SIAM J. OPTIM.
Vol. 22, No. 2, pp. 474–500
© 2012 Society for Industrial and Applied Mathematics

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

A SEQUENTIAL QUADRATIC PROGRAMMING ALGORITHM FOR NONCONVEX, NONSMOOTH CONSTRAINED OPTIMIZATION*

FRANK E. CURTIS† AND MICHAEL L. OVERTON‡

Abstract. We consider optimization problems with objective and constraint functions that may be nonconvex and nonsmooth. Problems of this type arise in important applications, many having solutions at points of nondifferentiability of the problem functions. We present a line search algorithm for situations when the objective and constraint functions are locally Lipschitz and continuously differentiable on open dense subsets of $\mathbb{R}^n$. Our method is based on a sequential quadratic programming (SQP) algorithm that uses an ℓ_1 penalty to regularize the constraints. A process of gradient sampling (GS) is employed to make the search direction computation effective in nonsmooth regions. We prove that our SQP-GS method is globally convergent to stationary points with probability one and illustrate its performance with a MATLAB implementation.

Key words. nonconvex optimization, nonsmooth optimization, constrained optimization, sequential quadratic programming, gradient sampling, exact penalization

AMS subject classifications. 49M37, 65K05, 65K10, 90C26, 90C30, 90C55

DOI. 10.1137/090780201

1. Introduction.

In this paper, we consider optimization problems of the form

$$\begin{array}{lll} \underset{x}{\text{minimize}} & (\min\limits_{x}) & f(x) \\ \text{subject to} & (\text{s.t.}) & c(x) \leq 0, \end{array} \tag{1.1}$$

where the objective $f : \mathbb{R}^n \to \mathbb{R}$ and constraint functions $c : \mathbb{R}^n \to \mathbb{R}^m$ are locally Lipschitz and continuously differentiable on open dense subsets of $\mathbb{R}^n$. Due to an exact penalization strategy that we apply to problem (1.1), the solution method that we propose can also be applied to problems with equality constraints if they are relaxed with slack variables. For ease of exposition, however, we focus on only inequality constraints in (1.1). Since we make no convexity assumptions about f and/or c, we are concerned only with finding local solutions to (1.1).

A wealth of research on the solution of *smooth constrained* optimization problems has been produced in recent decades. In particular, the methodology known as sequential quadratic programming (SQP) has had one of the longest and richest histories [21, 36]. Its many variations are still widely used and studied throughout the optimization community as new techniques are developed to confront the issues of nonconvexity, ill-conditioned constraints, and the solution of large-scale applications [3, 17]. At an iterate x_k, the main feature of traditional SQP algorithms is the following quadratic programming (QP) subproblem used to compute a search direction:

$$\begin{array}{l} \min\limits_{d} \; f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d \\ \text{s.t. } c^j(x_k) + \nabla c^j(x_k)^T d \leq 0, \quad j = 1, \ldots, m. \end{array} \tag{1.2}$$

*Received by the editors December 15, 2009; accepted for publication (in revised form) February 16, 2012; published electronically May 15, 2012.
http://www.siam.org/journals/siopt/22-2/78020.html

†Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18018 (frank.e.curtis@gmail.com). This author was supported by National Science Foundation grants DMS 0602235 and DMS 1016291.

‡Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (overton@cs.nyu.edu). This author was supported by National Science Foundation grants DMS 0714321 and DMS 1016325.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

This subproblem is constructed by forming local linearizations of the objective and constraint functions and by providing a symmetric, sufficiently positive definite, and bounded matrix H_k (preferably approximating the Hessian of the Lagrangian of (1.1)) to ensure that the solution is bounded and one of descent for an appropriate measure of progress, such as a penalty function or filter. This type of formulation makes the search direction calculation intuitively appealing as well as computationally effective.

There has also been a great deal of work on *nonsmooth unconstrained* optimization problems, i.e., problem (1.1) with $m = 0$. For instance, in a collection of recent papers [7, 8, 30], an algorithm known as gradient sampling (GS) was developed and analyzed for the minimization of locally Lipschitz f. The GS method is interesting theoretically in that convergence guarantees hold with probability one, but it is also widely applicable and robust and has been used to solve a variety of interesting problems [6, 14, 19, 20, 42]. It is worthwhile to note that GS does not require the user to compute subgradients as the iterates and sample points remain in the open dense set $\mathcal{D}^f \subset \mathbb{R}^n$ over which f is continuously differentiable. The main computational expenses at a given iterate x_k are $O(n)$ gradient evaluations and the computation of an approximate steepest descent direction. Letting $\operatorname{cl}\operatorname{conv} S$ denote the closure of the convex hull of a set S and defining the multifunction

$$\mathbb{G}_\epsilon^f(x_k) := \operatorname{cl}\operatorname{conv} \nabla f(\mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^f),$$

where

$$\mathbb{B}_\epsilon(x_k) := \{x \mid \|x - x_k\|_2 \le \epsilon\}$$

is the closed ball with radius ϵ centered at x_k, we have the representation

$$\overline{\partial} f(x_k) = \bigcap_{\epsilon > 0} \mathbb{G}_\epsilon^f(x_k)$$

of the Clarke subdifferential of $f(x)$ at x_k [11]. Then, with a finite randomly generated set of points $\mathcal{B}_{\epsilon,k}^f \subset \mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^f$ (including x_k), and $\mathcal{G}_{\epsilon,k}^f \subset \mathbb{G}_\epsilon^f(x_k)$ defined as the convex hull of the gradients of f evaluated at points in $\mathcal{B}_{\epsilon,k}^f$, the approximate steepest descent direction computed by GS is $d_k = -g_k/\|g_k\|_2$, where g_k can be obtained by solving either of the following subproblems (dual to each other):

$$\left\{ \begin{array}{l} \min\limits_{g} \ \frac{1}{2}\|g\|_2^2 \\ \text{s.t. } g \in \mathcal{G}_{\epsilon,k}^f \end{array} \right\}; \qquad \left\{ \begin{array}{l} \min\limits_{g,z} \ z + \frac{1}{2}\|g\|_2^2 \\ \text{s.t. } \nabla f(x)^T(-g) \le z \ \ \forall x \in \mathcal{B}_{\epsilon,k}^f \end{array} \right\}. \tag{1.3}$$

In fact, as shown in [30], the search direction need not be normalized, so a search direction d_k can alternatively be obtained as part of the solution to

$$\begin{array}{l} \min\limits_{d,z} \ z + \frac{1}{2} d^T H_k d \\ \text{s.t. } f(x_k) + \nabla f(x)^T d \le z \ \ \forall x \in \mathcal{B}_{\epsilon,k}^f. \end{array} \tag{1.4}$$

Here, as in (1.2), H_k is a symmetric, sufficiently positive definite, and bounded matrix that ensures boundedness of the solution of (1.4) and descent in f from x_k. We have added $f(x_k)$ (a constant at x_k) to the left-hand side of each constraint in (1.4) for consistency with (1.2) and other subproblems we define later on, noting that it affects the value of the optimal z_k but has no effect on the computed search direction d_k.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

The successes of these and other methods for smooth constrained and nonsmooth unconstrained problems have laid the groundwork for *nonsmooth constrained* optimization algorithms. Indeed, the purpose of this paper is to present and analyze such an approach. Our method combines techniques from SQP and GS for the development of an effective SQP-GS algorithm. At the heart of our discussion is the subproblem

$$\begin{aligned} &\min_{d,z} \; z + \tfrac{1}{2} d^T H_k d \\ &\text{s.t.} \; \begin{cases} f(x_k) + \nabla f(x)^T d \le z & \forall x \in \mathcal{B}^f_{\epsilon,k}, \\ c^j(x_k) + \nabla c^j(x)^T d \le 0 & \forall x \in \mathcal{B}^{c^j}_{\epsilon,k}, \quad j = 1, \ldots, m, \end{cases} \end{aligned} \tag{1.5}$$

which can be seen as a natural extension of subproblem (1.2) when combined with the GS techniques that led to (1.4). Defining $\mathcal{D}^{c^j}$ for each $j = 1, \ldots, m$ as the open dense subset of $\mathbb{R}^n$ over which c^j is locally Lipschitz and continuously differentiable, the finite set $\mathcal{B}^{c^j}_{\epsilon,k} \subset \mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^{c^j}$ (including x_k) is randomly generated as a set of points over which gradients of the jth constraint function are sampled in an ϵ-neighborhood of x_k. As for (1.2) in the case of smooth optimization, enhancements to subproblem (1.5) and other algorithmic features are necessary to ensure that the method is well defined and yields convergence guarantees, but we believe the underlying structure of this subproblem gives our approach both intuitive appeal as well as practical advantages.

We remark that there are two distinct differences between the constraints of subproblem (1.5) and *cutting planes* found in algorithms for nonsmooth optimization (e.g., bundle methods [23, 27]). We describe these differences by noting that in our notation the latter constraints would take the following form (assuming for comparison purposes that f and c^j, $j = 1, \ldots, m$, are differentiable at all points of interest):

$$\begin{cases} f(x) + \nabla f(x)^T d \le z & \forall x \in \mathcal{B}^f_k, \\ c^j(x) + \nabla c^j(x)^T d \le 0 & \forall x \in \mathcal{B}^{c^j}_k, \quad j = 1, \ldots, m. \end{cases} \tag{1.6}$$

The first main difference is that in (1.5), all the linearized constraints involve function values at x_k, as opposed to those in (1.6), which involve function values at the points at which the gradients are evaluated. The latter type of linearization has advantages for convex problems since the cutting planes provide lower bounding models for the true functions. However, for nonconvex problems, these types of cutting planes lose this essential property, causing complications in the development of effective algorithms. In contrast, as can be seen in previous GS methods and in the development of our approach, the linearized constraints in (1.5) lead to a useful search direction regardless of any assumption of convexity. The second main difference between the constraints in (1.5) and those commonly used in algorithms that employ cutting planes is that in the latter methods the sets $\mathcal{B}^f_k$ and $\mathcal{B}^{c^j}_k$ for $j = 1, \ldots, m$ would typically include only subsets of previous iterates. This is in contrast to (1.5), which, in the analysis in this paper, requires new sets of sampled gradients to be computed during every iteration. The cost of these additional gradient evaluations may be prohibitive for some applications. However, we believe that with the foundations provided by the algorithm developed in this paper, further extensions can readily be made that may allow a user to significantly reduce both the size of subproblem (1.5) and limit the number of gradient evaluations required during each iteration. Indeed, we mention some of these extensions along with our numerical experiments, as they have already been implemented in our software. It is clear that these gradient evaluations could

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

be performed in parallel, thus maintaining a clock time equal to that required for a single gradient evaluation if enough processors are available.

We emphasize that the GS procedure in our approach is necessary for ensuring theoretical convergence guarantees since a basic SQP strategy (without GS) will fail in theory and in practice. To understand that this is true, one need only consult the well-known examples of the failure of steepest descent algorithms for nonsmooth problems (e.g., see [23, 31, 41]) and note that in the case of $H_k = I$ with no active constraints at a solution, SQP is a steepest-descent-like strategy. The critical consequences of the GS procedure are that at remote points, it ensures that good search directions are produced, and in small neighborhoods of a solution it ensures that with probability one the algorithm will eventually recognize that an ϵ-stationary point is nearby. Thus, while it is true that a basic SQP method may be able to make some progress from remote starting points, this latter feature will not be attained if an insufficient number of gradients are sampled during each iteration. Moreover, even an SQP method that perturbs the iterates to enforce our requirement that, for all k,

$$x_k \in \mathcal{D} := \mathcal{D}^f \cap \mathcal{D}^{c^1} \cap \cdots \cap \mathcal{D}^{c^m}$$

will fail unless the perturbation strategy is sophisticated enough to ensure that the algorithm does not repeatedly approach suboptimal points of nondifferentiability. (We know of no such perturbation strategy, even for the unconstrained case, and believe that the development of one would not be as straightforward and efficient as our SQP-GS approach.) We briefly illustrate our claims about the necessity of the gradient sampling procedure along with our numerical results in section 5.

Before we close this introduction, we mention previous work on nonsmooth constrained optimization algorithms. One popular approach has been to solve constrained problems through unconstrained optimization techniques, either by using problem-specific reformulations (e.g., see section 5.9 in [31]) or penalty methods (e.g., see section 4.2 in [8], section 14.3 in [15], and [26, 35]). Researchers have also considered special classes of constrained problems for which methods such as the SQP approach in [16] or the techniques in [25, 32] can be applied. In contrast, our approach is designed for general problems of the form (1.1). Overall, our method is most similar to the bundle techniques in [24] and [28, 29], though these methods and ours differ significantly in detail, as revealed by the comments in the discussion above.

The paper is organized as follows. Motivation and a description of our algorithm are presented in section 2. The main components of the approach are the penalty parameter update, search direction computation, and line search. Global convergence guarantees to stationary points are provided in section 3. In section 4 we describe a MATLAB implementation of our algorithm. Our implementation involves certain practical enhancements to the basic framework discussed in section 2, including an update for a variable-metric Hessian approximation and techniques for reducing the sizes of the subproblems. Numerical results are presented in section 5, and concluding remarks are provided in section 6.

2. SQP with GS. In this section, we motivate and present our algorithm. We begin by developing a relaxed, yet still practical version of subproblem (1.5). Our complete algorithm is specified in detail at the end of this section, though some practical features that are inconsequential for our convergence analysis in section 3 are left for the discussion in section 4.

2.1. Search direction calculation. The search direction calculation in our algorithm is inspired by subproblem (1.5). This subproblem lacks safeguards to ensure

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

that it is well defined at any given iterate x_k, so we begin this subsection by proposing a particular safeguard: regularization through exact penalization. We also formally define the set of sample sets

$$\mathcal{B}_{\epsilon,k} := \{\mathcal{B}^f_{\epsilon,k}, \mathcal{B}^{c^1}_{\epsilon,k}, \ldots, \mathcal{B}^{c^m}_{\epsilon,k}\} \tag{2.1}$$

and a line search in such a way that the method possesses convergence guarantees.

Consider the ℓ_1 penalty function

$$\phi_\rho(x) := \rho f(x) + v(x),$$

where $\rho \geq 0$ is a penalty parameter and

$$v(x) := \sum_{j=1}^{m} \max\{c^j(x), 0\}$$

is a constraint violation measure. If x_* is a minimizer of ϕ_ρ for $\rho > 0$ and $v(x_*) = 0$, then x_* solves (1.1) [23, pp. 299]. In order to ensure that such a ρ exists, however, one must formally consider situations such as that described in the following theorem. In the theorem, we define a local stability condition known as *calmness* [11, 39]. We then state a result given as Proposition 1 and Theorem 1 in [40], the proofs of which draw from that of Proposition 6.4.3 in [11].

THEOREM 2.1. *Suppose f and c are locally Lipschitz on $\mathbb{R}^n$ and let x_* be a local minimizer of* (1.1)*. Moreover, suppose problem* (1.1) *is* calm *at x_* in the sense that there exists $\varepsilon > 0$ such that for all sequences $(x_k, s_k) \to (x_*, 0)$, with each pair (x_k, s_k) satisfying $c(x_k) \leq s_k$, we have $f(x_*) - f(x_k) \leq \varepsilon\|s_k\|$. Then, there exists $\rho_* > 0$ and $\epsilon_* > 0$ such that for all $x \in \mathbb{B}_{\epsilon_*}(x_*)$,*

$$\phi_\rho(x) \geq \phi_\rho(x_*) \quad \textit{whenever} \quad \rho \in [0, \rho_*]. \tag{2.2}$$

If, in addition, x_* is an isolated local minimizer, then $\rho_* > 0$ and $\epsilon_* > 0$ can be chosen so that the inequality in (2.2) *is strict for $x \neq x_*$.*

Theorem 2.1 leads to the notion of *exact penalization* that is also commonly discussed in smooth optimization; e.g., see section 17.2 in [34]. With this concept in mind, it is appropriate to use the minimization of ϕ_ρ as an algorithmic tool for solving (1.1). That is, under the assumption that (1.1) is calm at its minimizers and ρ is sufficiently small so that the result of Theorem 2.1 applies, the focus of our discussion and analysis will be on producing a monotonically decreasing sequence $\{\phi_\rho(x_k)\}$. It is worthwhile to note that calmness is a weak constraint qualification in our context as other qualifications, such as the Mangasarian–Fromowitz and Slater conditions, both imply calmness [11, section 6.4]. We also have the following result from Proposition 6.4.4 in [11], proving the existence of Lagrange multipliers at calm solutions to (1.1).

THEOREM 2.2. *Suppose f and c are locally Lipschitz on $\mathbb{R}^n$, let x_* be a calm local minimizer of* (1.1)*, and let $L(x, \lambda) := f(x) + \lambda^T c(x)$ be the Lagrangian of* (1.1)*. Then, there exists $\lambda_* \geq 0$ such that the primal-dual pair (x_*, λ_*) satisfies*

$$\begin{aligned} &\overline{\partial}_x L(x_*, \lambda_*) \ni 0, \\ &\quad \lambda_*^j c^j(x_*) = 0, \ j = 1, \ldots, m, \end{aligned}$$

where $\overline{\partial}_x L(x_*, \lambda_*)$ is the (partial) Clarke subdifferential [11, p. 48] *of $L(\cdot, \lambda_*)$ at x_*.*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Penalty functions in algorithms for constrained optimization problems have been studied extensively and have been widely implemented since the analyses by Han [21] and Powell [37]. Indeed, they have also been investigated in the context of nonsmooth applications [5, 4, 26, 28, 29, 40]. In our context, the two main benefits of the penalty function are that it provides a mechanism for judging progress in the algorithm and, when used in the derivation of the search direction computation, the penalty terms have important regularization effects on the subproblem formulation. For smooth optimization problems, it can be seen that the minimization of a quadratic model of the penalty function ϕ_ρ at a given iterate x_k is equivalent to the QP

$$\begin{aligned} &\min_{d,r} \ \rho(f(x_k) + \nabla f(x_k)^T d) + \sum_{j=1}^{m} r^j + \tfrac{1}{2} d^T H_k d \\ &\text{s.t. } c(x_k) + \nabla c(x_k)^T d \le r, \quad r \ge 0, \end{aligned} \tag{2.3}$$

where $r \in \mathbb{R}^m$ is a vector of auxiliary variables [9]. The solution component d_k of this subproblem is guaranteed to provide descent in the penalty function from the current iterate x_k, which means that (2.3) is consistent with the choice of judging progress by reductions in ϕ_ρ. Moreover, due to the presence of the auxiliary variables, this subproblem is always feasible, meaning that d_k is always well defined. Comparing this subproblem with (1.2), the only difference is that the constraints have been relaxed (with appropriate objective terms for penalizing violations in these constraints), so one may view (2.3) as a regularized version of subproblem (1.2).

In our algorithm, the search direction calculation is performed by solving a QP of the form (1.5) after appropriate penalty terms have been introduced. The subproblem may be viewed as a regularized version of subproblem (1.5) or, alternatively, as an augmented version of subproblem (2.3). To incorporate ideas from the GS algorithms described in [8, 30], we define

$$\begin{aligned} &\mathcal{B}^f_{\epsilon,k} := \{x^f_{k,0}, x^f_{k,1}, \ldots, x^f_{k,p}\}, \quad \text{where} \quad x^f_{k,0} := x_k, \\ \text{and} \quad &\mathcal{B}^{c^j}_{\epsilon,k} := \{x^{c^j}_{k,0}, x^{c^j}_{k,1}, \ldots, x^{c^j}_{k,p}\}, \quad \text{where} \quad x^{c^j}_{k,0} := x_k, \quad \text{for } j = 1, \ldots, m, \end{aligned} \tag{2.4}$$

as sets of independent and identically distributed random points sampled uniformly from $\mathbb{B}_\epsilon(x_k)$ for some sample size $p \ge n+1$. Then, as long as we ensure

$$\mathcal{B}^f_{\epsilon,k} \subset \mathcal{D}^f \quad \text{and} \quad \mathcal{B}^{c^j}_{\epsilon,k} \subset \mathcal{D}^{c^j} \quad \text{for } j = 1, \ldots, m, \tag{2.5}$$

the quadratic subproblem

$$\begin{aligned} &\min_{d,z,r} \ \rho z + \sum_{j=1}^{m} r^j + \tfrac{1}{2} d^T H_k d \\ &\text{s.t.} \begin{cases} f(x_k) + \nabla f(x)^T d \le z \ \forall x \in \mathcal{B}^f_{\epsilon,k} \\ c^j(x_k) + \nabla c^j(x)^T d \le r^j \ \forall x \in \mathcal{B}^{c^j}_{\epsilon,k}, \ r^j \ge 0, \ j = 1, \ldots, m, \end{cases} \end{aligned} \tag{2.6}$$

is well defined and provides a direction of descent for the penalty function ϕ_ρ at x_k; see Lemma 3.7. Indeed, let us define the local model

$$\begin{aligned} &q_\rho(d; x_k, \mathcal{B}_{\epsilon,k}, H_k) \\ &:= \rho \max_{x \in \mathcal{B}^f_{\epsilon,k}} \{f(x_k) + \nabla f(x)^T d\} + \sum_{j=1}^{m} \max_{x \in \mathcal{B}^{c^j}_{\epsilon,k}} \{\max\{c^j(x_k) + \nabla c^j(x)^T d, 0\}\} + \tfrac{1}{2} d^T H_k d \end{aligned}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

of the penalty function ϕ_ρ at x_k. If (d_k, z_k, r_k) solves (2.6), then d_k also solves

$$\min_d \; q_\rho(d; x_k, \mathcal{B}_{\epsilon,k}, H_k) \tag{2.7}$$

(e.g., see [9, 15]), and the two subproblems produce equal optimal objective values. Because of this equivalence, we refer to (2.6) and (2.7) interchangeably throughout the rest of our algorithmic development and analysis.

We remark that, as in other methods that have been proposed for nonsmooth constrained optimization (e.g., see [24]), it may be tempting to *aggregate* all the inequality constraints into the single constraint $\max_j\{c^j(x)\} \leq 0$ in order to reduce the (sub)problem size. We believe, however, that significant disadvantages result from this reformulation. First, this reformulation is not scale invariant, and so an algorithm may perform differently for different scalings of the constraints and poorly for certain choices of scalings. Second, such a reformulation disregards gradient information for constraints that do not currently yield the largest value. This can be detrimental to the optimization process, especially if numerous constraints are active or near-active at a solution point. In contrast, subproblem (2.6) maintains information about all constraints during all iterations and is less sensitive to poor scalings of the constraint functions. We also remark that it may be tempting to select the same sample sets for all functions (when allowed under $\mathcal{D}$), i.e., $\mathcal{B}^f_{\epsilon,k} = \mathcal{B}^{c^1}_{\epsilon,k} = \cdots = \mathcal{B}^{c^m}_{\epsilon,k}$, so that only p random points in $\mathbb{B}_\epsilon(x_k)$ need to be generated during iteration k. Indeed, this choice may work in practice. However, with such a choice, one cannot guarantee an optimal solution to (2.6) (which generally has $\mathcal{B}^f_{\epsilon,k} \neq \mathcal{B}^{c^1}_{\epsilon,k} \neq \cdots \neq \mathcal{B}^{c^m}_{\epsilon,k}$) on which all our convergence results in section 3 (in particular, Lemma 3.8) are based. Thus, such a selection is not recommended. This should not be viewed as a deficiency of our algorithm, however, since with even a rudimentary random number generator, the computational costs of generating distinct sample sets for each function are insignificant compared to the costs of evaluating gradients and solving (2.6).

2.2. Algorithm description. Our complete algorithm is presented as Algorithm 2.1. Similar to the methods in [8, 30], the algorithm progresses by locating points that are approximately ϵ-stationary with respect to the penalty function ϕ_ρ (see Definition 3.5) and then subsequently decreasing the sampling radius ϵ to zero so that, in the limit, a stationary point is produced (see Definition 3.4). The indicator for reducing ϵ is the reduction obtained in the model $q_\rho(\cdot; x_k, \mathcal{B}_{\epsilon,k}, H_k)$. For a computed d_k, we define this reduction to be

$$\begin{aligned}\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) &:= q_\rho(0; x_k, \mathcal{B}_{\epsilon,k}, H_k) - q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \\ &= \phi_\rho(x_k) - q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \geq 0.\end{aligned}$$

(The reduction is nonnegative because $d = 0$ yields an objective value of $\phi_\rho(x_k)$ for subproblem (2.7).) As shown in Lemma 3.6, $\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k)$ can be zero only if x_k is ϵ-stationary, so a sufficiently small reduction in $q_\rho(\cdot; x_k, \mathcal{B}_{\epsilon,k}, H_k)$ indicates that a decrease in ϵ is appropriate.

We also use a small reduction in $q_\rho(\cdot; x_k, \mathcal{B}_{\epsilon,k}, H_k)$ as the condition under which we consider a reduction in the penalty parameter ρ. Let K be the subsequence of iterations during which $\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k)$ is sufficiently small (see step 4) and define a dynamic constraint violation tolerance $\theta > 0$. If $v(x_k) \leq \theta$ for a given $k \in K$, then x_k is considered sufficiently feasible and θ is reduced to tighten the tolerance. Otherwise, if $v(x_k) > \theta$ for $k \in K$, then ρ is decreased to place a higher priority on reducing constraint violation in subsequent iterations. Note that in this manner, if

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

ρ is sufficiently small such that $v(x_k) \leq \theta$ for all large $k \in K$ and $\theta \to 0$, then the penalty parameter will remain bounded while the algorithm attains $\{v(x_k)\}_{k\in K} \to 0$. This means that in the limit, we obtain feasibility for (1.1) over the subsequence K.

The line search is performed by backtracking along the search direction in order to ensure progress in reducing the value of the penalty function ϕ_ρ during each iteration. The main condition that we enforce is the sufficient decrease condition

$$\phi_\rho(x_{k+1}) \leq \phi_\rho(x_k) - \eta\alpha_k \Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k), \tag{2.8}$$

where $x_{k+1} \leftarrow x_k + \alpha_k d_k$ for some $\alpha_k \in (0, 1]$ and $\eta \in (0, 1)$. By virtue of Lemmas 3.6 and 3.7, we find that the directional derivative of ϕ_ρ at x_k along d_k, namely,

$$\phi'_\rho(d_k; x_k) := \lim_{\alpha\to 0^+} \frac{\phi_\rho(x_k + \alpha d_k) - \phi_\rho(x_k)}{\alpha},$$

satisfies the relationships

$$\phi'_\rho(d_k; x_k) \leq -\tfrac{1}{2} d_k^T H_k d_k = -\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k),$$

so (2.8) is weaker than a standard Armijo condition and thus will be satisfied for all sufficiently small $\alpha > 0$. If as a result of the line search we have $x_{k+1} \in \mathcal{D}$, then we continue from x_{k+1}; else, we replace x_{k+1} with any point in $\mathcal{D}$ satisfying (2.8) and

$$\|x_k + \alpha_k d_k - x_{k+1}\| \leq \min\{\alpha_k, \epsilon\}\|d_k\|. \tag{2.9}$$

Since $x_{k+1} \notin \mathcal{D}$ is an unlikely event in exact arithmetic, it is argued in [8, 30] that we may always set $x_{k+1} \leftarrow x_k + \alpha_k d_k$ in practice, in which case (2.9) is trivially satisfied. However, consideration that the iterates remain in $\mathcal{D}$ is necessary for our analysis.

ALGORITHM 2.1. SQP-GS.

1: (Initialization): Choose a sampling radius $\epsilon > 0$, penalty parameter $\rho > 0$, constraint violation tolerance $\theta > 0$, sample size $p \geq n + 1$, line search constant $\eta \in (0, 1)$, backtracking constant $\gamma \in (0, 1)$, sampling radius reduction factor $\beta_\epsilon \in (0, 1)$, penalty parameter reduction factor $\beta_\rho \in (0, 1)$, constraint violation tolerance reduction factor $\beta_\theta \in (0, 1)$, and stationarity tolerance parameter $\nu > 0$. Choose an initial iterate $x_0 \in \mathcal{D}$ and set $k \leftarrow 0$.
2: (Gradient sampling): Generate $\mathcal{B}_{\epsilon,k}$ satisfying (2.1), (2.4), and (2.5).
3: (Search direction calculation): Choose $H_k \succ 0$ and compute (d_k, z_k, r_k) from (2.6).
4: (Parameter update): If $\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) > \nu\epsilon^2$, then go to step 5. Otherwise, if $v(x_k) \leq \theta$, then set $\theta \leftarrow \beta_\theta\theta$, but if $v(x_k) > \theta$, then set $\rho \leftarrow \beta_\rho\rho$. In either case, set $\epsilon \leftarrow \beta_\epsilon\epsilon$, $x_{k+1} \leftarrow x_k$, and $\alpha_k \leftarrow 0$ and go to step 6.
5: (Line search): Set α_k as the largest value in the sequence $\{1, \gamma, \gamma^2, \dots\}$ such that $x_{k+1} \leftarrow x_k + \alpha_k d_k$ satisfies (2.8). If $x_{k+1} \notin \mathcal{D}$, then replace x_{k+1} with any point in $\mathcal{D}$ satisfying (2.8) and (2.9).
6: (Iteration increment): Set $k \leftarrow k + 1$ and go to step 2.

We remark that the sampling in step 2 and the computation of $x_{k+1} \in \mathcal{D}$ satisfying (2.8) and (2.9) in step 5 can be implemented as finite processes. If, for example, the algorithm generates $x^f_{k,i} \notin \mathcal{D}^f$ for some $i \in \{1, \dots, p\}$, then this point may be discarded and replaced by another randomly generated point in $\mathbb{B}_\epsilon(x_k)$, which is then also checked for inclusion in $\mathcal{D}^f$. Since the points are sampled independently

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

and uniformly from $\mathbb{B}_\epsilon(x_k)$ and $\mathcal{D}^f$ is dense, this procedure terminates. Similarly, if $x_k + \alpha_k d_k \notin \mathcal{D}$, then we may sample x_{k+1} from a uniform distribution defined on

$$\{x \mid \|x - (x_k + \alpha_k d_k)\| \leq \min\{\alpha_k, \epsilon\}\|d_k\|/i\},$$

incrementing i by 1 each time until $x_{k+1} \in \mathcal{D}$ and (2.8) holds. As in [30], since α_k is chosen to satisfy (2.8), the continuity of ϕ_ρ implies that this procedure terminates.

It is worthwhile to note that in the context of smooth constrained optimization, Algorithm 2.1 with a sample size of $p = 0$ reduces to an Sℓ_1QP method [15], and in the context of nonsmooth unconstrained optimization with $m = 0$, the algorithm reduces to a variation of the GS algorithms in [8, 30]. Thus, Algorithm 2.1 generalizes each of these methods to nonsmooth constrained problems.

3. Global convergence. We make two assumptions throughout our global convergence analysis. The first relates to the properties of the problem functions themselves, though we believe that in practice Algorithm 2.1 may be a viable approach even when the problem functions are not necessarily locally Lipschitz.

Assumption 3.1. The functions $f, c^1, \ldots, c^m$ are locally Lipschitz and continuously differentiable on open dense subsets $\mathcal{D}^f, \mathcal{D}^{c^1}, \ldots, \mathcal{D}^{c^m}$, respectively, of $\mathbb{R}^n$.

The second assumption relates to the iterates generated in the algorithm.

Assumption 3.2. The sequences of iterates and sample points generated by Algorithm 2.1 are contained in a convex set over which the functions f and c and their first derivatives are bounded. In addition, there exist constants $\overline{\xi} \geq \underline{\xi} > 0$ such that $\underline{\xi}\|d\|^2 \leq d^T H_k d \leq \overline{\xi}\|d\|^2$ for all k and $d \in \mathbb{R}^n$.

We remark that restricting H_k to the space of symmetric, positive semidefinite, and bounded matrices is standard for SQP methods since otherwise one cannot be sure that a QP solver is able to compute a global solution to subproblem (2.6). Indeed, we go slightly further and require H_k to be positive definite with eigenvalues bounded below and above to ensure that the solution to the QP is bounded in norm with respect to an appropriate quantity (see Lemma 3.9). We also remark that our assumption on the boundedness of f eliminates consideration of cases where ϕ_ρ is unbounded below, though it is clear that in such cases either our convergence analysis will apply or the algorithm will produce $\phi_\rho(x_k) \to -\infty$.

The main result that we prove in this section is the following.

THEOREM 3.3. *Let $\{x_k\}$ be the (infinite) sequence of iterates generated by Algorithm* 2.1. *Then, with probability one, one of the following holds true:*

(a) *The penalty parameter satisfies $\rho = \rho_*$ for some $\rho_* > 0$ for all large k and every cluster point of $\{x_k\}$ is stationary for the penalty function ϕ_{ρ_*}. Moreover, with K defined as the (infinite) subsequence of iterations during which ϵ is decreased, we have $\{v(x_k)\}_{k\in K} \to 0$, meaning that all cluster points of $\{x_k\}_{k\in K}$ are feasible for problem* (1.1) *in addition to being stationary for ϕ_{ρ_*}.*

(b) *The penalty parameter satisfies $\rho \to \rho_* = 0$ and every cluster point of $\{x_k\}$ is stationary for the constraint violation measure v.*

We have from Theorem 3.3 that every cluster point of $\{x_k\}$ generated by Algorithm 2.1 is stationary for the penalty function ϕ_{ρ_*} for some $\rho_* \geq 0$. Specifically, along with Theorems 2.1 and 2.2, parts (a) and (b) of the theorem, respectively, tie cluster points of $\{x_k\}$ with first-order solutions and (potentially infeasible) stationary points of the constrained optimization problem (1.1).

Our notion of stationarity for ϕ_ρ differs from that used in GS methods for unconstrained optimization (namely, Clarke stationarity [11]). Rather, our definition resembles the notion of stationarity common in constrained optimization contexts

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

(e.g., see Theorem 3.2 in [9], which draws from the results in [22]) as it relates to the solution of a constrained subproblem. We begin our analysis by defining the notions of stationarity and ϵ-stationary that will be used throughout our analysis.

Consider any $\overline{x} \in \mathbb{R}^n$ and let

$$\overline{\mathcal{B}}_\epsilon := \{\overline{\mathcal{B}}_\epsilon^f, \overline{\mathcal{B}}_\epsilon^{c^1}, \ldots, \overline{\mathcal{B}}_\epsilon^{c^m}\} := \{\mathbb{B}_\epsilon(\overline{x}) \cap \mathcal{D}^f, \mathbb{B}_\epsilon(\overline{x}) \cap \mathcal{D}^{c^1}, \ldots, \mathbb{B}_\epsilon(\overline{x}) \cap \mathcal{D}^{c^m}\} \tag{3.1}$$

be the set of sets of points in an ϵ-ball about $\overline{x}$ over which the problem functions are differentiable. A local model of the penalty function $\phi_{\overline{\rho}}$ about $\overline{x}$ is given by

$$\begin{aligned} &q_{\overline{\rho}}(d; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) \\ &= \overline{\rho} \sup_{x \in \overline{\mathcal{B}}_\epsilon^f} \{f(\overline{x}) + \nabla f(x)^T d\} + \sum_{j=1}^m \sup_{x \in \overline{\mathcal{B}}_\epsilon^{c^j}} \{\max\{c^j(\overline{x}) + \nabla c^j(x)^T d, 0\}\} + \tfrac{1}{2} d^T H_k d, \end{aligned}$$

and a corresponding subproblem to minimize this model is given by

$$\inf_d \; q_{\overline{\rho}}(d; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k). \tag{3.2}$$

Given a solution $\overline{d}$ to (3.2), we have the reduction

$$\begin{aligned} \Delta q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) &:= q_{\overline{\rho}}(0; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) - q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) \\ &= \phi_{\overline{\rho}}(\overline{x}) - q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) \geq 0. \end{aligned}$$

We define the following notion of stationarity.

DEFINITION 3.4. *A point $\overline{x}$ is* stationary *for $\phi_{\overline{\rho}}$ if and only if for all $\epsilon > 0$ the solution to* (3.2) *is $\overline{d} = 0$.*

Our notion of ϵ-stationarity is similar but considers a fixed $\epsilon > 0$.

DEFINITION 3.5. *If for a given $\epsilon > 0$ the solution to* (3.2) *is $\overline{d} = 0$, then $\overline{x}$ is* ϵ-stationary *for $\phi_{\overline{\rho}}$.*

This latter definition leads to our first lemma. The result relates Algorithm 2.1 to the methods in [8, 30], both in terms of the first conditional statement in step 4 of the algorithm and with respect to the line search conditions (2.8) and (2.9). Precisely, considering nonnormalized search directions as in section 4.1 of [30], the algorithms in [8, 30] essentially update the sampling radius ϵ based on norms of the search directions, whereas in our case the value is updated based on $\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k)$. The lemma illustrates that these choices are consistent.

LEMMA 3.6. *The solution d_k to* (2.7) *yields*

$$\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) = \tfrac{1}{2} d_k^T H_k d_k. \tag{3.3}$$

Therefore, if $\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) = 0$, then x_k is ϵ-stationary for ϕ_ρ.

Proof. Equation (3.3) follows from the optimality conditions (2.6); e.g., see [34, eq. (16.37)]. Thus, due to the positive definiteness of H_k under Assumption 3.2, $d_k = 0$ if and only if $\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) = 0$. The second part of the lemma follows from the fact that if $d_k = 0$ solves (2.7), then it also solves (2.7) with min and $\mathcal{B}_{\epsilon,k}$ replaced by inf and $\{\mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^f, \mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^{c^1}, \ldots, \mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^{c^m}\}$, respectively. ∎

The next parts of our analysis proceed in the following manner. We illustrate that the line search procedure in step 5 is well defined so that the method will generate an infinite sequence of iterates $\{x_k\}$. We then show that if the iterates remain in a

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

neighborhood of a point $\overline{x}$ and $\rho = \overline{\rho}$ remains constant, then the algorithm eventually computes a search direction d_k that sufficiently approximates the solution $\overline{d}$ to (3.2). Similarly, if the iterates remain in a neighborhood of a point $\overline{x}$ and $\rho \to 0$, then the algorithm eventually computes a search direction d_k that sufficiently approximates the solution of $\overline{d}$ to (3.2) with $\overline{\rho} = 0$. These results, along with a lemma illustrating that the computed search directions $\{d_k\}$ are bounded in norm with respect to an appropriate quantity, will be used to prove our main theorem.

At x_k and for a given $d \in \mathbb{R}^n$, we respectively define the sets of active and violated linearized constraints as

$$\begin{aligned} \mathcal{A}(d; x_k) &:= \{j \in \{1, \ldots, m\} \mid c^j(x_k) + \nabla c^j(x_k)^T d = 0\} \\ \text{and} \quad \mathcal{V}(d; x_k) &:= \{j \in \{1, \ldots, m\} \mid c^j(x_k) + \nabla c^j(x_k)^T d > 0\}, \end{aligned}$$

so that for $x_k \in \mathcal{D}$ the directional derivative of ϕ_ρ at x_k is given by

$$\phi'_\rho(d; x_k) = \rho \nabla f(x_k)^T d + \sum_{\substack{j \in \mathcal{A}(0; x_k) \\ j \in \mathcal{V}(d; x_k)}} \nabla c^j(x_k)^T d + \sum_{j \in \mathcal{V}(0; x_k)} \nabla c^j(x_k)^T d. \tag{3.4}$$

Our next lemma shows that this directional derivative will always be negative whenever the line search in step 5 of the algorithm is encountered.

LEMMA 3.7. *If Algorithm* 2.1 *executes step* 5 *during iteration* k*, then*

$$\phi'_\rho(d_k; x_k) \leq -d_k^T H_k d_k < 0,$$

and hence there exists $\alpha_k > 0$ *such that condition* (2.8) *is satisfied.*

Proof. Since $x_k \in \mathcal{B}^f_{\epsilon,k} \cap \mathcal{B}^{c^1}_{\epsilon,k} \cap \cdots \cap \mathcal{B}^{c^m}_{\epsilon,k}$ by (2.4), we have from (3.4) that

$$\begin{aligned} &\phi'_\rho(d_k; x_k) \\ &\leq \rho \max_{x \in \mathcal{B}^f_{\epsilon,k}} \nabla f(x)^T d_k + \sum_{\substack{j \in \mathcal{A}(0; x_k) \\ j \in \mathcal{V}(d_k; x_k)}} \max_{x \in \mathcal{B}^{c^j}_{\epsilon,k}} \nabla c^j(x)^T d_k + \sum_{j \in \mathcal{V}(0; x_k)} \max_{x \in \mathcal{B}^{c^j}_{\epsilon,k}} \nabla c^j(x)^T d_k \\ &= \rho(z_k - f(x_k)) + \sum_{\substack{j \in \mathcal{A}(0; x_k) \\ j \in \mathcal{V}(d_k; x_k)}} r_k^j + \sum_{j \in \mathcal{V}(0; x_k)} (r_k^j - c^j(x_k)) \\ &\leq -\phi_\rho(x_k) + \rho z_k + \sum_{j=1}^m r_k^j \\ &= -\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) - \tfrac{1}{2} d_k^T H_k d_k. \end{aligned} \tag{3.5}$$

By Assumption 3.2 and Lemma 3.6, we then have

$$\phi'_\rho(d_k; x_k) \leq -\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) - \tfrac{1}{2} d_k^T H_k d_k = -d_k^T H_k d_k < 0.$$

Thus, d_k is a descent direction for ϕ_ρ at x_k, so $\exists \alpha_k > 0$ satisfying (2.8). ∎

We now turn to the algorithm's ability to approximate the solution to (3.2) when (ρ, x_k) is sufficiently close to $(\overline{\rho}, \overline{x})$. In particular, our goal is to show that if (ρ, x_k) lies in a sufficiently small neighborhood of $(\overline{\rho}, \overline{x})$, then there exists a set of sample sets that the algorithm may generate that will produce a search direction d_k that approximates the solution to subproblem (3.2) with any desired accuracy (though not

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

necessarily perfect accuracy). This result will be used in the proof of Theorem 3.3 since, for a fixed ϵ, it will lead to a contradiction to the supposition that x_k may converge to a point $\overline{x}$ that is not ϵ-stationary for $\phi_{\overline{\rho}}$ (if $\rho = \overline{\rho}$ remains constant) or $\phi_0 = v$ (if $\rho \to 0$). In addition, it will be used in the proof of that result to show that if an iterate is obtained that lies in a sufficiently small neighborhood of a point $\overline{x}$ that is ϵ-stationary for ϕ_ρ, then there is a positive probability that the algorithm will recognize its proximity to ϵ-stationarity.

Now to the formal definitions, statement, and proof of this result. According to Definition 3.4, a measure of the proximity of any point $\overline{x}$ to ϵ-stationarity of $\phi_{\overline{\rho}}$ is $\Delta q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k)$, where $\overline{d}$ solves (3.2). Thus, for a given x_k, we define

$$\mathcal{S}_\epsilon(x_k) := \left\{ \prod_0^p (\mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^f), \prod_0^p (\mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^{c^1}), \ldots, \prod_0^p (\mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^{c^m}) \right\}$$

and consider the set

$$\mathcal{T}_{\rho,\epsilon}(\overline{\rho}, \overline{x}, \omega; x_k) := \{\mathcal{B}_{\epsilon,k} \in \mathcal{S}_\epsilon(x_k) \mid \Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \le \Delta q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) + \omega\}.$$

In the definition of $\mathcal{S}_\epsilon(x_k)$ the notation $\prod_0^p$ indicates that we are taking the Cartesian product of $p+1$ instances of the given sets, and in the definition of $\mathcal{T}_{\rho,\epsilon}(\overline{\rho}, \overline{x}, \omega; x_k)$ we continue with the notation that d_k solves (2.7) and $\overline{d}$ solves (3.2).

We now show that for x_k sufficiently close to $\overline{x}$ the set $\mathcal{T}_{\rho,\epsilon}(\overline{\rho}, \overline{x}, \omega; x_k)$ is nonempty.

LEMMA 3.8. *For any $\omega > 0$, there exists $\zeta > 0$ and a nonempty set $\mathcal{T}$ such that for all $x_k \in \mathbb{B}_\zeta(x')$ and ρ such that $|\rho - \overline{\rho}| \le \zeta$ we have $\mathcal{T} \subset \mathcal{T}_{\rho,\epsilon}(\overline{\rho}, \overline{x}, \omega; x_k)$.*

Proof. We present a proof by illustrating that for a given $\omega > 0$, a nonempty set $\mathcal{T} \subset \mathcal{T}_{\rho,\epsilon}(\overline{\rho}, \overline{x}, \omega; x_k)$ can be constructed. Under Assumption 3.1 and since $\omega > 0$, there exists a vector d satisfying

$$\Delta q_{\overline{\rho}}(d; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) < \Delta q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) + \omega$$

such that for some

$$\begin{aligned} g^f &\in \operatorname{conv} \nabla f(\mathbb{B}_\epsilon(x') \cap \mathcal{D}^f) \\ \text{and} \quad g^{c^j} &\in \operatorname{conv} \nabla c^j(\mathbb{B}_\epsilon(x') \cap \mathcal{D}^{c^j}), \ j = 1, \ldots, m, \end{aligned}$$

we have

$$q_{\overline{\rho}}(d; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) = \rho(f(\overline{x}) + {g^f}^T d) + \sum_{j=1}^m \max\{c^j(\overline{x}) + {g^{c^j}}^T d, 0\} + \tfrac{1}{2} d^T H_k d.$$

Since the sample size p in Algorithm 2.1 satisfies $p \ge n+1$, Carathéodory's theorem [38] implies that there exists

$$\left\{ y_i^f \right\}_{i=1}^p \subset \mathbb{B}_\epsilon(\overline{x}) \cap \mathcal{D}^f$$

and a set of nonnegative scalars $\{\lambda_i^f\}_{i=1}^p$ such that

$$\sum_{i=1}^p \lambda_i^f = 1 \quad \text{and} \quad \sum_{i=1}^p \lambda_i^f \nabla f(y_i^f) = g^f.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Similarly, for $j \in \{1, \ldots, m\}$, the same theorem and conditions imply the existence of

$$\left\{ y_i^{c^j} \right\}_{i=1}^p \subset \mathbb{B}_\epsilon(\overline{x}) \cap \mathcal{D}^{c^j}$$

and nonnegative scalars $\{\lambda_i^{c^j}\}_{i=1}^p$ such that

$$\sum_{i=1}^p \lambda_i^{c^j} = 1 \quad \text{and} \quad \sum_{i=1}^p \lambda_i^{c^j} \nabla c^j(y_i^{c^j}) = g^{c^j}.$$

Since $f, c^1, \ldots, c^m$ are continuously differentiable in the open sets $\mathcal{D}^f, \mathcal{D}^{c^1}, \ldots, \mathcal{D}^{c^m}$, respectively, there exists $\zeta \in (0, \epsilon)$ such that the ordered set

$$\mathcal{T} := \left\{ \{x_k\} \times \prod_{i=1}^p \text{int } \mathbb{B}_\zeta(y_i^f), \{x_k\} \times \prod_{i=1}^p \text{int } \mathbb{B}_\zeta(y_i^{c^1}), \ldots, \{x_k\} \times \prod_{i=1}^p \text{int } \mathbb{B}_\zeta(y_i^{c^m}) \right\}$$

satisfies the following three properties:

- The first entry of $\mathcal{T}$ lies in $\mathbb{B}_{\epsilon-\zeta}(x') \cap \mathcal{D}^f$.
- The $j+1$ entry of $\mathcal{T}$ lies in $\mathbb{B}_{\epsilon-\zeta}(x') \cap \mathcal{D}^{c^j}$ for $j = 1, \ldots, m$.
- The solution d_k to (2.7) with $|\rho - \overline{\rho}| \leq \zeta$ and $\mathcal{B}_{\epsilon,k} \in \mathcal{T}$ satisfies

$$\Delta q_\rho(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \leq \Delta q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) + \omega.$$

Therefore, since for $x_k \in \mathbb{B}_\zeta(\overline{x})$ we have $\mathbb{B}_{\epsilon-\zeta}(\overline{x}) \subset \mathbb{B}_\epsilon(x_k)$, it follows that with $|\rho - \overline{\rho}| \leq \zeta$ the constructed set $\mathcal{T}$ above satisfies $\mathcal{T} \subset \mathcal{T}_{\rho,\epsilon}(\overline{\rho}, \overline{x}, \omega; x_k)$. ∎

The proof of Lemma 3.8 illustrates the need for sampling gradients of each function independently. Indeed, for d as described satisfying $\Delta q_{\overline{\rho}}(d; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) < \Delta q_{\overline{\rho}}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) + \omega$, the objective value of subproblem (3.2) is defined by up to $m+1$ distinct points at which the gradients of the functions (f and c^j for $j = 1, \ldots, m$) are to be evaluated. In order to guarantee (by Carathéodory's theorem) that these points are included in the convex hull of the sampled gradients, the sampling of each function must be performed independently.

A technical lemma related to the solution of (2.7) follows next.

LEMMA 3.9. *The solution of* (2.7) *is contained in* $\mathbb{B}_{\xi_1 \rho + \xi_2 \|\hat{d}_k\|}(0)$ *for some constants* $\xi_1, \xi_2 > 0$, *where* $\hat{d}_k$ *is the minimum-norm minimizer of*

$$l(d; x_k, \mathcal{B}_{\epsilon,k}) := \sum_{j=1}^m \max_{x \in \mathcal{B}_{\epsilon,k}^{c^j}} \max\{c^j(x_k) + \nabla c^j(x)^T d, 0\}.$$

Proof. The function $l(\cdot; x_k, \mathcal{B}_{\epsilon,k})$ is piecewise linear, convex, and bounded below, so it has a minimum-norm minimizer and we call it $\hat{d}_k$. Under Assumption 3.2, the vector $\hat{d}_k$ yields

$$\rho \max_{x \in \mathcal{B}_{\epsilon,k}^f} \{\nabla f(x)^T \hat{d}_k\} + \tfrac{1}{2} \hat{d}_k^T H_k \hat{d}_k \leq \rho \max_{x \in \mathcal{B}_{\epsilon,k}^f} \|\nabla f(x)\| \|\hat{d}_k\| + \tfrac{1}{2} \overline{\xi} \|\hat{d}_k\|^2. \tag{3.6}$$

Similarly, for an arbitrary vector $d \in \mathbb{R}^n$ with

$$\begin{aligned} &\tfrac{1}{4} \underline{\xi} \|d\| > \rho \max_{x \in \mathcal{B}_{\epsilon,k}^f} \|\nabla f(x)\| \\ &\text{and} \quad \tfrac{1}{4} \underline{\xi} \|d\|^2 > \rho \max_{x \in \mathcal{B}_{\epsilon,k}^f} \|\nabla f(x)\| \|\hat{d}_k\| + \tfrac{1}{2} \overline{\xi} \|\hat{d}_k\|^2, \end{aligned} \tag{3.7}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

we have

$$\begin{aligned}\rho \max_{x\in\mathcal{B}^f_{\epsilon,k}} \{\nabla f(x)^T d\} + \tfrac{1}{2} d^T H_k d &\geq -\rho \max_{x\in\mathcal{B}^f_{\epsilon,k}} \|\nabla f(x)\|\|d\| + \tfrac{1}{2}\underline{\xi}\|d\|^2 \\ &> \tfrac{1}{4}\underline{\xi}\|d\|^2 \\ &> \rho \max_{x\in\mathcal{B}^f_{\epsilon,k}} \|\nabla f(x)\|\|\hat{d}_k\| + \tfrac{1}{2}\overline{\xi}\|\hat{d}_k\|^2 \\ &\geq \rho \max_{x\in\mathcal{B}^f_{\epsilon,k}} \{\nabla f(x)^T \hat{d}_k\} + \tfrac{1}{2}\hat{d}_k^T H_k \hat{d}_k,\end{aligned}$$

where the last inequality follows from (3.6). Thus, for d satisfying (3.7), we have $q_\rho(d; x_k, \mathcal{B}_{\epsilon,k}, H_k) > q_\rho(\hat{d}_k; x_k, \mathcal{B}_{\epsilon,k}, H_k)$, which means that d cannot be a minimizer of $q_\rho(\cdot; x_k, \mathcal{B}_{\epsilon,k}, H_k)$. By Assumption 3.1, this means that we can define constants $\xi_1, \xi_2 > 0$ related to the quantities in (3.7) such that no solution of subproblem (2.7) lies outside $\mathbb{B}_{\xi_1\rho+\xi_2\|\hat{d}_k\|}(0)$. ∎

We are now ready to prove Theorem 3.3. Our proof follows [30, Theorem 3.3], except that we also need to consider cases when $\rho \to 0$.

Proof of Theorem 3.3. We begin with a proof of part (a), where it is supposed that $\rho \geq \rho_*$ for some $\rho_* > 0$ for all k. Since ρ is scaled by a fixed factor β_ρ if it is decreased during step 4, we have that $\rho = \rho_*$ for all $k \geq k_*$ for some $k_* \geq 0$. Moreover, by the conditions of step 4, it follows that $\{v(x_k)\}_{k\in K} \leq \theta \to 0$, where K is the subsequence of iterations during which ϵ is decreased.

The update condition (2.9) ensures

$$\|x_{k+1} - x_k\| \leq \min\{\alpha_k, \epsilon\}\|d_k\| + \alpha_k\|d_k\| \leq 2\alpha_k\|d_k\|. \tag{3.8}$$

This inequality holds trivially if the algorithm skips from step 4 to step 6 and holds by the triangle inequality if step 5 yields $x_{k+1} = x_k + \alpha_k d_k$. By condition (2.8), Lemma 3.6, Assumption 3.2, and (3.8), we have for $k \geq k_*$ that

$$\begin{aligned}\phi_{\rho_*}(x_k) - \phi_{\rho_*}(x_{k+1}) &\geq \eta\alpha_k \Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \\ &\geq \tfrac{1}{2}\eta\alpha_k\underline{\xi}\|d_k\|^2 \\ &\geq \tfrac{1}{4}\eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k\|. \qquad (3.9)\end{aligned}$$

Thus, since (2.8) and (3.9) hold for all k, we have by Assumption 3.2 that

$$\sum_{k=k_*}^{\infty} \alpha_k \Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) < \infty \quad \text{and} \tag{3.10a}$$

$$\sum_{k=k_*}^{\infty} \|x_{k+1} - x_k\|\|d_k\| < \infty. \tag{3.10b}$$

We continue by considering two cases, the first of which has two subcases.

Case 1. Suppose there exists $\overline{k} \geq 0$ such that $\epsilon = \overline{\epsilon}$ for some $\overline{\epsilon} > 0$ for all $k \geq \overline{k}$. According to step 4 of Algorithm 2.1, this can occur only if

$$\Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) > \nu(\overline{\epsilon})^2 \ \forall k \geq \max\{\overline{k}, k_*\}. \tag{3.11}$$

The inequality (3.11) in conjunction with (3.10a) means that $\alpha_k \to 0$. Similarly, by Lemma 3.6, inequality (3.11) implies that $\|d_k\|$ is bounded away from zero, which in conjunction with (3.10b) means that $x_k \to \overline{x}$ for some $\overline{x}$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Case 1a. If $\overline{x}$ is $\overline{\epsilon}$-stationary for ϕ_{ρ_*}, then we have $\Delta q_{\rho_*}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_{\overline{\epsilon}}, H_k) = 0$ for any $H_k \succ 0$. Thus, with $\omega = \nu(\overline{\epsilon})^2/2$ and $(\zeta, \mathcal{T})$ chosen as in Lemma 3.8, there exists $k' \geq \overline{k}$ such that $x_k \in \mathbb{B}_\zeta(\overline{x})$ and $\rho = \rho_* = \overline{\rho}$ for all $k \geq \max\{k', k_*\}$ and

$$\Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \leq \nu(\overline{\epsilon})^2/2 \quad \text{whenever } \mathcal{B}_{\epsilon,k} \in \mathcal{T}. \tag{3.12}$$

Together, (3.11) and (3.12) imply that $\mathcal{B}_{\epsilon,k} \notin \mathcal{T}$ for all $k \geq \max\{k', k_*\}$. However, this is a probability zero event since for all such k the points in $\mathcal{B}_{\epsilon,k}$ are sampled uniformly from $\mathcal{S}_\epsilon(x_k)$, which includes the nonempty set $\mathcal{T}$.

Case 1b. If $\overline{x}$ is not $\overline{\epsilon}$-stationary for ϕ_{ρ_*}, then for all $k \geq \max\{\overline{k}, k_*\}$, any α not satisfying the line search condition (2.8) yields

$$\phi_{\rho_*}(x_k + \alpha d_k) - \phi_{\rho_*}(x_k) > -\eta\alpha\Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k),$$

while (3.5) implies

$$\phi_{\rho_*}(x_k + \alpha d_k) - \phi_{\rho_*}(x_k) \leq -\alpha\Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) + \alpha^2 L_k \|d_k\|^2.$$

Here, L_k is an upper bound for $(\phi'_{\rho_*}(d_k; x_k + \alpha d_k) - \phi'_{\rho_*}(d_k; x_k))/(\alpha\|d_k\|)$ on the interval $[x_k, x_k + \alpha_k d_k]$ whose existence follows from Assumption 3.1. Combining these inequalities yields a lower bound on any α not satisfying (2.8), which, since the line search in Algorithm 2.1 has a backtracking factor of γ, yields the bound

$$\alpha_k > \gamma(1-\eta)\Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k)/(L_k\|d_k\|^2) \quad \text{whenever } k \geq \max\{\overline{k}, k_*\}.$$

However, with the tolerance $\omega = \Delta q_{\rho_*}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k)$ (which is strictly positive for any $H_k \succ 0$ since $\overline{x}$ is not $\overline{\epsilon}$-stationary for ϕ_{ρ_*}) and $(\zeta, \mathcal{T})$ again chosen as in Lemma 3.8, there exists $k' \geq \overline{k}$ such that $x_k \in \mathbb{B}_\zeta(\overline{x})$ and $\rho = \rho_* = \overline{\rho}$ for all $k \geq \max\{k', k_*\}$ and

$$\Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \leq 2\Delta q_{\rho_*}(\overline{d}; \overline{x}, \overline{\mathcal{B}}_\epsilon, H_k) \quad \text{whenever } \mathcal{B}_{\epsilon,k} \in \mathcal{T}. \tag{3.13}$$

Under Assumptions 3.1 and 3.2, the fact that $x_k \to \overline{x}$, Lemma 3.9, and (3.13) imply that for all sufficiently large k, $L_k\|d_k\|^2 \leq L$ for some constant $L > 0$, meaning that for all $k \geq \max\{k', k_*\}$ such that $\mathcal{B}_{\epsilon,k} \in \mathcal{T}$, α_k is bounded away from zero. This and the fact that $\alpha_k \to 0$ imply that $\mathcal{B}_{\epsilon,k} \notin \mathcal{T}$ for all $k \geq \max\{k', k_*\}$, which is a probability zero event.

Overall, we have shown in Case 1 that $\epsilon = \overline{\epsilon}$ for some $\overline{\epsilon} > 0$ for all k is a probability zero event. Thus, since every time the algorithm decreases ϵ it does so by multiplying it by a fraction $\beta_\epsilon \in (0, 1)$, the sampling radius ϵ converges to 0 with probability one.

Case 2. Suppose that $\epsilon \to 0$ and $\{x_k\}$ has a cluster point $\overline{x}$. First, we show that

$$\liminf_{k\to\infty} \max\{\|x_k - \overline{x}\|, \|d_k\|\} = 0. \tag{3.14}$$

If $x_k \to \overline{x}$, then by construction in the algorithm and Lemma 3.6, $\epsilon \to 0$ if and only if there is an infinite subsequence K of iterations where $\rho = \rho_*$ and

$$\tfrac{1}{2}\xi\|d_k\|^2 \leq \tfrac{1}{2}d_k^T H_k d_k = \Delta q_{\rho_*}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \leq \nu\epsilon^2.$$

Thus, since $\epsilon \to 0$,

$$\lim_{k\in K} \|d_k\| = 0$$

and (3.14) follows. On the other hand, if $x_k \not\to \overline{x}$, then we proceed by contradiction and suppose that (3.14) does not hold. Since $\overline{x}$ is a cluster point of $\{x_k\}$, there is an $\overline{\epsilon} > 0$ and an index $\overline{k} \geq 0$ such that the set $\overline{K} := \{k \mid k \geq \overline{k},\ \|x_k - \overline{x}\| \leq \overline{\epsilon},\ \|d_k\| > \overline{\epsilon}\}$

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

is infinite. By (3.10b), this means

$$\sum_{k\in\overline{K}} \|x_{k+1} - x_k\| < \infty. \tag{3.15}$$

Since $x_k \not\to \overline{x}$, there exists an $\epsilon > 0$ such that for all $k_1 \in \overline{K}$ with $\|x_{k_1} - \overline{x}\| \leq \overline{\epsilon}/2$ there is $k_2 > k_1$ satisfying $\|x_{k_1} - x_{k_2}\| > \epsilon$ and $\|x_k - \overline{x}\| \leq \overline{\epsilon}$ for all $k_1 \leq k < k_2$. Thus, by the triangle inequality, we have $\epsilon < \|x_{k_1} - x_{k_2}\| \leq \sum_{k=k_1}^{k_2-1} \|x_{k+1} - x_k\|$. However, for $k_1 \in \overline{K}$ sufficiently large, (3.15) implies that the right-hand side of this inequality must be strictly less than ϵ (a contradiction).

We have shown in Case 2 that (3.14) holds. Then, since for all k we have

$$\mathcal{B}_{\epsilon,k} \subset \{\mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^f, \mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^{c^1}, \ldots, \mathbb{B}_\epsilon(x_k) \cap \mathcal{D}^{c^m}\},$$

(3.14) and $\epsilon \to 0$ imply that the cluster point $\overline{x}$ is stationary for ϕ_{ρ_*}. Moreover, by the construction of the parameter update step 4 in Algorithm 2.1 and the discussion in the second paragraph of section 2.2, we have the existence of an infinite subsequence of iterations K yielding the remainder of the result for part (a) of the theorem.

The proof of part (b) of the theorem follows similarly to the proof of part (a). Let ρ_k be the value of the penalty parameter ρ during the line search of iteration k. Similar to (3.9), we find

$$\begin{aligned} \phi_{\rho_k}(x_k) - \phi_{\rho_k}(x_{k+1}) &\geq \eta\alpha_k \Delta q_{\rho_k}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) \\ &\geq \tfrac{1}{2}\eta\alpha_k \underline{\xi}\|d_k\|^2 \\ &\geq \tfrac{1}{4}\eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k\|. \end{aligned} \tag{3.16}$$

Thus, similar to (3.10), we have the inequalities

$$\sum_{k=0}^{\infty} \alpha_k \Delta q_{\rho_k}(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) < \infty \quad \text{and} \tag{3.17a}$$

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\|\|d_k\| < \infty. \tag{3.17b}$$

Equation (3.17a) can also be written as

$$\sum_{k=0}^{\infty} \alpha_k (\Delta q_0(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) + \rho_k \max_{x\in\mathcal{B}^f_{\epsilon,k}} \{f(x_k) + \nabla f(x)^T d_k\}) < \infty,$$

from which it follows by Lemma 3.9 that under Assumption 3.2 we have

$$\sum_{k=0}^{\infty} \alpha_k \Delta q_0(d_k; x_k, \mathcal{B}_{\epsilon,k}, H_k) < \infty.$$

With this inequality and (3.17b) in hand, the remainder of the proof follows similarly to Cases 1 and 2 in the proof of part (a). However, instead of invoking Lemma 3.8 with $\overline{\rho} = \rho_*$, one invokes it with $\overline{\rho} = 0$ and k sufficiently large such that for some $\zeta > 0$ and with $x_k \in \mathbb{B}_\zeta(\overline{x})$ and $\rho \leq \zeta$, the set $\mathcal{T}$ is nonempty and implies inequalities similar to (3.12) and (3.13), where q_{ρ_*} is replaced with q_0. Consequently, we have that whenever $\rho \to 0$, any cluster point $\overline{x}$ of $\{x_k\}$ is stationary for $\phi_0 = v$. □

4. An implementation. We have implemented Algorithm 2.1[1] in MATLAB. We tested a few QP solvers for subproblem (2.6) and overall found that MOSEK[2]

[1] Publicly available at http://coral.ie.lehigh.edu/~frankecurtis/software.

[2] Available at http://www.mosek.com.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

produced the best results in terms of reliability and efficiency. In this section we describe some of the remaining details of our implementation. In particular, we discuss our technique for obtaining Lagrange multipliers λ_k to approximate an element of the (partial) Clarke ϵ-subdifferential of $L_\rho(\cdot, \lambda_k) := \rho f(x) + \lambda_k^T c(x)$ at x_k, outline a strategy for computing H_k and for reducing the number of sampled gradients required during each iteration, and comment on our choices for the input parameters.

Consistent with the GS strategy, we approximate the minimum-norm element of the (partial) Clarke ϵ-subdifferential of $L_\rho(\cdot, \lambda_k)$ at x_k with the gradients sampled for the problem functions. With

$$(\lambda_{k,0}^f, \ldots, \lambda_{k,p}^f), (\lambda_{k,0}^{c^1}, \ldots, \lambda_{k,p}^{c^1}), \ldots, (\lambda_{k,0}^{c^m}, \ldots, \lambda_{k,p}^{c^m})$$

defined as the optimal multipliers for the linearized constraints obtained by solving (2.6), we define

$$g_k := \sum_{i=1}^{p} \lambda_{k,i}^f \nabla f(x_{k,i}^f) + \sum_{j=1}^{m} \sum_{i=1}^{p} \lambda_{k,i}^{c^j} \nabla c(x_{k,i}^{c^j}).$$

The optimality conditions of (2.6) guarantee that $\sum_{i=1}^{p} \lambda_{k,i}^f = \rho$, so g_k approximates the minimum-norm element of $\overline{\partial}_x L_\rho(x_k, \lambda_k)$, where $\overline{\partial}_x f(x_k)$, $\overline{\partial}_x c^j(x_k)$ for $j = 1, \ldots, m$, and the components of λ_k are defined implicitly through the computed sample gradients and optimal multipliers for subproblem (2.6).

The vector g_k is used for two purposes in our implementation. First, it is used as part of the following optimality error estimate:

$$E_k := \max \left\{ \|g_k\|_\infty, \max_j \{c^j(x_k)\}, \max_{j,i} |\lambda_{k,i}^{c^j} c(x_{k,i}^{c^j})| \right\}.$$

Observing Theorem 2.2, this quantity provides a reasonable optimality measure for problem (1.1) when $\rho > 0$. Specifically, for a given value of the sampling radius ϵ, our implementation keeps track of the smallest value of E_k obtained. Note that this may be a better measure of optimality error than each individual E_k value as each is highly dependent on the set of sample points; a poor sampling set for a given iteration may yield a large value of E_k despite the fact that x_k is nearly ϵ_k-stationary.

Our second use for g_k relates to the choice of H_k made during each iteration. In an attempt to achieve fast convergence, we set this matrix for each iteration k as an L-BFGS [33] approximation of the Hessian of L_ρ at (x_k, λ_k) with λ_k defined as above. At the end of iteration k, we initialize $H \leftarrow I$ and then perform the k_H updates

$$H \leftarrow H - \frac{H s_l s_l^T H^T}{s_l^T H s_l} + \frac{y_l y_l^T}{s_l^T y_l} \quad \text{for} \quad l = k, k-1, \ldots, k - k_H + 1, \tag{4.1}$$

setting $H_{k+1} \leftarrow H$ as the final matrix obtained from this process. Here, $s_l := x_l - x_{l-1}$ and $y_l := g_l - g_{l-1}$ are the displacements in x and in the approximations of the minimum-norm elements of $\overline{\partial}_x L_\rho$ obtained in the most recent k_H iterations. (Note that these updates lag one iteration behind a traditional L-BFGS update as g_{k+1} is not obtained until (2.6) is solved during iteration $k+1$.) An update in (4.1) is performed if and only if

$$\|s_l\| \leq \chi_s \epsilon, \quad \|y_l\| \leq \chi_y \epsilon, \quad \text{and} \quad s_l^T y_l \geq \chi_{sy} \epsilon^2$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

TABLE 4.1
Input values for the static parameters for Algorithm 2.1.

Parameter	Value	Parameter	Value	Parameter	Value
p	2n	β_ρ	5e-1	χ_s	1e+3
η	1e-8	β_θ	8e-1	χ_y	1e+3
γ	5e-1	ν	1e+1	χ_{sy}	1e-6
β_ϵ	5e-1	k_H	1e+1		

TABLE 4.2
Initial values for the dynamic parameters for Algorithm 2.1.

Parameter	Initial Value
ϵ	1e-1
ρ	1e-1
θ	1e-1

for given constants $(\chi_s, \chi_y, \chi_{sy}) > 0$. As shown in [12], these restrictions guarantee that there exist constants $\overline{\xi} \geq \underline{\xi} > 0$ such that $\underline{\xi}\|d\|^2 \leq d^T H_k d \leq \overline{\xi}\|d\|^2$ for all k and $d \in \mathbb{R}^n$. This makes $\{H_k\}$ comply with Assumption 3.2, required for our convergence analysis. (We remark, however, that in [31] the authors argue that BFGS updates can be effective for unconstrained nonsmooth optimization when the Hessian approximations are allowed to become ill-conditioned.)

A remark is necessary here with this choice of H_k. That is, it should be noted that changes in the penalty parameter value will have an effect on (4.1) in a manner that is difficult to quantify. Specifically, if ρ at the start of iteration $k-1$ differs from ρ at the start of iteration k, then g_{k-1} and g_k in the computation of y_k will have been computed with different values of ρ. The update (4.1), however, makes no distinction between this case and that case when ρ remains constant between iterations $k-1$ and k. In our implementation, we simply disregard this occurrence and update the Hessian according to (4.1) no matter if ρ has changed.

Another important feature that has been implemented in our code is a special handling of functions that are known to be smooth or that depend on a number of variables less than n. In the former case, if a function is known to be continuously differentiable everywhere in $\mathbb{R}^n$, then we conjecture that it is not necessary to sample its gradient at nearby points. Specifically, if the objective (jth constraint) is known to be smooth, then we set $\mathcal{B}^f_{\epsilon,k} = \{x_k\}$ ($\mathcal{B}^{c^j}_{\epsilon,k} = \{x_k\}$) for all k. This choice can have a significant impact on the practical performance of the algorithm as this effectively eliminates pn_s linear inequality constraints from the quadratic program (2.6), where n_s is the number of smooth functions; otherwise, the subproblems will always have $p(m+1)$ such constraints in addition to the m bounds on the auxiliary variables. As for functions that depend on fewer than n, say, $\widetilde{n}$, variables, for such functions we sample only $2\widetilde{n}$ points with the idea that the gradients essentially live in a space of dimension $\widetilde{n}$. Again, this choice reduces the sizes of the subproblems.

We use values for the static input parameters as indicated in Table 4.1, as we found them to yield the best results for the experiments in section 5. Initial values for the dynamic parameters are provided in Table 4.2. A sample size of $p = n+1$ is all that is required in our analysis, but as in [8], we found that the choice $p = 2n$ yielded faster convergence in terms of the number of nonlinear iterations required to yield a good result. The chosen values for η and γ are standard for line search methods. The remaining values were decided upon during the course of our experiments.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

5. Numerical experiments. In this section we discuss the results of our MATLAB implementation of Algorithm 2.1 applied to a set of test problems. The first problem is somewhat contrived but is still an interesting illustrative example in that it involves the minimization of a classically difficult type of objective function. All the remaining problems are derived from real applications. We remark that in a few cases, special-purpose solvers designed for the individual applications will most likely provide better results than are presented here, but it should be noted that ours is only a preliminary implementation and, in fact, there may be ways in which our approach can be tailored to each problem individually; see section 6. Thus, for our purposes here, we simply apply our general-purpose implementation of Algorithm 2.1 and illustrate that it is effective on a wide range of problems.

For each problem we ran our implementation with 10 different starting points, each sampled from a standard multivariate normal distribution. Because of the stochastic nature of the algorithm itself, we could have run the algorithm from the *same* starting point multiple times, expecting to find variations in the performance over each run. However, since we found that the algorithm behaved consistently even for different starting points, we provide the results for these runs with the idea that they more clearly indicate the robustness of the approach. In all cases we imposed a maximum iteration limit of 500. Our code includes an optimality check that terminates the solver when a sufficiently feasible ϵ-stationary point has been found (for any user-defined $\epsilon > 0$), but in our tests we turned this feature off and simply let the code run until the iteration limit was reached, allowing ϵ to decrease indefinitely.

Example 5.1. Find the minimizer of a nonsmooth Rosenbrock function subject to an inequality constraint on a weighted maximum value of the variables:

$$\min_{x_1,x_2} \; w|x_1^2 - x_2| + (1 - x_1)^2 \;\; \text{s.t.} \;\; \max\{c_1 x_1, c_2 x_2\} \leq 1. \tag{5.1}$$

The Rosenbrock function is a standard smooth, nonconvex optimization test problem, and so the nonsmooth variation in problem (5.1) is an interesting one to consider for our method. Defining the objective weight to be $w = 8$ and the coefficients to be $(c_1, c_2) = (\sqrt{2}, 2)$, we illustrate this problem in Figure 5.1. The curved contours of the objective illustrate that the unconstrained minimizer lies at $(1, 1)$. The feasible region, however, is the rectangle including the lower left-hand corner of the graph, so the solution of the constrained problem is $x_* = (\frac{\sqrt{2}}{2}, \frac{1}{2})$, a point at which both the objective and the constraint function are nondifferentiable. If the initial iterate is located in the upper left-hand corner of the graph, then the algorithm essentially needs to trace the classical curved valley to the solution with the added restriction that it must find the optimal solution located in a corner of the nonsmooth constraint.

In all our runs, Algorithm 2.1 converged rapidly to x_*. On the left-hand side of Figure 5.2, we plot $\|x_k - x_*\|$ with respect to k and note that in all cases this distance rapidly decreased to `1e-10`. The final penalty parameter value (ρ) was `1e-01` and the final sampling radius (ϵ) was approximately `6e-09` in all runs. Finally, we remark that over all runs the median optimality error estimate (E_{500}) was approximately `2e-12`.

As a brief numerical illustration of the difficulty of our instance of Example 5.1, we also plotted in Figure 5.2 (on the right-hand side) the results of a second set of experiments for the same set of starting points, this time obtained *without* sampling any gradients. That is, in these runs we ran Algorithm 2.1 with $p = 0$, violating the restriction imposed in our analysis that the number of sample points must be at least $n + 1$. It is clear in the plots of Figure 5.2 that the sampling of a sufficient number of gradients is necessary for our algorithm to be robust. (In these runs,

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

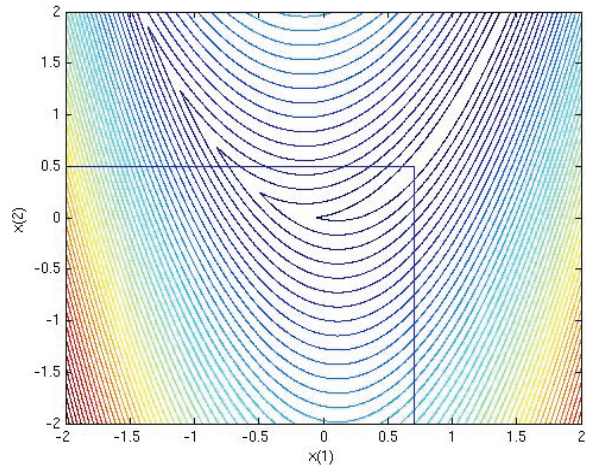

FIG. 5.1. *Illustration of objective contours and the feasible region for an instance of Example* 5.1.

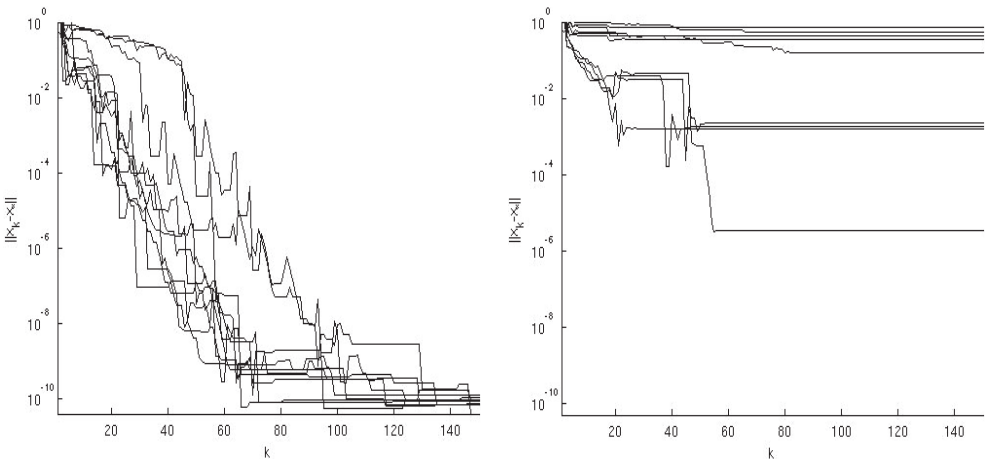

FIG. 5.2. *Plots of* $\|x_k - x_*\|$ *with respect to* k *for two sets of* 10 *runs of Algorithm* 2.1 *applied to the instance of Example* 5.1 *illustrated in Figure* 5.1. *The plots on the left-hand side are the results obtained with our implementation of Algorithm* 2.1 *and the plots on the right-hand side are the results obtained with the same implementation, but with* $p = 0$ *so that GS is turned off. The lack of sampling in the latter set of experiments causes the algorithm to perform poorly.*

the final penalty parameter (ρ) was always `1e-01`, the final sampling radius (ϵ) was always around `2e-03`, and the median optimality error estimate (E_{500}) was `8e-01`.) In fact, similarly poor results were obtained for all our remaining test problems when no sampling was performed (i.e., when $p = 0$), providing strong evidence that the GS procedure is critical for the effectiveness of our approach.

Example 5.2. Find an $N \times N$ matrix X with normalized columns so that the product of the K largest eigenvalues of the matrix $A \circ X^T X$ is minimized:

$$\min_X \ \ln\left(\prod_{j=1}^{K} \lambda_j(A \circ X^T X)\right) \quad \text{s.t. } \|X_j\|^2 = 1,\ j = 1, \ldots, N, \tag{5.2}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

TABLE 5.1
Results for Example 5.2 *for various values of* N.

N	K	n	f_{500}	$v(x_{500})$	ϵ	E_{500}
2	1	4	1.0000e+00	3.1752e-14	5.9605e-09	7.6722e-12
4	2	16	7.4630e-01	2.8441e-07	4.8828e-05	1.1938e-04
6	3	36	6.3359e-01	2.1149e-06	9.7656e-05	8.7263e-02
8	4	64	5.5832e-01	2.0492e-05	9.7656e-05	2.7521e-03
10	5	100	2.1841e-01	9.8364e-06	7.8125e-04	9.6041e-03
12	6	144	1.2265e-01	1.8341e-04	7.8125e-04	6.0492e-03
14	7	196	8.4650e-02	1.6692e-04	7.8125e-04	7.1461e-03
16	8	256	6.5051e-02	6.4628e-04	1.5625e-03	3.1596e-03

where $\lambda_j(M)$ is the jth largest eigenvalue of M, A is a real symmetric $N \times N$ matrix, $\circ$ denotes the Hadamard matrix product, and X_j is the jth column of X.

Example 5.2 is a nonconvex relaxation of an entropy minimization problem arising in an environmental application [1]. We remark that this problem is one of the examples in [8], where the variables were defined as the off-diagonal entries in $M = X^T X$ and M was forced to be positive semidefinite through a penalty term in the objective, and in [31], where the constraint is enforced through a redefinition of the objective function. In our case, such reformulations are unnecessary as we can simply impose constraints requiring that the columns of X are normalized.

Table 5.1 shows the results of Algorithm 2.1 applied to problem (5.2) for various values of N, where in each case we choose $K = N/2$. We set A to be the $N \times N$ leading submatrix of a 63×63 covariance data matrix.[3] The data we provide are the number of optimization variables (n) and the final values obtained in the run yielding the best optimality error estimate for the smallest final sampling radius. Specifically, for that run we provide the final objective value (f_{500}), infeasibility measure value ($v(x_{500})$), sampling radius (ϵ), and optimality error estimate (E_{500}) corresponding to the final sampling radius. The values are comparable to those obtained in [8].

Example 5.3. Find the minimum ℓ_q-norm vector that approximates the solution of a linear system insofar as the residual is within a given tolerance:

$$\min_x \ \|x\|_q \ \text{ s.t. } \|Rx - y\| \leq \delta, \tag{5.3}$$

where $q > 0$, $R \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^m$, and $\delta \geq 0$ is an error tolerance.

Example 5.3 is a problem of interest in compressed sensing, where the goal is to reconstruct a sparse vector x_* of length n from $m < n$ observations. In our test, we choose $m = 32$ and $n = 256$ and generate R by taking the first m rows of the $n \times n$ discrete cosine transform matrix. The vector we aim to recover, x_*, is chosen to take entries equal to 0 with probability 0.90 and entries that have a standard normal distribution with probability 0.1. The vector x_* that we generated has 19 nonzero entries; see the left-hand side of Figure 5.3.

We consider four instances of problem (5.3), each with the same R and x_*. In the first two instances, $y = Rx_*$ and $\delta = 0$, but two different values of q are chosen. First, for $q = 1$, we have the common l_1-norm minimization problem; e.g., see [10, 13]. On the right-hand side of Figure 5.3 we plot the best solution obtained by Algorithm 2.1. (Note that for this and all other instances of problem (5.3), we define the best solution as that yielding the lowest objective value.) The final feasibility violation ($v(x_{500})$), sampling radius (ϵ), and optimality error estimate (E_{500}) corre-

[3] Publicly available at http://www.cs.nyu.edu/faculty/overton/papers/gradsamp/probs.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

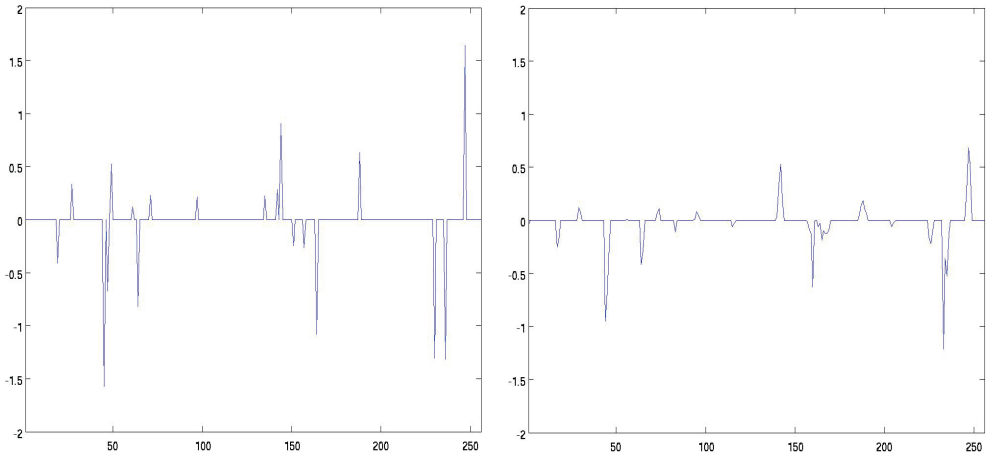

FIG. 5.3. *x_* (left) and x_{500} (right) for Example 5.3 with $q = 1$ and exact data.*

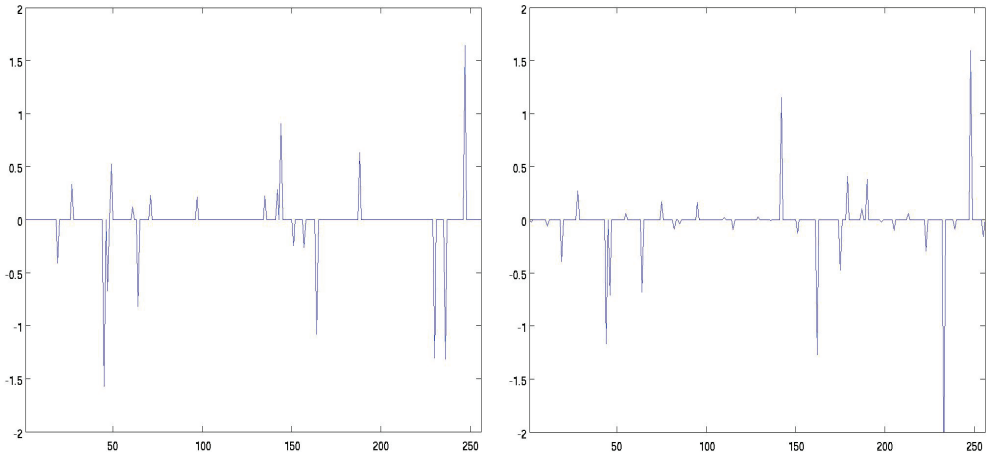

FIG. 5.4. *x_* (left) and x_{500} (right) for Example 5.3 with $q = 0.5$ and exact data.*

sponding to this solution were `2e-12`, `4e-04`, and `1e-03`, respectively. It can be seen in Figure 5.3 that the solution vector x_* is not recovered exactly with this choice of q. Indeed, defining a nonzero entry as any having an absolute value greater than `1e-4`, the illustrated solution has 66 nonzero entries. Thus, in an attempt to steer the algorithm toward sparser solutions, our second set of runs was performed with $q = 0.5$; again, see [10, 13]. In this case, the best solution obtained by Algorithm 2.1 is plotted along with x_* in Figure 5.4. The final feasibility violation, sampling radius, and optimality error estimate corresponding to this solution were `3e-05`, `5e-05`, and `8e-01`, respectively. Note that with $q < 1$, the objective is nonconvex, with multiple local minimizers, and non-Lipschitz. However, we do consistently find that a solution with few nonzero entries is found; e.g., the solution in Figure 5.4 has only 32.

In our other two instances of problem (5.3), $y = Rx_* + e$, where e is a noise vector with entry e_j distributed uniformly in $[R_j x_* - 0.005|R_j x_*|, R_j x_* + 0.005|R_j x_*|]$; i.e., there is up to a 0.5% difference (componentwise) between y and Rx_*. We assume that this error level is overestimated and so choose $\delta = 2\|Rx_* - y\|$. For $q = 1$ and

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

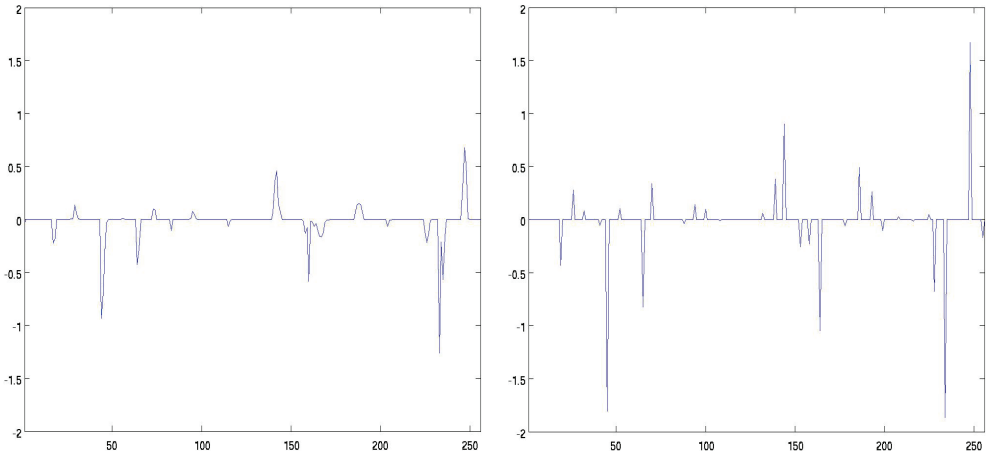

FIG. 5.5. x_{500} *when* $q = 1.0$ *(left) and* x_{500} *when* $q = 0.5$ *(right) for Example* 5.3 *with noisy data.*

$q = 0.5$, respectively, the best solutions obtained are plotted in Figure 5.5. For $q = 1$, the illustrated solution has 134 nonzero entries with $v(x_{500})$, ϵ, and E_{500} equal to approximately `0e-00`, `2e-03`, and `2e-03`, respectively. For $q = 0.5$, the illustrated solution has only 29 nonzero entries with $v(x_{500})$, ϵ, and E_{500} equal to approximately `0e-00`, `6e-06`, and `2e-01`, respectively. Again, note that with $q < 1$, the objective is nonconvex, with multiple local minimizers, and non-Lipschitz.

Example 5.4. Find the robust minimizer of a linear objective function subject to an uncertain convex quadratic constraint:

$$\min_x \ f^T x \ \text{ s.t. } x^T A x + b^T x + c \leq 0 \ \forall (A, b, c) \in \mathcal{U}, \tag{5.4}$$

where $f \in \mathbb{R}^n$ and for each (A, b, c) in the uncertainty set $\mathcal{U}$, A is an $n \times n$ positive semidefinite matrix, b is a vector in $\mathbb{R}^n$, and c is a real scalar value.

Example 5.4 is a problem of interest in robust optimization. In this modeling framework, the primary concern is that the parameters defining an optimization problem are in practice estimated and are therefore subject to measurement and statistical errors. Since the solution is typically sensitive to perturbations in these parameters, it is important to take data uncertainty into account in the optimization process. For instance, in problem (5.4), we require that any solution vector x_* satisfy the convex quadratic constraint no matter which instance in the uncertainty set $\mathcal{U}$ is realized. Problems of this type have applications in, for example, robust mean-variance portfolio selection [18], least-squares problems [2], and data-fitting [43].

We generated random data for an instance of problem (5.4) with $n = 100$ unknowns. The uncertainty set was defined to be

$$\mathcal{U} := \left\{ (A, b, c) : (A, b, c) = (A^{(0)}, b^{(0)}, c^{(0)}) + \sum_{i=1}^{10} u^i (A^{(i)}, b^{(i)}, c^{(i)}), \ u^T u \leq 1 \right\},$$

where $A^{(i)}$, $i = 0, \ldots, 10$, were generated randomly using MATLAB's built-in `sprandsym` function. In particular, $A^{(0)}$ was a symmetric 100×100 matrix with approximately $0.1n^2$ nonzero entries, generated by a shifted sum of outer products so that the condition number was approximately 10 (i.e., $A^{(0)}$ was set by

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

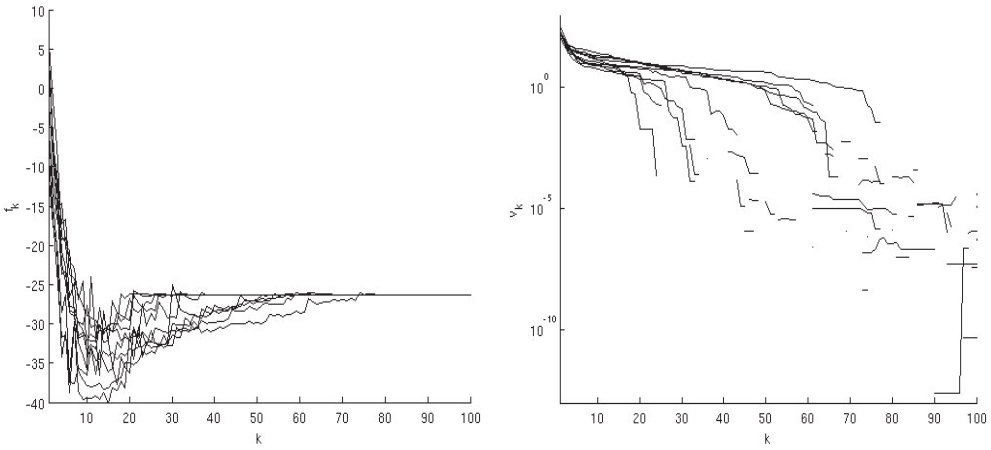

FIG. 5.6. *Plots of $f(x_k)$ (left, linear scale) and $v(x_k)$ (right, log scale) with respect to k for* 10 *runs of Algorithm* 2.1 *on an instance of Example* 5.4.

`sprandsym(100,0.1,0.1,2)`), and $A^{(i)}$ for $i = 1, \ldots, 10$ was generated in the same way but multiplied by the scalar factor 0.01. The entries of all the $b^{(i)}$ vectors had a standard normal distribution. Finally, $c^{(0)} = -10$, while $c^{(i)}$ for $i = 1, \ldots, 10$ were uniformly distributed on $[-1, 1]$. The feasible region was verified to be nonempty as it included the origin. At a given x_k, the corresponding u_k in $\mathcal{U}$ was computed by minimizing a linear function over the closed unit ball, with which the constraint value and gradient was obtained. We remark that as described in [2], this instance can be reformulated and solved as an equivalent semidefinite program. However, it is worth noting that our algorithm can also be applied to more general robust nonlinear programs that do not necessarily have convenient reformulations.

Algorithm 2.1 converges rapidly from all starting points on this instance of problem (5.4). In Figure 5.6, on the left-hand side we plot the objective function value $f(x_k)$ with respect to k (on a linear scale), and on the right-hand side we plot the feasibility violation measure $v(x_k)$ with respect to k (on a log scale). In all instances, the function values converge to the optimal value and the violation measures converge to zero (or around at most `1e-5`) comfortably in under 100 iterations. (Note that in the plot of violation measures, gaps are found when the violation dropped to exactly zero.) The median final penalty parameter value (ρ) was `1e-01`, the median final sampling radius (ϵ) was `6e-07`, and the median final optimality error estimate (E_{500}) was `2e-08`. These results indicate that Algorithm 2.1 can successfully solve robust optimization problems along the lines of Example 5.4.

6. Conclusion. We have proposed and analyzed an algorithm for nonconvex, nonsmooth constrained optimization. The method is based on an SQP framework where a GS technique is used to create augmented linear models of the objective and constraint functions around the current iterate. We have shown that if the problem functions are locally Lipschitz and continuously differentiable almost everywhere in $\mathbb{R}^n$, then the algorithm successfully locates stationary points of a penalty function. Preliminary numerical experiments illustrate that the algorithm is effective and applicable to numerous types of problems.

We believe that our approach can be fine-tuned for individual applications. For example, for Example 5.3 with $\delta = 0$, it would be ideal if at any feasible iterate the

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

search direction is computed in the null space of R. Moreover, in such cases, one may consider sampling gradients only at points in this space, and one may find that fewer points are needed overall; e.g., $n - m + 1$ as opposed to $n + 1$. More generally, if $f, c^1, \ldots, c^m$ are convex, then subproblem (2.6) may be replaced by

$$\begin{aligned} &\min_{d,z} \; z + \tfrac{1}{2} d^T H_k d \\ &\text{s.t.} \begin{cases} f(x_k) + \nabla f(x)^T d \le z \;\forall x \in \mathcal{B}^f_{\epsilon,k}, \\ c^j(x_k) + \nabla c^j(x)^T d \le 0 \;\forall x \in \mathcal{B}^{c^j}_{\epsilon,k}, \;\; j = 1, \ldots, m \end{cases} \end{aligned} \tag{6.1}$$

(i.e., the auxiliary variables r may be fixed to zero), after which the penalty parameter can be decreased, if necessary, to ensure that the computed search direction is one of descent for the penalty function. The main idea here is to reduce the size of the QP subproblem, as its solution is the main computational expense of the approach.

Finally, we remark that the SQP-GS algorithm discussed in this paper can easily be altered into a sequential linear programming algorithm with GS (SLP-GS). In particular, the quadratic term in (2.6) can be replaced by a trust region constraint on the search direction, yielding the linear programming subproblem

$$\begin{aligned} &\min_{d,z,r} \; \rho z + \sum_{j=1}^m r^j \\ &\text{s.t.} \begin{cases} f(x_k) + \nabla f(x)^T d \le z \;\forall x \in \mathcal{B}^f_{\epsilon,k}, \\ c^j(x_k) + \nabla c^j(x)^T d \le r^j \;\forall x \in \mathcal{B}^{c^j}_{\epsilon,k}, \;\; r^j \ge 0, \;\; j = 1, \ldots, m, \\ \|d\|_\infty \le \Delta_k, \end{cases} \end{aligned} \tag{6.2}$$

where $\Delta_k > 0$ is a trust region radius. Our MATLAB implementation contains an option for running SLP-GS instead of SQP-GS. However, although the computation time per iteration will generally be less for SLP-GS, the rate of convergence is typically much slower when compared to SQP-GS. There are also a variety of details that must be worked out to prove the convergence of such an algorithm, in particular with respect to the value and updates for the trust region radius.

Acknowledgments. The authors would like to thank Adrian S. Lewis and the anonymous referees for many useful comments and suggestions that significantly improved the paper. They are also indebted to Krzysztof C. Kiwiel, whose careful, generous, and insightful comments helped to improve the convergence analysis.

REFERENCES

[1] K. M. Anstreicher and J. Lee, *A masked spectral bound for maximum-entropy sampling*, in MODA 7: Advances in Model-Oriented Design and Analysis, A. di Bucchianico, H. Läuter, and H. P. Wynn, eds., Springer-Verlag, Berlin, 2004, pp. 1–10.

[2] A. Ben-Tal and A. Nemirovski, *Robust optimization: Methodology and applications*, Math. Program. Ser. B, 92 (2002), pp. 453–480.

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

[4] J. V. Burke, *An exact penalization viewpoint of constrained optimization*, SIAM J. Control Optim., 29 (1991), pp. 968–998.

[5] J. V. Burke, *Calmness and exact penalization*, SIAM J. Control Optim., 29 (1991), pp. 493–497.

[6] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton, *HIFOO: A MATLAB package for fixed-order controller design and H-infinity optimization*, in Proceedings of the IFAC Symposium on Robust Control Design, Haifa, Israel, 2006.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

[7] J. V. Burke, A. S. Lewis, and M. L. Overton, *Approximating subdifferentials by random sampling of gradients*, Math. Oper. Res., 27 (2002), pp. 567–584.
[8] J. V. Burke, A. S. Lewis, and M. L. Overton, *A robust gradient sampling algorithm for nonsmooth, nonconvex optimization*, SIAM J. Optim., 15 (2005), pp. 751–779.
[9] R. H. Byrd, G. Lopez-Calva, and J. Nocedal, *A line search exact penalty method using steering rules*, Math. Program., DOI: 10.1007/s10107-010-0408-0.
[10] E. J. Candès and T. Tao, *Near optimal signal recovery from random projections: Universal encoding strategies*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5406–5425.
[11] F. H. Clarke, *Optimization and Nonsmooth Analysis*, CMS Ser. Monogr. Adv. Texts, John Wiley, New York, 1983.
[12] F. E. Curtis and X. Que, *An adaptive gradient sampling algorithm for nonsmooth optimization*, Lehigh ISE/COR@L Technical Report 11T-008, Optim. Methods Softw., in review.
[13] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.
[14] D. Dotta, A. S. De Silva, and I. C. Decker, *Design of power system controllers by nonsmooth, nonconvex optimization*, in Proceedings of the IEEE Power and Energy Society General Meeting, Calgary, Alberta, Calgary, 2009.
[15] R. Fletcher, *Practical Methods of Optimization*, John Wiley, New York, 1987.
[16] M. Fukushima, *A successive quadratic programming method for a class of constrained nonsmooth optimization problems*, Math. Program., 49 (1990), pp. 231–251.
[17] P. E. Gill and E. Wong, *Sequential quadratic programming methods*, in Mixed Integer Nonlinear Programming, J. Lee and S. Leyffer, eds., IMA Vol. Math. Appl. 154, Springer-Verlag, New York, 2012, pp. 147–224.
[18] D. Goldfarb and G. Iyengar, *Robust portfolio selection problems*, Math. Oper. Res., 28 (2003), pp. 1–38.
[19] S. Guenter, M. Sempf, P. Merkel, E. Strumberger, and C. Tichmann, *Robust control of resistive wall modes using pseudospectra*, New J. Phys., 11 (2009), pp. 1–40.
[20] S. Gumussoy, D. Henrion, M. Millstone, and M. L. Overton, *Multiobjective robust control with HIFOO* 2.0, in Proceedings of the IFAC Symposium on Robust Control Design, Haifa, Israel, 2009.
[21] S. P. Han, *A globally convergent method for nonlinear programming*, J. Optim. Theory Appl., 22 (1977), pp. 297–309.
[22] S. P. Han and O. L. Mangasarian, *Exact penalty functions in nonlinear programming*, Math. Program., 17 (1979), pp. 251–269.
[23] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms* II, Grundlehren Math. Wiss., Springer-Verlag, New York, 1993.
[24] E. W. Karas, A. Ribeiro, C. A. Sagastizábal, and M. V. Solodov, *A bundle-filter method for nonsmooth convex constrained optimization*, Math. Program., 116 (2007), pp. 297–320.
[25] N. Karmitsa and M. M. Makela, *Limited memory bundle method for large bound constrained nonsmooth optimization: Convergence analysis*, Optim. Methods Softw., 25 (2010), pp. 895–916.
[26] K. C. Kiwiel, *An exact penalty function algorithm for non-smooth convex constrained minimization problems*, IMA J. Numer. Anal., 5 (1985), pp. 111–119.
[27] K. C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Math. 1133, Springer-Verlag, New York, 1985.
[28] K. C. Kiwiel, *A constraint linearization method for nondifferentiable convex minimization*, Numer Math., 51 (1987), pp. 395–414.
[29] K. C. Kiwiel, *Exact penalty functions in proximal bundle methods for constrained convex nondifferentiable minimization*, Math. Program., 52 (1991), pp. 285–302.
[30] K. C. Kiwiel, *Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization*, SIAM J. Optim., 18 (2007), pp. 379–388.
[31] A. S. Lewis and M. L. Overton, *Nonsmooth optimization via quasi-Newton methods*, Math. Program., DOI 10.1007/s10107-012-0514-2.
[32] R. Mifflin, *An algorithm for constrained optimization with semismooth functions*, Math. Oper. Res., 2 (1977), pp. 191–207.
[33] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comput., 35 (1980), pp. 773–782.
[34] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer Ser. Oper. Res., New York, 2006.
[35] E. Polak, D. Q. Mayne, and Y. Wardi, *On the extension of constrained optimization algorithms from differentiable to nondifferentiable problems*, SIAM J. Control Optim., 21 (1983), pp. 179–203.
[36] M. J. D. Powell, *A fast algorithm for nonlinearly constrained optimization calculations*, in

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Numerical Analysis, Lecture Notes in Math. 630, Springer, New York, 1978, pp. 144–157.
[37] M. J. D. Powell, *Variable metric methods for constrained optimization*, in Mathematical Programming: The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer-Verlag, New York, 1983, pp. 288–311.
[38] R. T. Rockafellar, *Convex Analysis*, Princeton Landmarks Math. Phys., Princeton University Press, Princeton, NJ, 1970.
[39] R. T. Rockafellar, *Lagrange multipliers and subderivatives of optimal value functions in nonlinear programming*, in Math. Program. Study 17, 1982, pp. 28–66.
[40] E. Rosenberg, *Exact penalty functions and stability in locally Lipschitz programming*, Math. Program., 30 (1984), pp. 340–356.
[41] A. Ruszczynski, *Nonlinear Optimization*, Princeton University Press, Princeton, NJ, 2006.
[42] J. Vanbiervliet, K. Verheyden, W. Michiels, and S. Vandewalle, *A nonsmooth optimisation approach for the stabilisation of time-delay systems*, ESAIM Control Optim. Calc. Var., 14 (2008), pp. 478–493.
[43] G. A. Watson, *Data fitting problems with bounded uncertainties in the data*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1274–1293.

Downloaded 07/17/13 to 128.122.253.212. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.