	3	5	0	0.00e+00	0.00e+00	1	2	1.0e+01	1	0	0	0
HS52	3	5	0	5.33e+00	2.22e-15	1	2	1.0e+01	1	0	0	0
HS53	3	5	0	4.09e+00	1.33e-15	1	2	1.0e+01	1	0	0	0
HS54	1	6	0	-8.61e-01	6.37e-12	700	701	1.0e+01	699	1	0	0
HS55	6	6	0	6.67e+00	2.22e-16	1	2	1.0e+01	0	1	0	0
HS56	4	7	0	9.11e-12	2.76e-06	17	24	3.2e+01	13	4	0	0
HS57	1	2	0	3.06e-02	0.00e+00	1	2	1.0e+01	1	0	0	0
HS59	3	2	0	-7.80e+00	2.96e-11	12	15	1.0e+01	8	4	0	0
HS6	1	2	0	0.00e+00	8.88e-15	2	3	1.0e+01	1	1	0	0
HS60	1	3	0	3.26e-02	1.20e-07	6	7	1.0e+01	6	0	0	0
HS61	2	3	0	-1.44e+02	2.85e-12	6	8	1.0e+01	1	3	1	1
HS62	1	3	0	-2.63e+04	1.32e-16	5	8	1.0e+01	5	0	0	0
HS63	2	3	0	9.62e+02	1.81e-08	7	8	1.0e+01	2	5	0	0
HS64	1	3	0	6.30e+03	2.26e-12	15	16	2.6e+03	10	5	0	0
HS65	1	3	0	9.54e-01	5.05e-07	4	5	1.0e+01	1	3	0	0
HS66	2	3	0	5.18e-01	3.88e-09	4	5	1.0e+01	2	2	0	0
HS68	2	4	0	-9.20e-01	1.04e-10	13	21	1.0e+01	9	4	0	0
HS69	2	4	0	-9.57e+02	6.41e-12	10	21	6.2e+01	5	5	0	0
HS7	1	2	0	-1.73e+00	7.92e-08	14	17	1.0e+01	3	11	0	0
HS70	1	4	0	1.86e-01	0.00e+00	26	35	1.0e+01	26	0	0	0
HS71	2	4	0	1.70e+01	6.70e-10	5	6	1.0e+01	0	5	0	0
HS72	2	4	0	7.28e+02	1.51e-11	12	13	1.6e+04	0	12	0	0
HS73	3	4	0	2.99e+01	5.01e-07	3	4	1.0e+01	2	1	0	0
HS74	5	4	0	5.13e+03	3.60e-07	5	6	1.0e+01	0	5	0	0
HS75	5	4	0	5.17e+03	2.36e-11	6	7	2.0e+01	0	6	0	0
HS76	3	4	0	-4.68e+00	0.00e+00	1	2	1.0e+01	1	0	0	0
HS76I	3	4	0	-4.68e+00	0.00e+00	1	2	1.0e+01	1	0	0	0
HS77	2	5	0	2.42e-01	5.64e-10	9	13	1.0e+01	6	1	1	1
HS78	3	5	0	-2.92e+00	1.51e-09	4	5	1.0e+01	0	4	0	0
HS79	3	5	0	7.88e-02	1.19e-08	4	5	1.0e+01	4	0	0	0
HS8	2	2	0	-1.00e+00	1.86e-10	5	6	1.0e+01	0	5	0	0
HS80	3	5	0	5.39e-02	4.85e-10	7	8	1.0e+01	2	5	0	0
HS81	3	5	0	5.39e-02	1.50e-09	7	8	1.0e+01	2	5	0	0

HS83	3	5	0	-3.07e+04	4.63e-09	3	4	1.1e+03	1	2	0	0
HS84	3	5	0	-5.28e+06	2.33e-10	3	4	1.0e+01	3	0	0	0
HS86	10	5	0	-3.23e+01	2.66e-15	3	4	1.0e+01	3	0	0	0
HS88	1	2	0	1.36e+00	4.73e-15	16	24	1.1e+03	2	12	1	1
HS89	1	3	0	1.36e+00	1.51e-13	16	29	1.9e+03	1	11	2	2
HS9	1	2	0	-5.00e-01	1.78e-15	5	10	1.0e+01	5	0	0	0
HS90	1	4	-1	0.00e+00	1.33e-01	4	37	5.0e+01	2	0	1	1
HS91	1	5	0	1.36e+00	1.01e-12	19	22	1.3e+03	5	14	0	0
HS92	1	6	0	1.36e+00	4.21e-11	21	56	1.3e+03	5	14	1	1
HS93	2	6	0	1.35e+02	1.50e-13	6	7	9.9e+01	2	4	0	0
HS95	4	6	0	1.56e-02	0.00e+00	1	2	1.0e+01	0	1	0	0
HS96	4	6	0	1.56e-02	0.00e+00	1	2	1.0e+01	0	1	0	0
HS97	4	6	0	4.07e+00	0.00e+00	10	15	1.0e+01	7	3	0	0
HS98	4	6	0	4.07e+00	0.00e+00	10	15	1.0e+01	7	3	0	0
HS99	2	7	0	-8.31e+08	9.04e-11	4	5	5.4e+02	1	3	0	0
HUBFIT	1	2	0	1.69e-02	0.00e+00	1	2	1.0e+01	1	0	0	0
HYDCAR20	99	99	0	0.00e+00	4.89e-06	8	11	1.0e+01	0	8	0	0
HYDCAR6	29	29	0	0.00e+00	2.37e-06	5	8	1.0e+01	0	5	0	0
HYPCIR	2	2	0	0.00e+00	1.31e-09	4	7	1.0e+01	0	4	0	0
KIWCRESC	2	3	0	-7.24e-07	1.45e-06	9	11	1.0e+01	1	8	0	0
LAUNCH	28	25	0	9.00e+00	6.34e-13	6	6	3.0e+01	5	0	0	0
LEWISPOL	9	6	-8	1.16e+00	5.79e-05	14	35	9.6e+03	3	9	1	1
LIN	2	4	0	-1.96e-02	1.17e-16	16	17	1.0e+01	16	0	0	0
LINSPANH	33	97	0	-7.70e+01	5.07e-13	1	2	1.0e+01	0	1	0	0
LOADBAL	31	31	0	4.53e-01	4.40e-14	7	8	1.0e+01	7	0	0	0
LOOTSMA	2	3	-1	6.00e+00	4.00e+00	0	1	1.0e+01	0	0	0	0
LOTSCHD	7	12	0	2.40e+03	1.46e-13	1	2	2.0e+01	0	1	0	0
LSNNODOC	4	5	0	1.23e+02	1.78e-15	6	6	2.0e+01	5	0	0	0
LSQFIT	1	2	0	3.38e-02	0.00e+00	1	2	1.0e+01	1	0	0	0
MADSEN	6	3	0	6.16e-01	1.21e-09	9	12	1.0e+01	4	5	0	0
MAKELA1	2	3	0	-1.41e+00	2.26e-12	6	7	1.0e+01	3	3	0	0
MAKELA2	3	3	0	7.20e+00	7.99e-15	4	5	1.0e+01	1	3	0	0
MAKELA3	20	21	0	1.51e-14	1.89e-05	15	16	1.0e+01	1	14	0	0
MAKELA4	40	21	0	6.16e-15	1.07e-14	211	211	3.0e+01	209	1	0	0
MARATOS	1	2	0	-1.00e+00	1.45e-06	13	24	1.0e+01	3	10	0	0
MATRIX2	2	6	0	9.54e-07	0.00e+00	10	11	1.0e+01	10	0	0	0
MESH	48	41	0	-1.48e-04	5.30e-06	5	7	1.0e+01	4	1	0	0
METHANB8	31	31	0	0.00e+00	3.07e-07	2	3	1.0e+01	0	2	0	0
METHANL8	31	31	0	0.00e+00	4.36e-06	4	5	1.0e+01	0	4	0	0
MIFFLIN1	2	3	0	-1.00e+00	2.85e-08	8	9	1.0e+01	2	6	0	0
MIFFLIN2	2	3	0	-1.00e+00	9.63e-08	7	8	1.0e+01	2	5	0	0
MINMAXBD	20	5	0	1.16e+02	1.26e-08	17	46	1.0e+01	6	11	0	0
MINMAXRB	4	3	0	0.00e+00	0.00e+00	4	6	1.0e+01	2	2	0	0

MISTAKE	13	9	0	-1.00e+00	1.01e-05	8	9	1.0e+01	2	6	0	0
MRIBASIS	55	36	0	1.82e+01	2.18e-09	4	5	1.0e+01	1	3	0	0
MSS1	73	90	-9	-1.50e+01	5.86e-08	8460	642673	Inf	26	77	8	1826
MWRIGHT	3	5	0	2.50e+01	1.04e-12	7	10	1.0e+01	4	3	0	0
NASH	24	72	-1	0.00e+00	4.37e+01	2623	2624	2.0e+01	0	2623	0	0
NYSTROM5	20	18	0	0.00e+00	5.30e-07	23	36	3.0e+01	1	22	0	0
ODFITS	6	10	0	-2.38e+03	1.14e-13	7	8	1.0e+01	6	1	0	0
OPTCNTRL	20	32	0	5.50e+02	1.30e-09	3	4	1.3e+02	0	3	0	0
OPTPRLOC	30	30	0	-1.64e+01	6.21e-13	5	6	1.0e+01	1	4	0	0
ORTHREGB	6	27	0	8.67e-18	8.71e-09	2	3	1.0e+01	0	2	0	0
PENTAGON	15	6	0	1.37e-04	5.55e-17	10	14	1.0e+01	10	0	0	0
PFIT1	3	3	0	0.00e+00	3.77e-15	257	4079	1.2e+78	0	256	0	0
PFIT2	3	3	0	0.00e+00	1.02e-14	131	1720	1.7e+39	0	128	1	1
PFIT3	3	3	0	0.00e+00	2.94e-07	203	2917	3.5e+14	0	203	0	0
PFIT4	3	3	0	0.00e+00	1.03e-13	144	1743	3.4e+39	0	143	0	0
POLAK1	2	3	0	2.72e+00	4.38e-11	8	9	1.0e+01	2	6	0	0
POLAK2	2	11	0	5.46e+01	2.98e-13	10	15	1.0e+01	2	8	0	0
POLAK3	10	12	0	5.93e+00	2.25e-10	11	16	1.0e+01	6	5	0	0
POLAK4	3	3	0	-9.21e-19	1.73e-18	4	5	1.0e+01	0	4	0	0
POLAK5	2	3	0	5.00e+01	0.00e+00	3	4	2.0e+01	0	3	0	0
POLAK6	4	5	0	-4.40e+01	2.57e-11	19	45	1.0e+01	10	9	0	0
PORTFL1	1	12	0	2.05e-02	2.22e-16	1	2	1.0e+01	1	0	0	0
PORTFL2	1	12	0	2.97e-02	1.11e-16	1	2	1.0e+01	1	0	0	0
PORTFL3	1	12	0	3.27e-02	2.78e-16	1	2	1.0e+01	1	0	0	0
PORTFL4	1	12	0	2.63e-02	1.67e-16	1	2	1.0e+01	1	0	0	0
PORTFL6	1	12	0	2.58e-02	8.33e-17	1	2	1.0e+01	1	0	0	0
POWELLBS	2	2	0	0.00e+00	2.76e-06	53	386	1.0e+01	0	53	0	0
POWELLSQ	2	2	1	0.00e+00	9.02e+00	10001	370442	1.0e+01	0	9993	4	4
PRODPL0	29	60	0	5.88e+01	6.65e-10	10	12	2.0e+01	1	9	0	0
PRODPL1	29	60	0	3.57e+01	1.71e-13	9	10	2.0e+01	2	7	0	0
QC	4	9	-9	-8.61e+02	1.25e-10	10001	279975	1.0e+01	2	0	0	0
QCNEW	3	9	0	-8.07e+02	0.00e+00	1	2	1.0e+01	1	0	0	0
QPCBLEND	74	83	0	-7.84e-03	3.18e-16	1	2	1.0e+01	1	0	0	0
QPNBLEND	74	83	0	-9.14e-03	1.50e-15	13	14	1.0e+01	13	0	0	0
RECIPE	3	3	0	0.00e+00	5.96e-06	11	12	1.0e+01	0	11	0	0
RES	14	20	0	0.00e+00	1.95e-14	0	1	1.0e+01	0	0	0	0
RK23	11	17	0	8.33e-02	1.49e-13	8	10	1.0e+01	3	5	0	0
ROBOT	2	14	0	6.59e+00	7.97e-10	6	7	1.0e+01	1	5	0	0
ROSENMMX	4	5	0	-4.40e+01	1.41e-07	10	23	1.0e+01	2	8	0	0
RSNBRNE	2	2	0	0.00e+00	0.00e+00	8	37	1.0e+01	0	8	0	0
S268	5	5	0	-2.55e-11	0.00e+00	1	2	1.0e+01	1	0	0	0
S316-322	1	2	-1	8.00e+02	1.00e+00	0	1	1.0e+01	0	0	0	0
SIMPLLPA	2	2	0	1.00e+00	0.00e+00	3	4	1.0e+01	2	1	0	0

SIMPLLPB	3	2	0	1.10e+00	0.00e+00	3	4	1.0e+01	2	1	0	0
SINVALNE	2	2	0	0.00e+00	0.00e+00	5	42	1.0e+01	0	5	0	0
SNAKE	2	2	0	-2.17e-14	2.17e-18	5	6	1.0e+01	3	2	0	0
SPANHYD	33	97	0	2.40e+02	1.54e-12	5	6	1.0e+01	5	0	0	0
SPIRAL	2	3	0	-1.64e-10	3.27e-10	45	64	1.0e+01	32	13	0	0
SUPERSIM	2	2	0	6.67e-01	1.11e-16	1	2	1.0e+01	0	1	0	0
SWOPF	92	83	0	6.79e-02	4.30e-10	8	9	1.0e+01	1	7	0	0
SYNTHES1	6	6	0	7.59e-01	1.32e-10	4	6	1.0e+01	1	1	1	1
SYNTHES2	14	11	0	-5.54e-01	1.67e-16	5	6	1.0e+01	4	1	0	0
SYNTHES3	23	17	0	1.51e+01	4.16e-16	5	6	1.0e+01	5	0	0	0
TAME	1	2	0	0.00e+00	0.00e+00	1	2	1.0e+01	0	1	0	0
TENBARS1	9	18	0	2.30e+03	1.72e-11	45	76	4.0e+01	3	38	2	2
TENBARS2	8	18	0	2.30e+03	7.60e-07	49	98	2.0e+01	2	39	4	4
TENBARS3	8	18	0	2.25e+03	1.07e-06	46	104	1.0e+01	3	39	2	2
TENBARS4	9	18	0	3.68e+02	5.09e-09	63	109	3.6e+01	48	15	0	0
TRIGGER	6	7	0	0.00e+00	6.09e-07	38	257	1.0e+01	0	38	0	0
TRO3X3	13	30	-1	-5.04e+04	1.00e+00	11	12	6.7e+01	8	3	0	0
TRY-B	1	2	0	1.00e+00	2.51e-06	6	7	1.0e+01	6	0	0	0
TWOBARS	2	2	0	1.51e+00	7.66e-06	6	7	1.0e+01	1	5	0	0
WACHBIEG	2	3	-1	-1.00e+00	1.50e+00	5	6	1.0e+01	0	5	0	0
WATER	10	31	0	1.05e+04	3.69e-13	10	11	3.5e+02	6	2	1	1
WOMFLET	3	3	0	2.49e-18	1.20e-05	30	231	1.0e+01	3	27	0	0
YFITNE	17	3	0	0.00e+00	8.41e-06	19	76	1.0e+01	0	19	0	0
ZANGWIL3	3	3	0	0.00e+00	0.00e+00	1	2	1.0e+01	0	1	0	0
ZECEVIC2	2	2	0	-4.12e+00	2.22e-16	1	2	1.0e+01	1	0	0	0
ZECEVIC3	2	2	0	9.73e+01	7.64e-08	8	10	2.0e+01	3	5	0	0
ZECEVIC4	2	2	0	7.56e+00	0.00e+00	5	6	1.0e+01	2	3	0	0
ZY2	2	3	0	2.00e+00	0.00e+00	4	5	1.0e+01	4	0	0	0

Table A.2: Results for PenSQO on the CUTEst problems of size $m \geq 1$ and $\max\{m, n\} \leq 100$.

prob	m	n	status	f	v	iters	feval	σ
ACOPP14	68	38	0	8.08e+03	5.45e-09	5	41	1.2e+03
ACOPR14	82	38	0	8.08e+03	4.12e-05	7	45	1.4e+02
AIRCRFTA	5	8	0	0.00e+00	6.04e-06	2	3	1.0e+01
AIRPORT	42	84	0	4.80e+04	4.00e-13	16	131	2.7e+03
ALLINITA	4	4	-9	3.33e+01	8.47e-13	10001	906005	Inf
ALLINITC	1	4	0	3.05e+01	1.10e-13	43	631	8.4e+07
ALSOTAME	1	2	0	8.21e-02	2.92e-13	4	5	1.0e+01
ANTWERP	10	27	1	2.44e+04	7.28e-12	10001	10002	1.0e+01
ARGAUSS	15	3	-8	0.00e+00	3.38e-04	2	3	1.0e+01
AVGASA	10	8	0	-4.63e+00	2.83e-15	1	2	1.0e+01

AVGASB	10	8	0	-4.48e+00	1.11e-16	1	2	1.0e+01
AVION2	15	49	-9	9.47e+07	1.38e-12	10001	319727	1.4e+04
BATCH	73	48	0	2.59e+05	6.30e-07	8	28	7.2e+01
BIGGSC4	7	4	0	-2.44e+01	0.00e+00	3	5	1.0e+01
BOOTH	2	2	0	0.00e+00	0.00e+00	1	2	1.0e+01
BT1	1	2	0	-9.99e-01	9.79e-06	2043	43284	1.5e+03
BT10	2	2	0	-1.00e+00	5.57e-09	6	7	1.0e+01
BT11	3	5	0	8.25e-01	2.57e-08	7	8	1.0e+01
BT12	3	5	0	6.19e+00	5.15e-06	3	4	1.0e+01
BT13	1	5	0	0.00e+00	7.82e-06	15	16	1.0e+01
BT2	1	3	0	3.26e-02	9.10e-07	11	12	1.0e+01
BT3	3	5	0	4.09e+00	1.07e-14	1	2	1.0e+01
BT4	2	3	0	-4.55e+01	1.54e-09	7	27	1.0e+01
BT5	2	3	0	9.62e+02	5.59e-06	5	8	1.0e+01
BT6	2	5	0	2.77e-01	5.87e-10	11	20	1.0e+01
BT7	3	5	0	3.60e+02	3.83e-12	40	306	2.4e+03
BT8	2	5	0	1.00e+00	7.63e-06	9	10	1.0e+01
BT9	2	4	0	-1.00e+00	6.71e-08	8	9	1.0e+01
BURKEHAN	1	1	-1	0.00e+00	1.00e+00	0	1	1.0e+01
BYRDSPHR	2	3	0	-4.68e+00	6.60e-10	8	21	4.0e+01
CANTILVR	1	5	0	1.34e+00	5.94e-07	10	11	1.0e+01
CB2	3	3	0	1.95e+00	9.37e-12	6	7	1.0e+01
CB3	3	3	0	2.00e+00	1.43e-12	6	7	1.0e+01
CHACONN1	3	3	0	1.95e+00	7.33e-08	4	5	1.0e+01
CHACONN2	3	3	0	2.00e+00	3.16e-12	6	7	1.0e+01
CLUSTER	2	2	0	0.00e+00	7.22e-06	7	8	1.0e+01
CONGIGMZ	5	3	0	2.80e+01	2.95e-10	4	5	1.0e+01
COOLHANS	9	9	0	0.00e+00	2.35e-08	199	212	1.0e+01
CRESC4	8	6	-1	1.59e-08	7.50e-01	19	26	3.2e+02
CRESC50	100	6	0	7.86e-01	6.85e-12	67	181	1.3e+06
CSFI1	4	5	0	-4.91e+01	9.61e-11	6	9	1.0e+01
CSFI2	4	5	0	5.50e+01	8.81e-13	8	9	1.0e+01
CUBENE	2	2	0	0.00e+00	0.00e+00	4	15	1.0e+01
DALLASS	31	46	0	-3.24e+04	5.37e-13	13	21	1.0e+01
DEGENLPA	15	20	0	3.06e+00	6.55e-07	5	5	3.0e+01
DEGENLPB	15	20	0	-3.07e+01	5.88e-07	3	6	3.0e+01
DEMBO7	20	16	1	1.75e+02	8.38e-13	10001	257134	Inf
DEMYMALO	3	3	0	-3.00e+00	0.00e+00	8	8	2.0e+01
DIPIGRI	4	7	0	6.81e+02	1.11e-09	12	47	1.0e+01
DISC2	23	29	0	1.56e+00	4.60e-06	19	100	1.0e+01
DIXCHLNG	5	10	0	2.47e+03	2.34e-09	9	49	6.8e+02
DNIEPER	24	61	0	1.87e+04	2.10e-07	7	50	1.5e+03
DUAL1	1	85	0	3.50e-02	1.08e-16	1	2	1.0e+01

DUAL2	1	96	0	3.37e-02	2.93e-16	1	2	1.0e+01
DUAL4	1	75	0	7.46e-01	4.36e-16	1	2	1.0e+01
EQC	3	9	1	-8.28e+02	0.00e+00	10001	19004	1.0e+01
ERRINBAR	9	18	1	2.83e+01	3.60e-06	10001	59171	8.0e+01
EXPFITA	22	5	0	1.14e-03	0.00e+00	14	18	1.0e+01
EXTRASIM	1	2	0	1.00e+00	0.00e+00	1	2	1.0e+01
FCCU	8	19	0	1.11e+01	1.78e-14	1	2	1.0e+01
FLETCHER	4	4	-1	4.00e+00	1.00e+00	1	2	1.0e+01
FLT	2	2	-6	0.00e+00	4.86e-05	8	9	1.0e+01
GENHS28	8	10	0	9.27e-01	3.89e-15	1	2	1.0e+01
GIGOMEZ1	3	3	0	-3.00e+00	3.43e-11	6	11	1.0e+01
GIGOMEZ2	3	3	0	1.95e+00	1.63e-07	5	6	1.0e+01
GIGOMEZ3	3	3	0	2.00e+00	9.62e-13	6	7	1.0e+01
GOFFIN	50	51	0	7.28e-14	0.00e+00	26	26	2.0e+01
GOTTFR	2	2	0	0.00e+00	2.24e-10	5	10	1.0e+01
GOULDQP1	17	32	0	-3.49e+03	0.00e+00	1	2	1.0e+01
GROWTH	12	3	-1	0.00e+00	2.44e+00	20	75	1.0e+01
HAIFAS	9	13	0	-4.50e-01	4.01e-06	21	219	1.0e+01
HALDMADS	42	6	0	1.22e-04	1.96e-09	6	11	1.0e+01
HATFLDF	3	3	1	0.00e+00	9.50e-03	10001	69110	4.0e+01
HATFLDG	25	25	0	0.00e+00	4.26e-05	6	31	1.0e+01
HATFLDH	7	4	0	-2.44e+01	8.88e-16	1	2	1.0e+01
HEART6	6	6	0	0.00e+00	6.80e-07	33	234	5.4e+09
HEART8	8	8	0	0.00e+00	1.01e-08	12	55	5.1e+03
HIMMELBA	2	2	0	0.00e+00	0.00e+00	1	2	1.0e+01
HIMMELBC	2	2	0	0.00e+00	1.72e-08	5	8	1.0e+01
HIMMELBD	2	2	1	0.00e+00	4.67e+01	10001	400990	1.0e+01
HIMMELBE	3	3	0	0.00e+00	2.22e-16	2	3	1.0e+01
HIMMELBI	12	100	0	-1.74e+03	8.08e-14	9	10	1.0e+01
HIMMELBJ	14	45	-10	-3.04e+01	1.01e+02	0	2	1.0e+01
HIMMELBK	14	24	0	5.18e-02	4.11e-10	8	12	1.0e+01
HIMMELP2	1	2	0	-6.40e+00	0.00e+00	94	1267	1.0e+01
HIMMELP3	2	2	0	-5.90e+01	0.00e+00	3	5	1.0e+01
HIMMELP4	3	2	0	-5.90e+01	0.00e+00	3	6	1.0e+01
HIMMELP5	3	2	0	-5.90e+01	0.00e+00	7	10	1.0e+01
HIMMELP6	5	2	0	-5.90e+01	0.00e+00	6	10	1.0e+01
HONG	1	4	0	2.26e+01	1.67e-16	6	7	1.0e+01
HS10	1	2	0	-1.00e+00	4.32e-10	9	10	1.0e+01
HS100	4	7	0	6.81e+02	1.11e-09	12	47	1.0e+01
HS100LNP	2	7	0	6.81e+02	1.52e-09	7	30	1.0e+01
HS100MOD	4	7	0	6.79e+02	1.12e-11	8	43	1.0e+01
HS101	5	7	0	1.81e+03	2.50e-16	21	137	7.3e+03
HS102	5	7	0	9.12e+02	2.78e-17	18	68	3.8e+03

HS103	5	7	0	5.44e+02	7.56e-16	16	32	1.0e+03
HS104	5	8	0	3.95e+00	4.53e-09	12	88	1.0e+01
HS105	1	8	0	1.04e+03	0.00e+00	15	22	1.0e+01
HS106	6	8	0	7.05e+03	3.77e-08	522	2498	1.0e+01
HS107	6	9	0	5.06e+03	5.45e-10	5	28	1.5e+03
HS108	13	9	0	-8.66e-01	9.40e-06	16	38	1.0e+01
HS109	10	9	0	5.36e+03	5.03e-08	11	47	3.4e+03
HS11	1	2	0	-8.50e+00	6.71e-08	5	6	1.0e+01
HS111	3	10	0	-4.78e+01	1.96e-07	16	32	2.0e+01
HS111LNP	3	10	0	-4.78e+01	1.02e-06	15	27	2.0e+01
HS112	3	10	0	-4.78e+01	4.97e-14	11	12	1.0e+01
HS113	8	10	0	2.43e+01	2.88e-13	5	6	1.0e+01
HS114	11	10	0	-1.77e+03	1.24e-12	39	87	2.0e+02
HS116	14	13	0	9.76e+01	5.81e-14	13	29	5.0e+01
HS117	5	15	0	3.23e+01	0.00e+00	13	24	1.0e+01
HS118	17	15	0	6.65e+02	0.00e+00	1	2	1.0e+01
HS119	8	16	0	2.45e+02	6.92e-15	7	8	1.0e+01
HS12	1	2	0	-3.00e+01	1.25e-10	6	19	1.0e+01
HS13	1	2	0	1.00e+00	0.00e+00	26	26	2.0e+01
HS14	2	2	0	1.39e+00	7.67e-13	5	6	1.0e+01
HS15	2	2	0	3.06e+02	0.00e+00	7	61	6.6e+03
HS16	2	2	0	2.31e+01	0.00e+00	4	5	1.0e+01
HS17	2	2	0	1.00e+00	0.00e+00	7	11	1.0e+01
HS18	2	2	0	5.00e+00	5.54e-06	5	6	1.0e+01
HS19	2	2	0	-6.96e+03	7.26e-09	6	37	2.5e+02
HS20	3	2	0	4.02e+01	0.00e+00	3	4	1.0e+01
HS21	1	2	0	-1.00e+02	0.00e+00	1	2	1.0e+01
HS21MOD	1	7	0	-9.60e+01	0.00e+00	1	2	1.0e+01
HS22	2	2	0	1.00e+00	2.10e-09	4	5	1.0e+01
HS23	5	2	0	2.00e+00	0.00e+00	5	6	1.0e+01
HS24	3	2	0	-1.00e+00	0.00e+00	6	10	2.0e+01
HS26	1	3	0	6.77e-11	4.63e-06	14	16	1.0e+01
HS268	5	5	0	-2.55e-11	0.00e+00	1	2	1.0e+01
HS27	1	3	0	4.00e-02	2.31e-11	152	1987	1.0e+01
HS28	1	3	0	1.23e-32	4.44e-16	1	2	1.0e+01
HS29	1	3	0	-2.26e+01	3.77e-10	7	15	1.0e+01
HS30	1	3	0	1.00e+00	0.00e+00	9	10	1.0e+01
HS31	1	3	0	6.00e+00	1.64e-08	5	6	1.0e+01
HS32	2	3	0	1.00e+00	0.00e+00	1	2	1.0e+01
HS33	2	3	0	-4.00e+00	0.00e+00	4	5	1.0e+01
HS34	2	3	0	-8.34e-01	3.84e-06	47	469	1.0e+01
HS35	1	3	0	1.11e-01	0.00e+00	1	2	1.0e+01
HS35I	1	3	0	1.11e-01	0.00e+00	1	2	1.0e+01

HS35MOD	1	3	0	2.50e-01	8.88e-16	1	2	1.0e+01
HS36	1	3	0	-3.30e+03	0.00e+00	2	4	1.0e+01
HS37	2	3	0	-3.46e+03	1.78e-14	4	5	1.0e+01
HS39	2	4	0	-1.00e+00	6.71e-08	8	9	1.0e+01
HS40	3	4	0	-2.50e-01	2.18e-06	3	4	1.0e+01
HS41	1	4	0	1.93e+00	2.22e-16	5	6	1.0e+01
HS42	2	4	0	1.39e+01	2.58e-11	5	6	1.0e+01
HS43	3	4	0	-4.40e+01	8.83e-09	6	13	1.0e+01
HS44	6	4	0	-3.00e+00	0.00e+00	1	2	1.0e+01
HS44NEW	6	4	0	-1.50e+01	0.00e+00	5	6	1.0e+01
HS46	2	5	0	5.86e-10	6.03e-06	16	24	1.0e+01
HS47	3	5	0	3.53e-09	3.41e-06	14	18	1.0e+01
HS48	2	5	0	0.00e+00	0.00e+00	1	2	1.0e+01
HS49	2	5	0	3.52e-08	1.33e-15	14	15	1.0e+01
HS50	3	5	0	6.38e-13	9.33e-15	8	9	1.0e+01
HS51	3	5	0	0.00e+00	0.00e+00	1	2	1.0e+01
HS52	3	5	0	5.33e+00	2.22e-15	1	2	1.0e+01
HS53	3	5	0	4.09e+00	1.33e-15	1	2	1.0e+01
HS54	1	6	0	-8.61e-01	6.37e-12	700	701	1.0e+01
HS55	6	6	0	6.67e+00	2.22e-16	1	2	1.0e+01
HS56	4	7	0	-2.51e-16	9.67e-06	37	149	9.4e+01
HS57	1	2	0	3.06e-02	0.00e+00	1	2	1.0e+01
HS59	3	2	0	-7.80e+00	0.00e+00	23	167	1.0e+01
HS6	1	2	0	0.00e+00	7.63e-11	10	55	1.0e+01
HS60	1	3	0	3.26e-02	1.20e-07	6	7	1.0e+01
HS61	2	3	0	-1.44e+02	2.94e-07	4	6	1.0e+01
HS62	1	3	0	-2.63e+04	1.32e-16	5	8	1.0e+01
HS63	2	3	0	9.62e+02	8.36e-07	7	11	1.0e+01
HS64	1	3	0	6.30e+03	2.26e-12	15	16	2.6e+03
HS65	1	3	0	9.54e-01	3.22e-09	10	25	1.0e+01
HS66	2	3	0	5.18e-01	4.07e-07	6	17	1.0e+01
HS68	2	4	0	-9.20e-01	2.85e-06	14	38	1.0e+01
HS69	2	4	0	-9.57e+02	4.29e-13	9	22	6.2e+01
HS7	1	2	0	-1.73e+00	6.88e-07	7	32	1.0e+01
HS70	1	4	0	1.86e-01	0.00e+00	26	35	1.0e+01
HS71	2	4	0	1.70e+01	6.70e-10	5	6	1.0e+01
HS72	2	4	0	7.28e+02	3.50e-09	17	76	2.2e+04
HS73	3	4	0	2.99e+01	9.13e-08	4	9	1.0e+01
HS74	5	4	0	5.13e+03	3.60e-07	5	6	1.0e+01
HS75	5	4	0	5.17e+03	1.93e-12	8	21	4.0e+01
HS76	3	4	0	-4.68e+00	0.00e+00	1	2	1.0e+01
HS76I	3	4	0	-4.68e+00	0.00e+00	1	2	1.0e+01
HS77	2	5	0	2.42e-01	4.20e-09	12	45	1.0e+01

HS78	3	5	0	-2.92e+00	1.51e-09	4	5	1.0e+01
HS79	3	5	0	7.88e-02	1.19e-08	4	5	1.0e+01
HS8	2	2	0	-1.00e+00	1.86e-10	5	6	1.0e+01
HS80	3	5	0	5.39e-02	4.85e-10	7	8	1.0e+01
HS81	3	5	0	5.39e-02	1.20e-08	8	13	1.0e+01
HS83	3	5	0	-3.07e+04	2.66e-12	4	7	1.1e+03
HS84	3	5	0	-5.28e+06	2.33e-10	3	4	1.0e+01
HS86	10	5	0	-3.23e+01	2.66e-15	3	4	1.0e+01
HS88	1	2	0	1.36e+00	3.11e-13	17	78	1.8e+03
HS89	1	3	0	1.36e+00	2.72e-14	17	179	1.2e+03
HS9	1	2	0	-5.00e-01	1.78e-15	5	10	1.0e+01
HS90	1	4	0	1.36e+00	4.29e-13	19	137	1.1e+03
HS91	1	5	0	1.36e+00	3.06e-12	21	204	2.1e+03
HS92	1	6	0	1.36e+00	3.11e-12	21	119	1.7e+03
HS93	2	6	0	1.35e+02	1.38e-14	7	12	9.9e+01
HS95	4	6	0	1.56e-02	0.00e+00	1	2	1.0e+01
HS96	4	6	0	1.56e-02	0.00e+00	1	2	1.0e+01
HS97	4	6	0	4.07e+00	0.00e+00	10	15	1.0e+01
HS98	4	6	0	4.07e+00	0.00e+00	10	15	1.0e+01
HS99	2	7	0	-8.31e+08	3.38e-10	5	32	5.4e+02
HUBFIT	1	2	0	1.69e-02	0.00e+00	1	2	1.0e+01
HYDCAR20	99	99	0	0.00e+00	4.89e-06	8	11	1.0e+01
HYDCAR6	29	29	0	0.00e+00	2.37e-06	5	8	1.0e+01
HYPCIR	2	2	0	0.00e+00	1.31e-09	4	7	1.0e+01
KIWCRESC	2	3	0	-3.28e-06	6.57e-06	9	37	1.0e+01
LAUNCH	28	25	0	9.00e+00	4.33e-13	7	13	3.0e+01
LEWISPOL	9	6	-8	1.16e+00	5.79e-05	25	135	7.8e+03
LIN	2	4	0	-1.96e-02	1.17e-16	16	17	1.0e+01
LINSPANH	33	97	0	-7.70e+01	5.07e-13	1	2	1.0e+01
LOADBAL	31	31	0	4.53e-01	4.40e-14	7	8	1.0e+01
LOOTSMA	2	3	-1	6.00e+00	4.00e+00	0	1	1.0e+01
LOTSCHD	7	12	0	2.40e+03	1.40e-13	2	5	2.0e+01
LSNNODOC	4	5	0	1.23e+02	1.78e-15	6	6	2.0e+01
LSQFIT	1	2	0	3.38e-02	0.00e+00	1	2	1.0e+01
MADSEN	6	3	0	6.16e-01	7.53e-10	12	29	1.0e+01
MAKELA1	2	3	0	-1.41e+00	8.33e-10	6	8	1.0e+01
MAKELA2	3	3	0	7.20e+00	7.99e-15	4	5	1.0e+01
MAKELA3	20	21	0	-7.28e-14	6.78e-05	15	34	1.0e+01
MAKELA4	40	21	0	6.16e-15	1.07e-14	211	211	3.0e+01
MARATOS	1	2	0	-1.00e+00	2.45e-08	4	45	1.0e+01
MATRIX2	2	6	0	9.54e-07	0.00e+00	10	11	1.0e+01
MESH	48	41	-2	-1.00e+37	4.22e-14	448	3324	2.0e+01
METHANB8	31	31	0	0.00e+00	3.07e-07	2	3	1.0e+01

METHANL8	31	31	0	0.00e+00	4.36e-06	4	5	1.0e+01
MIFFLIN1	2	3	0	-1.00e+00	6.51e-06	13	78	1.0e+01
MIFFLIN2	2	3	0	-1.00e+00	2.94e-09	8	25	1.0e+01
MINMAXBD	20	5	0	1.16e+02	1.70e-06	12	51	1.0e+01
MINMAXRB	4	3	0	1.39e-17	0.00e+00	97	648	1.0e+01
MISTAKE	13	9	0	-1.00e+00	7.61e-09	7	11	1.0e+01
MRIBASIS	55	36	0	1.82e+01	2.18e-09	4	5	1.0e+01
MSS1	73	90	-9	-1.60e+01	2.82e-07	9422	675629	Inf
MWRIGHT	3	5	0	2.50e+01	1.04e-12	7	10	1.0e+01
NASH	24	72	-1	0.00e+00	4.37e+01	2623	2624	2.0e+01
NYSTROM5	20	18	0	0.00e+00	5.21e-07	23	39	3.0e+01
ODFITS	6	10	0	-2.38e+03	1.14e-13	7	8	1.0e+01
OPTCNTRL	20	32	0	5.50e+02	5.13e-06	5	20	8.0e+01
OPTPRLOC	30	30	0	-1.64e+01	5.58e-13	19	94	1.0e+01
ORTHREGB	6	27	0	8.67e-18	8.71e-09	2	3	1.0e+01
PENTAGON	15	6	0	1.37e-04	5.55e-17	10	14	1.0e+01
PFIT1	3	3	0	0.00e+00	3.77e-15	257	4079	1.2e+78
PFIT2	3	3	0	0.00e+00	7.11e-15	131	1719	1.7e+39
PFIT3	3	3	0	0.00e+00	2.94e-07	203	2917	3.5e+14
PFIT4	3	3	0	0.00e+00	1.03e-13	144	1743	3.4e+39
POLAK1	2	3	0	2.72e+00	5.47e-09	11	28	1.0e+01
POLAK2	2	11	0	5.46e+01	4.34e-06	5	22	1.0e+01
POLAK3	10	12	0	5.93e+00	1.41e-10	7	16	1.0e+01
POLAK4	3	3	0	-9.21e-19	1.73e-18	4	5	1.0e+01
POLAK5	2	3	0	5.00e+01	0.00e+00	3	4	2.0e+01
POLAK6	4	5	0	-4.40e+01	2.86e-10	14	85	1.0e+01
PORTFL1	1	12	0	2.05e-02	2.22e-16	1	2	1.0e+01
PORTFL2	1	12	0	2.97e-02	1.11e-16	1	2	1.0e+01
PORTFL3	1	12	0	3.27e-02	2.78e-16	1	2	1.0e+01
PORTFL4	1	12	0	2.63e-02	1.67e-16	1	2	1.0e+01
PORTFL6	1	12	0	2.58e-02	8.33e-17	1	2	1.0e+01
POWELLBS	2	2	0	0.00e+00	2.76e-06	53	386	1.0e+01
POWELLSQ	2	2	1	0.00e+00	9.02e+00	10001	370438	1.0e+01
PRODPL0	29	60	0	5.88e+01	7.13e-14	9	14	2.0e+01
PRODPL1	29	60	0	3.57e+01	1.05e-15	9	14	2.0e+01
QC	4	9	-9	-8.61e+02	1.25e-10	10001	279975	1.0e+01
QCNEW	3	9	0	-8.07e+02	0.00e+00	1	2	1.0e+01
QPCBLEND	74	83	0	-7.84e-03	3.18e-16	1	2	1.0e+01
QPNBLEND	74	83	0	-9.14e-03	1.50e-15	13	14	1.0e+01
RECIPE	3	3	0	0.00e+00	5.96e-06	11	12	1.0e+01
RES	14	20	0	0.00e+00	1.95e-14	0	1	1.0e+01
RK23	11	17	0	8.33e-02	8.48e-12	10	37	1.0e+01
ROBOT	2	14	0	6.59e+00	2.22e-15	7	10	1.0e+01

ROSENMMX	4	5	0	-4.40e+01	6.55e-06	8	37	1.0e+01
RSNBRNE	2	2	0	0.00e+00	0.00e+00	8	37	1.0e+01
S268	5	5	0	-2.55e-11	0.00e+00	1	2	1.0e+01
S316-322	1	2	-1	8.00e+02	1.00e+00	0	1	1.0e+01
SIMPLLPA	2	2	0	1.00e+00	0.00e+00	3	4	1.0e+01
SIMPLLPB	3	2	0	1.10e+00	0.00e+00	3	4	1.0e+01
SINVALNE	2	2	0	0.00e+00	0.00e+00	5	42	1.0e+01
SNAKE	2	2	0	1.56e-14	0.00e+00	8	23	1.0e+01
SPANHYD	33	97	0	2.40e+02	1.54e-12	5	6	1.0e+01
SPIRAL	2	3	0	-6.24e-08	1.25e-07	179	1306	1.0e+01
SUPERSIM	2	2	0	6.67e-01	1.11e-16	1	2	1.0e+01
SWOPF	92	83	0	6.79e-02	4.72e-11	466	930	1.0e+01
SYNTHES1	6	6	0	7.59e-01	1.32e-10	4	5	1.0e+01
SYNTHES2	14	11	0	-5.54e-01	1.67e-16	5	6	1.0e+01
SYNTHES3	23	17	0	1.51e+01	4.16e-16	5	6	1.0e+01
TAME	1	2	0	0.00e+00	0.00e+00	1	2	1.0e+01
TENBARS1	9	18	0	2.30e+03	1.57e-05	50	123	2.0e+01
TENBARS2	8	18	0	2.30e+03	2.71e-08	49	91	2.0e+01
TENBARS3	8	18	0	2.25e+03	4.58e-11	189	432	2.0e+01
TENBARS4	9	18	1	3.69e+02	1.99e-05	10001	121686	5.7e+02
TRIGGER	6	7	0	0.00e+00	6.09e-07	38	257	1.0e+01
TRO3X3	13	30	0	9.00e+00	2.18e-12	85	184	6.7e+01
TRY-B	1	2	0	1.00e+00	2.51e-06	6	7	1.0e+01
TWOBARS	2	2	0	1.51e+00	4.70e-09	6	9	1.0e+01
WACHBIEG	2	3	-1	-1.00e+00	1.50e+00	5	6	1.0e+01
WATER	10	31	0	1.05e+04	2.56e-13	10	12	3.5e+02
WOMFLET	3	3	1	2.75e-06	2.21e-07	10001	992903	Inf
YFITNE	17	3	0	0.00e+00	8.41e-06	19	76	1.0e+01
ZANGWIL3	3	3	0	0.00e+00	0.00e+00	1	2	1.0e+01
ZECEVIC2	2	2	0	-4.12e+00	2.22e-16	1	2	1.0e+01
ZECEVIC3	2	2	0	9.73e+01	2.34e-11	6	13	1.0e+01
ZECEVIC4	2	2	0	7.56e+00	0.00e+00	5	6	1.0e+01
ZY2	2	3	0	2.00e+00	0.00e+00	4	5	1.0e+01

Table A.3: Results for FiSQO on the CUTEst problems of size $m \geq 1$ and $100 < \max\{m,n\} \leq 1000$.

prob	m	n	status	f	v	iters	feval	σ	#o	#v	#b	#p
ACOPP118	608	344	0	1.30e+05	4.17e-08	6	7	9.3e+02	2	4	0	0
ACOPP30	142	72	0	5.77e+02	4.09e-12	6	9	7.4e+01	1	5	0	0
ACOPP57	274	128	0	4.17e+04	4.90e-10	6	7	3.8e+03	0	6	0	0
ACOPR118	726	344	0	1.30e+05	1.81e-04	612	3591	4.4e+02	443	167	1	1
ACOPR30	172	72	0	5.77e+02	1.49e-08	306	609	4.9e+01	287	19	0	0
ACOPR57	331	128	0	4.17e+04	1.58e-06	7	16	3.8e+03	0	5	1	1
AGG	488	163	2	-2.05e+06	9.35e+06	8096	8097	4.0e+01	8096	0	0	0
C-RELOAD	284	342	0	-1.02e+00	2.26e-09	62	63	1.0e+01	56	6	0	0
CRESC100	200	6	0	7.44e-01	2.69e-10	37	48	8.2e+04	5	32	0	0
DALLASL	667	906	0	-2.03e+05	5.68e-09	15	16	3.0e+01	15	0	0	0
DALLASM	151	196	0	-4.82e+04	1.64e-12	12	13	1.0e+01	12	0	0	0
DUAL3	1	111	0	1.36e-01	2.66e-15	1	2	1.0e+01	0	1	0	0
DUALC1	215	9	0	6.16e+03	4.16e-17	1	2	4.0e+03	0	1	0	0
DUALC2	229	7	0	3.55e+03	3.89e-16	1	2	4.8e+03	0	1	0	0
DUALC5	278	8	0	4.27e+02	7.22e-16	1	2	7.9e+02	0	1	0	0
DUALC8	503	8	0	1.83e+04	0.00e+00	1	2	6.8e+03	0	1	0	0
ELATTAR	102	7	1	6.64e+01	4.36e-08	10001	19990	1.0e+01	9987	14	0	0
EXPFITB	102	5	0	5.02e-03	1.42e-14	13	14	1.0e+01	13	0	0	0
EXPFITC	502	5	0	2.33e-02	2.84e-14	13	15	1.0e+01	13	0	0	0
FEEDLOC	259	90	0	0.00e+00	2.37e-06	6	8	1.0e+01	2	4	0	0
GMNCASE1	300	175	0	2.67e-01	4.04e-16	1	3	1.0e+01	0	0	1	0
GMNCASE4	350	175	0	5.95e+03	1.95e-11	1	2	6.6e+01	0	1	0	0
HAIFAM	150	99	0	-4.50e+01	1.56e-06	10	23	1.0e+01	3	7	0	0
HIE1372D	525	637	0	2.78e+02	2.11e-11	3	4	1.0e+01	2	1	0	0
HYDROELM	504	505	2	-3.57e+06	2.73e-12	1223	1224	1.0e+01	1223	0	0	0
HYDROELS	168	169	0	-3.58e+06	2.88e-13	2002	2034	1.0e+01	2001	0	0	0
KTMODEL	450	726	-7	0.00e+00	1.93e+04	28	60	2.0e+04	0	28	0	0
LEAKNET	153	156	0	8.05e+00	8.75e-08	6	7	4.0e+01	4	2	0	0
LEUVEN7	946	360	0	6.95e+02	3.67e-14	2	3	1.0e+01	0	0	1	1
LHAIFAM	150	99	-6	-Inf	0.00e+00	3	4	1.0e+01	3	0	0	0
PRIMAL1	85	325	0	-3.50e-02	6.72e-14	1	2	1.0e+01	1	0	0	0
PRIMAL2	96	649	0	-3.37e-02	3.31e-13	1	2	1.0e+01	1	0	0	0
PRIMAL3	111	745	0	-1.36e-01	4.75e-13	1	2	1.0e+01	1	0	0	0
PRIMALC1	9	230	0	-6.16e+03	6.21e-11	2	3	1.0e+01	2	0	0	0
PRIMALC2	7	231	0	-3.55e+03	0.00e+00	1	2	1.0e+01	1	0	0	0
PRIMALC5	8	287	0	-4.27e+02	3.95e-12	1	2	1.0e+01	1	0	0	0
PRIMALC8	8	520	0	-1.83e+04	1.16e-10	8	9	1.0e+01	8	0	0	0
QPCBOEI1	351	384	0	1.15e+07	2.74e-10	12	17	9.5e+05	2	8	1	1
QPCBOEI2	166	143	0	8.17e+06	1.95e-12	12	13	2.4e+04	0	12	0	0

QPCSTAIR	356	467	0	6.20e+06	5.23e-12	9	28	6.5e+04	1	6	1	1
QPNBOEI1	351	384	0	6.74e+06	4.14e-11	31	33	2.5e+06	20	11	0	0
QPNBOEI2	166	143	0	1.37e+06	2.83e-12	23	45	1.1e+05	10	9	2	2
QPNSTAIR	356	467	0	5.15e+06	2.87e-12	19	20	2.3e+05	11	8	0	0
READING6	50	102	0	-1.45e+02	1.69e-11	397	466	1.0e+01	395	2	0	0
S365	5	7	0	0.00e+00	2.81e-10	18	50	1.0e+01	9	9	0	0
S365MOD	5	7	-1	2.50e-01	2.50e+00	16	293	2.0e+04	2	5	1	7
SAWPATH	774	583	-5	1.47e+03	3.56e+00	0	1	1.0e+01	0	0	0	0
SMBANK	64	117	0	-7.13e+06	8.53e-10	77	78	1.0e+01	77	0	0	0
SMMPSF	263	720	0	1.03e+06	2.92e-09	11	16	7.7e+01	2	9	0	0
SSEBLIN	72	194	1	1.99e+07	2.64e-10	10001	10002	6.2e+03	9466	535	0	0
SSEBNLN	96	194	0	1.62e+07	1.14e-13	5	5	3.3e+03	2	2	0	0
STATIC3	96	434	0	-1.53e+03	8.70e-16	1	2	1.0e+01	1	0	0	0
STEENBRA	108	432	0	1.70e+04	1.22e-12	3	4	1.0e+01	2	1	0	0
STEENBRB	108	468	0	1.11e+04	1.09e-12	875	1084	1.0e+03	873	2	0	0
STEENBRC	126	540	2	2.77e+04	7.66e-14	2616	2814	2.7e+03	2613	3	0	0
STEENBRE	126	540	2	2.86e+04	5.63e-13	2608	2781	2.3e+09	2604	2	1	1
STEENBRF	108	468	0	9.91e+03	6.52e-13	369	487	1.0e+03	367	2	0	0
STEENBRG	126	540	2	2.85e+04	1.80e-12	2646	2891	2.7e+03	2643	3	0	0
TABLE7	230	624	2	5.96e+04	7.32e-11	1212	68715	Inf	24	2	1	72
TARGUS	63	162	0	1.08e+03	5.61e-07	79	80	1.0e+01	78	1	0	0
TRIMLOSS	75	142	0	9.06e+00	7.44e-12	7	8	2.0e+01	3	4	0	0
TRO21X5	201	540	-5	5.00e+01	9.81e-01	8	28	2.1e+04	0	8	0	0
ZAMB2-10	96	270	0	-1.58e+00	3.64e-12	11	12	1.0e+01	9	2	0	0
ZAMB2-11	96	270	0	-1.12e+00	8.29e-10	5	6	1.0e+01	5	0	0	0
ZAMB2-8	48	138	0	-1.53e-01	4.73e-10	11	19	1.0e+01	9	0	1	1
ZAMB2-9	48	138	0	-3.55e-01	2.61e-05	8	9	1.0e+01	8	0	0	0

Table A.4: Results for PenSQO on the CUTEst problems of size $m \geq 1$ and $100 < \max\{m,n\} \leq 1000$.

prob	m	n	status	f	v	iters	feval	σ
ACOPP118	608	344	0	1.30e+05	4.09e-08	7	22	6.0e+02
ACOPP30	142	72	0	5.77e+02	1.10e-11	7	22	6.2e+01
ACOPP57	274	128	0	4.17e+04	2.39e-11	5	38	3.8e+03
ACOPR118	726	344	2	1.30e+05	5.83e-08	1189	39284	7.4e+07
ACOPR30	172	72	0	5.77e+02	1.50e-08	313	625	4.4e+01
ACOPR57	331	128	0	4.17e+04	1.96e-06	9	49	3.8e+03
AGG	488	163	2	-2.16e+06	9.34e+06	8569	8570	4.0e+01
C-RELOAD	284	342	0	-1.02e+00	2.02e-09	359	1405	1.0e+01
CRESC100	200	6	0	7.44e-01	2.53e-11	49	80	3.3e+05
DALLASL	667	906	0	-2.03e+05	1.39e-10	16	29	3.0e+01
DALLASM	151	196	0	-4.82e+04	1.64e-12	12	13	1.0e+01
DUAL3	1	111	0	1.36e-01	2.66e-15	1	2	1.0e+01
DUALC1	215	9	0	6.16e+03	1.67e-16	3	52	7.9e+03
DUALC2	229	7	0	3.55e+03	1.05e-15	2	41	4.8e+03
DUALC5	278	8	0	4.27e+02	6.11e-16	2	37	7.9e+02
DUALC8	503	8	0	1.83e+04	1.67e-16	4	33	2.7e+04
ELATTAR	102	7	1	5.82e+01	8.90e-08	10001	23277	1.0e+01
EXPFITB	102	5	0	5.02e-03	1.42e-14	13	14	1.0e+01
EXPFITC	502	5	0	2.33e-02	2.84e-14	13	15	1.0e+01
FEEDLOC	259	90	0	0.00e+00	2.37e-06	6	8	1.0e+01
GMNCASE1	300	175	0	2.67e-01	3.96e-14	1	2	1.0e+01
GMNCASE4	350	175	0	5.95e+03	1.86e-11	2	43	6.6e+01
HAIFAM	150	99	0	-4.50e+01	9.16e-06	21	76	1.0e+01
HIE1372D	525	637	0	2.78e+02	2.11e-11	3	4	1.0e+01
HYDROELM	504	505	2	-3.57e+06	1.99e-12	1224	1225	1.0e+01
HYDROELS	168	169	0	-3.58e+06	2.88e-13	2002	2034	1.0e+01
KTMODEL	450	726	-10	0.00e+00	3.41e+05	0	2	1.0e+01
LEAKNET	153	156	0	8.05e+00	1.37e-08	7	9	4.0e+01
LEUVEN7	946	360	0	6.95e+02	3.67e-14	2	3	1.0e+01
LHAIFAM	150	99	-10	6.93e-01	0.00e+00	0	3	1.0e+01
LHAIFAM	150	99	-10	6.93e-01	0.00e+00	0	3	1.0e+01
PRIMAL1	85	325	0	-3.50e-02	6.72e-14	1	2	1.0e+01
PRIMAL2	96	649	0	-3.37e-02	3.31e-13	1	2	1.0e+01
PRIMAL3	111	745	0	-1.36e-01	4.75e-13	1	2	1.0e+01
PRIMALC1	9	230	0	-6.16e+03	6.21e-11	2	3	1.0e+01
PRIMALC2	7	231	0	-3.55e+03	0.00e+00	1	2	1.0e+01
PRIMALC5	8	287	0	-4.27e+02	3.95e-12	1	2	1.0e+01
PRIMALC8	8	520	0	-1.83e+04	1.16e-10	8	9	1.0e+01
QPCBOEI1	351	384	0	1.15e+07	2.00e-11	19	74	1.0e+06

QPCBOEI2	166	143	0	8.17e+06	1.10e-12	21	128	5.8e+04
QPCSTAIR	356	467	0	6.20e+06	3.82e-12	15	106	5.8e+04
QPNBOEI1	351	384	0	6.78e+06	8.24e-10	25	92	2.5e+06
QPNBOEI2	166	143	0	1.38e+06	8.27e-13	23	84	1.1e+05
QPNSTAIR	356	467	0	5.15e+06	1.81e-12	26	119	5.3e+04
READING6	50	102	0	-1.45e+02	1.42e-12	2196	4065	1.0e+01
S365	5	7	-10	1.25e+00	8.26e+00	1	3	1.0e+01
S365MOD	5	7	-1	2.50e-01	2.50e+00	16	293	2.0e+04
SAWPATH	774	583	-5	1.47e+03	3.56e+00	0	1	1.0e+01
SMBANK	64	117	0	-7.13e+06	8.53e-10	77	78	1.0e+01
SMMPSF	263	720	0	1.03e+06	1.40e-06	10	45	1.5e+02
SSEBLIN	72	194	1	1.99e+07	2.64e-10	10001	10002	6.2e+03
SSEBNLN	96	194	0	1.62e+07	1.14e-13	5	5	3.3e+03
STATIC3	96	434	0	-1.53e+03	8.70e-16	1	2	1.0e+01
STEENBRA	108	432	0	1.70e+04	1.22e-12	3	4	1.0e+01
STEENBRB	108	468	0	9.08e+03	0.00e+00	401	489	1.0e+03
STEENBRC	126	540	2	2.84e+04	8.79e-13	2600	2758	1.4e+03
STEENBRE	126	540	2	2.85e+04	4.88e-13	2555	2714	1.7e+16
STEENBRF	108	468	0	8.99e+03	0.00e+00	401	490	1.0e+03
STEENBRG	126	540	2	2.82e+04	3.27e-12	2655	2813	1.4e+03
TABLE7	230	624	2	5.96e+04	3.76e-11	1218	65858	Inf
TARGUS	63	162	0	1.08e+03	5.61e-07	79	80	1.0e+01
TRIMLOSS	75	142	0	9.06e+00	3.67e-09	9	13	2.0e+01
TRO11X3	61	150	-5	1.43e+01	9.91e-01	26	159	5.4e+05
TRO21X5	201	540	-6	6.80e+00	9.97e-01	32	296	1.1e+10
ZAMB2-10	96	270	0	-1.58e+00	4.55e-05	20	62	1.0e+01
ZAMB2-11	96	270	0	-1.12e+00	8.29e-10	5	6	1.0e+01
ZAMB2-8	48	138	0	-1.53e-01	1.77e-09	170	1927	1.0e+01
ZAMB2-9	48	138	0	-3.55e-01	6.63e-06	10	13	1.0e+01

Bibliography

[1] A. Wächter, L. T. Biegler, Y.-D. Lang, and A. Raghunathan, "IPOPT: An interior point algorithm for large-scale nonlinear optimization," http://www.coin-or.org, 2002.

[2] N. I. Gould, Y. Loh, and D. P. Robinson, "A filter method with unified step computation for nonlinear optimization," *SIAM Journal on Optimization*, vol. 24, no. 1, pp. 175–209, 2014.

[3] ——, "A nonmonotone filter sqp method: Local convergence and numerical results," *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1885–1911, 2015.

[4] N. I. M. Gould and Ph. L. Toint, "Global convergence of a non-monotone trust-region SQP-filter algorithm for nonlinear programming," in *Multiscale Optimization Methods and Applications*, W. Hager and O. A. Prokopyev, Eds. "Dordrecht, The Netherlands: Kluwer Academic Publishers, 2005.

BIBLIOGRAPHY

[5] C. Shen, S. Leyffer, and R. Fletcher, "A nonmonotone filter method for nonlinear optimization," *Computational Optimization and Applications*, vol. 52, no. 3, pp. 583–607, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10589-011-9430-2

[6] K. Su and D. Pu, "A nonmonotone filter trust region method for nonlinear constrained optimization," *Journal of Computational and Applied Mathematics*, vol. 223, no. 1, pp. 230–239, 2009.

[7] N. I. M. Gould, , D. Orban, and Ph. L. Toint, "CUTEst : a constrained and unconstrained testing environment with safe threads for mathematical optimization," *Computational Optimization and Applications*, vol. 60, no. 3, pp. 545–557, 2015.

[8] H. Mohy-ud-Din and D. P. Robinson, "A solver for nonconvex bound-constrained quadratic optimization," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2385–2407, 2015. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/15M1022100

[9] Z. Dostál and J. Schöberl, "Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination," *Comput. Optim. Appl.*, vol. 30, no. 1, pp. 23–43, 2005. [Online]. Available: http://dx.doi.org/10.1007/s10589-005-4557-7

[10] A. Barclay, P. E. Gill, and J. B. Rosen, "SQP methods in optimal control,"

in *12th Conference on Variational Calculus, Optimal Control and Applications*, ser. International Series of Numerical Mathematics, R. Bulirsch, L. Bittner, W. Schmidt, and K. Heier, Eds. Basel: Birkhäuser, 1997.

[11] ——, "SQP methods in optimal control," in *Variational Calculus, Optimal Control and Applications*, ser. International Series of Numerical Mathematics, R. Bulirsch, L. Bittner, W. H. Schmidt, and K. Heier, Eds., vol. 124. Basel, Boston and Berlin: Birkhäuser, 1998, pp. 207–222.

[12] A. Bryson and Y. Ho, *Applied Optimal Control*. New York: Taylor and Francis, 1975.

[13] P. E. Gill, L. O. Jay, M. W. Leonard, L. R. Petzold, and V. Sharma, "An SQP method for the optimal control of large-scale dynamical systems," *J. Comput. Appl. Math.*, vol. 120, no. 1-2, pp. 197–213, 2000, SQP-based direct discretization methods for practical optimal control problems.

[14] D. Kraft, "On converting optimal control problems into nonlinear programming problems," in *Computational Mathematical Programming*, ser. NATO ASI Series F: Computer and Systems Sciences 15, K. Schittkowski, Ed. Berlin and Heidelberg: Springer Verlag, 1985, pp. 261–280.

[15] G. Alarcón, C. Torres, and L. Gómez, "Global optimization of gas allocation to a group of wells in artificial lift using nonlinear constrained

programming," *Journal of energy resources technology*, vol. 124, p. 262, 2002.

[16] T. Johansen, T. Fossen, and S. Berge, "Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming," *Control Systems Technology, IEEE Transactions on*, vol. 12, no. 1, pp. 211–216, 2004.

[17] S. Flåm and A. Antipin, "Equilibrium programming using proximal-like algorithms," *Math. Program.*, vol. 78, no. 1, pp. 29–41, 1996.

[18] F. Murphy, H. Sherali, and A. Soyster, "A mathematical programming approach for determining oligopolistic market equilibrium," *Math. Program.*, vol. 24, no. 1, pp. 92–106, 1982.

[19] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley and Sons, 2006.

[20] P. Liu and A. Der Kiureghian, "Optimization algorithms for structural reliability," *Structural safety*, vol. 9, no. 3, pp. 161–177, 1991.

[21] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[22] F. E. Curtis, O. Schenk, and A. Wächter, "An interior-point algorithm for large-scale nonlinear optimization with inexact step computations,"

SIAM Journal on Scientific Computing, vol. 32, no. 6, pp. 3447–3475, 2010.

[23] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program. A.*, vol. 106, no. 1, pp. 25–57, 2006.

[24] M. Ulbrich, S. Ulbrich, and L. Vicente, "A globally convergent primal-dual interior-point filter method for nonlinear programming," *Math. Program.*, vol. 100, no. 2, pp. 379–410, 2004.

[25] R. Waltz, J. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Math. Program.*, vol. 107, no. 3, pp. 391–408, 2006.

[26] R. H. Byrd, J. Nocedal, and R. A. Waltz, "KNITRO optimization software package," http://www.ziena.com/knitro/kindex.htm.

[27] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, `LANCELOT`*: a Fortran package for large-scale nonlinear optimization (Release A)*, ser. Lecture Notes in Computation Mathematics 17. Berlin, Heidelberg, New York, London, Paris and Tokyo: Springer Verlag, 1992.

[28] B. A. Murtagh and M. A. Saunders, "MINOS 5.5 User's Guide," Depart-

ment of Operations Research, Stanford University, Stanford, CA, Report SOL 83-20R, Revised 1998.

[29] E. G. Birgin, R. Castillo, and J. M. Martínez, "Numerical comparison of augmented Lagrangian algorithms for nonconvex problems," *Computational Optimization and Applications*, vol. 31, no. 1, pp. 31–55, 2005.

[30] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM J. Numer. Anal.*, vol. 28, pp. 545–572, 1991.

[31] ——, "Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization," *Math. Program. A.*, vol. 73, no. 1, pp. 73–110, 1996.

[32] F. E. Curtis, H. Jiang, and D. P. Robinson, "An adaptive augmented Lagrangian method for large-scale constrained optimization," *Math. Program.*, vol. 151, no. 1-2, pp. 201–245, 2015.

[33] M. Kočvara and M. Stingl, "PENNON: a generalized augmented Lagrangian method for semidefinite programming," in *High performance algorithms and software for nonlinear optimization (Erice, 2001)*, ser. Appl. Optim. Norwell, MA: Kluwer Acad. Publ., 2003, vol. 82, pp. 303–321. [Online]. Available: http://dx.doi.org/10.1007/978-1-4613-0241-4_14

[34] G. Di Pillo and L. Grippo, "Exact penalty functions in constrained optimization," *SIAM Journal on control and optimization*, vol. 27, no. 6, pp. 1333–1360, 1989.

[35] P. E. Gill and D. P. Robinson, "A primal-dual augmented Lagrangian," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 1–25, 2012.

[36] V. M. Zavala and M. Anitescu, "Scalable nonlinear programming via exact differentiable penalty functions and trust-region newton methods," *SIAM J. Optim.*, vol. 24, no. 1, pp. 528–558, 2014.

[37] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numer.*, vol. 4, pp. 1–51, 1995.

[38] F. E. Curtis, N. I. Gould, D. P. Robinson, and P. L. Toint, "An interior-point trust-funnel algorithm for nonlinear optimization," *accepted to Mathematical Programming*, 2016.

[39] F. E. Curtis, T. Johnson, D. P. Robinson, and A. Wächter, "An inexact sequential quadratic optimization algorithm for large-scale nonlinear optimization," *SIAM J. Optim.*, vol. 24, no. 3, pp. 1041–1074, 2014.

[40] R. Fletcher, "An ℓ_1 penalty method for nonlinear constraints," in *Numer-*

ical Optimization 1984, P. T. Boggs, R. H. Byrd, and R. B. Schnabel, Eds. Philadelphia: SIAM, 1985, pp. 26–40.

[41] R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," *Math. Program.*, vol. 91, no. 2, Ser. A, pp. 239–269, 2002. [Online]. Available: http://dx.doi.org/10.1007/s101070100244

[42] R. Fletcher, S. Leyffer, and Ph. L. Toint, "On the global convergence of a filter-SQP algorithm," *SIAM J. Optim.*, vol. 13, no. 1, pp. 44–59, 2002. [Online]. Available: http://dx.doi.org/10.1137/S105262340038081X

[43] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, pp. 99–131, 2005.

[44] N. I. M. Gould and D. P. Robinson, "A second derivative SQP method: Global convergence," *SIAM J. Optim.*, vol. 20, no. 4, pp. 2023–2048, 2010.

[45] ——, "A second derivative SQP method: Local convergence and practical issues," *SIAM J. Optim.*, vol. 20, no. 4, pp. 2049–2079, 2010.

[46] ——, "A second derivative SQP method with a "trust-region-free" predictor step," *IMA J. Numer. Anal.*, vol. 32, no. 2, pp. 580–601, 2012.

[47] J. Morales, J. Nocedal, and Y. Wu, "A sequential quadratic programming

algorithm with an additional equality constrained phase," *IMA Journal of Numerical Analysis*, vol. DOI: 10.1093/imanum/drq037, 2011.

[48] M. J. D. Powell and Y.-X. Yuan, "A recursive quadratic programming algorithm that uses differentiable exact penalty functions," *Math. Program.*, vol. 35, no. 3, pp. 265–278, 1986.

[49] E. O. Omojokun, "Trust region algorithms for nonlinear equality and inequality constraints," Ph.D. dissertation, Department of Computer Science, University of Colorado, Boulder, 1989.

[50] O. L. Mangasarian and S. P. Han, "Exact penalty functions in nonlinear programming," *Math. Program.*, vol. 17, no. 1, pp. 251–269, 1979.

[51] R. Fletcher, S. Leyffer, and Ph. L. Toint, "On the global convergence of an SLP-filter algorithm," Department of Mathematics, University of Namur, Belgium, Report 98/13, 1998.

[52] C. M. Chin and R. Fletcher, "On the global convergence of an SLP-filter algorithm that takes EQP steps," *Math. Program.*, vol. 96, no. 1, Ser. A, pp. 161–177, 2003. [Online]. Available: http://dx.doi.org/10.1007/s10107-003-0378-6

[53] C. M. Chin, A. H. A. Rashid, and K. M. Nor, "A combined filter line search

and trust region method for nonlinear programming," *WSEAS Trans. Math.*, vol. 5, no. 6, pp. 656–662, 2006.

[54] ——, "Global and local convergence of a filter line search method for nonlinear programming," *Optim. Methods Softw.*, vol. 22, no. 3, pp. 365–390, 2007. [Online]. Available: http://dx.doi.org/10.1080/10556780600565489

[55] R. Fletcher, N. I. M. Gould, S. Leyffer, P. L. Toint, and A. Wächter, "Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming," *SIAM J. Optim.*, vol. 13, no. 3, pp. 635–659, 2002. [Online]. Available: http://dx.doi.org/10.1137/S1052623499357258

[56] A. Wachter and L. T. Biegler, "Line search filter methods for nonlinear programming: Local convergence," *SIAM J. Optim.*, vol. 16, no. 1, pp. 32–48, 2005.

[57] ——, "Line search filter methods for nonlinear programming: motivation and global convergence," *SIAM J. Optim.*, vol. 16, no. 1, pp. 1–31, 2006.

[58] F. E. Curtis and J. Nocedal, "Flexible penalty functions for nonlinear constrained optimization," *IMA J. Numer. Anal.*, vol. 28, no. 4, pp. 749–769, 2008. [Online]. Available: http://dx.doi.org/10.1093/imanum/drn003

[59] L. Chen and D. Goldfarb, "An interior-point piecewise linear penalty

method for nonlinear programming," *Math. Program.*, vol. 128, no. 1-2, pp. 73–122, 2011.

[60] N. Maratos, "Exact penalty function algorithms for finite-dimensional and control optimization problems," Ph.D. dissertation, Department of Computing and Control, University of London, 1978.

[61] M. J. Powell, "Convergence properties of algorithms for nonlinear optimization," *Siam Review*, vol. 28, no. 4, pp. 487–500, 1986.

[62] S. Ulbrich, "On the superlinear local convergence of a filter-SQP method," *Math. Program.*, vol. 100, no. 1, pp. 217–245, 2004.

[63] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Steering exact penalty methods for nonlinear programming," *Optim. Methods Softw.*, vol. 23, no. 2, pp. 197–213, 2008. [Online]. Available: http://dx.doi.org/10.1080/10556780701394169

[64] J. V. Burke, F. E. Curtis, and H. Wang, "A Sequential Quadratic Optimization Algorithm with Fast Infeasibility Detection," COR@L Laboratory, Department of ISE, Lehigh University, Tech. Rep. 12T-012, 2012. [Online]. Available: http://coral.ie.lehigh.edu/~frankecurtis/wp-content/papers/BurkCurtWang12.pdf

[65] F. E. Curtis, T. Johnson, D. P. Robinson, and A. Wachter, "An inexact

sequential quadratic optimization algorithm for large-scale optimization," Johns Hopkins University, Departmetn of Applied Mathematic and Statisics, Baltimore, MD, Technical Report TR-1-13, 2013.

[66] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[67] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, pp. 503–528, 1989.

[68] P. E. Gill and W. Murray, "Newton-type methods for unconstrained and linearly constrained optimization," *Math. Program.*, vol. 7, pp. 311–350, 1974.

[69] R. B. Schnabel and E. Eskow, "A new modified Cholesky factorization," *SIAM J. Sci. and Statist. Comput.*, vol. 11, pp. 1136–1158, 1990.

[70] R. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.

[71] R. H. Byrd, G. Lopez-Calva, and J. Nocedal, "A line search exact penalty method using steering rules," *Math. Program.*, vol. 133, no. 1-2, pp. 39–73, 2012.

[72] S. M. Robinson, "Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms," *Math. Program.*, vol. 7, no. 1, pp. 1–16, 1974.

[73] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2000.

[74] O. L. Mangasarian and S. Fromovitz, "The Fritz John necessary optimality conditions in the presence of equality and inequality constraints," *J. Math. Anal. Appl.*, vol. 17, pp. 37–47, 1967.

[75] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz, "On the convergence of successive linear-quadratic programming algorithms," *SIAM J. Optim.*, vol. 16, no. 2, pp. 471–489, 2006.

[76] R. Chamberlain, M. Powell, C. Lemarechal, and H. Pedersen, "The watchdog technique for forcing convergence in algorithms for constrained optimization," in *Algorithms for Constrained Minimization of Smooth Nonlinear Functions*. Springer, 1982, pp. 1–17.

[77] L. Grippo, F. Lampariello, and S. Lucidi, "A class of nonmonotone stabilization methods in unconstrained optimization," *Numerische Mathematik*, vol. 59, no. 1, pp. 779–805, 1991.

[78] IBM, "ILOG CPLEX: High-performance software for mathematical programming and optimization," 2006. [Online]. Available: http://www.ilog.com/products/cplex

BIBLIOGRAPHY

[79] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, Ser. A, pp. 201–213, 2002.

[80] P. E. Gill, V. Kungurtsev, and D. P. Robinson, "A stabilized SQP method: Global convergence," University of California, San Diego, Center for Computational Mathematics Report CCoM 13-04, 2013.

[81] ——, "A stabilized SQP method: Superlinear convergence," University of California, San Diego, Center for Computational Mathematics Report CCoM 14-01, 2014.

[82] P. E. Gill and D. P. Robinson, "A globally convergent stabilized SQP method," *SIAM J. Optim.*, vol. 23, no. 4, pp. 1983–2010, 2013.

[83] R. H. Byrd, F. E. Curtis, and J. Nocedal, "Infeasibility detection and SQP methods for nonlinear optimization," *SIAM J. Optim.*, vol. 20, no. 5, pp. 2281–2299, 2010. [Online]. Available: http://link.aip.org/link/?SJE/20/2281/1

[84] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

[85] M. J. Saltzman, "Coin-or: an open-source library for optimization," in

Programming languages and systems in computational economics and finance. Springer, 2002, pp. 3–32.

[86] C. Guéret, C. Prins, M. Sevaux, and S. Heipcke, *Applications of optimization with Xpress-MP*. Dash Optimization Blisworth, UK, 2002.

[87] E. Andersen, B. Jensen, J. Jensen, R. Sandvik, and U. Worsøe, "Mosek version 6," Technical Report TR–2009–3, MOSEK, Tech. Rep., 2009.

[88] E. Ellison, M. Hajian, R. Levkovitz, I. Maros, G. Mitra, and D. Sayers, "Fortmp manual," *OptiRisk Systems and Brunel University*, 1999.

[89] J. Gondzio, "Matrix-free interior point method," *Computational Optimization and Applications*, vol. 51, no. 2, pp. 457–480, 2012.

[90] J. J. Moré and G. Toraldo, "On the solution of large quadratic programming problems with bound constraints," *SIAM Journal on Optimization*, vol. 1, no. 1, pp. 93–113, 1991.

[91] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[92] N. J. Higham, "Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation," *ACM Trans-*

actions on Mathematical Software (TOMS), vol. 14, no. 4, pp. 381–396, 1988.

[93] ——, "Experience with a matrix norm estimator," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 4, pp. 804–809, 1990.

[94] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[95] E. D. Dolan, J. J. Moré, and T. S. Munson, "Benchmarking optimization software with cops 3.0," *Argonne National Laboratory Research Report*, 2004.

[96] T. F. Coleman and A. Liao, "An efficient trust region method for unconstrained discrete-time optimal control problems," *Computational Optimization and Applications*, vol. 4, no. 1, pp. 47–66, 1995.

[97] J. C. Gilbert, "On the realization of the wolfe conditions in reduced quasi-newton methods for equality constrained optimization," *SIAM Journal on Optimization*, vol. 7, no. 3, pp. 780–813, 1997.

[98] L. Lukšan and J. Vlcek, "Sparse and partially separable test problems for unconstrained and equality constrained optimization," 1999.

[99] Z. Dostl, *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[100] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[101] Y. Saad, "A flexible inner-outer preconditioned gmres algorithm," *SIAM Journal on Scientific Computing*, vol. 14, no. 2, pp. 461–469, 1993.

[102] M. T. Jones and P. E. Plassmann, "An improved incomplete cholesky factorization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 21, no. 1, pp. 5–17, 1995.

[103] L. Eldén and V. Simoncini, "Solving ill-posed linear systems with gmres and a singular preconditioner," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1369–1394, 2012.

[104] H. A. Van der Vorst and K. Dekker, "Conjugate gradient type methods and preconditioning," *Journal of Computational and Applied Mathematics*, vol. 24, no. 1-2, pp. 73–87, 1988.

[105] L. Bergamaschi, J. Gondzio, and G. Zilli, "Preconditioning indefinite sys-

tems in interior point methods for optimization," *Computational Optimization and Applications*, vol. 28, no. 2, pp. 149–171, 2004.

[106] Y. Saad, "Preconditioning techniques for nonsymmetric and indefinite linear systems," *Journal of Computational and Applied Mathematics*, vol. 24, no. 1-2, pp. 89–105, 1988.

[107] C. Durazzi and V. Ruggiero, "Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems," *Numerical linear algebra with applications*, vol. 10, no. 8, pp. 673–688, 2003.

[108] S. Gratton and J. Tshimanga, "An observation-space formulation of variational assimilation using a restricted preconditioned conjugate gradient algorithm," *Quarterly Journal of the Royal Meteorological Society*, vol. 135, no. 643, pp. 1573–1585, 2009.

[109] S. Gratton, S. Gürol, and P. L. Toint, "Preconditioning and globalizing conjugate gradients in dual space for quadratically penalized nonlinear-least squares problems," *Computational Optimization and Applications*, vol. 54, no. 1, pp. 1–25, 2013.

[110] S. Gürol, A. Weaver, A. Moore, A. Piacentini, H. Arango, and S. Gratton, "B-preconditioned minimization algorithms for variational data assimi-

lation with the dual formulation," *Quarterly Journal of the Royal Meteorological Society*, vol. 140, no. 679, pp. 539–556, 2014.

Vita

Yueling Loh was born in 1985 in Singapore. She attended the National Cathedral School in Washington, DC from 2000 to 2003. She then joined the Jerome Fisher Program in Management and Technology at the University of Pennsylvania, where she graduated *magna cum laude* in 2007 with a Bachelor of Science in Economics and a Bachelor of Applied Science. From 2009 to 2010, she attended the Nanyang Technological University in Singapore and received a Master of Science in Systems and Project Management.

She enrolled in the Ph.D. program at the Department of Applied Mathematics and Statistics at the Johns Hopkins University in the Fall of 2010.