SIAM J. Optim.
Vol. 25, No. 3, pp. 1885–1911

A NONMONOTONE FILTER SQP METHOD: LOCAL CONVERGENCE AND NUMERICAL RESULTS*

NICHOLAS I. M. GOULD[†], YUELING LOH[‡], AND DANIEL P. ROBINSON[‡]

Abstract. The work by Gould, Loh, and Robinson [*SIAM J. Optim.*, 24 (2014), pp. 175–209] established global convergence of a new filter line search method for finding local first-order solutions to nonlinear and nonconvex constrained optimization problems. A key contribution of that work was that the search direction was computed using the same procedure during every iteration from subproblems that were always feasible and computationally tractable. This contrasts previous filter methods that require a separate restoration phase based on subproblems solely designed to reduce infeasibility. In this paper, we present a nonmonotone variant of our previous algorithm that inherits the previously established global convergence property. In addition, we establish local superlinear convergence of the iterates and provide the results of numerical experiments. The numerical tests validate our method and highlight an interesting numerical trade-off between accepting more (on average lower quality) steps versus fewer (on average higher quality) steps.

Key words. filter, restoration, penalty function, sequential quadratic programming, nonlinear programming, nonconvex

AMS subject classifications. 49M05, 49M15, 65K05, 65K10, 65K15

DOI. 10.1137/140996677

1. Introduction.

In this paper, we present a new algorithm for finding local solutions of nonconvex optimization problems of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x) \quad \text{subject to} \quad c(x)\ge 0, \tag{1.1}$$

where 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. Many algorithms have been developed for this task and include interior-point [13, 45, 46, 42, 47], augmented Lagrangian [1, 6, 7, 10, 34], primal-dual penalty [15, 23, 48], and sequential quadratic optimization (SQO) methods [2, 9, 11, 17, 18, 19, 22, 28, 29, 30, 38, 39] (commonly called sequential quadratic programming (SQP) methods). Each class of methods has advantages and disadvantages. Interior-point methods may be used to solve very-large-scale sparse problems since the predominate cost per iteration is a single symmetric indefinite matrix factorization. However, interior-point methods are currently ineffective at utilizing a good initial estimate of a solution, i.e., at being warm-started. Augmented Lagrangian methods may be warm-started and implemented matrix-free and thus may be applied to extreme-scale problems. Unfortunately, too frequently they are 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. In summary, each class of algorithms serves a distinct and vital role in solving real-life optimization problems.

*Received by the editors November 21, 2014; accepted for publication (in revised form) July 27, 2015; published electronically September 22, 2015. This research was supported by the National Science Foundation grant DMS-1217153 and EPSRC grant EP/I013067/1.

http://www.siam.org/journals/siopt/25-3/99667.html

[†]Scientific Computing Department, Rutherford Appleton Laboratory, Chilton, OX11-0QX, UK (nick.gould@stfc.ac.uk).

[‡]Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD 21218 (yueling.loh@jhu.edu, daniel.p.robinson@gmail.com).

Each class of methods discussed in the previous paragraph contains a variety of algorithms that may be distinguished by their details. For example, some of these variations include whether a line search or trust region is used or whether globalization is attained via the use of a merit function, a filter, or a step classification scheme. It is no surprise that each of these variants has advantages and disadvantages and serves a vital role in optimization. In this paper, we address 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 paper may be used (to various degrees) by future filter-based algorithms.

The new SQO algorithm is a nonmonotone variant of the method that we presented in [26]. (Nonmonotone strategies are commonly used by SQO-like algorithms to avoid the Maratos effect [37] and ultimately for proving superlinear convergence of the iterates.) The motivation for that work was to overcome the dissatisfying fact that previous filter methods require a special restoration phase to handle various scenarios that would otherwise lead to failure. For example, some methods formulate subproblems that may be infeasible [18, 19, 46], while others may fail as a result of an ineffective line search [46]. In all of these cases, the restoration phase temporarily ignores the object function and iterates toward the feasible region until the issue that triggered the restoration phase is resolved. Since the objective function is ignored during these iterations, a significant decrease in performance is often observed on problems for which restoration plays a notable role. In contrast, the filter line search method developed in [26] computes a search direction from the same procedure during *every* iteration, uses a practical penalty phase in lieu of a traditional restoration phase, incorporates an improved definition for the filter margin, and is globally convergent.

Here, we present a nonmonotone variant (section 2) of our previously introduced algorithm [26] that inherits the global convergence property (section 3) previously established. We also show that no additional mechanism, e.g., a shadow/nonmonotone filter [31, 40, 41], is needed to establish local superlinear convergence (section 4) of the iterates. Finally, we provide numerical results (section 5) on problems from the CUTEst [25] test set.

Before proceeding, we briefly describe aspects of some closely related algorithms. The nonmonotone (sometimes called watchdog) approach that we use is a common way for exact penalty and filter methods to address the Maratos effect (e.g., see [31, 41, 40]) and thereby establish superlinear convergence under standard assumptions. Interestingly, the second of two consecutive steps in a nonmonotone approach is an example of a second-order correction step. Second-order correction steps, such as those used by Wächter and Biegler [44], are computed from systems of linear equations defined from quantities that the user is free to choose, subject to satisfying certain conditions. Consequently, the sum of two consecutive trial steps in our nonmontone approach is equivalent to the sum of their trial and second-order correction step for one particular choice of the correction step. In terms of the definition of the filter, we use the same one, although in our case acceptability is based on weaker conditions. In [40], Shen, Leyffer, and Fletcher used a nonmonotone filter to establish local convergence, while a standard filter is used to prove global convergence. In particle, they show that a point acceptable to the local filter is obtained after a constant number of nonmonotone steps. Another approach for establishing local convergence and avoiding the Maratos effect is to use the Lagrangian as an element of the filter in place of the objective function [43]. We note that all of these related methods require a traditional restoration phase.

1.1. Notation and preliminaries. 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 componentwise. 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 (1.1) is $L(x, y) := f(x) - c(x)^T y$, where y is an m-vector of 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 (1.1). For convenience, 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 vector $v \in \mathbb{R}^n$, 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. We say that x is a KKT point for problem (1.1) with associated Lagrange multiplier vector y if and only if

$$F_{\text{KKT}}(x, y) := \begin{pmatrix} g(x) - J(x)^T y \\ \min\left\{c(x), y\right\} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{1.2}$$

and we say that (x, y) is a KKT pair. The primary goal of most optimization algorithms is to find a KKT pair. Our trial step computation uses the ℓ_1-penalty function

$$\phi(x; \sigma) := f(x) + \sigma v(x) \quad \text{with } v(x) := \left\|[c(x)]^-\right\|_1 \text{ and } \sigma > 0, \tag{1.3}$$

where σ is called the penalty parameter.

2. A nonmonotone filter SQO method.

In this section, we present our nonmonotone line search filter SQO method. We begin with an overview of the algorithm in section 2.1, give details of the search direction in section 2.2, discuss step acceptance in section 2.3, and formally state the complete algorithm in section 2.4.

2.1. Algorithm overview. Our method is a line search algorithm. During the kth iteration, a search direction s_k is computed and used within a backtracking line search procedure. Although the direction s_k is sufficient to guarantee global convergence, we also allow for the computation of an additional search direction s_k^a that promotes faster convergence; in fact, we establish (under common assumptions) the local superlinear convergence of the iterates. Our algorithm is not typical since we perform a backtracking line search along both s_k and s_k^a (in parallel) with preference given to s_k^a.

Every line search algorithm terminates when some specified set of conditions is satisfied. In our case, the precise conditions used depend on the current mode of our algorithm; we utilize a filter mode and a penalty mode. The filter mode is similar to that used by previous filter methods, although some new enhancements will be introduced. The penalty mode is based on the penalty function (1.3) and used in lieu of a traditional restoration phase, which we find appealing from a practical perspective. Importantly, the calculations performed to obtain the search directions s_k and s_k^a are exactly the same for both modes.

In filter mode, there are three sets of conditions that can trigger the line search to terminate. Which of these sets of conditions are satisfied determines whether we call that iterate an o-, a v-, or a b-iterate; the precise conditions will be given in section 2.3. The conditions that define o- and v-iterates are similar in spirit to previous filter methods, but with some enhancements. By contrast, the conditions that define a b-iterate indicate progress toward solving (1.1) and, in addition, that a penalty mode should be entered. Roughly, the properties of a b-iterate are that (i) the conditions that define a v- and an o-iterate are not satisfied, (ii) the constraint violation is decreased, and (iii) the penalty function is decreased. This is a situation that commonly occurs when the entries that define the filter (to be discussed in section 2.3) are blocking additional progress and would typically trigger a restoration phase. Therefore, one may view our penalty mode as a replacement for a traditional restoration phase that is triggered by intuitive conditions.

In penalty mode, a single set of conditions is used to terminate the line search. The iterates computed to satisfy those conditions are called p-iterates since they obtain sufficient decrease in the penalty function. We choose to return to filter mode as soon as any iterate is computed that is acceptable to the filter. (See section 2.3 for additional details.) Again, the calculations of each trial step in the penalty mode are precisely the same as those performed in the filter mode.

The search direction s_k is computed from subproblems that are always feasible and promote convergence to the feasible region in an efficient manner. When x_k is infeasible, the direction s_k is constructed to be a descent direction for v at x_k. Moreover, the penalty parameter is adjusted to make s_k a descent direction for ϕ at x_k. These key properties associated with s_k ensure that the line search will terminate finitely with a point that decreases both the constraint violation and penalty function, i.e., with a b-iterate, when x_k is infeasible. If x_k is feasible, finite termination of the line search will be guaranteed by the conditions that define an o-iterate. Importantly, unlike some algorithms [46], we do not use a heuristic for recognizing failure of the line search as a trigger to enter restoration (or penalty) mode.

The second search direction s_k^a has been previously used [28, 29, 30, 38]. In short, it accelerates convergence by using an active set estimation to form an equality constrained QP subproblem defined with exact second derivatives. Other acceleration steps are possible [28] but will not be considered here. We also suspect that a superlinear rate of convergence could be established based on the steps s_k (i.e., without computing and using accelerator steps s_k^a) if the matrices B_k were chosen to satisfy the Dennis–More [14] condition. However, since it is difficult to construct such a sequence (especially in the large-scale case), we prefer to use the accelerator step s_k^a to achieve the same fast convergence. Also, in our experience, the step s_k^a often improves global performance, even though no such result can be proved.

Finally, to ensure superlinear convergence, we must incorporate either a second-order correction step [8, section 10.4.2] or a nonmonotone strategy [8, section 10.1.1]. Here, we choose to use a nonmonotone strategy in which we temporarily accept the acceleration step s_k^a even when it does not satisfy the conditions normally required to terminate the line search. In fact, we allow this to continue for a prespecified number of iterations in what is called a nonmonotone phase. If, unfortunately, appropriate conditions are not satisfied within the prespecified 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. Thus, it is more accurately categorized as a watchdog strategy [5, 32].

2.2. Search direction computation. In this section, we provide a compact description of the calculations for computing the search directions s_k and s_k^a, which are the same as in [26, section 2].

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

$$\underset{s\in\mathbb{R}^n}{\text{minimize}}\ \ \ell^v(s;x_k) := \left\|[c_k + J_k s]^-\right\|_1 \quad \text{subject to} \quad \|s\|_\infty \le \delta_k \tag{2.1}$$

or the equivalent linear program

$$\underset{(s,r)\in\mathbb{R}^{n+m}}{\text{minimize}}\ \ e^T r \quad \text{subject to} \ \ c_k + J_k s + r \ge 0, \ \ r \ge 0, \ \ \|s\|_\infty \le \delta_k, \tag{2.2}$$

where $c_k := c(x_k)$, $J_k := J(x_k)$, $\delta_k \in [\delta_{\min}, \delta_{\max}]$, and $0 < \delta_{\min} \le \delta_{\max} < \infty$. Once the steering step s_k^s has been computed, we calculate the change in the linearized constraint violation given by

$$\Delta\ell^v(s_k^s;x_k) := \ell^v(0;x_k) - \ell^v(s_k^s;x_k) = \left\|[c_k]^-\right\|_1 - \left\|[c_k + J_k s_k^s]^-\right\|_1, \tag{2.3}$$

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. Specifically, if

$$v_k := v(x_k) > 0 \ \text{ and } \ \Delta\ell^v(s_k^s;x_k) = 0, \tag{2.4}$$

then x_k is an infeasible stationary point for v.

The computation of the predictor step s_k^p involves the quadratic model of the objective function

$$q^f(s;x,M) := f(x) + g(x)^T s + \tfrac{1}{2}s^T M s$$

for any symmetric matrix M, and the piecewise quadratic model of ϕ given by

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

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

$$s_k^p = \begin{cases} \underset{s\in\mathbb{R}^n}{\operatorname{argmin}}\ q^f(s;x_k,B_k) \ \text{ subject to } \ c_k + J_k s \ge 0, & \text{if } \Delta\ell^v(s_k^s;x_k) = v_k, \quad \text{(2.5a)} \\ \underset{s\in\mathbb{R}^n}{\operatorname{argmin}}\ q^\phi(s;x_k,B_k,\sigma_k) & \text{otherwise,} \quad \text{(2.5b)} \end{cases}$$

where $\sigma_k > 0$ is the kth penalty parameter value and B_k a positive-definite approximation of $\nabla^2_{xx}L(x_k,y_k)$.

The search direction s_k is then defined as

$$s_k := (1-\tau_k)s_k^s + \tau_k s_k^p, \tag{2.6}$$

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

$$\Delta\ell^v(s_k;x_k) \ge \eta_v \Delta\ell^v(s_k^s;x_k) \ge 0 \ \text{ for some } \eta_v \in (0,1). \tag{2.7}$$

This makes s_k a descent direction for v when $\Delta\ell^v(s_k^s;x_k) > 0$ (see [26, Lemma 2.6]).

Next, the penalty parameter σ_{k+1} is updated so that s_k is also a decent direction for the penalty function ϕ (see [26, Lemma 2.10]) at x_k. Specifically, 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 set

$$\sigma_{k+1} \leftarrow \begin{cases} \sigma_k & \text{if } \Delta\ell_k^\phi \ge \sigma_k \eta_\sigma \Delta\ell_k^v, \\ \max\left\{\sigma_k + \sigma_{\text{inc}},\, \frac{-\Delta\ell^f(s_k;x_k)}{\Delta\ell^v(s_k;x_k) - \eta_\sigma \Delta\ell^v(s_k^s;x_k)}\right\} & \text{otherwise} \end{cases} \tag{2.8}$$

for some $\sigma_{\text{inc}} > 0$ and $0 < \eta_\sigma < \eta_v < 1$ with η_v defined in (2.7), and where

$$\Delta\ell^f(s_k; x_k) := -g_k^T s_k \quad \text{and} \quad \Delta\ell^\phi(s_k; x_k, \sigma_k) := \Delta\ell^f(s_k; x_k) + \sigma_k \Delta\ell^v(s_k; x_k)$$

with $g_k := g(x_k)$.

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 (2.5), and $H_k := \nabla_{xx}^2 L(x_k, y_k^p)$, we compute the Cauchy-$f$ step

$$s_k^{cf} := \alpha_k^f s_k \quad \text{with} \quad \alpha_k^f := \underset{0\le\alpha\le1}{\operatorname{argmin}}\; q^f(\alpha s_k; x_k, H_k) \tag{2.9}$$

and the Cauchy-ϕ step

$$s_k^{c\phi} := \alpha_k^\phi s_k \quad \text{with} \quad \alpha_k^\phi := \underset{0\le\alpha\le1}{\operatorname{argmin}}\; q^\phi(\alpha s_k; x_k, H_k, \sigma_{k+1}). \tag{2.10}$$

These Cauchy steps 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. To be precise, we define a predicted change in the objective function given by

$$\Delta q^f(s_k^{cf}; x_k, H_k) := q^f(0; x_k, H_k) - q^f(s_k^{cf}; x_k, H_k) = -g_k^T s_k^{cf} - \tfrac{1}{2} s_k^{cf^T} H_k s_k^{cf} \tag{2.11}$$

and a predicted change in the merit function given by

$$\begin{aligned} &\Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1}) \\ &\qquad := q^\phi(0; x_k, H_k, \sigma_{k+1}) - q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1}) \\ &\qquad = -g_k^T s_k^{c\phi} - \tfrac{1}{2} s_k^{c\phi^T} H_k s_k^{c\phi} + \sigma_{k+1}\big(\big\|[c_k]^-\big\|_1 - \big\|[c_k + J_k s_k^{c\phi}]^-\big\|_1\big). \end{aligned} \tag{2.12}$$

Finally, to accelerate convergence, we compute an accelerator step as

$$s_k^a := s_k^p + s_k^{a'}, \tag{2.13}$$

where $s_k^{a'}$ is computed as

$$s_k^{a'} := \underset{s\in\mathbb{R}^n}{\operatorname{argmin}}\; q^f(s_k^p + s; x_k, H_k) \;\text{ subject to }\; [J_k s]_{\mathcal{A}_k} = 0, \;\; \|s\|_2 \le \delta_a, \tag{2.14}$$

the set

$$\mathcal{A}_k := \mathcal{A}(s_k^p) := \{i : [c_k + J_k s_k^p]_i = 0\} \tag{2.15}$$

gives a prediction of those constraints active at a solution to (1.1), and $\delta_a > 0$ is a trust-region radius.

2.3. 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, but were already outlined in section 2.1.

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, which 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 2.1 (acceptable to $\mathcal{F}_k$). *The point x is acceptable to $\mathcal{F}_k$ if its associated ordered pair $(v(x), f(x))$ satisfies*

$$\begin{aligned} v(x) &\leq \max\left\{v_i - \alpha_i \eta_v \Delta\ell^v(s_i^s; x_i),\ \beta v_i\right\} \quad \textit{or} \\ f(x) &\leq f_i - \gamma \min\left\{v_i - \alpha_i \eta_v \Delta\ell^v(s_i^s; x_i),\ \beta v_i\right\} \end{aligned} \tag{2.16}$$

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 (2.16) 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 2.2 (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* 2.1 *and* (2.16) *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; an overview of these two modes was discussed in section 2.1. At this point, the reader only needs to know that step acceptance in filter mode (section 2.3.1) is driven by acceptability to the filter, whereas step acceptance in penalty mode (section 2.3.2) is driven by reducing the penalty function.

In the following two sections, the names used to denote various pairs of the form (α, s) for some step length α and search direction s (e.g., see Definition 2.3) are the same as in [26]. This decision emphasizes that the definitions in this paper are generalizations of the former that account for nonmonotonicity.

2.3.1. 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 2.3 (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*

$$\Delta \ell^f(s_{R(k)}; x_{R(k)}) < \gamma_v \Delta \ell^v(s_{R(k)}; x_{R(k)}) \quad \textit{for some } \gamma_v \in (0,1). \tag{2.17}$$

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 deemed an o-pair based on the following.

DEFINITION 2.4 (o-pair). *The pair (α, s) constitutes an o-pair if $x_k + \alpha s$ is acceptable to $\mathcal{F}_{R(k)}$,*

$$\Delta \ell^f(s_{R(k)}; x_{R(k)}) \geq \gamma_v \Delta \ell^v(s_{R(k)}; x_{R(k)}), \quad \textit{and} \tag{2.18a}$$

$$f(x_k + \alpha s) \leq f(x_{R(k)}) - \gamma_f \alpha \rho^f_{R(k)}, \tag{2.18b}$$

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

$$\rho^f_{R(k)} := \min\left\{\Delta \ell^f(s_{R(k)}; x_{R(k)}),\ \Delta q^f(s^{cf}_{R(k)}; x_{R(k)}, H_{R(k)})\right\}. \tag{2.19}$$

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 the pair (α_k, s_k) is a b-pair.

DEFINITION 2.5 (b-pair). *The pair (α, s) constitutes a b-pair if*

$$v(x_k + \alpha s) < v(x_{R(k)}) \tag{2.20}$$

and

$$\phi(x_k + \alpha s; \sigma_{k+1}) \leq \phi(x_{R(k)}; \sigma_{R(k)+1}) - \gamma_\phi \alpha \rho^\phi_{R(k)} \quad \textit{for some } \gamma_\phi \in (0,1), \tag{2.21}$$

where

$$\rho^\phi_{R(k)} := \min\left\{ \Delta \ell^\phi(s_{R(k)}; x_{R(k)}, \sigma_{R(k)+1}),\ \Delta q^\phi(s^{c\phi}_{R(k)}; x_{R(k)}, H_{R(k)}, \sigma_{R(k)+1}) \right\}. \tag{2.22}$$

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.

2.3.2. 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 2.6 (p-pair). *The pair* (α, s) *is a p-pair if* (2.21) *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 $x_{R(k)}$.

2.4. The formal statement of the algorithm. Our method is stated as Algorithm 1. 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 line 4 tests false so that the search directions s_k and s_k^a are computed in lines 7–15 as described in section 2.2. We now consider two possible scenarios. First, suppose that $\mathcal{P}$-*mode* has the value false in line 17, 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 17. 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 5) and then perform a backtracking line search. To give more details, first suppose that $\mathcal{P}$-*mode* has the value false in line 17. Then, the backtracking loop starts in line 29 and proceeds until either a valid v-, o-, or b-pair is found. Note that in line 31, 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 $\mathcal{P}$-*mode* has the value true in line 17. Then, the backtracking loop starts in line 18 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 25 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 $\mathcal{P}$-*mode* to false.

Finally, at the end of every iteration and regardless of the current mode, we choose to increase the penalty parameter if

$$\Delta q^{\phi}(s_k; x_k, B_k, \sigma_{k+1}) < \eta_\phi \Delta q^{\phi}(s_k^p; x_k, B_k, \sigma_{k+1}) \quad \text{for some } \eta_\phi \in (0,1). \tag{2.23}$$

The satisfaction of (2.23) indicates that the contribution of s_k^p to the definition of s_k in (2.6) 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 (2.6) and (2.7)) 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 46) to promote linearized feasibility of the predictor step during the next iteration.

3. Global convergence.

In this section, we establish the same global convergence result as in [26] under the same assumptions, which we restate here.

Assumption 3.1. The iterates $\{x_k\}$ lie in an open, bounded, and convex set $\mathcal{X}$.

Assumption 3.2. The problem functions $f(x)$ and $c(x)$ are twice continuously differentiable on $\mathcal{X}$.

Assumption 3.3. The matrices B_k are uniformly positive definite and bounded, i.e., there exist values $0 < \lambda_{\min} < \lambda_{\max} < \infty$ such that $\lambda_{\min} \|s\|_2^2 \le s^T B_k s \le \lambda_{\max} \|s\|_2^2$ for all $s \in \mathbb{R}^n$ and all B_k.

Assumption 3.4. The matrices H_k are uniformly bounded, i.e., $\|H_k\|_2 \le \mu_{\max}$ for some $\mu_{\max} \ge 1$.

Using these assumptions, we may state our global convergence result, which is identical to [26, Theorem 4.1] and uses the Mangasarian–Fromovitz constraint qualification (MFCQ) [36]. Since the proof is essentially the same, here we only describe the differences that result from the nonmonotonicity of Algorithm 1.

THEOREM 3.1. *If Assumptions* 3.1–3.4 *hold, then one of the following must occur.*

(i) *Algorithm* 1 *terminates finitely with either a first-order KKT point or an infeasible stationary point in lines* 12 *or* 9*, respectively, for problem* (1.1).

(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* (1.1).

(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 proof for the monotone variant [26, Algorithm 1] hinges on guaranteeing sufficient progress during every iteration as measured by conditions placed on the (α, s) pairs. In this paper, we have formulated conditions on (α, s) pairs (see section 2.3) that generalize the conditions used in [26]. The key difference is that the conditions in this paper 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 1, our method reduces to the monotone variant analyzed in [26].

We establish global convergence of Algorithm 1 by walking the reader through the analysis of [26] 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 (2.4)) or a KKT point in a finite number of iterations (see lines 9 and 12 of Algorithm 1).

Algorithm 1. A nonmonotone filter SQO algorithm.

```
1:  Input an initial primal-dual pair $(x_0, y_0)$.
2:  Choose parameters $\{\eta_v, \eta_\sigma, \eta_\phi, \sigma_{\text{inc}}, \beta, \gamma, \gamma_v, \gamma_f, \gamma_\phi, \xi\} \subset (0,1)$, $0 < \delta_{\min} \le \delta_{\max} \le \delta_a < \infty$, and $0 \le max\_fails \in \mathbb{N}$, set $fails \leftarrow 0$, $\mathcal{S} \leftarrow \{0\}$, $\mathcal{U} \leftarrow \emptyset$, $k \leftarrow 0$, $\mathcal{F}_0 \leftarrow \emptyset$, and $\mathcal{P}$-mode $\leftarrow$ false, and then choose initializations $\sigma_0 > 0$ and $\delta_0 \in [\delta_{\min}, \delta_{\max}]$.
3:  loop
4:      if $fails > max\_fails$ then
5:          $x_k \leftarrow x_{R(k)}$, $s_k \leftarrow s_{R(k)}$, $s_k^a \leftarrow s_{R(k)}^a$, $y_k^p \leftarrow y_{R(k)}^p$, $\sigma_{k+1} \leftarrow \sigma_{R(k)+1}$, $H_k \leftarrow H_{R(k)}$.
6:      else
7:          Compute $s_k^s$ as a solution of (2.2), and then calculate $\Delta\ell^v(s_k^s; x_k)$ from (2.3).
8:          if (2.4) holds then
9:              return with the infeasible stationary point $x_k$ for problem (1.1).
10:         Choose $B_k \succ 0$ and compute $s_k^p$ as the unique solution of (2.5) with multiplier $y_k^p$.
11:         if $\Delta q^\phi(s_k^p; x_k, \sigma_k) = v(x_k) = 0$, then
12:             return with the KKT point $(x_k, y_k^p)$ for problem (1.1).
13:         Compute $s_k = (1-\tau_k)s_k^s + \tau_k s_k^p$ from (2.6) such that (2.7) is satisfied.
14:         Compute the new weight $\sigma_{k+1}$ from (2.8).
15:         Evaluate $H_k = \nabla_{xx}^2 L(x_k, y_k^p)$. Solve (2.13) and (2.14) to get $s_k^a$ and $y_k^a$.
16:     Compute $s_k^{c\phi}$ from (2.10) and then calculate $\Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1})$ from (2.12).
17:     if $\mathcal{P}$-mode then
18:         for $j = 0, 1, 2, \ldots$ do
19:             $\alpha_k \leftarrow \xi^j$.
20:             for $\hat{s}_k \in \{s_k^a, s_k\}$ do
21:                 if $(\alpha_k, \hat{s}_k)$ is a p-pair then
22:                     $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$ and go to line 25.                  $\triangleright\ k+1 \in \mathcal{S}$
23:                 if $fails \le max\_fails$ and $max\_fails > 0$ then
24:                     $fails \leftarrow fails + 1$, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, go to line 45.   $\triangleright\ k+1 \in \mathcal{U}$
25:         if $x_k + \alpha_k \hat{s}_k$ is acceptable to $\mathcal{F}_k$ then
26:             $\mathcal{P}$-mode $\leftarrow$ false.
27:     else
28:         Compute $s_k^{cf}$ from (2.9) and then calculate $\Delta q^f(s_k^{cf}; x_k, H_k)$ from (2.11).
29:         for $j = 0, 1, 2, \ldots$ do
30:             $\alpha_k \leftarrow \xi^j$.
31:             for $\hat{s}_k \in \{s_k^a, s_k\}$ do
32:                 if $(\alpha_k, \hat{s}_k)$ is a v-pair then
33:                     $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_{R(k)}, f_{R(k)})\}$, go to line 44.     $\triangleright\ k+1 \in \mathcal{S}$
34:                 if $(\alpha_k, \hat{s}_k)$ is an o-pair then
35:                     $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, go to line 44.                       $\triangleright\ k+1 \in \mathcal{S}$
36:                 if $fails \le max\_fails$ and $max\_fails > 0$ then
37:                     if $(\alpha_k, \hat{s}_k)$ is a b-pair then
38:                         $\mathcal{P}$-mode $\leftarrow$ true.                                     $\triangleright\ k+1 \in \mathcal{S}$
39:                         $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_{R(k)}, f_{R(k)})\}$, go to line 44.
40:                     else
41:                         $fails \leftarrow fails + 1$, $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k$, go to line 45.   $\triangleright\ k+1 \in \mathcal{U}$
42:             if $(\alpha_k, s_k)$ is a b-pair then
43:                 $\mathcal{F}_{k+1} \leftarrow \mathcal{F}_k \cup \{(v_{R(k)}, f_{R(k)})\}$, $\mathcal{P}$-mode $\leftarrow$ true, go to line 44.  $\triangleright\ k+1 \in \mathcal{S}$
44:     $fails \leftarrow 0$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{k+1\}$.
45:     if (2.23) is satisfied then
46:         $\sigma_{k+1} \leftarrow \sigma_{k+1} + \sigma_{\text{inc}}$.
47:     $x_{k+1} \leftarrow x_k + \alpha_k \hat{s}_k$, $y_{k+1} \leftarrow y_k^a$, $\delta_{k+1} \in [\delta_{\min}, \delta_{\max}]$, $k \leftarrow k+1$.
```

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 [26] since they only depend on Assumptions 3.1–3.4, 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 [26], we may again deduce that [26, Lemmas 4.19 and 4.20, Theorem 4.21] 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 [26] 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 are stated as [26, Definition 2.14] but are restated here: $x_k + \alpha s$ is acceptable to the filter $\mathcal{F}_k$ augmented by x_k, and $\Delta\ell^f(s_k; x_k) < \gamma_v \Delta\ell^v(s_k; x_k)$. In our nonmonotone Algorithm 1, 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 2.3 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 1, 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 (2.16), 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 (see [26, Lemma 4.16]).

The second difference arises in the definition of o-iterates and the associated o-pairs. In the monotone algorithm [26], a pair (α, s) constituted an o-pair at iteration k if the following conditions (see [26, Definition 2.15]) 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 1, we again use the last successful iterate $x_{R(k)}$, which leads to our Definition 2.4. 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 needed 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 [26, Lemma 4.12] and the penalty function [26, Lemma 4.13].

The third difference is the definition of b-pairs. In the monotone algorithm, the conditions for that define a b-pair (see [26, Definition 2.16]) 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 2.5. 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 [26, Lemma 4.17(ii)].

The fourth difference is the definition of a p-pair. In the monotone algorithm, the condition that defines a p-pair (see [26, Definition 2.17]) is that $\phi(x_k + \alpha s; \sigma_{k+1}) \le \phi(x_k; \sigma_{k+1}) - \gamma_\phi \alpha \rho_k^\phi$, where $\gamma_\phi \in (0,1)$ is a chosen fixed constant and we defined $\rho_k^\phi := \min\left\{\Delta\ell^\phi(s_k; x_k, \sigma_{k+1}),\, \Delta q^\phi(s_k^{c\phi}; x_k, H_k, \sigma_{k+1})\right\}$. This condition ensured that the penalty function was sufficiently reduced. In the nonmonotone Algorithm 1, we have again adjusted our conditions to be based on the last successful iterate $R(k)$ as given by Definition 2.6. 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 [26, Lemma 4.10 and Theorem 4.11] 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 (2.4)).

A fifth difference is that the monotone algorithm only searches along the direction s_k for a b-pair, whereas our nonmonotone Algorithm 1 additionally checks whether $(1, s_k^a)$ is a b-pair in line 37. This impacts the proof of [26, Lemma 4.8(iii)], which is described only for the step s_k. In turn, [26, Lemma 4.8(iii)] is used in the proofs of [26, Lemmas 4.13 and 4.17(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 37 of Algorithm 1, 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., [26, Lemma 4.8]). ∎

4. Local convergence.

We show that Algorithm 1 is Q-quadratically convergent by making use of the following additional assumption.

Assumption 4.1. Algorithm 1 generates an infinite sequence of iterates $\{x_k\}$ that converges to a KKT-point x_* for problem (1.1) 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 $s^T H_* s \ge \lambda_{\min} \|s\|_2^2$ 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 used in Assumption 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 SQP 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 1, by considering the filter and penalty mode separately. In particular, if $\mathcal{P}$-*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 $\mathcal{P}$-*mode* = **true** during iteration k, we show that one of the two points must be a p-iterate. Theorem 4.7 ties all of these facts together.

We begin by first showing that under Assumption 4.1, the penalty parameter is bounded and that infinitely many iterations occur in filter mode.

LEMMA 4.1. *If Assumption* 4.1 *holds, then* (i) *the penalty parameter $\sigma_k = \bar{\sigma} < \infty$ for all k sufficiently large, and* (ii) *$\mathcal{P}$-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 [26, Theorem 4.21] that $\sigma_k = \bar{\sigma} < \infty$ for all k sufficiently large, which proves part (i). Moreover, since (i) has been established, it follows as in [26, Lemma 4.10 and Theorem 4.11] that if all sufficiently large successful iterations are p-iterates, then x_* is an infeasible stationary point (see (2.4)). This contradicts the fact that x_* is a KKT point (in particular that it is feasible), and therefore 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 [8, Theorem 14.5.1].

LEMMA 4.2. *If Assumptions* 3.3 *and* 4.1 *hold, then $\sigma_k \equiv \bar{\sigma} \geq \|y_*\|_\infty$ for all sufficiently large k.*

Proof. By Lemma 4.1, 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 [8, Theorem 14.5.2] and the fact that (x_*, y_*) is a first-order KKT pair for problem (1.1) that x_* is not a local minimizer of $\phi(x; \bar{\sigma})$.

Since x_* is a KKT-point, we know from [26, Theorem 4.11] that not all iterations are p-iterates for sufficiently large k. It then follows from [26, Theorem 4.9] 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 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 (2.5a) 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 4.1(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 (2.1)) for all k sufficiently large; therefore, $\Delta\ell^v(s_k^s; x_k) = v_k$ and the predictor step is computed from (2.5a) 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 [4, Theorem 3.2(a)] that $s_*^p \neq 0$. Using this fact, that s_k^p is computed from (2.5a) for all sufficiently large k, and Assumptions 3.2 and 4.1, we know that $\lim_{k\in\mathcal{K}} s_k^p = s_*^p \neq 0$, and therefore we have that $\lim_{k\in\mathcal{K}} \Delta q^\phi(s_k^p; x_k, B_k, \bar{\sigma}) \neq 0$. In the first case above, i.e., iterates are o-iterates for sufficiently large k, this contradicts [26, Lemmas 4.13]; in the second, this contradicts [26, Lemmas 4.17(i)]; and in the third, this contradicts [26, Lemmas 4.17(ii)]. ☐

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

Assumption 4.2. For all sufficiently large k, we have $\sigma_k \equiv \bar{\sigma} > \|y_*\|_\infty$.

Next, we present some key results from [29].

LEMMA 4.3. *Let $w_* = (x_*, y_*)$ be the minimizer for problem* (1.1) *that satisfies Assumption* 4.1*, let Assumptions* 3.3 *and* 4.2 *hold, and let $\gamma_{c\phi} \in \big(\max\{\gamma_f, \gamma_\phi\}, 1\big)$ with γ_f and γ_ϕ defined in Definitions* 2.4 *and* 2.5*. 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* (2.15) *and Assumption* 4.1*);*

(ii) s_k^a *is the minimum norm solution to*

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; g_k^T s + \tfrac{1}{2} s^T H_k s \;\; \text{subject to} \;\; c_k + J_k s \geq 0, \tag{4.1}$$

which is the traditional SQP subproblem; 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 [29] directly apply here since the step computation is the same. In particular, part (i) is equivalent to [29, Lemma 3.7(ii)], part (ii) is established by [29, proof of Theorem 3.12], and part (iii) follows from [29, proof of Theorem 3.12 and equations (2.7) and (2.9)]. ◻

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

LEMMA 4.4. *Let Assumptions* 3.3*,* 4.1*, and* 4.2 *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

$$\lim_{k\in\mathcal{S}_2\to\infty} \frac{\phi(x_k; \bar\sigma) - \phi(x_k + s_k^a + s_{k+1}^a; \bar\sigma)}{\Delta q^\phi(s_k^a; x_k, H_k, \bar\sigma)} = 1 \tag{4.2}$$

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 (4.2) follows from [8, Theorem 15.3.7] and from the fact that $\Delta q^\phi(s_k^a; x_k, H_k, \bar\sigma) > 0$ for all $k \in \mathcal{S}$ sufficiently large can be found in the first line of the proof of [8, Theorem 15.3.7]. ◻

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

LEMMA 4.5. *Let $w_* = (x_*, y_*)$ be the minimizer for problem* (1.1) *that satisfies Assumption* 4.1*. Also, let Assumptions* 3.3 *and* 4.2 *hold, and $\gamma_{c\phi}$ and $\delta > 0$ be as defined in Lemma* 4.3*. 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* (2.18a) *and* (2.18b) *are satisfied, with k replaced by $k+1$, by the pair $(1, s_{k+1}^a)$.*

Proof. Since $k \in \mathcal{S}$, Algorithm 1 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 $k+1 \notin \mathcal{S}$, $R(k+1) = k$.

It follows from (2.7), $v(x_k) = 0$, 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 [26, Lemma 2.7(ii)] the inequalities $\Delta\ell^\phi(s_k; x_k, \bar\sigma) \geq \frac{1}{2} s_k^{p\,T} B_k s_k^p \geq 0$, 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 (2.18a) 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 (2.5a) 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 (2.6)) 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 4.3). Then, it follows from definition of ϕ, $x_{k+1} = x_k + s_k^a$, $v(x_k) = 0$, $v(x_{k+1} + s_{k+1}^a) \geq 0$, Lemmas 4.4 and 4.3(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 (2.18b), 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 4.6. *Let $w_* = (x_*, y_*)$ be the minimizer for problem* (1.1) *that satisfies Assumption* 4.1. *Also, let Assumptions* 3.3 *and* 4.2 *hold and $\gamma_{c\phi}$ and $\delta > 0$ be defined as in Lemma* 4.5. *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 follows that if* (2.18a) *is satisfied and* (2.18b) *is violated (both with k replaced by $k+1$) by the pair $(1, s_{k+1}^a)$, then $x_{k+1} = x_k + s_k^a$, $(1, s_{k+1}^a)$ is a b-pair during iteration $k+1$, $k+2 \in \mathcal{S}$, and $x_{k+2} = x_k + s_k^a + s_{k+1}^a$.*

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

We first show that (2.21) is satisfied, with k replaced by $k+1$, by the pair $(1, s_{k+1}^a)$. It follows from Lemmas 4.4 and 4.3(iii), the choice of κ, and (2.22) that

$$\begin{aligned}
\phi(x_k; \bar\sigma) - \phi(x_{k+1} + s_{k+1}^a; \bar\sigma) &\geq \kappa\Delta q^\phi(s_k^a; x_k, H_k, \bar\sigma) \\
&\geq \kappa\gamma_{c\phi}\Delta q^\phi(s_k^{c\phi}; x_k, H_k, \bar\sigma) \geq \gamma_\phi \rho_k^\phi
\end{aligned} \tag{4.3}$$

for sufficiently large k, which shows that (2.21), with k replaced by $k+1$, is satisfied by $(1, s_{k+1}^a)$.

We next show that (2.20) is satisfied, with k replaced by $k+1$, by the pair $(1, s_{k+1}^a)$. Define $\mathcal{G} = \{i : c_i(x_k) < 0\}$ and $\mathcal{H} = \{i : c_i(x_k) \geq 0\}$ and observe that Lemma 4.3(i)–(ii) and the definition of s_k^a (see (2.13) and (2.14)) imply that $c_k + J_k s_k^p \geq 0$. We also know that $s_k = s_k^p$ since $\tau_k = 1$ (see (2.6)) 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 $[c_i(x_k)]^- - \big[c_i(x_k) + \alpha\nabla c_i(x_k)^T s_k\big]^- \geq 0$ 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

$$\Delta \ell^v(s_k^{cf}; x_k) = \big\| [c_k]^- \big\|_1 - \big\| \big[c_k + J_k s_k^{cf}\big]^- \big\|_1 \geq 0. \tag{4.4}$$

Now, choose any $\kappa \in (\max\{\gamma_f, \gamma_\phi\}/\gamma_{c\phi}, 1)$, which is possible since $\max\{\gamma_f, \gamma_\phi\} < \gamma_{c\phi} < 1$ by the definition of $\gamma_{c\phi}$ in Lemma 4.3. 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 4.4, the supposition that (2.18b) is violated (with k replaced by $k+1$) by the pair $(1, s_{k+1}^a)$, $R(k+1) = k$, Lemma 4.3(iii), the definitions of ρ_k^f, $s_k^{c\phi}$, s_k^{cf}, and Δq^ϕ, (4.4), the selection of κ, and $\Delta q^f(s_k^{cf}; x_k, H_k) > 0$ to conclude that

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

so that (2.20) 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 know that $\Delta \ell^v(s_k^s; x_k) > 0$ because otherwise Algorithm 1 would have exited in line 9. It then follows from the definition of ϕ, the fact that we have already shown that (2.21) (with k replaced by $k+1$) is satisfied by $(1, s_{k+1}^a)$, $R(k+1) = k$, the assumption that (2.18b) 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, [26, Lemma 2.8], $\Delta \ell^v(s_k^s; x_k) > 0$, and (2.10) that

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

so that (2.20) is again satisfied, with k replaced by $k+1$, by the pair $(1, s_{k+1}^a)$.

Since we have shown that (2.20) and (2.21) (with k replaced by $k+1$) are satisfied by $(1, s_{k+1}^a)$, and we know from assumption that (2.18a) is satisfied and (2.18b) 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 1 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 4.7. *Let $w_* = (x_*, y_*)$ be the minimizer for problem* (1.1) *that satisfies Assumption* 4.1. *Furthermore, let Assumptions* 3.3 *and* 4.2 *hold, $\delta > 0$ be given as in Lemma* 4.5, *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 $\nabla_{xx}^2 L(x, y)$ 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 $\mathcal{P}$-*mode* = **false** during 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 $k \in \mathcal{S}$, *fails* = 0, $max\_fails > 0$, and the structure of Algorithm 1. 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 [44, Lemmas 4.5 and 4.6, Theorem 4.7] to establish that $x_k + s_k^a + s_{k+1}^a$ 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 [44]. Specifically, if a point is acceptable to the filter given by [44, equation (10)], then it is also acceptable to our filter given by Definition 2.1. This is easy to see since the inequalities in Definition 2.1 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 [44, Lemmas 4.5 and 4.6, Theorem 4.7] to show that $x_k + s_k^a + s_{k+1}^a$ is acceptable to the augmented filter under the current assumptions.

We now consider three cases.

Case 1. Condition (2.18a) is not satisfied at iteration $k+1$. Since (2.17) 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 (2.18a) and (2.18b) 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 (2.18a) holds, but condition (2.18b) is violated at iteration $k+1$. Under the current assumptions, it follows from Lemma 4.5 that $v(x_k) > 0$, or else there would be a contradiction. We may now use Lemma 4.6 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 $\mathcal{P}$-*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 that a similar result holds when $\mathcal{P}$-*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 1. To prove the rest, we suppose that (i) does not hold and proceed to prove that (ii) holds. When (i) does not hold, then the same argument that led to (4.3) 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 (4.1), this means that Algorithm 1 accepts the traditional SQP step at (x_k, y_k) for all k sufficiently large. Since this was the precise condition required to establish [29, Theorem 3.12], we have the same conclusions as in that theorem, which completes the proof of Theorem 4.7. ◻

5. Numerical results.

We present numerical experiments performed with a basic MATLAB implementation of Algorithm 1 (henceforth called FiSQO) on the set of low-dimensional CUTEst [25] 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.

5.1. Implementation details. During each iteration, FiSQO requires the solution of a linear and a quadratic subproblem in order to obtain the steering step s_k^s and the predictor step s_k^p in lines 7 and 10, respectively. In our implementation, they were obtained by using the primal simplex active-set solver in Cplex [33], which we generally found to be reliable. The formulation of the predictor subproblem (2.5) 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 [35], 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 13 was obtained by performing backtracking (starting with an initial guess of $\tau_k = 1$) until condition (2.7) was satisfied. We note that although Algorithm 1 states that τ_k should be computed as the *largest* value on $[0,1]$ that satisfies (2.7), 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 paper (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 15. We defined s_k^a via (2.13), where $s_k^{a'}$ was computed

TABLE 1
Control parameters and initial values required by FiSQO *and* PenSQO.

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

from subproblem (2.14) in the following way. We first used the backslash operator in MATLAB in an attempt to solve the linear system

$$\begin{pmatrix} H_k & [J_k]_{\mathcal{A}_k}^T \\ [J_k]_{\mathcal{A}_k} & 0 \end{pmatrix} \begin{pmatrix} s_k^{a'} \\ -[y_k]_{\mathcal{A}_k} \end{pmatrix} = - \begin{pmatrix} g_k + H(x_k, y_k^p) s_k^p \\ 0 \end{pmatrix}, \tag{5.1}$$

where $\mathcal{A}_k$ is defined by (2.15). 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 (2.14). Of course, a solution to (5.1) may not exist, and even when it does exist, it is a solution to (2.14) 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 a norm bounded by δ_a, i.e., to make it satisfy the trust-region constraint in (2.14), 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 (2.14) 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 4.1), our procedure for computing $s_k^{a'}$ will asymptotically give the unique minimizer to problem (2.14). In particular, this means that our local convergence theory remains valid.

The computations just described, as well as most of the other steps in FiSQO, use control parameters and require the choice of initial values; we used the values in Table 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 1, in our implementation we imposed an iteration limit of 10,000 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 line 9 of Algorithm 1, if it satisfied

$$v_k \geq 100\,\tau_{\text{stop}} \quad \text{and} \quad \Delta\ell^v(s_k^s; x_k) \leq 10^{-12}, \tag{5.2}$$

where the value of the termination tolerance τ_{stop} is given in Table 1. It is clear that these conditions were motivated by (2.4) but designed to account for numerical error. Finally, we concluded that x_k is a solution to (1.1) in line 12 if

$$v_k \leq \tau_{\text{stop}} \ \text{ and } \ \Delta q^\phi(s_k^p; x_k, \sigma_k) \leq 10^{-12} \tag{5.3}$$

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

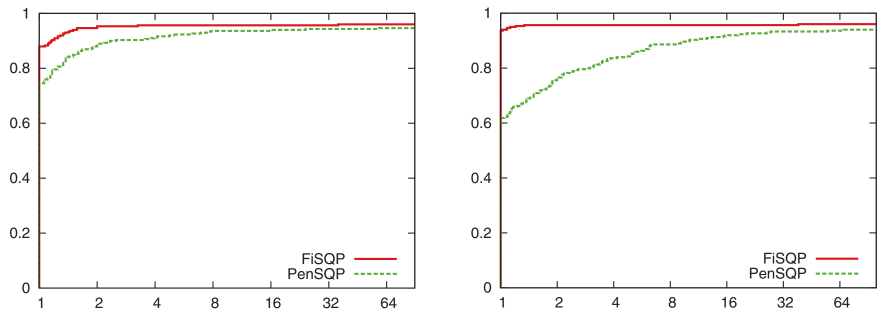

FIG. 1. *Performance profiles on the CUTEst problems with* $m \geq 1$ *and* $\max\{m,n\} \leq 100$ *for the number of iterations (left) and function evaluations (right).*

condition

$$\|F_{\mathrm{KKT}}(x_k, y_k)\|_\infty \leq \tau_{\mathrm{stop}}. \tag{5.4}$$

In our implementation, we set $y_k \leftarrow y_k^p$ if $\|F_{\mathrm{KKT}}(x_k, y_k^p)\|_\infty < \|F_{\mathrm{KKT}}(x_k, y_k^a)\|_\infty$, 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 25 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.

5.2. Collection of CUTEst test problems. We tested our MATLAB implementations of FiSQO and PenSQO on two subsets of problems from the CUTEst [25] 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*, *hs*99*exp*, *lakes*, *tr*04*x*4, *tr*06*x*2, *truspyr*1, and *truspyr*2 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 our accompanying technical report [27, Appendix, Tables 1.3 and 1.4].

Here, to illustrate the performance of our software, we use performance profiles as introduced by Dolan and Moré [16] to give visual comparisons of numerical performance. Consider a performance profile that measures performance 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 β% 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 solved if an approximate infeasible stationary point satisfying (5.2) or an approximate KKT pair (x_k, y_k) satisfying either (5.3) or (5.4) was found.

Figure 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

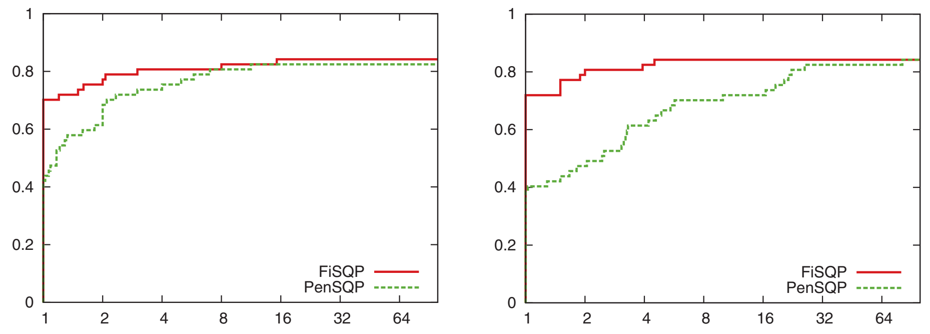

FIG. 2. *Performance profiles on the CUTEst problems with* $m \geq 1$ *and* $100 < \max\{m, n\} \leq 1000$ *for the number of iterations (left) and function evaluations (right).*

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 (overall) progress toward a solution when compared to PenSQO.

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 [12, 20, 21, 24], 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 [27, Appendix, Tables 1.5 and 1.6] for a detailed presentation of the results on a problem-by-problem basis.)

Figure 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 led 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 significantly reduces the number of function evaluations needed by the line search procedure, while the decrease in the number of iterations/gradient evaluations is mild.

5.3. 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 [27, Appendix, Tables 1.3 and 1.5] 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 [27, Appendix, Tables 1.3 and 1.5] that the number of p-pairs is typically very small; notable exceptions are problems *allinita*, *hatfldf*, *mss*1, and *table*7, 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 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 (*himmelp*2, *hs*111*lnp*, and *hs*27) required a single extra iteration and one of which (*hs*92) required 5 additional iterations. We also

TABLE 2

Results for FiSQO/modFiSQO *on the CUTEst problems of size* $1 \le m \le \max\{m,n\} \le 1000$ *for which at least one b-pair was needed.*

Problem	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

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; *acopp*14 (42/11), *bt*7 (106/77), *qpcboei*1 (17/16), *qpcstair* (28/12), and *qpnboei*2 (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 evaluations, 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., *hs*101 (72/1879), *tenbars*1 (76/525), and *acopr*57 (16/101), to name a few.

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.

6. Conclusions and discussion. This paper considered the local convergence properties and numerical performance of FiSQO: a nonmonotone variant of the filter line search algorithm proposed in [26] 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. Next, 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 clearly 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 paper 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 [18], would typically perform better on infeasible problems, a feature directly attributed to feasibility restoration. We believe, however, that recent advances in feasibility detection [3] could be used within our framework and perhaps reduce, if not entirely mitigate, this disadvantage.

REFERENCES

[1] E. G. Birgin, R. Castillo, and J. M. Martínez, *Numerical comparison of augmented Lagrangian algorithms for nonconvex problems*, Comput. Optim. Appl., 31 (2005), pp. 31–55.

[2] P. T. Boggs and J. W. Tolle, *Sequential quadratic programming*, Acta Numer., 4 (1995), pp. 1–51.

[3] R. H. Byrd, F. E. Curtis, and J. Nocedal, *Infeasibility detection and SQP methods for nonlinear optimization*, SIAM J. Optim., 20 (2010), pp. 2281–2299.

[4] R. H. Byrd, G. Lopez-Calva, and J. Nocedal, *A line search exact penalty method using steering rules*, Math. Program., 133 (2012), pp. 39–73.

[5] 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, New York, 1982, pp. 1–17.

[6] 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., 28 (1991), pp. 545–572.

[7] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization*, Math. Program., 73 (1996), pp. 73–110.

[8] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, 2000.

[9] F. E. Curtis, N. I. M. Gould, D. P. Robinson, and Ph. L. Toint, *An Interior-Point Trust-Funnel Algorithm for Nonlinear Optimization*, preprint, RAL-P-2014-001, 2014.

[10] F. E. Curtis, H. Jiang, and D. P. Robinson, *An adaptive augmented Lagrangian method for large-scale constrained optimization*, Math. Program., 151 (2015), pp. 201–245.

[11] 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., 24 (2014), pp. 1041–1074.

[12] F. E. Curtis and J. Nocedal, *Flexible penalty functions for nonlinear constrained optimization*, IMA J. Numer. Anal., 28 (2008), pp. 749–769.

[13] F. E. Curtis, O. Schenk, and A. Wächter, *An interior-point algorithm for large-scale nonlinear optimization with inexact step computations*, SIAM J. Sci. Comput., 32 (2010), pp. 3447–3475.

[14] J. Dennis and J. J. Moré, *A characterization of superlinear convergence and its application to quasi-Newton methods*, Math. Comp., 28 (1974), pp. 549–560.

[15] G. Di Pillo and L. Grippo, *Exact penalty functions in constrained optimization*, SIAM J. Control Optim., 27 (1989), pp. 1333–1360.

[16] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[17] R. Fletcher, *An ℓ_1 penalty method for nonlinear constraints*, in Numerical Optimization 1984, P. T. Boggs, R. H. Byrd, and R. B. Schnabel, eds., SIAM, Philadelphia, 1985, pp. 26–40.

[18] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Program., 91 (2002), pp. 239–269.

[19] R. Fletcher, S. Leyffer, and Ph. L. Toint, *On the global convergence of a filter-SQP algorithm*, SIAM J. Optim., 13 (2002), pp. 44–59.

[20] P. E. Gill, V. Kungurtsev, and D. P. Robinson, *A Stabilized SQP Method: Global Convergence*, Center for Computational Mathematics Report CCoM 13-04, University of California, San Diego, 2013.

[21] P. E. Gill, V. Kungurtsev, and D. P. Robinson, *A Stabilized SQP Method: Superlinear Convergence*, Center for Computational Mathematics Report CCoM 14-01, University of California, San Diego, 2014.

[22] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Rev., 47 (2005), pp. 99–131.

[23] P. E. Gill and D. P. Robinson, *A primal-dual augmented Lagrangian*, Comput. Optim. Appl., 51 (2012), pp. 1–25.

[24] P. E. Gill and D. P. Robinson, *A globally convergent stabilized SQP method*, SIAM J. Optim., 23 (2013), pp. 1983–2010.

[25] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEst: A constrained and unconstrained testing environment with safe threads for mathematical optimization*, Comput. Optim. Appl., 60 (2015), pp. 545–557.

[26] N. I. M. Gould, Y. Loh, and D. P. Robinson, *A filter method with unified step computation for nonlinear optimization*, SIAM J. Optim., 24 (2014), pp. 175–209.

[27] N. I. M. Gould, Y. Loh, and D. P. Robinson, *A Nonmonotone Filter SQP Method: Local Convergence and Numerical Results*, preprint, RAL-P-2014-012R, 2014.

[28] N. I. M. Gould and D. P. Robinson, *A second derivative SQP method: Global convergence*, SIAM J. Optim., 20 (2010), pp. 2023–2048.

[29] N. I. M. Gould and D. P. Robinson, *A second derivative SQP method: Local convergence and practical issues*, SIAM J. Optim., 20 (2010), pp. 2049–2079.

[30] N. I. M. Gould and D. P. Robinson, *A second derivative SQP method with a "trust-region-free" predictor step*, IMA J. Numer. Anal., 32 (2012), pp. 580–601.

[31] 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., Kluwer Academic, Dordrecht, The Netherlands, 2005.

[32] L. Grippo, F. Lampariello, and S. Lucidi, *A class of nonmonotone stabilization methods in unconstrained optimization*, Numer. Math., 59 (1991), pp. 779–805.

[33] IBM, *ILOG CPLEX: High-Performance Software for Mathematical Programming and Optimization*, 2006.

[34] M. Kočvara and M. Stingl, *PENNON: A generalized augmented Lagrangian method for semidefinite programming*, in High Performance Algorithms and Software for Nonlinear Optimization, Appl. Optim. 82, Kluwer Academic, Dordrecht, The Netherlands, 2003, pp. 303–321.

[35] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program., 45 (1989), pp. 503–528.

[36] O. L. Mangasarian and S. Fromovitz, *The Fritz John necessary optimality conditions in the presence of equality and inequality constraints*, J. Math. Anal. Appl., 17 (1967), pp. 37–47.

[37] N. Maratos, *Exact Penalty Function Algorithms for Finite-Dimensional and Control Optimization Problems*, Ph.D. thesis, Department of Computing and Control, University of London, 1978.

[38] J. Morales, J. Nocedal, and Y. Wu, *A sequential quadratic programming algorithm with an additional equality constrained phase*, IMA J. Numer. Anal., 32 (2012), pp. 553–579.

[39] M. J. D. Powell and Y.-X. Yuan, *A recursive quadratic programming algorithm that uses differentiable exact penalty functions*, Math. Program., 35 (1986), pp. 265–278.

[40] C. Shen, S. Leyffer, and R. Fletcher, *A nonmonotone filter method for nonlinear optimization*, Comput. Optim. Appl., 52 (2012), pp. 583–607.

[41] K. Su and D. Pu, *A nonmonotone filter trust region method for nonlinear constrained optimization*, J. Comput. Appl. Math., 223 (2009), pp. 230–239.

[42] M. Ulbrich, S. Ulbrich, and L. Vicente, *A globally convergent primal-dual interior-point filter method for nonlinear programming*, Math. Program., 100 (2004), pp. 379–410.

[43] S. Ulbrich, *On the superlinear local convergence of a filter-SQP method*, Math. Program., 100 (2004), pp. 217–245.

[44] A. Wachter and L. T. Biegler, *Line search filter methods for nonlinear programming: Local convergence*, SIAM J. Optim., 16 (2005), pp. 32–48.

[45] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2006), pp. 25–57.

[46] 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).

[47] 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., 107 (2006), pp. 391–408.

[48] V. M. Zavala and M. Anitescu, *Scalable nonlinear programming via exact differentiable penalty functions and trust-region Newton methods*, SIAM J. Optim., 24 (2014), pp. 528–558.