A Pattern Search Filter Method for Nonlinear Programming without Derivatives

Charles Audet
J.E. Dennis Jr.

{charlesa, dennis}@caam.rice.edu
Rice University
Department of Computational
and Applied Mathematics
6100 South Main Street - MS 134
Houston, Texas, 77005-1892 USA

March 3, 2000

Abstract: This paper presents and analyzes a pattern search method for general constrained optimization based on filter methods for step acceptance. Roughly, a filter method accepts a step that either improves the objective function value or the value of some function that measures the constraint violation. The new algorithm does not compute or approximate any derivatives, penalty constants or Lagrange multipliers. It reduces trivially to the Torczon GPS (generalized pattern search) algorithm when there are no constraints, and indeed, it is formulated here to reduce to the version of GPS designed to handle finitely many linear constraints if they are treated explicitly. A key feature is that it preserves the useful division into SEARCH and POLL steps. Assuming local smoothness, the algorithm produces a KKT point for a problem related to the original problem.

Key words Pattern search algorithm, filter algorithm, surrogate-based optimization, derivative-free convergence analysis, constrained optimization, nonlinear programming.

Acknowledgments: Work of the first author was supported by NSERC (Natural Sciences and Engineering Research Council) fellowship PDF-207432-1998 and both authors were supported by DOE DE-FG03-95ER25257, AFOSR F49620-98-1-0267, The Boeing Company, Sandia LG-4253, Exxon/Mobil and CRPC CCR-9120008.

1 Introduction

The generalized pattern search algorithm class (GPS) designed by Torczon [14] unify a wide class of useful derivative-free algorithms for unconstrained optimization. Lewis and Torczon extended the GPS framework to bound constrained optimization [10] and more generally for problems with a finite number of linear constraints [11]. Audet and Dennis [1] identify a specific set of limit points that allow the application of Clarke's [3] generalized derivatives[1] to strengthen and simplify the Lewis and Torczon theory. No more smoothness is needed for the general nonlinear programming algorithm presented here.

Assuming continuous differentiability, Lewis and Torczon also gave a satisfying theoretical treatment of a derivative-free procedure to handle general constraints [12]. In their procedure, GPS for bound constraints is used to carry out the specified inexact minimizations of the sequence of augmented Lagrangian subproblems formulated by Conn, Gould, and Toint [5]. Our algorithm is a direct GPS alternative to their method, but we do not claim that it is to be preferred for every problem.

We extend the unconstrained GPS algorithm to general nonlinear programming problems in a way that reduces transparently to GPS for unconstrained problems or for problems where a subset of the constraints consisting of a finite number of linear constraints are to be treated directly by rejecting infeasible points for those constraints. We do this without the need for any penalty constants or Lagrange multiplier estimates by formulating a step acceptance rule based on filter methods. This is a contribution to pattern search methods, but our main interest is in using our algorithm as a meta algorithm in the surrogate management framework for general nonlinear programming [2]. Thus, a key issue for us is that we preserve the division into SEARCH and POLL steps of the statement of GPS given in [2]. Another key issue for us is to make assumptions on the problem functions that conform to the instances we meet in practice. They may be discontinuous and even take on infinite values.

Filter algorithms were introduced by Fletcher and Leyffer [6] as a way to globalize SQP and SLP without using any merit function that would require a troublesome parameter to be provided by the user for weighting the relative merits of improving feasibility and optimality. A filter algorithm introduces a function that aggregates constraint violations and then treats the resulting biobjective problem. A step is accepted if it either reduces the objective function or the constraint violation. Although this clearly is less parameter dependent than a penalty function, still we acknowledge that specifying a constraint violation function implies assigning relative weights to reducing each constraint.

Fletcher, Leyffer and Toint [7] show convergence of the filter method that uses sequential linear programming to suggest steps. Fletcher, Gould, Leyffer and Toint [8] show convergence of the filter method that uses sequential quadratic programming to suggest steps. In both cases, convergence is to a point satisfying Fritz John optimality conditions [9]. Thus, previous filter algorithms require explicit use of the derivatives of both the objective and the

[1] An appendix on generalized derivatives is included for the convenience of the reader.

constraints. They also require more than simple decrease to accept a step.

We suggest a derivative-free pattern search version of the filter approach that converges using any user-defined finite SEARCH procedure to suggest steps, as long as that is successful, and falls back on a special POLL step when the user defined procedure stalls. Our algorithms preserves the desirable GPS property of requiring only simple decrease.

Pattern search methods are appropriate when some of the functions defining the problem are given as black boxes that do not guarantee enough accuracy to approximate derivatives, or, when the user would rather not find only the nearby local optimizer found by derivative-based methods, but instead is willing to use some function values to explore the domain more thoroughly. We use the term "semi-global", rather than "global", for this common situation because for a black box function global optimization is impossible, even to the point that if we had the global optimum we could not know it [13]. A feature of our algorithm useful for semi-global optimization is that it will move away from most constrained local optimizers.

Pattern search methods are generally useful for both these cases, and they are supported by a growing body of theory. In the papers cited above, Torczon showed that GPS methods produce a limit point for which the gradient of the objective function is zero, and Lewis and Torczon showed that their adaptations produce a Karush-Kuhn-Tucker point for bound constrained and linearly constrained problems under standard constraint qualifications. For general constraints, we will not be able to prove such strong results, but we do not require derivatives to define subproblems, and our approach is suited to the use of surrogates that do not match derivatives.

We show that our pattern search filter class of algorithms guarantees subsequence convergence to points, which, if the functions are locally smooth, are first-order critical points. We identify a perturbed problem for which such a limit point satisfies KKT conditions. Without going into the technical aspects of this KKT result here, the generators of the cone in which we can guarantee the negative objective gradient to lie depend on the search directions we use. Since we have no derivatives to guide our choice of pattern search directions, our results seem appropriate. We do not use Lagrange multipliers or linearized constraints. Thus, we do not need any constraint qualifications.

The optimization problem considered in this paper is

$$\begin{array}{ll} \min\limits_{x \in \Re^n} & f(x) \\ \text{s.t.} & C(x) \leq 0 \end{array}$$

where $f : \Re^n \to \Re \cup \{\infty\}$ and $C : \Re^n \to (\Re \cup \{\infty\})^m$ are functions with $C = (c_1, \ldots, c_m)^T$. The feasible region is denoted by Ω.

The paper is divided as follows. In Sections 2 and 3 we present a brief description of pattern search and filter algorithms. Then, in Section 4 we present and begin the analysis of a new algorithm that combines their features. Specifically, without any smoothness assumptions on the problem, we show the existence of some promising limit points. In Section 5, we show that if the constraint violation is a locally smooth function at such a limit point,

then it satisfies first order optimality conditions, and if the objective is locally smooth there, then we provide a perturbed KKT result.

In Section 6, we make three main points through illustrative examples. We show the value of a filter method as opposed to a "barrier" method, which sets $f(x) = \infty$ if $x \notin \Omega$, [11], [1]. We show the advantages for our algorithm of a squared ℓ_2 over an ℓ_1 measure of constraint violations, and we show that our KKT result is tight. These examples show the strength and limitations of our approach. We do not give any of the numerical results we have generated because all they show is that the method is satisfactorily robust, and that indeed ℓ_2 is more robust than ℓ_1 to measure infeasibility in this context. In our approach, efficiency is the job of the SEARCH procedure.

2 Pattern search algorithms

Pattern search algorithms for unconstrained minimization generate a sequence of iterates $\{x_k\}$ in $\Re^n$ with nonincreasing objective function values. At each iteration, the objective function is evaluated at a finite number of points on a mesh (a discrete subset of $\Re^n$ defined below) to try to find one that yields a lower objective function value than the feasible incumbent. Any strategy may be used to select the mesh points that are to be candidates for the next iteration, as long as only a finite number of points (possibly none) are selected. This is called the SEARCH step. Before declaring the iteration unsuccessful, refining the mesh, and setting $x_{k+1} = x_k$, it is required that the neighboring mesh points be POLLED to see if any one yields a lower function value. Only after a failed poll of the neighbors can an iteration be declared unsuccessful. The situation for our constrained version is going to be a bit more complex, but it is consistent in spirit.

The reader may wonder why we do not simply include the set of neighbors in the POLL step in the SEARCH step. The reason is that this would require every SEARCH step to compute $\mathcal{O}(n)$ function values, regardless of how cleverly the user might define it, and our experience is that our users understand their problems and implement clever searches, which greatly enhance the quality of the final solution and the efficiency of the overall algorithm.

If the unconstrained iteration is successful, then the new point $x_{k+1} \neq x_k$ has a strictly lower objective function value, the mesh size parameter is kept the same or increased, and the process is reiterated. Indeed, as long as the SEARCH steps are succeeding, one would likely choose trial points on a coarser submesh than the current mesh. Our experience with surrogate-based SEARCH steps is that a great deal of progress can be made with few function values, and $\mathcal{O}(n)$ function values are needed only for unsuccessful POLL steps, which indicate that the mesh needs to be refined.

The following notation will be useful for the constrained algorithm to be given later. Let $\mathcal{S}$ be a finite set of positive spanning matrices in $\Re^n$, i.e., the non negative linear combinations of their columns span $\Re^n$. Moreover, for technical convergence reasons, assume that every column of each positive spanning matrix is be constructed from a single basis matrix and

from a finite set of integer generating matrices (see Torczon [14] for the formal definition of the basis and generating matrices). The current mesh M_k is defined through the lattices spanned by the matrices of $\mathcal{S}$: $M_k = \{x_k + \Delta_k S z : z \in \mathcal{Z}^{n_S}, S \in \mathcal{S}\}$, where $\Delta_k > 0$ is the mesh size parameter, and n_S is the number of columns of the matrix S.

Before declaring an iteration unsuccessful, the objective function must be evaluated at the mesh points that neighbor some selected point on the mesh. In the unconstrained case this *poll center* is x_k, the current iterate. However in the filter algorithm we will present, we may not poll about the current iterate, but instead we will poll about some possibly different p_k on its own current mesh. This defines the poll set $P_k = \{p_k + \Delta_k s : s \in S_k\}$ for some positive spanning matrix $S_k \in \mathcal{S}$, and where $s \in S_k$ signifies that s is a column of the matrix S_k (we will refer to $p_k + \Delta_k s$ as polling in the direction s). Thus the points of the poll set are the neighbors of p_k on its own current mesh.

If the iteration is unsuccessful, then the mesh is refined. More precisely, Δ_{k+1} is set to $\tau^{m_k^-}\Delta_k$ for $0 < \tau^{m_k^-} < 1$ where $\tau > 1$ is a rational number that remains constant over all iterations and $m_k^- \leq -1$ is an integer bounded below by $-m_{max} \leq 0$.

If the iteration is successful, then the mesh parameter is updated $\Delta_{k+1} \geq \Delta_k$. One may choose to coarsen the search to carry out far reaching and inexpensive SEARCH steps. In this case, one searches on a submesh coarsened by the rule $\tau^{m_k^+}\Delta_k$ for some $\tau^{m_k^+} \geq 1$ where $m_k^+ \geq 0$ is an integer bounded above by $m_{max} \geq 0$.

By modifying the mesh size parameters this way, it follows that for any $k \geq 0$, there exists an integer $r_k \in \mathcal{Z}$ such that $\Delta_k = \tau^{r_k}\Delta_0$, and the next iterate x_{k+1} can always be written as $x_0 + \sum_{i=1}^{k} \Delta_i S_i z_i$ for some z_i in $\mathcal{Z}^{n_{S_i}}$. This observation, together with the definition of the positive spanning matrices, will be essential to the proof of Theorem 4.2.

We denote by T_k the set of trial points that may be explored at iteration k. This set contains all the search points obtained by the search strategy as well as the poll set. In a successful iteration, one may stop evaluating the function at points in T_k as soon as a new iterate is found.

3 Filter algorithms

Filter algorithms treat the optimization problem as biobjective - one wishes to minimize both the objective function f and a nonnegative continuous aggregate constraint violation function h. Filter algorithms attempt to minimize both functions, but clearly priority must be given to h, at least until a feasible iterate is found. Fletcher *et al.* [6], [7] and [8] do this via a *restoration* phase, and this priority appears also in our algorithm in the definition of the poll set.

The terminology used in this paper differs slightly from Fletcher *et al.*'s. Our notation is more compact for our class of algorithms, and so it simplifies the presentation of our results. In addition, since our plan is soon to provide a truly multiobjective GPS algorithm, it is

best to conform to standard terminology in multiobjective optimization.

Fletcher *et al.*'s definition of *dominance* makes it a reflexive relation, which simplifies the definition of a filter, but we will forgo that convenience to adhere to standard multiobjective terminology. The point is that the reader familiar with the filter literature should read this section carefully. We will end up with almost the standard notion of a filter, but we will define it slightly differently using the standard multiobjective notion of dominance: For a pair of vectors w, w', with finite components, w *dominates* w', written $w \prec w'$, if and only if $w_i \leq w'_i$ for each i, and $w \neq w'$. Since we allow our problem functions to take on the value ∞, we will make the convention that any vector whose components are all finite dominates any vector with an infinite component. We will use $w \preceq w'$ to indicate that either $w \prec w'$, or that $w = w'$, which is the notion of dominance used in earlier filter papers.

The constraint violation function always satisfies the following properties: $h(x) \geq 0$, and $h(x) = 0$ if and only if x is feasible (and thus $h(x) > 0$ if and only if x is infeasible). For example, we could set $h(x) = \|C(x)_+\|$ where $\|\cdot\|$ is a vector norm. We show in Section 5 that the more locally smooth h is, the better it is able to exploit the positive spanning sets that are used. Our analysis and the examples in Section 6 indicate that $h(x) = \|C(x)\|_2^2$ is a sound choice. By convention, $h(x) = \infty$ if any component of $C(x)$ is infinite.

There should be no confusion with defining a special meaning of dominance for the vector arguments of our problem functions h, f, and this will simplify our terminology rather than to use some other symbol like, $\prec_{(h,f)}$. Thus, a point $x_k \in \Re^n$ is said to *dominate* $x \in \Re^n$, $x_k \prec x$ if and only if $(h(x_k), f(x_k))^T \prec (h(x), f(x))^T$. Two points are equivalent if they generate an identical pair of h and f values. As above, $x \preceq x'$ indicates that either $x \prec x'$, or x and x' are equivalent.

A filter $\mathcal{F}$ is a finite set of points in $\Re^n$ such that no pair x, x' in the filter are in the relation $x \prec x'$. A point x' is said to be filtered if either $x' \succeq x$ for some $x \in \mathcal{F}$, or if $h(x') \geq h_{max}$ for some positive finite upper bound h_{max} on allowable aggregate infeasibility, or if it is feasible and its objective function value is greater than or equal to the feasible incumbent value f^F (i.e., the least function value found so far at a feasible solution). It is unfiltered otherwise. Of course, the first feasible point with a finite value of f we have is automatically the initial feasible incumbent solution. The set of filtered points $\overline{\mathcal{F}}$ is denoted in standard notation as:

$$\overline{\mathcal{F}} \;=\; \bigcup_{x \in \mathcal{F}} \{x' : x' \succeq x\} \;\cup\; \{x' : h(x') \geq h_{max}\} \;\cup\; \{x' : h(x') = 0, f(x') \geq f^F\}.$$

Observe that our notation implies that the current incumbent solution is filtered at the current iteration. This means that trial points that tie the incumbent's functions values do not produce a successful iteration.

Unfiltered points are accepted as they are generated, and filtered ones are rejected. Feasible ones improve the incumbent value f^F, and infeasible ones are added to $\mathcal{F}$. Whether a point is filtered can depend on when it is generated. This temporal property causes "block-

ing entries" [6]. In order to avoid the problem of blocking entries, the filter contains only infeasible points. The incumbent feasible solution is treated separately.

4 A pattern search filter algorithm

We now define our algorithm, which is a pattern search, but the test for accepting a successful iterate is not solely based on the decrease of the objective function value when there are constraints. The notion of successful iteration is defined using the current filter. We assume that we start with at least one point with a finite value of f and h. We do not accept any points for which either value is infinite.

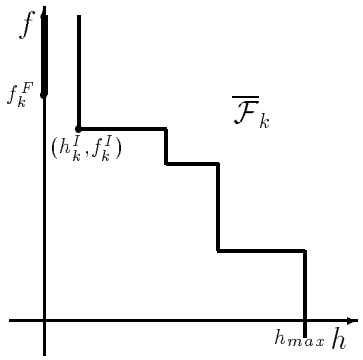
4.1 Algorithm

As in the pattern search algorithms presented in Section 2, each iteration produced by our method is either declared *successful* or *unsuccessful*. Successful ones are those where an unfiltered trial point is found. The SEARCH and POLL step may be terminated without any more function or constraint evaluations if such a point is found. Unsuccessful iterations are those where all trial points are filtered, thus they leave the filter unmodified. The mesh size parameter is either increased or kept constant in successful iterations, and it is decreased in unsuccessful ones. Unlike Fletcher *et al.*'s filter algorithms, there is no "envelope" added to the filter to guarantee sufficient decrease. We will postpone an explanation of the explicit treatment of some linear constraints because the algorithm is the same.

We define two types of incumbent solutions: the feasible ones, and the most feasible infeasible ones (see Figure 1). Let f_k^F represent the feasible incumbent value, i.e., the smallest objective function value (for feasible solutions) found by the algorithm up to iteration k. Let $h_k^I > 0$ be the least positive constraint violation function value found up to iteration k, and let f_k^I denote the smallest objective function value of the points found whose constraint violation function value are equal to h_k^I. The superscript F stands for *feasible* and I for *infeasible*. Our filter algorithm does not have a sufficient decrease condition. Any unfiltered trial point (i.e., not in $\overline{\mathcal{F}}$) guarantees a successful iteration.

The poll set for any iteration is the set of mesh neighbors of p_k. The poll center p_k is either chosen in the set of feasible incumbent solutions, or it must be one of the most feasible infeasible incumbent solutions. The current iterate x_k is used to seed the SEARCH step, and the poll center is used for the POLL step.

Note that there will usually be a single value in each of the sets of incumbents. This means that p_k either satisfies $(h(p_k), f(p_k)) = (0, f_k^F)$, or, $(h(p_k), f(p_k)) = (h_k^I, f_k^I)$. Our class of algorithms and their analysis are completely flexible about the choice between these two alternative poll centers p_k at a given iteration. It is up to the user to select the strategy defining the poll center. This flexibility may seem tedious to the reader of this paper, but



Feasible region:	$\Omega = \{x \in \Re^n : h(x) = 0\}$
Trial set:	T_k
Filtered points:	$\overline{\mathcal{F}}_k$
Successful iterations:	$T_k \setminus \overline{\mathcal{F}}_k \neq \emptyset$
Unsuccessful iterations:	$T_k \subset \overline{\mathcal{F}}_k$

Figure 1: The three possible types of iterations.

the user may have a clear preference based on the results of the search. Also, there is meant to be in this flexibility some reasonable approach to constraint tolerances.

Even if we already have a feasible incumbent solution, we may wish to poll around one of the least infeasible points, which might have a lower objective function value, in order to try to find and explore a different part of the feasible region Ω. Also, this is what allows our filter algorithm to avoid stalling in Lewis and Torczon [10] example, which is detailed here in Section 6.1.

The most useful successful iterations are those that produce an unfiltered feasible iterate x_{k+1}. This improves the feasible incumbent value to $f_{k+1}^F = f(x_{k+1}) < f_k^F$. Then there are the successful iterations that do not produce a feasible iterate, but improve the most feasible infeasible incumbent solution: either both $0 < h_{k+1}^I = h(x_{k+1}) < h_k^I$ and $f_{k+1}^I = f(x_{k+1})$ or both $h_{k+1}^I = h_k^I$ and $f_{k+1}^I = f(x_{k+1}) < f_k^I$. Finally there are the other successful iterations. They leave the incumbents unchanged $h_{k+1}^F = h_k^F, h_{k+1}^I = h_k^I$ and $f_{k+1}^I = f_k^F$, but they add some elements to the filter.

The unsuccessful iterations usually require more function evaluations than successful ones. These iterations are such that all points in the trial set T_k are filtered. Regardless of the feasibility of x_k, if the iteration is unsuccessful, the next iterate x_{k+1} is set to x_k, and the mesh size parameter is decreased: $\Delta_{k+1} < \Delta_k$. The next poll center p_{k+1} need not be fixed to p_k.

Our algorithm for constrained optimization is stated formally below. We allow for the fact that in some applications, a set of initial points with finite values of f and h may be available from solving similar problems, and they can used to seed the filter. Without any loss of generality we assume that any such points, or at least the undominated ones, are on the initial mesh and that they have been "filtered" to be consistent with our initialization step in the sense that x_0 will not be filtered by the other seed points. An easy way to assure this would be to take x_0 to be the most feasible seed point and to break ties by taking one with the smallest objective function value.

A standard trick in engineering design when seed designs are available is to go further and make linear combinations of the seed points be the new design space (see e.g., Vanderplaats [15]). This certainly makes sense, can lead to a big reduction in the dimension of the search space, and it makes the initial mesh be defined by the seed points in a simple way.

Algorithm

- INITIALIZATION:
  Let x_0 be an undominated point of a set of initial solutions. Include all these solutions in the filter $\mathcal{F}_0$, together with $h_{max} > h(x_0)$. Fix $\Delta_0 > 0$ and set the iteration counter k to 0.

- DEFINITION OF INCUMBENT SOLUTIONS:
  Define (if possible)
  f_k^F: the smallest objective function value for all feasible solutions found so far;
  $h_k^I > 0$: the least positive constraint violation function value found so far;
  f_k^I: the smallest objective function value of the points found so far whose constraint violation function value are equal to h_k^I.

- SEARCH AND POLL STEPS:
  Perform the SEARCH and possibly the POLL steps (or only part of the steps) until an unfiltered trial point x_{k+1} is found, or when it is shown that all trial points are filtered by $\mathcal{F}_k$.
  - SEARCH STEP: Evaluate the functions h and f on a set of trial points on the current mesh M_k (the strategy that gives the set of points is usually provided by the user).
  - POLL STEP: Evaluate the functions h and f on the poll set around p_k, where p_k satisfies either $(h(p_k), f(p_k)) = (0, f_k^F)$ or $(h(p_k), f(p_k)) = (h_k^I, f_k^I)$.

- PARAMETER UPDATE:
  If the SEARCH or the POLL step produced an unfiltered iterate $x_{k+1} \in \mathcal{F}_{k+1}$, then declare the iteration *successful* and update $\Delta_{k+1} \geq \Delta_k$.
  Otherwise, set $x_{k+1} = x_k$, declare the iteration *unsuccessful* and update $\Delta_{k+1} < \Delta_k$.
  Increase $k \leftarrow k + 1$ and go back the definition of the incumbents.

It is understood in the algorithm that the filter $\mathcal{F}_{k+1}$ is constructed at iteration k by adding all infeasible unfiltered points (with respect to $\mathcal{F}_k$) found during the SEARCH and POLL on T_k. Moreover, points now dominated by new points are removed from $\mathcal{F}_{k+1}$.

In pattern search algorithms, one role of the POLL step is to guarantee convergence. This is why it is rigidly defined through the positive spanning sets. In practice, the largest improvements in the incumbent solution are obtained in the SEARCH step (e.g., see [2] where a surrogate of an expensive function is constructed). The SEARCH step is usually the one that drives the iterates away from a local optimum. In a SEARCH implementation, it might be a good idea to try some points that are near points of the filter. Paul Frank made the

interesting suggestion that SEARCH might include polling around the next most feasible filter point, i.e., $x \in \mathcal{F}_k$ with the least value of $h(x) > h^I$. The objective here again is to attempt to find and then explore a different part of the feasible region.

4.2 Infinite refinement of the mesh

In this section, we will identify a set of limit points of the algorithm that exist even without assuming any problem smoothness. We need only that all the problem functions are finite at x_0. Later we will show some optimality conditions hold at these limit points for which the problem is locally smooth.

The convergence analysis of our algorithm is based on the standard (see [5], [6], [7] and [8]) assumption that all trial points produced by the algorithm lie in a compact set. A consequence of this is that since the mesh size parameter does not decrease at successful iterations ($\Delta_{k+1} \geq \Delta_k$), then it follows that there can only be a finite number of *consecutive* successful iterations.

We will be concerned here only with the poll centers p_k about which some unsuccessful poll directions were tried, even if others were successful. Remember that unsuccessful iterations are those at which complete polling about p_k failed and the mesh size parameter is reduced ($\Delta_{k+1} < \Delta_k$).

Our first result is that there is a subsequence of iterations for which the mesh size parameter goes to zero. In order to prove it we require the following lemma from Torczon [14] or Audet and Dennis [1].

Lemma 4.1 *The mesh size parameters Δ_k are bounded above by a positive constant independent of the iteration number k.*

Combining this lemma with the assumption that all iterates lie in a compact set implies the following result. Its proof is omitted since it is identical to that of the same result in Torczon [14].

Theorem 4.2 *The mesh size parameters satisfy* $\liminf_{k \to +\infty} \Delta_k = 0$.

Since the mesh size parameter shrinks only at unsuccessful iterations, Theorem 4.2 guarantees that there are infinitely many unsuccessful iterations. An unsuccessful iteration k is such that the entire trial set (which includes the poll set) is filtered. Therefore, for each direction $s \in S_k$ there exists some element x in the filter $\mathcal{F}_k$ such that both $f(p_k + \Delta_k s) \geq f(x)$ and $h(p_k + \Delta_k s) \geq h(x)$. Clearly then, the unsuccessful poll centers have limit points. We will study them in the next section.

We introduce the following definition to identify the unsuccessful polling directions for which we can draw conclusions. We will make the convention, which is implied by the

algorithm, that for p_k to be a poll center implies that polling was at least initiated at iteration k, although polling may have been successful in finding a better point, thus making the iteration SUCCESSFUL.

Definition 4.3 *A convergent subsequence of poll centers $\{p_k\}_{k\in K}$ (for some subset of indices K) is said to be a* refining subsequence *if $\lim_{k\in K} \Delta_k = 0$. A polling direction $s \in \Re^n$ is said to be* associated *with the refining subsequence if for every term of the subsequence, the poll step $p_k + \Delta_k s$ did not produce an unfiltered point. A positive spanning set consisting of directions associated with a given refining subsequence is said to be an* associated *positive spanning set.*

Most of the results in this paper deal with limit points of refining subsequences and their associated directions. The following results shows the existence of interesting points and directions.

Lemma 4.4 *There exist refining subsequences of poll centers. Any refining subsequence $\{p_k\}_{k\in K}$ for unsuccessful iterations has at least one limit point $\hat{x}$ and at least one associated positive spanning set $\hat{S}$.*

Proof. Theorem 4.2 guarantees that there exists a subsequence of iterations whose mesh size parameter goes to zero. Moreover, the assumption that all trial points (thus all poll centers) are in a compact set implies the first part of the result.

To prove the second part, we first note that Δ only decreases after a completely unsuccessful poll step in directions constituting a positive spanning set. Thus, there exists a refining subsequence consisting of such poll centers. Finiteness of the set of positive spanning sets $\mathcal{S}$ ensures that at least one of them is associated with that refining subsequence. ∎

5 Optimality conditions

Note that not every refining subsequence has an associated direction, much less an associated positive spanning set. However all of them do except for the unlikely event that a refining subsequence might have been successful in its first polling direction every time. The finiteness of the set of all vectors in any member of $\mathcal{S}$ does guarantee at least one associated direction for any other refining subsequence since there would have had to be at least one unsuccessful direction tried at every term of the refining subsequence. Furthermore, as in the last result of the previous section, it is the refining subsequences for which the corresponding poll steps resulted in unsuccessful iterations that are certain to have associated positive spanning sets. Thus, the reader may wonder why we do not restrict ourselves to consider limit points of unsuccessful refining sequences. The answer is that when the f and h are smooth at $\hat{x}$, then the more lenient and general definition enriches the set of directions whose polar cone contains $-\nabla f(\hat{x})$ (see Corollary 5.6).

We will begin with results that only consider the behavior of h, and then move on to complete our results by considering the effect of the filter on the objective function f.

5.1 Results for the constraint violation function

At this point, we have shown the existence of limit points of refining subsequences of the algorithm without any assumptions on the objectives or constraints except that the values $f(x_0)$ and $h(x_0)$ are both finite. Now we will show that if f and h are smooth in a neighborhood of such a limit point $\hat{x}$, then it has certain optimality properties. As in [1], Clarke's [3] generalized derivatives are the key to our convergence analysis. To use this powerful tool, we assume from here on that h is Lipschitz near such a limit point, i.e., that h is Lipschitz in a neighborhood of $\hat{x}$.

Theorem 5.1 *Let $\hat{x}$ be the limit point of a refining subsequence $\{p_k\}_{k\in K}$ with associated direction s. The generalized directional derivative of h at $\hat{x}$ in the direction s is nonnegative, i.e., $h^\circ(\hat{x};s) \geq 0$.*

Proof. If the limit point $\hat{x}$ of the subsequence $\{p_k\}_{k\in K}$ is feasible, then $h(\hat{x}) = 0$ and nonnegativity of the function h implies the result for the associated direction s.

Consider the case where $\hat{x}$ is infeasible and $h(p_k + \Delta_k s) \neq 0$ for each k in a subset K' of K. From Clarke [3], we have by definition of the generalized directional derivative that:

$$h^\circ(\hat{x};s) \;=\; \limsup_{y\to\hat{x},\; t\downarrow 0} \frac{h(y+ts)-h(y)}{t} \;\geq\; \limsup_{k\in K'} \frac{h(p_k+\Delta_k s)-h(p_k)}{\Delta_k}.$$

For each $k \in K'$, the quotient $\frac{h(p_k+\Delta_k s)-h(p_k)}{\Delta_k}$ is nonnegative since the poll step $p_k + \Delta_k s$ is unsuccessful. Thus the limit superior of the right-hand side is nonnegative. ∎

Note that if the function h is regular at $\hat{x}$ then the usual directional derivative $h'(x;s)$ exists and coincides with the generalized directional derivative.

The last result has the following easy corollary, in which we still assume that h is Lipschitz near $\hat{x}$, and in addition, that the generalized gradient of h at $\hat{x}$ is a singleton. This is equivalent to assuming that h is strictly differentiable at $\hat{x}$ (see [3], Proposition 2.2.1 or Proposition 2.2.4).

Corollary 5.2 *Let $\hat{x}$ be the limit point of a refining subsequence $\{p_k\}_{k\in K}$ with an associated positive spanning set $\hat{S}$. If h is strictly differentiable at $\hat{x}$ then $\nabla h(\hat{x}) = 0$.*

Proof. Again from [3], if h is strictly differentiable at $\hat{x}$, then $h^\circ(\hat{x};w) = \nabla h(\hat{x})^T w$ for any direction $w \neq 0$. Now by Theorem 5.1, for each $s_i \in \hat{S}$, $0 \leq h^\circ(\hat{x};s_i) = \nabla h(\hat{x})^T s_i$. Thus, if we write w as a nonnegative linear combination of the elements of $\hat{S}$, then we see immediately

that $\nabla h(\hat{x})^T w \geq 0$. But the same construction for $-w$ shows that $-\nabla h(\hat{x})^T w \geq 0$ and so $\nabla h(\hat{x}) = 0$. ∎

This means that using $h_2(x) = \|C(x)_+\|_2^2$ gives nice results since it is continuously differentiable whenever C is (see Dennis, El Alem and Williamson [4] for a compact formulation of ∇h_2). Our preliminary tests indicate that it works better than other common choice, at least for SQP, of $h_1(x) = \|C(x)_+\|_1$. Thus, the question arises as to the differentiability of h_1. The answer, which implies that it is rarely strictly differentiable at $\hat{x}$, is given by the following result. In Section 6.2 we will see an example showing the cost of this lack of smoothness.

Proposition 5.3 *If C is regular at every x, then so is h_1. Let $V(x) = \{i : c_i(x) > 0\}$ and $A(x) = \{i : c_i(x) = 0\}$ be the violated and active sets at x. Then the generalized gradients are related by*

$$\partial h_1(x) \;=\; \sum_{i \in V(x)} \partial c_i(x) + \left\{ \sum_{i \in A(x)} \gamma_i \zeta_i \;:\; \gamma_i \in [0,1], \zeta_i \in \partial c_i(x), i \in A(x) \right\}.$$

The generalized directional derivatives of h_1 and C in a direction w at x are related by

$$h_1^\circ(x;w) \;=\; \sum_{i \in V(x)} c_i^\circ(x;w) + \sum_{i \in A(x)} (c_i^\circ(x;w))_+.$$

Thus, if C is strictly differentiable at x, then

$$h_1^\circ(x;w) \;=\; \sum_{i \in V(x)} \nabla c_i(x)^T w + \sum_{i \in A(x)} (\nabla c_i(x)^T w)_+.$$

Proof. The proof follows from various results in [3], and some simple observations. Clarke's Propositions 2.3.12 and 2.3.6 guarantee that $c_i(x)_+$ thus $h_1(x)$ are regular at x whenever $c_i(x)$ is.

The third corollary to Proposition 2.3.3 implies that $\partial h_1(x) = \sum_i \partial c_i(x)_+$, where this means all possible sums of an element from each $\partial c_i(x)_+$. Propositions 2.3.12 implies that

$$\partial c_i(x)_+ \;=\; \begin{cases} \partial c_i(x) & \text{if } c_i(x) > 0, \\ co\{\partial c_i(x), \partial 0(x)\} = \{\gamma_i \zeta_i : \gamma_i \in [0,1], \zeta_i \in \partial c_i(x)\} & \text{if } c_i(x) = 0, \\ \partial 0(x) = \{0\} & \text{if } c_i(x) < 0, \end{cases}$$

where *co* denotes the convex hull.

The generalized directional derivative in any direction w can be written: $h_1^\circ(x;w) = \sum_i (c_i(x;w)_+)^\circ$. If $c_i(x) > 0$ then $(c_i(x;w)_+)^\circ = \max\{w^T\zeta : \zeta \in \partial c_i(x)\} = c_i^\circ(x;w)$. If

$c_i(x) < 0$ then $(c_i(x;w)_+)^\circ = \max\{w^T\zeta : \zeta \in \partial 0(x)\} = 0$. And if $c_i(x) = 0$, then

$$\begin{aligned}
(c_i(x;w)_+)^\circ &= \max\{w^T\eta : \eta \in \partial c_i(x)_+\} \\
&= \max\left\{w^T\eta : \eta \in \{\gamma_i\zeta_i : \gamma_i \in [0,1], \zeta_i \in \partial c_i(x)\}\right\} \\
&= \begin{cases} 0 & \text{if } \max\{w^T\zeta_i \in \partial c_i(x)\} \le 0 \\ \max\{w^T\zeta_i \in \partial c_i(x)\} & \text{otherwise} \end{cases} \\
&= (\max\{w^T\zeta_i \in \partial c_i(x)\})_+ \quad = \quad (c_i^\circ(x;w))_+.
\end{aligned}$$

The last part of the result follows by definition of strict differentiability. ∎

Note that the above result could be slightly rewritten by using one-sided directional derivatives instead of generalized directional derivatives. Indeed, if C is regular at x, then $c_i^\circ(x;w)$ coincides with the one-sided directional derivative $c_i'(x;w)$, and $h_1^\circ(x;w)$ coincides with $h_1'(x;w)$.

Having the form of $h_1^\circ(\hat{x};w)$ allows us to prove as much as we can expect, if not as much as we would like, about the feasibility of $\hat{x}$ as defined above. In particular, we can show that the algorithm with constraint violation function h_1 produces limit points at which directional first order conditions hold for feasibility or for a local minimum of $h_1(x)$. This is another reason we prefer the square norm h_2 to measure the constraint violation for our filter algorithm.

5.2 Results for the objective function

We have shown above that the limit point for a refining subsequence generated by the algorithm satisfies local optimality conditions for the constraint violation function. We now derive some results for the objective function. The following theorem gives some conditions that guarantee that the generalized directional derivative in an associated direction is zero. For the remainder of the paper we will assume that f is Lipschitz near the limit point in question.

Theorem 5.4 *Let $\hat{x}$ be the limit point of a refining subsequence $\{p_k\}_{k\in K}$ with associated direction s. If $h(p_k) = h(p_k + \Delta_k s)$ for each k in K, then $f^\circ(\hat{x};s) \ge 0$.*

Proof. Suppose by contradiction that $f^\circ(\hat{x};s) < 0$. Since f is Lipschitz near $\hat{x}$, then $\limsup_{k\in K} \frac{f(p_k+\Delta_k s)-f(p_k)}{\Delta_k}$ is bounded above by the negative value $f^\circ(\hat{x};s)$ and it follows that $f(p_k + \Delta_k s) < f(p_k)$ for large k in K. Therefore, for these large indices, the assumption that $h(p_k) = h(p_k + \Delta_k s)$ implies that the iteration is successful since $p_k + \Delta_k s$ dominates the poll center p_k. This is a contradiction. ∎

The result does not hold if $h(p_k) < h(p_k + \Delta_k s)$. This is illustrated by the example in Section 6.3. Also, since the poll step in the direction s is unsuccessful at iteration k, then it is impossible that $h(p_k) > h(p_k + \Delta_k s) > 0$.

Theorem 5.4 holds for any associated direction s. This means that one of the advantage of using a large number of positive spanning directions is that the set of associated directions will be larger.

An obvious corollary to this result is that this local optimality condition necessarily holds at limit points interior to the feasible region.

Corollary 5.5 *Let $\hat{x}$ be the limit point of a refining subsequence $\{p_k\}_{k\in K}$ and s an associated direction. If $\hat{x}$ is strictly feasible, then $f^\circ(\hat{x};s) \geq 0$.*

Proof. If $\hat{x}$ is strictly feasible, then there exists an $\epsilon > 0$ such that $h(x) = 0$ for every x satisfying $\|x - \hat{x}\| \leq \epsilon$. If $k \in K$ is large enough, then both $h(p_k) = 0$ and $h(p_k + \Delta_k s) = 0$. Theorem 5.4 applies and yields the result. ∎

The next result shows which problem we have solved, or at least found a KKT point for. As an example in the next section will show, it does not necessarily have the same solution as the problem we set out to solve. Of course, this result is pessimistic because a key point in our analysis is to provide maximum flexibility in the search step. Indeed, the price we pay for this important practical consideration is that although we can not assume the search step gives us any help, we are limited in what we can prove by the possibility for an unlucky search procedure to find an unfiltered point that causes a later step to be filtered when it would have been advantageous. Consequently, we believe that in this paper we have proved as much as we can without hypotheses on the search step. For specific application domains, this may be a fruitful direction for future research.

We cannot guarantee with this algorithm that when $\hat{x}$ is on the boundary of the feasible region Ω, $-\nabla f(\hat{x})$ belongs to the normal cone $N_\Omega(\hat{x})$ to the feasible region at $\hat{x}$ (this cone is also the polar of the tangent cone $T_\Omega(\hat{x})$ to the feasible region at $\hat{x}$). However, this second corollary provides a cone that contains $N_\Omega(\hat{x})$ as well as $-\nabla f(\hat{x})$. This result is illustrated in Section 6.

Corollary 5.6 *Let C_s be the cone generated by all the associated directions of all refining subsequences that converge to the limit point $\hat{x}$ and that satisfy the conditions of Theorem 5.4. If f is strictly differentiable at $\hat{x}$, then $-\nabla f(\hat{x})$ belongs to the polar C_s° of C_s.*

Proof. Theorem 5.4 guarantees that $f^\circ(\hat{x};s) \geq 0$ for any vector s that generates C_s. Strict differentiability of f ensures that $\nabla f(\hat{x})^T s \geq 0$. From the definition of C_s any vector y from $\hat{x}$ pointing into C_s is a nonnegative linear combination of these associated directions, which means that $\nabla f(\hat{x})^T y \geq 0$. The result follows from the definition of a polar cone: $-\nabla f(\hat{x}) \in C_s^\circ = \{x \in \Re^n : x^T y \leq 0 \ \forall y \in C\}$. ∎

By using a filter based step acceptance criterion, we have overcome a difficulty in applying pattern search algorithms to constrained optimization. Specifically, that the objective function descent directions in the positive spanning set S may be infeasible. Lewis and Torczon [10] give an example where a nonfilter based version of the pattern search algorithm

stalls (all subsequent iterations are unsuccessful) when the positive spanning matrix is poorly chosen at a feasible point where the gradient of f is not zero.

The following result shows that, under assumptions on the smoothness of the functions, our algorithm will not stall at any point x for which $\nabla f(x) \neq 0$ regardless of the choice of positive spanning set. The implication is that there eventually will be a successful iteration that will move the poll center away from the current one. This is an essential ingredient of any method with ambitions to find more than a single local minimizer.

Proposition 5.7 *Let h Lipschitz near the poll center p_k and f be strictly differentiable at p_k. If $\nabla f(p_k) \neq 0$, then there cannot be infinitely many consecutive unsuccessful poll steps around p_k.*

Proof. Assume that $\nabla f(p_k) \neq 0$ and that there are infinitely many consecutive unsuccessful poll steps around p_k. Let s be an associated direction of the (constant) subsequence of unsuccessful poll centers such that $\nabla f(p_k)^T s < 0$. Let ϵ_f be such that $f(p_k + \Delta s) < f(p_k)$ whenever $0 < \Delta < \epsilon_f$.

First consider the case where p_k is feasible. Since h is Lipschitz near p_k we can define $\epsilon_h > 0$ to be such that $h(p_k + \Delta s) < h_k^I$ whenever $0 < \Delta < \epsilon_h$ (h_k^I is equal to h_{max} if $\mathcal{F}_k$ is empty). Thus, when $0 < \Delta_\ell < \min\{\epsilon_f, \epsilon_h\}$ polling around $p_\ell = p_k$ is successful, which is a contradiction.

Consider now the case where p_k is infeasible. If $\mathcal{F}_k = \{p_k\}$, then since h is Lipschitz near p_k we can define $\epsilon_h > 0$ to be such that $0 < h(p_k + \Delta s) < h_{max}$ whenever $0 < \Delta < \epsilon_h$. Otherwise, $\mathcal{F}_k$ is not a singleton and since h is Lipschitz near p_k we can define $\epsilon_h > 0$ to be such that $0 < h(p_k + \Delta s) < \tilde{h}$ whenever $0 < \Delta < \epsilon_h$, where $\tilde{h}$ is the optimal value of $\min_{x \in \mathcal{F}_k}\{h(x) : h(x) > h(p_k)\}$. Therefore if $0 < \Delta_\ell < \min\{\epsilon_f, \epsilon_h\}$ then polling around $p_\ell = p_k$ is successful, which is a contradiction. ∎

5.3 Remaining feasible for linear constraints

When some of the constraints are linear, it is frequently advantageous to treat them separately from the others and to ask that every iterate satisfy those linear constraints. This is especially true of linear equality or bound constraints. For any derivative-free algorithm, one should surely use linear equality constraints to eliminate variables. This means for us that the mesh would be constructed in the linear manifold defined by the linear equality constraint.

It is simple to remain feasible with respect to a finite number of linear constraints, i.e., to remain in a polyhedron X defined by those constraints, and to treat the remaining constraints by our filter algorithm. Specifically, we apply the algorithm not to f, but to the function $f_X = f + \psi_X$, where ψ_X is the indicator function for X. It is zero on X and ∞ elsewhere.

This automatically handles the selected linear constraints the same way as in [10], [11] and [1]. Moreover, the algorithm can take into account the geometry of the region X just as suggested by these references, i.e., when the poll center is within a given tolerance of a point x' on the boundary of X, then the positive spanning directions that define the poll set are chosen to contain the ones that span the tangent cone to X at x' (see [11] for a precise way of identifying these directions).

The other nonlinear constraints would be handled by the filter as described in the previous sections. The constraint violation function h need only be defined with respect to the nonlinear constraints, (i.e., using Ω and not X). We assume that the initial iterate x_0 is in the polyhedron X and that $f_X(x_0)$ and $h(x_0)$ are both finite.

With this version of the algorithm, the results of Section 5.1 for the minimization of the constraint violation function and Section 5.2 for the objective function still hold for the function f, even if the minimization is done with respect to f_X. As above, the quality of the solution at the resulting limit point $\hat{x}$ would depend of the smoothness of the function f there, and not of the smoothness of f_X, which is not even Lipschitz on the boundary of X.

The reason is that taking positive spanning sets to contain the directions defined in [11] near the boundary of X yields limit points $\hat{x}$ at which the candidates for associated directions that span the cone C_s include directions that span the cone tangent to X at $\hat{x}$.

We have carefully verified that there is no snag in this approach, but there seems little reason to burden the reader with the rather unenlightening verification of this intuitively obvious argument.

6 Illustration of our results

We now illustrate the behavior of our algorithm on three examples. The first is due to Lewis and Torczon [10]. However, the filter approach, unlike the barrier approach in [10], can converge even with a badly chosen positive spanning set. Indeed, one might even think it will converge with most choices of positive spanning sets. We believe that this shows the value of our filter approach.

The second example justifies our choice of the squared ℓ_2 norm over the ℓ_1 norm in the definition of the constraint violation function. The non-smoothness of the latter may not provide descent on h_1 in some of the poll directions for which h_2 does descend. The example shows that since h_1 does not allow movement, using it can result in stalling at an infeasible point.

The third example shows the limitations of our results; there is more left to do. This example uses all the flexibility we provide in the SEARCH as a loophole to avoid a desirable outcome. Even with the squared ℓ_2 norm, it is still possible to choose the positive spanning sets, and to be unlucky, in a way that the limit point $\hat{x}$ solves an problem which we wish were a better approximation than it is to the problem we set out to solve. For this example,

there is a strictly feasible descent direction for the objective function f from $\hat{x}$. This does not contradict our results, but it does show their limitations without a suitable search scheme.

6.1 Example of Lewis and Torczon

Consider the linear program [10]

$$\begin{aligned} \min_{x=(a,b)^T} \quad & -a-2b \\ \text{s.t.} \quad & 0 \le a \le 1 \\ & b \le 0. \end{aligned}$$

The optimal solution is $\hat{x} = (1,0)^T$. Let us apply our algorithm with initial solution $x_0 = (0,0)^T$ and initial mesh size parameter $\Delta_0 = 1$ and with a single positive spanning matrix S constructed with the four directions $\pm(1,1)^T$ and $\pm(1,-1)^T$. We will not use any SEARCH step for this example. It is pointed out in [10] that all iterations of a "barrier" pattern search algorithm (that assigns an objective function value of $+\infty$ to infeasible solutions) remain at the origin since the polling directions that yield decrease in the objective function are infeasible. They show that it is sufficient to take positive spanning sets that conform to the geometry of the feasible region to overcome this difficulty.

However, our algorithm finds the optimal solution even with this poorly chosen spanning set since it accepts an infeasible step if it is unfiltered. Figure 2 displays the first few iterations. The poll centers are underlined, and the functions values are between brackets: $[h(x), f(x)]$. The points in the poll set are joined to the poll center by dotted lines.

Figure 2(a) illustrates the iterations for which $\Delta_k = 1$. The shaded area is the feasible region. Starting at $x_0 = p_0 = (0,0)^T$ the algorithm evaluates both functions at $\pm(1,1)$ and $\pm(1,-1)$. Only the trial point $(1,-1)$ is feasible; it is however dominated by x_0. The point $(1,1)$ dominates the two other trial points and is unfiltered. Iteration 0 is therefore a successful iteration.

Let $x_1 = p_1 = (1,1)^T$. The functions are evaluated at the four points around x_1 and two unfiltered points are found: $x_2 = (0,2)^T$ and $(2,2)^T$. Even if the iteration is successful, the poll center p_2 remains at p_1. Polling around p_2 yields filtered points, thus iteration 2 is unsuccessful. Figure 3 displays the filters corresponding to the iterates in Figure 2.

Figure 2(b) starts at iteration 2 with $p_3 = (1,1)^T$ and $\Delta_3 = \frac{1}{2}$. Two consecutive successful iterations lead to $p_4 = (\frac{3}{2}, \frac{1}{2})^T$ then $p_5 = (1,0)^T$, which is the optimal solution.

However, since the gradient is nonzero at this point, Proposition 5.7 ensures that polling around this point will eventually be successful. Indeed, as shown in Figure 2(c), iteration 5 is unsuccessful, but iteration 6 is successful, and it generates a new most feasible infeasible incumbent.

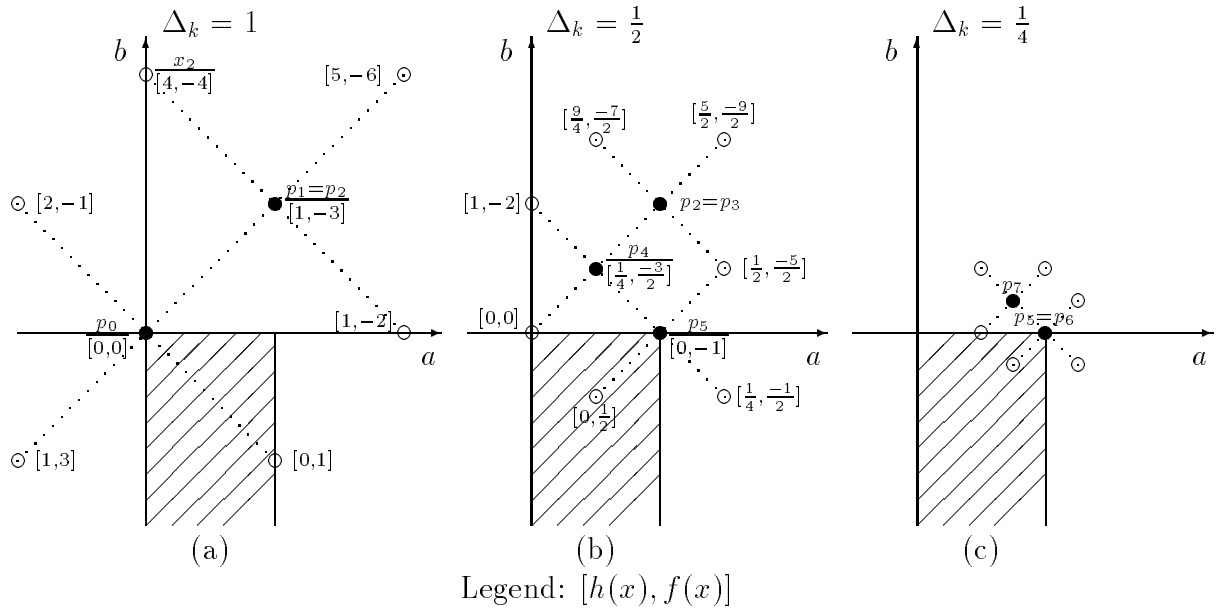


Figure 2: First iterations on example from Lewis and Torczon.

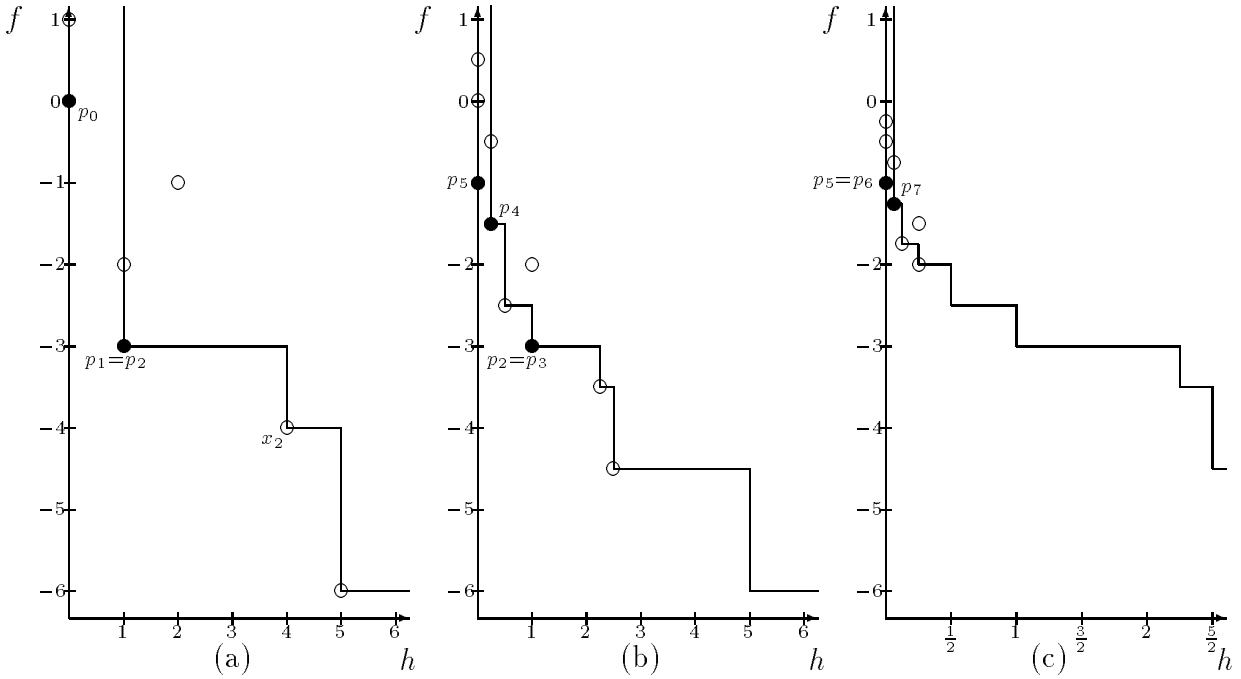


Figure 3: Filter for the example from Lewis and Torczon.

Observe that the choice of the poll center is the key to the quality of the limit points the algorithm finds. Indeed, in this example, if one always define the current center to be the best feasible incumbent, then the iterates will stall at x_0. Choosing to poll around the most feasible infeasible incumbent moves the iterates away from x_0. A poll center selection strategy could be to alternate between these two incumbents every time the poll is unsuccessful. Polling around the most feasible infeasible one is mostly interesting when f^I is less than f^F since it might move the iterates away from a local optimum, or toward a more interesting part of the feasible region. That way, there will be infinitely many poll steps around both types of incumbents. It is also worthwhile to change the positive spanning set to enrich the set of associated directions.

Our results show that any ad hoc strategy will work for choosing p_k, but we suspect that this point in the algorithm provides an important hook for user interaction. Again, we are indebted to Paul Frank for allowing us to observe him interact with the algorithm on exactly this issue.

6.2 Choice of the constraint violation norm

We observed through small examples that choice of the norm in the definition of the constraint violation function $h(x) = \|C(x)_+\|$ affects the convergence behavior.

We prefer the squared ℓ_2 norm over the ℓ_1 norm since it is differentiable whenever the constraint function C is (see [4] for an explicit formulation of the gradient). This means that if there is a descent direction, then a positive spanning set will detect it with the ℓ_2 norm, but ℓ_1 might miss it. This is illustrated in the following simple example.

Consider the linear program

$$\begin{aligned} \min_{x=(a,b)^T} \quad & b \\ \text{s.t.} \quad & -b \leq 3a \leq b \\ & b \geq 1. \end{aligned}$$

The ℓ_1 constraint violation function is

$$h_1(a,b) = \max(3a - b, 0) + \max(-3a - b, 0) + \max(1 - b, 0).$$

Let the algorithm start at the infeasible point $x_0 = p_0 = (0,0)$, and let the positive spanning set be $\{(1,1)^T, (1,-1)^T, (0,-1)^T\}$. The iterates and the filter are depicted in Figure 4.

Every even iteration k is successful (let $\alpha = \Delta_k$): The trial point $x_{k+1} = p_k + \Delta_k(1,1)^T$ is unfiltered by $\mathcal{F}_k$. Every odd iteration is unsuccessful: The three trial points are filtered since $p_{k+1} = p_k = (0,0)^T$. Therefore, the mesh size parameter is reduced at each odd iteration.

The sequence of poll centers stalls at $(0,0)^T$. This means that the non-differentiability of h_1 hides the descent directions for the constraint violation function.

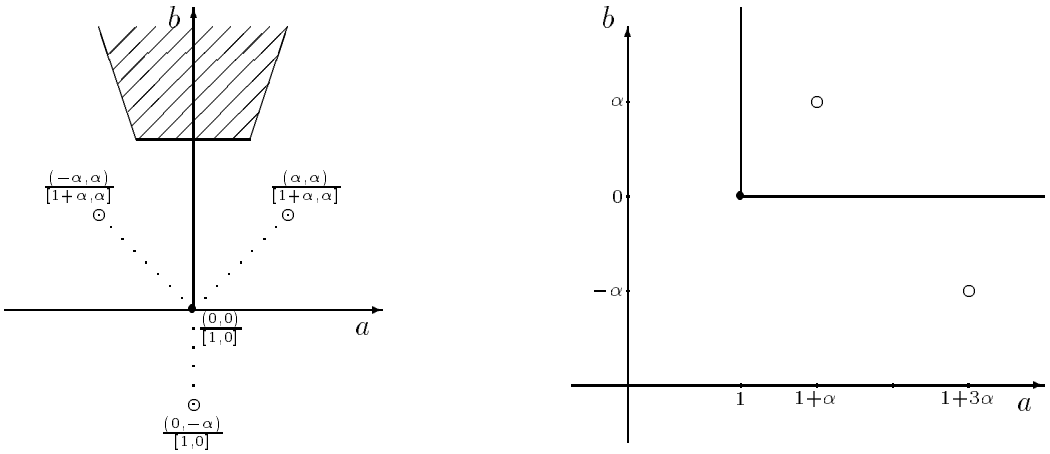


Figure 4: Alternation of successful and unsuccessful iterations.

If the squared ℓ_2 norm is used instead of ℓ_1, then the poll center moves away from $(0,0)$ to $(\alpha,\alpha)^T$ as soon as the mesh size parameter drops below $\frac{2}{3}$ since $h_2(\alpha,\alpha) = (2\alpha)^2 + 0^2 + (1-\alpha)^2 = 1 - 2\alpha + 3\alpha^2$ is less then 1 whenever $0 < \alpha < \frac{2}{3}$.

6.3 Illustration of the limitation of the results

Consider the optimization problem

$$\begin{aligned} \min_{x=(a,b)^T} \quad & b \\ \text{s.t.} \quad & a(1-a) - b \leq 0. \end{aligned}$$

The SEARCH and POLL strategies described below are such that the algorithm goes through infinitely many consecutive cycles. Each cycle contains 4 iterations, that alternate between successful and unsuccessful ones. We admit that the flexibility in SEARCH is exploited to lead to a weak result, but our point is that it can happen.

Every cycle starts with a nonnegative parameter $\alpha < \frac{1}{16}$ and with $x_k = (16\alpha, 0)^T$ and $\Delta_k = 2\alpha$. The function values are $(h(x_k), f(x_k)) = (16(1-16\alpha)\alpha, 0)$. The cycle goes as follows:

- First it does a successful SEARCH step at

  $$x_k + (-4,4)^T\Delta_k = (8\alpha, 8\alpha)^T, \qquad (h,f) = (0, 8\alpha)$$

  which produces the next iterate x_{k+1} with $\Delta_{k+1} = \Delta_k$.

- Second, it performs an unsuccessful POLL step around $p_{k+1} = x_{k+1}$ at the trial points

  $$x_{k+1} + (4,-4)^T\Delta_{k+1} = (16\alpha, 0)^T, \qquad \text{which equals } x_k$$

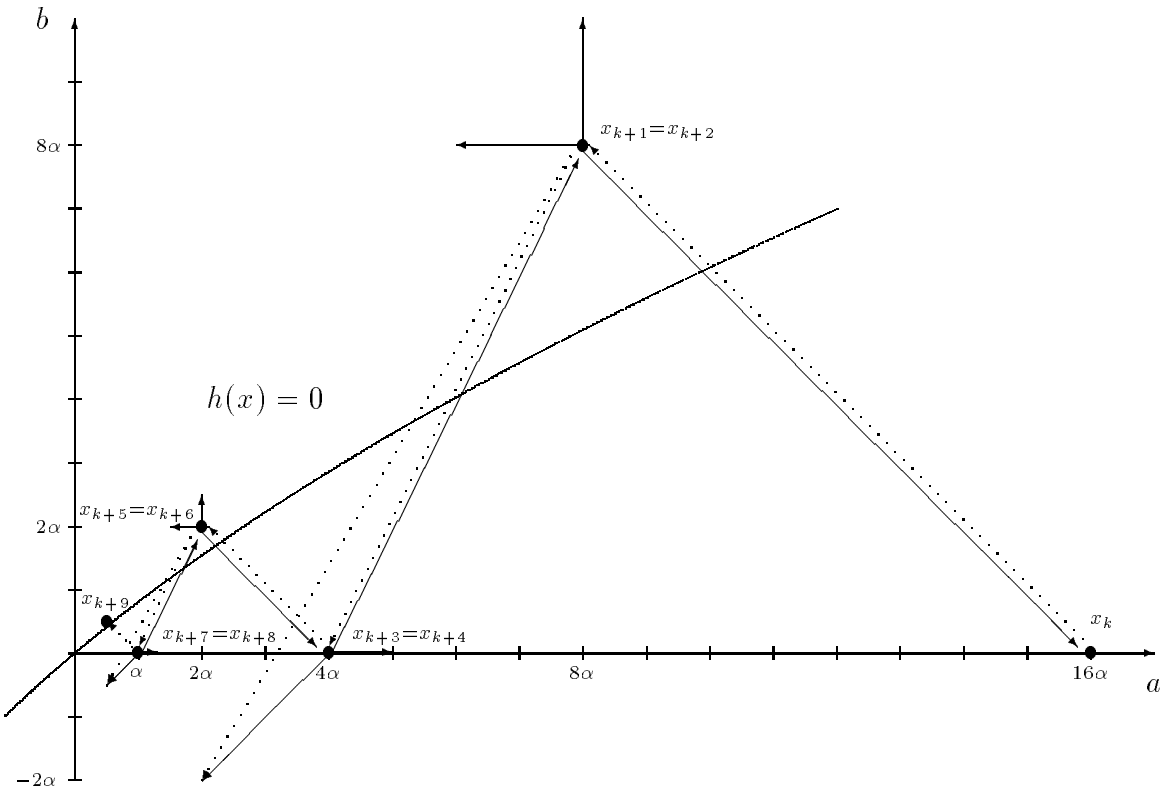


Figure 5: Two cycles of four iterations.

$$x_{k+1} + (0,1)^T\Delta_{k+1} = (8\alpha, 10\alpha)^T, \qquad (h,f) = (0, 10\alpha), \text{ which is filtered by } x_k$$
$$x_{k+1} + (-1,0)^T\Delta_{k+1} = (6\alpha, 8\alpha)^T, \qquad (h,f) = (0, 8\alpha), \text{ which is filtered by } x_k$$

- Third, at $x_{k+2} = x_{k+1}$ and $\Delta_{k+2} = \frac{\Delta_{k+1}}{2}$ the successful SEARCH looks at two trial points

$$x_{k+2} + (-6,-10)^T\Delta_{k+2} = (2\alpha, -2\alpha)^T, \qquad (h,f) = (4(1-\alpha)\alpha, -2\alpha)$$
$$x_{k+2} + (-4,-8)^T\Delta_{k+2} = (4\alpha, 0)^T, \qquad (h,f) = (4(1-4\alpha)\alpha, 0).$$

  Both trial points are unfiltered, x_{k+3} is chosen to be the most feasible one (i.e., the second one) and $\Delta_{k+3} = \Delta_{k+2}$.

- Fourth, it performs an unsuccessful POLL step around $p_{k+3} = x_{k+3}$ at the trial points

$$x_{k+3} + (1,0)^T\Delta_{k+3} = (5\alpha, 0)^T, \qquad (5(1-5\alpha)\alpha, 0), \text{ which is filtered by } x_{k+3}$$
$$x_{k+3} + (-2,-2)^T\Delta_{k+3} = (2\alpha, -2\alpha)^T, \qquad \text{which is the first trial point of step 3}$$
$$x_{k+3} + (4,8)^T\Delta_{k+3} = (8\alpha, 8\alpha)^T, \qquad \text{which is } x_{k+1}.$$

The cycle terminates, feeding the next one with $x_{k+4} = (2\alpha, 0)^T = \frac{x_k}{4}$ and $\Delta_{k+4} = \frac{\alpha}{2} = \frac{\Delta_k}{4}$. The iterates in the cycle are illustrated in Figure 5.

The sequence of unsuccessful iterates (i.e., all iterates corresponding to the second and fourth step of the cycles) converges to the limit point $\hat{x} = (0, 0)^T$. There are no other limit points.

The results of Section 5.1 are clearly satisfied since $\hat{x}$ is feasible. However, there is a strictly feasible associated direction which is also a descent direction for the objective function.

Using Definition 4.3, we see that the set of associated directions is $\{(4, -4)^T, (0, 1)^T, (-1, 0)^T, (1, 0), (-2, -2)^T, (4, 8)^T\}$. Theorem 5.4 applies, but the result $f^\circ(\hat{x}; s)$ holds only for the directions s along which the constraint violation function is constant, i.e., for s in $\{(0, 1)^T, (-1, 0)^T\}$. The strictly feasible direction $(-1, -1)^T$ does not belong to that set. Corollary 5.6 correctly says that the polar cone (whose generators are $(1, 0)^T$ and $(0, -1)^T$) contains $-\nabla f(\hat{x}) = -(0, 1)^T$.

References

[1] Audet C. and Dennis J.E.Jr.(2000), "Analysis of generalized pattern searches," *TR00-07* Department of Computational & Applied Mathematics, Rice University, Houston TX.

[2] Booker A.J., Dennis J.E.Jr, Frank P.D., Serafini D.B., Torczon V. and Trosset M.W.(1999), "A rigorous framework for optimization of expensive functions by surrogates," *Structural Optimization* Vol.17 No.1, 1-13.

[3] Clarke, Frank H.(1990) "Optimization and nonsmooth analysis" SIAM Classics in Applied Mathematics Vol.5, Philadelphia.

[4] Dennis J.E.Jr, El-Alem M. and Williamson K..(1999), "A trust-region approach to nonlinear systems of qualities and inequalities," *SIAM Journal on Optimization* Vol.9 No.2, 291–315.

[5] Conn A.R., Gould N.I.M. and Toint Ph.L.(1991) "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM Journal on Numerical Analysis* 28, 545–572.

[6] Fletcher R. and Leyffer S.(1997), "Nonlinear programming without a penalty function," Dundee University, Dept. of Mathematics, Report NA/171.

[7] Fletcher R, Leyffer S. and Toint Ph.L.(1998), "On the global convergence of an SLP-Filter algorithm," Dundee University, Dept. of Mathematics, Report NA/183.

[8] Fletcher R, Gould N.I.M., Leyffer S. and Toint Ph.L.(1999), "On the global convergence of trust-region SQP-Filter algorithms for general nonlinear programming," Department of Mathematics, FUNDP, Namur (B), Report 99/03.

[9] John F.(1948), "Extremum problems with inequalities as subsidiary conditions," in *Studies and essays: Courant Anniversary Volume*, (eds Friedrichs K.O., Neugebauer O.E. and Stoker J.J.), Wiley-Interscience, New-York, 187–204.

[10] Lewis R.M. and Torczon V.(1996), "Pattern search algorithms for bound constrained minimization," *SIAM Journal on Optimization*, Vol.9 No.4, 1082-1099.

[11] Lewis R.M. and Torczon V.(1998), "Pattern search methods for linearly constrained minimization," *ICASE NASA Langley Research Center* TR 98-3. To appear in *SIAM Journal on Optimization.*

[12] Lewis R.M. and Torczon V.(1998), "A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds," *ICASE NASA Langley Research Center* TR 98-31.

[13] Stephens C.P. and Baritompa, W.(1998), "Global optimization requires global information," *Journal of Optimization Theory and Applications* Vol.96 No.3, 575–588.

[14] Torczon V.(1997), "On the convergence of pattern search algorithms," *SIAM Journal on Optimization* Vol.7 No.1, 1–25.

[15] Vanderplaats G.N.(1984), *Numerical optimization techniques for engineering desing: with applications,* McGraw-Hill, New-York, London.

Appendix: Generalized gradients

The contents of this appendix are taken from Clarke [3]. We include it for the convenience of the reader.

Let Y be a subset of $\Re^n$. A function $f : Y \to \Re$ is said to satisfy a *Lipschitz condition on* Y if there exists a nonnegative scalar K such that $|f(y) - f(y')| \leq K\|y - y'\|$ for all y, y' in Y. Let $x \in Y$. The function is said to be *Lipschitz near* x if it satisfies a Lipschitz condition on a ball of radius ϵ around x, for some $\epsilon > 0$.

Let f be Lipschitz near a given point x and let v be a vector in $\Re^n$. The *generalized directional derivative* of f at x in the direction v is

$$f^\circ(x; v) \ := \ \limsup_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t}.$$

The *generalized gradient* of f at x is defined to be the set

$$\partial f(x) \ := \ \{\zeta \in \Re^n : f^\circ(x; v) \geq v^T \zeta \text{ for all } v \in \Re^n\}.$$

The generalized directional derivative may be obtained from the generalized gradient as follows $f^\circ(x; v) = \max\{v^T \zeta : \zeta \in \partial f(x)\}$.

When f is continuously differentiable, $\partial f(x)$ reduces to the singleton $\{\nabla f(x)\}$, and when f is convex, it coincides with the subdifferential.

f is said to be *regular* at x if for all v, the one-sided directional derivative exists and coincides with $f^\circ(x; v)$.

The function f is said to be *strictly differentiable* at x if for all v, $\lim_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t} = \nabla f(x)^T v$.

If f is strictly differentiable at x, then f is Lipschitz near x and $\partial f(x) = \{\nabla f(x)\}$

If f is Lipschitz near x and $\partial f(x)$ reduces to a singleton $\{\zeta\}$, then f is strictly differentiable at x and $\nabla f(x) = \zeta$.

If f is Lipschitz near x and is strictly differentiable at x, then f is regular at x.

If f is Lipschitz near x and is convex, then f is regular at x.