USING SAMPLING AND SIMPLEX DERIVATIVES IN PATTERN SEARCH METHODS

A. L. CUSTÓDIO [*] AND L. N. VICENTE [†]

Abstract. Pattern search methods can be made more efficient if past function evaluations are appropriately reused. In this paper we will introduce a number of ways of reusing previous evaluations of the objective function based on the computation of simplex derivatives (*e.g.*, simplex gradients) to improve the efficiency of a pattern search iteration.

At each iteration of a pattern search method, one can attempt to compute an accurate simplex gradient by identifying a sampling set of previous iterates with good geometrical properties. This simplex gradient computation can be done using only past successful iterates or by considering all past function evaluations.

The simplex gradient can then be used, for instance, to reorder the evaluations of the objective function associated with the positive spanning set or positive basis used in the poll step. But it can also be used to update the mesh size parameter according to a sufficient decrease criterion. None of these modifications demands new function evaluations. A search step can also be tried along the negative simplex gradient at the beginning of the current pattern search iteration.

We will present these procedures in detail and show how promising they are to enhance the practical performance of pattern search methods.

Key words. derivative free optimization, pattern search methods, simplex gradient, simplex Hessian, multivariate polynomial interpolation, poisedness

AMS subject classifications. 65D05, 90C30, 90C56

1. Introduction.

We are interested in this paper in designing efficient pattern search methods for derivative free nonlinear optimization problems. Although most of the strategies introduced here apply to the constrained case when constraints derivatives are available, we will focus our attention on unconstrained optimization problems of the form $\min_{x\in\mathbb{R}^n} f(x)$, where f is a continuous differentiable function.

The curve representing the objective function value as a function of the number of function evaluations frequently exhibits an L-shape for pattern search runs. This class of methods, perhaps because of their directional features, is relatively good in improving the initial guess, quickly decreasing the initial function value. However, these methods tend to be quite slow during the course of the iterations and especially towards stationarity, when the frequency of unsuccessful iterations increases.

There has not been too much effort in trying to develop efficient serial implementations of pattern search methods for the minimization of general functions. Some attention has been paid to parallel pattern search (see Hough, Kolda, and Torczon [11]). Other authors have considered particular instances where the problem structure can be exploited efficiently. Price and Toint [15] examined how to take advantage of partial separability. Alberto *et al* [2] have shown ways of incorporating user provided function evaluations. Abramson, Audet, and Dennis [1] looked at the case where some incomplete form of gradient information is available.

The goal of this paper is to develop a number of strategies for improving the efficiency of a current pattern search iteration, based on function evaluations calcu-

[*]Departamento de Matemática, FCT-UNL, Quinta da Torre 2829-516 Caparica, Portugal (`alcustodio@fct.unl.pt`). Support for this author was provided by Centro de Matemática e Aplicações da Universidade Nova da Lisboa.

[†] Departamento de Matemática, Universidade de Coimbra, 3001-454 Coimbra, Portugal (`lnv@mat.uc.pt`). Support for this author was provided by Centro de Matemática da Universidade de Coimbra and by FCT under grant POCTI/35059/MAT/2000.

lated at previous iterations. We make no use or assumption about the structure of the objective function, so that one can apply the techniques here to any functions (in particular those resulting from running black-box codes or performing physical experiments). More importantly, these strategies (i) require no extra function evaluation (except for the one in the search step where the payoff is clear) and (ii) have no interference in the global convergence requirements typically imposed in these methods.

The paper is organized as follows. Section 2 describes the pattern search framework over which we introduce the material of this paper. Section 3 summarizes geometrical features of sample sets (Λ–poisedness) and simplex derivatives, like simplex gradients and simplex Hessians.

The key ideas of this paper are reported in Section 4, where we show how to use sample sets of points previously evaluated in pattern search to compute simplex derivatives. The sample sets can be built by storing points where the function has been evaluated or by storing only points which lead to a decrease. The main destination of this computation is the efficient ordering of the vectors in the positive spanning set or positive basis used for polling. A descent indicator direction (like a negative simplex gradient) can be used to order the polling directions according to a simple angle criterion.

In Section 5 we describe one way of ensuring sample sets with adequate geometry at iterations succeeding unsuccessful ones. We study the pruning properties of negative simplex gradients in Section 6. Other uses of simplex derivatives in pattern search are suggested in Section 7, namely ways of performing a search step and of updating the mesh size parameter according to a sufficient decrease condition.

Most of these ideas were tested in a set of problems collected from papers on derivative free optimization. The numerical results are presented and discussed in Section 8 and show the effectiveness of using sampling-based simplex derivatives in pattern search. The paper is ended in Section 9 with some concluding remarks and prospects of future work.

2. Pattern search.

Pattern search methods are directional methods that make use of a finite number of directions with appropriate descent properties. In the unconstrained case, these directions must positively span $\mathbb{R}^n$. A positive spanning set is guaranteed to contain one positive basis, but it can contain more. A positive basis is a positive spanning set which has no proper subset positively spanning $\mathbb{R}^n$. Positive bases have between $n+1$ and $2n$ elements. Properties and examples of positive bases can be found in [2, 8, 13]. It is known that pattern search methods exhibit global convergence to stationary points (in the $\liminf$ sense) if one or more than one positive basis are used as long as the number of such bases remains finite.

We present pattern search methods in their generalized format introduced by Audet and Dennis [3]. The positive spanning set used by a pattern search method is represented by D (and its cardinal by $|D|$). It is convenient to view D as an $n \times |D|$ matrix whose columns store the positive generators. A positive basis in D is denoted by B and is also viewed as a matrix (an $n \times |B|$ column submatrix of D).

At each iteration k of a pattern search method, the next iterate x_{k+1} is selected among the points of a mesh M_k, defined as

$$M_k = \{x_k + \alpha_k D z : z \in Z\},$$

where Z is a subset of $\mathbb{Z}^{|D|}$ (containing the canonical basis of $\mathbb{R}^{|D|}$). This mesh is centered at the current iterate x_k and its discretization size is defined by the mesh size parameter α_k. Each direction $d \in D$ must be of the form $d = G\bar{z},\ \bar{z} \in \mathbb{Z}^n$, where G

is a nonsingular (generating) matrix. This property is crucial for global convergence, ensuring that the mesh has only a finite number of points in a compact set (provided that the mesh size parameter is also updated according to some rational requirements, as we will point out later).

The process of finding a new iterate $x_{k+1} \in M_k$ can be described in two phases (the search step and the poll step). The search step is optional and unnecessary for the convergence properties of the method. It consists of evaluating the objective function at a finite number of points in the mesh M_k. The choice of points in M_k is totally arbitrary as long as its number remains finite. The points could be chosen according to specific application properties or following some heuristic algorithm. The search step is declared successful if a new mesh point x_{k+1} is found such that $f(x_{k+1}) < f(x_k)$.

The poll step is only performed if the search step has been unsuccessful. It consists of a local search around the current iterate, exploring the points in the mesh neighborhood defined by Δ_k and by a positive basis $B_k \subset D$:

$$P_k \ = \ \{x_k + \alpha_k b : \ b \in B_k\} \ \subset \ M_k.$$

We call the points $x_k + \alpha_k b \in P_k$ the polling points and the vectors $b \in B_k$ the polling vectors.

The purpose of the poll step is to ensure decrease of the objective function for sufficiently small mesh size parameters. One knows that the poll step must be eventually successful unless the current iterate is a stationary point. In fact, given any vector w in $\mathbb{R}^n$ there exists at least one vector b in B_k such that $w^\top b > 0$. If one selects $w = -\nabla f(x_k)$, one is guaranteed the existence of a descent direction in B_k.

The polling vectors (or points) are ordered according to some criterion in the poll step. In most papers and implementations this ordering is the ordering in which they are originally stored and it is never changed during the course of the iterations. Another ordering that we are aware of consists of bringing into the first column (in B_{k+1}) the polling vector b_k associated to a successful polling iterate ($f(x_k + \alpha_k b_k) < f(x_k)$). We will return to this issue later. Our presentation of pattern search considers that the polling vectors are ordered in some given form before polling starts.

If the poll step also fails to produce a point where the objective function is lower than $f(x_k)$ than both the poll step and the iteration are declared unsuccessful. In this circumstance the mesh size parameter is typically decreased. On the contrary, the mesh size parameter is typically increased if in either the search step or in the poll step a new iterate is found yielding objective function decrease.

The class of pattern search methods used in this paper is described in Figure 2.1. Our description follows the one given in [3] for the generalized pattern search. We leave three procedures undetermined in the statement of the method: the `search` procedure in the search step, the determination of the `order` of the polling vectors, and the procedure `mesh` that updates the mesh size parameter. These procedures are called within squared brackets for better visibility.

The `search` and `order` routines are not asked to meet any requirements for global convergence purposes (rather than finiteness in the search). The `mesh` procedure, however, must update the mesh size parameter as described in Figure 2.2 (this `mesh` procedure is called `mesh-classical` to be distinguished from the one introduced in Section 7).

The global convergence analysis for this class of pattern search methods is divided in two parts. First it is shown that a subsequence of mesh size parameters goes to zero. This result was first proved by Torczon in [17] and it is stated here as Proposition 1.

Pattern Search Method

Initialization
Choose x_0 and $\alpha_0 > 0$. Choose all constants needed for procedures `[search]`, `[order]`, and `[mesh]`. Set $k = 0$.

Search step
Call `[search]` to try to compute a point $x \in M_k$ with $f(x) < f(x_k)$ by evaluating the function only at a finite number of points in M_k. If such a point is found then set $x_{k+1} = x$, declare the iteration as successful, and skip the poll step.

Poll step
Choose a positive basis $B_k \subset D$. Call `[order]` to order the polling set $P_k = \{x_k + \alpha_k b : b \in B_k\}$. Start evaluating f at the polling points following the order determined. If a polling point $x_k + \alpha_k b_k$ is found such that $f(x_k + \alpha_k b_k) < f(x_k)$ then stop polling, set $x_{k+1} = x_k + \alpha_k b_k$, and declare the iteration as successful. Otherwise declare the iteration as unsuccessful and set $x_{k+1} = x_k$.

Updating the mesh size parameter
Call `[mesh]` to compute α_{k+1}. Increment k by one and return to the search step.

FIG. 2.1. *Class of pattern search methods used in this paper.*

`procedure mesh-classical`

If the iteration was successful then maintain or expand mesh by taking $\alpha_{k+1} = \tau^{m_k^+}\alpha_k$, with $m_k^+ \in \{0, 1, 2, \ldots, m_{max}\}$. Otherwise contract mesh, by decreasing the mesh size parameter $\alpha_{k+1} = \tau^{m_k^-}\alpha_k$, with $m_k^- \in \{-1, -2, \ldots\}$.

FIG. 2.2. *Updating the mesh size parameter (for rational lattice requirements). The constant τ must satisfy $\tau \in \mathbb{N}$ and $\tau > 1$ and should be initialized at iteration $k = 0$.*

PROPOSITION 1. *Consider a sequence of iterates $\{x_k\}$ generated by a pattern search method. Assume that $L(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is compact. Then the sequence of the mesh size parameters satisfies $\liminf_{k \to +\infty} \alpha_k = 0$.*

The second part of the proof can be found, for instance, in Audet and Dennis [3] for the generalized pattern search framework. We formalize it here for unconstrained minimization in the continuous differentiable case.

THEOREM 1. *Consider a sequence of iterates $\{x_k\}$ generated by a pattern search method. Assume that $L(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded and that f is a continuously differentiable function in an open set of $\mathbb{R}^n$ containing $L(x_0)$. Then there exists at least one convergent subsequence $\{x_k\}_{k \in K}$ (with limit point x_*) of unsuccessful iterates for which the corresponding subsequence of the mesh size parameters $\{\alpha_k\}_{k \in K}$ converges to zero. For this subsequence $\lim_{k \in K} \nabla f(x_k) = 0$, i.e., $\nabla f(x_*) = 0$.*

Pattern search and direct search methods are surveyed in the excellent SIAM Review paper of Kolda, Lewis, and Torczon [13]. Although we only focus on unconstrained optimization in this paper, we point out that a number of papers have dealt recently with pattern search methods for linearly and nonlinearly constrained optimization (see [4], [13], [14], [16] and references therein).

3. Simplex derivatives.

3. Simplex derivatives. Simplex derivatives of order one are known as simplex gradients. Simplex gradients are used in the implicit filtering method of Bortz and Kelley [5], where the major step of each iteration is a line search along the negative simplex gradient. A simplex gradient is calculated by first computing or selecting a set of sample points. The geometrical properties of the sample set determine the quality of the corresponding simplex gradient as an approximation to the exact gradient of the objective function. In this paper, we use (determined) simplex gradients as well as underdetermined and overdetermined simplex gradients.

A simplex gradient in the determined case is computed by first sampling the objective function at $n+1$ points. The convex hull of a set of $n+1$ points $\{y^0, y^1, \ldots, y^n\}$ is called a simplex. The $n+1$ points are called vertices. The simplex is said to be nonsingular if the matrix $S = [\, y^1 - y^0 \;\cdots\; y^n - y^0 \,]$ is nonsingular. Given a nonsingular simplex of vertices $y^0, y^1, \ldots, y^n$, the simplex gradient at y^0 is defined as $\nabla_S f(y^0) = S^{-\top}\delta(f;S)$ with $\delta(f;S) = [\, f(y^1) - f(y^0) \;\cdots\; f(y^n) - f(y^0) \,]^\top$.

The simplex gradient is intimately related to linear multivariate polynomial interpolation. In fact, it is easy to see that the linear model $m(y) = f(y^0) + \nabla_S f(y^0)^\top (y - y^0)$ centered at y^0 interpolates f at the points $y^1, \ldots, y^n$.

In practical instances one might have less then or more then $n+1$ points. We will see instances in this paper where the number of points available for a simplex gradient calculation is different from $n+1$. The definition given in the next paragraph describes the extension of simplex gradients to underdetermined and overdetermined cases.

A sample set is said to be *poised* for the simplex gradient calculation if S is full rank, *i.e.*, if $\mathrm{rank}(S) = \min\{n, q\}$. Given the sample set $\{y^0, y^1, \ldots, y^q\}$, the simplex gradient $\nabla_S f(y^0)$ of f at y^0 can be defined as the "solution" of the system

$$S^\top g \;=\; \delta(f;S),$$

where $S = [\, y^1 - y^0 \;\cdots\; y^q - y^0 \,]$ and $\delta(f;S) = [\, f(y^1) - f(y^0) \;\cdots\; f(y^q) - f(y^0) \,]^\top$. This system is solved in the least-squares sense if $q > n$. A minimum norm solution is computed if $q < n$. This definition includes the determined case ($q = n$) as a particular case.

The formulae for the under and over determined simplex gradients can be expressed using the singular value decomposition of the matrix $S^\top$. However, to deal with the geometrical properties of the poised sample set and to better express the error bound for the corresponding gradient approximation, it is appropriated to take the SVD of a scaled form of $S^\top$. For this purpose, let

$$\Delta \;=\; \max_{1 \le i \le q} \|y^i - y^0\|$$

be the radius of the smallest enclosing ball of $\{y^0, y^1, \ldots, y^q\}$ centered at y^0. Now we write the SVD of the scaled matrix $S^\top/\Delta = U \Sigma V^\top$, which corresponds to a sample set in a ball of radius one centered around y^0. The underdetermined and overdetermined simplex gradients are both given by $\nabla_S f(y^0) = V \Sigma^{-1} U^\top \delta(f;S)/\Delta$.

The accuracy of simplex gradients is summarized in the following theorem. The proof of the determined case ($q = n$) is given, for instance, in Kelley [12]. The extension of the analysis to the nondetermined cases is developed in Conn, Scheinberg, and Vicente [6].

Theorem 2. *Let $\{y^0, y^1, \ldots, y^q\}$ be a poised sample set for a simplex gradient calculation in $\mathbb{R}^n$. Consider the enclosing (closed) ball $B(y^0;\Delta)$ of this sample set,* *centered at y^0, where $\Delta = \max_{1\le i\le q} \|y^i - y^0\|$. Let $S = [\, y^1 - y^0 \ \cdots \ y^q - y^0 \,]$ and let $U\Sigma V^\top$ be the SVD of $S^\top/\Delta$.*

Assume that f is continuously differentiable in an open domain Ω containing $B(y^0;\Delta)$ and that ∇f is Lipschitz continuous in Ω with constant $\gamma > 0$.

Then the error of the simplex gradient at y^0, as an approximation to $\nabla f(y^0)$, satisfies

$$\|V^\top[\nabla f(y^0) - \nabla_S f(y^0)]\| \ \le \ \left(q^{\frac{1}{2}} \frac{\gamma}{2} \|\Sigma^{-1}\| \right) \Delta,$$

where $V = I$ if $q \ge n$.

Notice that the error difference is projected over the null space of $S^\top/\Delta$. Unless we have enough points ($q + 1 \ge n + 1$), there is no guarantee of accuracy for the simplex gradient. Despite this observation, underdetermined simplex gradients contain relevant gradient information for q close to n and might be of some value in computations where the number of sample points is relatively low.

The quality of the error bound of Theorem 2 depends on the size of the constant $\sqrt{q}\gamma\|\Sigma^{-1}\|/2$ which multiplies Δ. This constant, in turn, depends essentially on the Lipschitz constant γ (which is unknown) and on $\|\Sigma^{-1}\|$ (which is associated to the sample set).

Conn, Scheinberg, and Vicente [7] introduced an algorithmic framework for building and maintaining sample sets with good geometry. They have suggested the notion of a Λ–poised sample set, where Λ is a positive constant. The notion of Λ–poisedness is closely related to Lagrange interpolation in the determined case. If a sample set $\{y^0, y^1, \ldots, y^q\}$ is Λ–poised in the sense of [7] then one can prove that $\|\Sigma^{-1}\|$ is bounded by a multiple of Λ. For the purpose of this paper, it is enough to consider $\|\Sigma^{-1}\|$ as a measure of the well-poisedness (quality of the geometry) of our sample sets. We will therefore say that a poised sample set is Λ–poised if $\|\Sigma^{-1}\| \le \Lambda$, for some positive constant Λ.

We do not need algorithms to build or maintain Λ–poised sets. Rather, we are given a sample set at each iteration of a pattern search method, and our goal is just to identify a poised subset of it that is Λ–poised. The constant $\Lambda > 0$ is chosen at iteration $k = 0$.

The notion of simplex gradient can be extended to higher order derivatives [6]. One can consider the computation of a simplex Hessian, by extending the linear system $S^\top g = \delta(f;S)$ to

$$(y^i - y^0)^\top g + \frac{1}{2}(y^i - y^0)^\top H (y^i - y^0) \ = \ f(y^i) - f(y^0), \quad i = 1, \ldots, p.$$

The number of points in the sample set $Y = \{y^0, y^1, \ldots, y^p\}$ must be equal to $p + 1 = (n+1)(n+2)/2$ if one wants to compute a full symmetric simplex Hessian. Similarly to the linear case, the simplex gradient $g = \nabla_S f(y^0)$ and the simplex Hessian $H = \nabla^2_S f(y^0)$ computed from the above system with $p+1 = (n+1)(n+2)/2$ points coincide with the coefficients of the quadratic multivariate polynomial interpolation model associate with Y. The notions of poisedness and Λ–poisedness and the derivation of the error bounds for simplex Hessians in determined and nondetermined cases is reported in [6].

In our application to pattern search we are interested in using sample sets with a relatively low number of points. One alternative is to consider less points than coefficients and to compute solutions in the minimum norm sense. Another process is to choose to approximate only some portions of the simplex Hessian. For instance, if one is given $2n+1$ points one can compute the n components of a simplex gradient and an approximation to the n diagonal terms of a simplex Hessian. The system to be solved in this case is of the form

$$\left[\begin{array}{ccc} y^1 - y^0 & \cdots & y^{2n} - y^0 \\ (1/2)(y^1 - y^0).\hat{\ }2 & \cdots & (1/2)(y^{2n} - y^0).\hat{\ }2 \end{array}\right]^\top \left[\begin{array}{c} g \\ \mathrm{diag}(H) \end{array}\right] \;=\; \delta(f;S),$$

where $\delta(f;S) = [\, f(y^1) - f(y^0) \;\cdots\; f(y^{2n}) - f(y^0)\,]^\top$ and the notation $.\hat{\ }2$ stands for component-wise squaring. Once again, if the number of points is lower than $2n+1$ a minimum norm solution can be computed.

4. Ordering the polling in pattern search. A pattern search method generates a number of function evaluations at each iteration. One can store some of these points and corresponding objective function values during the course of the iterations. Thus, at the beginning of each iteration of a pattern search method, one can try to identify a subset of these points with some desirable geometrical properties (Λ–poisedness in our context).

If we are successful in such an attempt, we compute some form of simplex derivatives. For instance, we can calculate a simplex gradient. Using these simplex derivatives, we can compute a direction of potential descent or of potential steepest descent (a negative simplex gradient for example). We call such direction a *descent indicator*. It cost us no additional function evaluation to compute a descent indicator. There might be iterations (especially at the beginning) where we fail to compute a descent indicator but such failures cost no extra function evaluations.

We adapt the description of pattern search to follow the approach described above. The class of pattern search methods remains essentially the same and it is spelled out in Figure 4.1. A new procedure named `store` is called every time a new function evaluation is made. The algorithm maintains a list of points X_k of maximum size p_{max}. The points are added (or not) to this list by `store`.

A new step is included at the beginning of each iteration to take care of the simplex derivatives calculation. In this step, the algorithm attempts first to extract from X_k a sample set Y_k with appropriate size and desirable geometrical properties. The points in Y_k must be within a distance of Δ_k to the current iterate x_k. The size of the radius Δ_k is chosen such that $B(x_k;\Delta_k)$ contains all the points in $P_k = \{x_k + \alpha_k b : \; b \in B_k\}$, where B_k is the polling positive basis to be chosen in the poll step. In other words, we choose

$$\Delta_k \;=\; \sigma\, \alpha_k \max_{b \in B_k} \|b\|,$$

where $\sigma \geq 1$ is a constant fixed *a priori* for all iterations. All the modifications to the algorithm reported in Figure 2.1 are marked in italic in Figure 4.1 for better identification. The fact that B_k has not be chosen yet is not restrictive. We could have set $\Delta_k = \sigma\, \alpha_k \max_{b \in B_{k-1}} \|b\|$ and nothing would have changed in this paper. We keep B_k instead of B_{k-1} mostly because we want to highlight the fact that the sample set Y_k is part of a ball of the same radius (when $\sigma = 1$) of the smallest enclosing ball containing the polling set P_k.

It is possible to implement different criteria for deciding whether to store or not a point where the function has been evaluated. In this paper, we consider the two following simple ways of storing points:

Pattern Search Method — Using Sampling and Simplex Derivatives

Initialization
Choose x_0 and $\alpha_0 > 0$. Choose all constants needed for procedures `[search]`, `[order]`, and `[mesh]`. Set $k = 0$. *Set* $X_0 = [x_0]$ *to initialize the list of points maintained by* `[store]`. *Choose a maximum* p_{max} *number of points that can be stored. Choose also the minimum* s_{min} *and the maximum* s_{max} *number of points involved in any simplex derivatives calculation. Choose* $\Lambda > 0$ *and* $\sigma > 0$.

Identifying a Λ–poised sample set and computing simplex derivatives
Skip this step if there are not enough points, i.e., if $|X_k| < s_{min}$. *Set* $\Delta_k = \sigma\,\alpha_k \max_{b\in B_k}\|b\|$. *Try to identify a set of points* Y_k *in* $X_k \cap B(x_k;\Delta_k)$, *with as many points as possible (up to* s_{max}*) and such that* Y_k *is* Λ*–poised and includes the current iterate* x_k. *If* $|Y_k| \geq s_{min}$ *compute some form of simplex derivatives based on* Y_k *(and from that compute a descent indicator* d_k*).*

Search step
Call `[search]` to try to compute a point $x \in M_k$ with $f(x) < f(x_k)$ by evaluating the function only at a finite number of points in M_k *and calling* `[store]` *each time a point is evaluated.* If such a point is found then set $x_{k+1} = x$, declare the iteration as successful, and skip the poll step.

Poll step
Choose a positive basis $B_k \subset D$. Call `[order]` to order the polling set $P_k = \{x_k + \alpha_k b : b \in B_k\}$. Start evaluating f at the polling points following the order determined *and calling* `[store]` *each time a point is evaluated.* If a polling point $x_k + \alpha_k b_k$ is found such that $f(x_k + \alpha_k b_k) < f(x_k)$ then stop polling, set $x_{k+1} = x_k + \alpha_k b_k$, and declare the iteration as successful. Otherwise declare the iteration as unsuccessful and set $x_{k+1} = x_k$.

Updating the mesh size parameter
Call `[mesh]` to compute α_{k+1}. Increment k by one and return to the simplex derivatives step.

FIG. 4.1. *Class of pattern search methods used in this paper, adapted now for identifying Λ–poised sample sets and computing simplex derivatives.*

- `store-successful`: in this case `store` keeps only the successful iterates x_{k+1} (for which $f(x_{k+1}) < f(x_k)$). The points in the list X_k are therefore ordered by decreasing objective function values.
- `store-all`: corresponds to the case where every point (for which the objective function is computed) is stored, independently of increasing or decreasing $f(x_k)$.

In both cases, the incoming points are added to X_k at the end of the list. When (and if) X_k has reached its predetermined size p_{max}, we must remove a point first before adding a new one. We assume that points are removed from the beginning of the list. Note that both variants store successful iterates x_{k+1} (for which $f(x_{k+1}) < f(x_k)$). It is thus obvious that the current iterate x_k is always in X_k, when `store-successful` is chosen. However, if one chooses `store-all`, and without any further provision, the current iterate x_k could had been removed from the list if a number of unsuccessful iterates would occurred consecutively. We must therefore assume that the current

```
procedure order
    Compute $\cos(d_k, b)$ for all $b \in B_k$. Order the columns in $B_k$ according to decreasing
    values of the corresponding cosines.
```

FIG. 4.2. *Ordering the polling vectors according to their angle distance to the descent indicator.*

iterate is never removed from the list in the `store-all` variant.

Having a descent indicator d_k at hands, we can order the polling vectors according to the increasing amplitudes of the angles between d_k and the polling vectors. So, the first polling point to be evaluated is the one corresponding to the polling vector that lead to the angle of smallest amplitude. We describe such procedure `order` in Figure 4.2 and illustrate it in Figure 4.3.

The descent indicator could be a negative simplex gradient $d_k = -\nabla_{S_k} f(x_k)$, where $S_k = [\, y_k^1 - x_k \ \cdots \ y_k^{q_k} - x_k \,]$ is formed from the sample set $Y_k = [\, y_k^0 \ y_k^1 \ \cdots \ y_k^{q_k} \,]$, with $q_k + 1 = |Y_k|$ and $y_k^0 = x_k$. This way of calculating the simplex derivatives and the descent indicator will be designated by `sgradient`.

Another possibility is to compute $d_k = -H_k^{-1} g_k$, where g_k is a simplex gradient and H_k approximates a simplex Hessian (in this paper we will test numerically the diagonal simplex Hessians described at the end of Section 3). This way of calculating the simplex derivatives and the descent indicator will be designated by `shessian`.

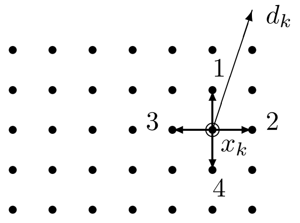

FIG. 4.3. *Ordering the polling vectors using a descent indicator. The positive basis considered is $B_k = [\, -I \ I \,]$.*

5. Geometry of the sample sets. If the points where the objective function is evaluated are added to the list X_k according to the `store-all` criterion, it is possible to guarantee the quality of the sample sets Y_k used to compute the simplex derivatives after the occurrence of unsuccessful iterations.

Let us focus on the case where our goal is to compute simplex gradients. To simplify the presentation we assume that there exists only one positive basis B in the positive spanning set D. Let us assume too that s_{min} has been chosen so that $s_{min} \geq |B|$, in other words, that we require at least $|B|$ points in X_k with appropriate geometry to compute a simplex gradient.

If the iteration $k-1$ was unsuccessful then there were at least $|B|$ points added to the list (the polling points $x_{k-1} + \alpha_{k-1} b$, for all $b \in B$). Such points are part of X_k as well as the current iterate $x_k = x_{k-1}$.

As we show now, the sample set $Y_k \subset X_k$ formed by x_k and by these $|B|$ points is poised for a simplex gradient calculation. Let us write $Y_k = [\, y_k^0 \ y_k^1 \ \cdots \ y_k^{q_k} \,]$ with $q_k + 1 = |Y_k| = |B| + 1$ and $y_k^0 = x_k$. Then

$$S_k \;=\; [\, y_k^1 - x_k \ \cdots \ y_k^{q_k} - x_k \,] \;=\; [\, \alpha_{k-1} b_1 \ \cdots \ \alpha_{k-1} b_{|B|} \,] \;=\; \alpha_{k-1} B.$$

The matrix B has rank n since any positive spanning set or positive basis linearly spans $\mathbb{R}^n$. If we choose $\sigma = 2$ in the formula for Δ_k, we get

$$\frac{1}{\Delta_k} S_k \;=\; \frac{\alpha_{k-1}}{2\,\alpha_k \max_{b \in B} \|b\|} B \;=\; \frac{1}{\max_{b \in B} \|b\|} B.$$

Thus, the geometry constant associated with this sample set Y_k is constant and given by

$$\|\Sigma^{-1}\| \qquad \text{with} \qquad \frac{1}{\max_{b \in B} \|b\|} B \;=\; U \Sigma V^\top.$$

If we choose the poisedness constant such that $\Lambda \geq \|\Sigma^{-1}\|$ then we are guaranteed to identify a Λ-poised sample set after each unsuccessful iteration.

The sample set $Y_k \subset X_k$ formed by x_k and by only $|B| - 1$ of the points $x_{k-1} + \alpha_{k-1} b$, where $b \in B$, is also poised for a simplex gradient calculation. In this case $q_k + 1 = |Y_k| = |B|$ and

$$S_k \;=\; \alpha_{k-1} B_{|B|-1},$$

where $B_{|B|-1}$ is some column submatrix of B with $|B| - 1$ columns. Since B is a positive spanning set then $B_{|B|-1}$ linearly spans $\mathbb{R}^n$ (see [8, Theorem 3.7]). It results that the matrix $B_{|B|-1}$ has rank n. We take $\sigma = 2$ as before in the formula for Δ_k. The difference now is that we must take into consideration all submatrices $B_{|B|-1}$ of B. Thus, if we choose the poisedness constant such that

$$\Lambda \;\geq\; \max \left\{ \|\Sigma^{-1}\| : \; \frac{1}{\max_{b \in B} \|b\|} B_{|B|-1} \;=\; U \Sigma V^\top, \quad \forall\, B_{|B|-1} \subset B \right\},$$

we are also guaranteed to identify a Λ-poised sample set after each unsuccessful iteration.

6. Pruning the polling directions. Abramson, Audet, and Dennis [1] have shown for a special choice of positive spanning set D that rough approximations to the gradient of the objective function can reduce the polling step to a single function evaluation. The gradient approximations considered were ϵ–approximations to the large components of the gradient vector.

Let g be a nonzero vector in $\mathbb{R}^n$ and $\epsilon \geq 0$. Consider

$$J^\epsilon(g) \;=\; \left\{ i \in \{1, \ldots, n\} : |g_i| + \epsilon \geq \|g\|_\infty \right\},$$

and for every $i \in \{1, \ldots, n\}$ let

$$d^\epsilon(g)_i \;=\; \begin{cases} \operatorname{sign}(g_i) & \text{if } i \in J^\epsilon(g), \\ 0 & \text{otherwise.} \end{cases}$$

The vector g is said to be an ϵ–approximation to the large components of a nonzero vector $v \in \mathbb{R}^n$ if and only if $i \in J^\epsilon(g)$ whenever $|v_i| = \|v\|_\infty$ and $\operatorname{sign}(g_i) = \operatorname{sign}(v_i)$ for every $i \in J^\epsilon(g)$.

The question that arises now is whether a descent indicator d_k, and in particular a negative simplex gradient $-\nabla_{S_k} f(x_k)$, is an ϵ–approximation to the large components of $-\nabla f(x_k)$, for some $\epsilon > 0$. We show in the next theorem that the answer is affirmative provided that the mesh size parameter α_k is sufficiently small, an issue we will return at the end of this section. We will use the notation previously introduced in this paper. We consider a sample set Y_k and the corresponding matrix S_k. The set Y_k is included in the ball $B(x_k;\Delta_k)$ centered at x_k with radius $\Delta_k = \sigma\,\alpha_k\,\max_{b\in B_k}\|b\|$, where B_k is the positive basis used for polling.

THEOREM 3. *Let Y_k be a Λ–poised sample set (for simplex gradients) computed at iteration k of a pattern search method.*

Assume that f is continuously differentiable in an open domain Ω containing $B(x_k;\Delta_k)$ and that ∇f is Lipschitz continuous in Ω with constant $\gamma > 0$.

Then, if

$$\alpha_k \;\le\; \frac{\|\nabla f(x_k)\|_\infty}{\sqrt{q}\gamma\Lambda\sigma\max_{b\in B_k}\|b\|}, \tag{6.1}$$

the negative simplex gradient $-\nabla_S f(x_k)$ is an ϵ_k–approximation to the large components of $-\nabla f(x_k)$, where

$$\epsilon_k \;=\; \left(q^{\frac{1}{2}}\gamma\Lambda\sigma\max_{b\in B_k}\|b\|\right)\alpha_k.$$

Proof. For i in the index set

$$I_k \;=\; \{i\in\{1,\ldots,n\}:\;|\nabla f(x_k)_i| = \|\nabla f(x_k)\|_\infty\},$$

we get from Theorem 2 that

$$\begin{aligned}
\|\nabla_{S_k} f(x_k)\|_\infty \;&\le\; \|\nabla f(x_k) - \nabla_{S_k} f(x_k)\|_\infty + |\nabla f(x_k)_i| \\
&\le\; 2\|\nabla f(x_k) - \nabla_{S_k} f(x_k)\| + |\nabla_{S_k} f(x_k)_i| \\
&\le\; q^{\frac{1}{2}}\gamma\Lambda\Delta_k + |\nabla_{S_k} f(x_k)_i| \\
&=\; \epsilon_k + |\nabla_{S_k} f(x_k)_i|.
\end{aligned}$$

From Theorem 2 we also know that

$$-\nabla_{S_k} f(x_k)_i \;=\; -\nabla f(x_k)_i + \xi_{k,i}, \quad \text{where} \quad |\xi_{k,i}| \le q^{\frac{1}{2}}\frac{\gamma}{2}\Lambda\Delta_k.$$

If $-\nabla f(x_k)_i$ and $\xi_{k,i}$ are equally signed so are $-\nabla f(x_k)_i$ and $-\nabla_{S_k} f(x_k)_i$. Otherwise, they are equally signed if

$$|\xi_{k,i}| \;\le\; q^{\frac{1}{2}}\frac{\gamma}{2}\Lambda\Delta_k \;\le\; \|\nabla f(x_k)\|_\infty \;=\; \frac{1}{2}|\nabla f(x_k)_i|.$$

The proof is concluded using the expression for Δ_k and the bound for α_k given in the statement of the theorem. $\square$

Theorem 4 in Abramson, Dennis, and Audet [1] shows that an ϵ–approximation prunes the set of the polling directions to a singleton, when considering

$$D \;=\; \{-1,0,1\}^n$$

and the positive spanning set

$$D_k \;=\; \{d^\epsilon(g_k)\}\cup\mathbb{A}(-\nabla f(x_k)),$$

where g_k is an ϵ–approximation to $-\nabla f(x_k)$ and

$$\mathbb{A}(-\nabla f(x_k)) \;=\; \{d \in D : -\nabla f(x_k)^\top d < 0\}$$

represents the set of the ascents directions in D. The pruning is to the singleton $\{d^\epsilon(g_k)\}$, meaning that $d^\epsilon(g_k)$ is the only vector d in D_k such that $-\nabla f(x_k)^\top d \geq 0$.

So, under the hypotheses of Theorem 3, the negative simplex gradient $-\nabla_{S_k} f(x_k)$ prunes the positive spanning set of $\mathbb{R}^n$,

$$D_k \;=\; \{d^{\epsilon_k}(-\nabla_{S_k} f(x_k))\} \cup \mathbb{A}(-\nabla f(x_k)),$$

to a singleton, namely $\{d^{\epsilon_k}(-\nabla_{S_k} f(x_k))\}$, where ϵ_k is given in Theorem 3.

Now we analyze in more detail the role of condition (6.1). There is no guarantee that this condition on α_k can be satisfied assymptoticaly. Condition (6.1) gives us only an indication of the pruning effect of the negative simplex gradient, and it is more likely to be satisfied at points where the gradient is relatively large. What is known is actually a condition that shows that α_k dominates $\|\nabla f(x_k)\|$ at unsuccessful iterations k:

$$\|\nabla f(x_k)\| \;\leq\; \left(\gamma \kappa(B_k)^{-1} \max_{b \in B_k} \|b\|\right) \alpha_k,$$

where

$$\kappa(B_k) \;=\; \min_{d \in \mathbb{R}^n} \max_{b \in B_k} \frac{d^\top b}{\|d\|\|b\|} \;>\; 0$$

is the cosine measure of the positive basis B_k (see [13, Theorem 3.3]). Since only a finite number of positive bases is used, $\kappa(B_k)^{-1}$ is uniformly bounded. So, one can be assured that at unsuccessful iterations the norm of the gradient is bounded by a constant times α_k.

However, it has been observed in [9] for some problems that α_k goes to zero typically faster than $\|\nabla f(x_k)\|$. Our numerical experience with pattern search has also pointed us in this direction. It is harder however to sharply verify condition (6.1) since it depends on the Lipschitz constant of ∇f. A detailed numerical study of these asymptotic behaviors is out of the scope of this paper.

7. Other uses for simplex derivatives. Having computed before some form of simplex derivatives, one can use the available information for purposes rather then just ordering the polling vectors. In this section, we suggest two other uses for simplex derivatives in pattern search: the computation of a search step and the update of the mesh size parameter.

There are many possibilities for a search step. One possibility is to first form a surrogate model $m_k(y)$ based on some form of simplex derivatives computed using the sample set Y_k, and then to minimize this model in $B(x_k; \Delta_k)$. At the end we would project the minimizer onto the mesh M_k. If the model $m_k(y)$ is linear and purely based on the descent indicator, *i.e.*, if $m_k(y) = f(x_k) - d_k^\top (y - x_k)$ then this procedure is described in Figure 7.1. A natural choice for d_k is $-\nabla_{S_k} f(x_k)$ but other descent indicators d_k could be used. As we said before, we could set $d_k = -H_k^{-1} g_k$, where g_k is a simplex gradient and H_k approximates a simplex Hessian.

The model $m_k(y)$ could be used for imposing a sufficient decrease condition on the update of the mesh size parameter α_k. We describe one such procedure in Figure 7.2,

```
procedure search
Compute
    $$x = \text{proj}\left(x_k + \frac{\Delta_k}{\|d_k\|} d_k\right) \quad \text{with} \quad \Delta_k = \sigma \alpha_k \max_{b \in B_k} \|b\|,$$
where proj(·) represents the projection operator onto the mesh $M_k$.
```

FIG. 7.1. *A search step based on the descent indicator.*

```
procedure mesh
If the iteration was successful then compute
    $$\rho_k = \frac{f(x_k) - f(x_{k+1})}{m_k(x_k) - m_k(x_{k+1})}.$$
    If $\rho_k > \gamma_2$                    then  $\alpha_{k+1} = \tau^{m_k^+} \alpha_k$,
    If $\gamma_1 < \rho_k \leq \gamma_2$      then  $\alpha_{k+1} = \alpha_k$,
    If $\rho_k \leq \gamma_1$                 then  $\alpha_{k+1} = \tau^{m_k^-} \alpha_k$.
Otherwise contract mesh, by decreasing the mesh size parameter $\alpha_{k+1} = \tau^{m_k^-} \alpha_k$.
The exponents satisfy $m_k^- \in \{-1, -2, \ldots\}$ and $m_k^+ \in \{0, 1, 2, \ldots\}$.
```

FIG. 7.2. *Updating the mesh size parameter (using sufficient decrease but meeting rational lattice requirements). The constants τ, γ_1, and γ_2 must satisfy $\tau \in \mathbb{N}$, $\tau > 1$, and $\gamma_2 > \gamma_1 > 0$ and should be initialized at iteration $k = 0$.*

where the sufficient decrease is only applied to successful iterations. For a linear model computed using a simplex gradient, we get

$$\rho_k = \frac{f(x_k) - f(x_{k+1})}{m_k(x_k) - m_k(x_{k+1})} = \frac{f(x_k) - f(x_{k+1})}{-\nabla_{S_k} f(x_k)^\top (x_{k+1} - x_k)}.$$

If x_{k+1} is computed in the poll step then $x_{k+1} - x_k = \alpha_k b_k$. Since the expansion and contraction parameters are restricted to integer powers of τ and the contraction rules match what was given in the mesh procedure of Figure 2.2, this modification has no influence on the global convergence properties of the underlying pattern search method.

8. Implementation and numerical results.

We have implemented a basic pattern search algorithm of the form given in Figure 2.1, without any search step and using the classical update of the mesh size parameter reported in Figure 2.2. We have tried positive spanning sets with only one positive basis, setting $D = B$. The order of the evaluations in the poll step of the basic implementation is the so-called consecutive poll ordering, where the columns in B are never reordered during the course of the iterations, staying always ordered as originally. We have tested two positive bases: $[-e\ I]$ and $[-I\ I]$. The first one, with $n + 1$ elements, was according to our tests one of the most efficient in terms of function evaluations. The second one, with $2n$ elements, corresponds to coordinate search and provides more accurate final iterates.

As we have mentioned in Section 2, another ordering for polling vectors consists of bringing into the first column (in B_{k+1}) the polling vector b_k associated to a successful polling iterate $(f(x_k + \alpha_k b_k) < f(x_k))$. This ordering procedure has been called dynamic polling (see [4]). Our numerical tests have shown that dynamic polling is worse that consecutive polling for ordering the polling vectors. On our test set, dynamic polling took more 1.75% iterations when using $B = [\,-e\ I\,]$ and more 0.48% iterations when using $B = [\,-I\ I\,]$, compared to consecutive polling. (The quality of the final iterates in terms of the optimal objective gap was similar.) As a result, we decided to use consecutive polling in our basic version of pattern search.

We have tested a number of pattern search methods of the form described in Figure 4.1. The strategies `order` (Figure 4.2), `search` (Figure 7.1), and `mesh` (Figure 7.2) were run in four different modes according to the way of storing points (`store-successful` or `store-all`) and to the way of computing simplex derivatives and descent indicators (`sgradient` or `shessian`). Each of these 28 combined strategies was compared against the basic pattern search method (`consecutive polling & mesh-classical`), for the bases $[\,-e\ I\,]$ and $[\,-I\ I\,]$.

The algorithms were coded in MATLAB and tested on a set of 27 problems of the CUTEr collection [10], with dimensions mostly equal to 10 and 20 (see Table 8.1). The starting points used were those reported in CUTEr. The stopping criterion consisted of the mesh size parameter becoming lower than 10^{-5} or a maximum number of 10000 iterations being reached.

problem	**dimension**
arwhead	10, 20
bdqrtic	10, 20
bdvalue	10, 20
biggs6	6
brownal	10, 20
broydn3d	10, 20
integreq	10, 20
penalty1	10, 20
penalty2	10, 20
powellsg	12, 20
srosenbr	10, 20
tridia	10, 20
vardim	10, 20
woods	10, 20

TABLE 8.1
Problem set used for numerical tests.

The simplex derivatives were computed based on Λ-poised sets Y_k, where $\Lambda = 100$ and $\sigma = 2$. The values for the parameters s_{min}, s_{max}, and p_{max} are given in Tables 8.2 and 8.3. We started all runs with the mesh size parameter $\alpha_0 = 1$. The contraction factor was set to $\tau^{m_k^-} = 0.5$ and the expansion factor to $\tau^{m_k^+} = 2$. In the `mesh` strategy of Figure 7.2, we set γ_1 and γ_2 to 0.25 and 0.75, respectively.

size	`store-successful`	`store-all`
p_{max}	$2(n+1)$	$4(n+1)$
s_{min}	$(n+1)/2$	$n+1$
s_{max}	$n+1$	$n+1$

TABLE 8.2
Sizes of the list X_k and the set Y_k for `sgradient`.

size	`store-successful`	`store-all`
p_{max}	$4(n+1)$	$8(n+1)$
s_{min}	n	$2n+1$
s_{max}	$2n+1$	$2n+1$

TABLE 8.3
Sizes of the list X_k and the set Y_k for `shessian`.

The numerical results are reported in Tables 8.4 and 8.5 for the basis $[-e\ I]$ and in Tables 8.6 and 8.7 for the basis $[-I\ I]$. Our main conclusions are summarized below.

1. 47 out of the 56 versions tried lead to an average decrease in the number of iterations. In 9 of these 47 versions (marked with a $*$ in the tables) the algorithm decreased (or maintained) the number of iterations *for all the problems* in the test set.
2. In terms of the quality of the answer obtained, we observe that the simplex derivatives versions performed generally better than the `consecutive polling & mesh-classical` version to obtain final iterates distancing less than 10^{-4} to the optimal objective value. In the $n+1$ basis $[-e\ I]$, the simplex derivatives versions did not obtain so high accurate answers (optimal gap under 10^{-7}), although we see an improvement from `sgradient` to `shessian`. In the $2n$ basis $[-I\ I]$, the situation is different, where some increase in function evaluations gave rise to higher accurate final iterates.
3. The effect of the poll ordering is more visible using the basis $[-I\ I]$ due to the larger number of polling vectors. In this case, the decrease in function evaluations reached $12-13\%$ in the `sgradient` case without any deterioration of the quality of the final iterate.
4. The performance of the update of the mesh size parameter using sufficient decrease (`mesh`) was a surprise to us. Even when applied individually lead to a significant decrease in function evaluations for medium quality answers. The best strategies over all included `mesh`.
5. The `search` strategy made a clear positive impact when using the smaller basis $[-e\ I]$, both in terms of cost and of quality. This effect was lost in the larger basis $[-I\ I]$, where the fine quality of the final iterates was already quite good without any search step.

The most promising strategy seems to be the one that combines `order`, `mesh`, and `search` in the variants `store-all` and `sgradient`. The strategy `mesh` and the strategy `mesh` and `order` also provided good overall results, especially when tried in the `sgradient` mode.

We picked some of these problems and ran several versions for $n = 40$ and $n = 80$. Our conclusions remain essentially the same. The ratios of improvement in the number of function evaluations and the quality of the final iterates do not change significantly with the increase of the dimension of the problem, but rather with the increase of the number of polling vectors in the positive spanning set (as we have seen from $[-e\ I]$ to $[-I\ I]$).

9. Conclusions.

We have proposed the use of simplex derivatives in pattern search methods in three ways: ordering the polling vectors, updating the mesh size

strategy	**number of evaluations**	**optimal gap**		
		10^{-7}	10^{-4}	10^{-1}
`consecutive polling & mesh-classical`	—	29.63%	44.44%	66.67%
`order (store-successful)`	-3.22%	29.63%	44.44%	66.67%
`mesh (store-successful)`	-35.50%	3.70%	40.74%	66.67%
`order,mesh (store-successful)`	-49.79%*	7.41%	44.44%	66.67%
`search (store-successful)`	-22.30%	33.33%	51.85%	74.07%
`order,search (store-successful)`	-24.07%	33.33%	48.15%	70.37%
`mesh,search (store-successful)`	-45.13%	14.81%	48.15%	74.07%
`order,mesh,search (store-successful)`	-50.40%*	22.22%	48.15%	70.37%
`order (store-all)`	5.48%	29.63%	44.44%	66.67%
`mesh (store-all)`	-49.21%*	22.22%	48.15%	66.67%
`order,mesh (store-all)`	-55.43%	18.52%	48.15%	66.67%
`search (store-all)`	-5.58%	33.33%	48.15%	74.07%
`order,search (store-all)`	-24.32%	33.33%	55.56%	77.78%
`mesh,search (store-all)`	-64.49%	18.52%	48.15%	77.78%
`order,mesh,search (store-all)`	-67.94%*	18.52%	48.15%	77.78%

TABLE 8.4

Variation in the number of function evaluations by comparison to the basic pattern search method (second column) and cumulative optimal gaps for final iterates (third to fifth columns). Case `sgradient` *and* $B = [\,-e\ I\,]$.

strategy	**number of evaluations**	**optimal gap**		
		10^{-7}	10^{-4}	10^{-1}
`consecutive polling & mesh-classical`	—	29.63%	44.44%	66.67%
`order (store-successful)`	-1.27%	29.63%	44.44%	66.67%
`mesh (store-successful)`	-25.27%*	14.81%	44.44%	66.67%
`order,mesh (store-successful)`	-32.25%*	11.11%	44.44%	66.67%
`search (store-successful)`	-15.99%	33.33%	48.15%	66.67%
`order,search (store-successful)`	-18.10%	33.33%	48.15%	66.67%
`mesh,search (store-successful)`	-30.10%*	25.93%	48.15%	70.37%
`order,mesh,search (store-successful)`	-32.32%	25.93%	48.15%	66.67%
`order (store-all)`	7.33%	29.63%	44.44%	66.67%
`mesh (store-all)`	-15.29%	22.22%	44.44%	66.67%
`order,mesh (store-all)`	-20.50%	18.52%	44.44%	66.67%
`search (store-all)`	-1.91%	33.33%	44.44%	66.67%
`order,search (store-all)`	4.77%	29.63%	44.44%	66.67%
`mesh,search (store-all)`	-3.61%	22.22%	48.15%	70.37%
`order,mesh,search (store-all)`	-13.61%	18.52%	44.44%	66.67%

TABLE 8.5

Variation in the number of function evaluations by comparison to the basic pattern search method (second column) and cumulative optimal gaps for final iterates (third to fifth columns). Case `shessian` *and* $B = [\,-e\ I\,]$.

parameter, and performing a search step. For the calculation of the simplex derivatives, we considered sample sets constructed in two variants: storing only all recent successful iterates, or storing all recent points where the objective function was evaluated. Finally, we studied two types of simplex derivatives: simplex gradients and

strategy	**number of evaluations**	**optimal gap**		
		10^{-7}	10^{-4}	10^{-1}
`consecutive polling & mesh-classical`	—	44.44%	62.96%	81.48%
`order (store-successful)`	-12.20%	44.44%	66.67%	77.78%
`mesh (store-successful)`	-4.53%*	44.44%	70.37%	81.48%
`order,mesh (store-successful)`	-15.20%	44.44%	70.37%	77.78%
`search (store-successful)`	12.67%	37.04%	66.67%	81.48%
`order,search (store-successful)`	-13.67%	44.44%	70.37%	81.48%
`mesh,search (store-successful)`	16.93%	37.04%	66.67%	81.48%
`order,mesh,search (store-successful)`	-5.98%	44.44%	70.37%	77.78%
`order (store-all)`	-13.10%	44.44%	66.67%	77.78%
`mesh (store-all)`	-28.28%	29.63%	70.37%	81.48%
`order,mesh (store-all)`	-48.45%	29.63%	66.67%	88.89%
`search (store-all)`	22.36%	37.04%	70.37%	81.48%
`order,search (store-all)`	26.20%	33.33%	70.37%	85.19%
`mesh,search (store-all)`	-18.85%	33.33%	70.37%	85.19%
`order,mesh,search (store-all)`	-34.25%	33.33%	70.37%	88.89%

TABLE 8.6

Variation in the number of function evaluations by comparison to the basic pattern search method (second column) and cumulative optimal gaps for final iterates (third to fifth columns). Case `sgradient` *and $B = [\,-I\ \ I\,]$.*

strategy	**number of evaluations**	**optimal gap**		
		10^{-7}	10^{-4}	10^{-1}
`consecutive polling & mesh-classical`	—	44.44%	62.96%	81.48%
`order (store-successful)`	-8.78%	44.44%	66.67%	81.48%
`mesh (store-successful)`	-1.11%	44.44%	70.37%	81.48%
`order,mesh (store-successful)`	-10.98%	44.44%	66.67%	81.48%
`search (store-successful)`	-3.64%	44.44%	66.67%	81.48%
`order,search (store-successful)`	-9.96%	44.44%	66.67%	77.78%
`mesh,search (store-successful)`	-5.46%	44.44%	70.37%	81.48%
`order,mesh,search (store-successful)`	-11.29%	44.44%	66.67%	77.78%
`order (store-all)`	-10.48%	44.44%	66.67%	77.78%
`mesh (store-all)`	-16.28%	33.33%	70.37%	77.78%
`order,mesh (store-all)`	-44.00%*	37.04%	66.67%	85.19%
`search (store-all)`	45.09%	37.04%	66.67%	81.48%
`order,search (store-all)`	19.48%	40.74%	66.67%	88.89%
`mesh,search (store-all)`	-7.89%	37.04%	70.37%	81.48%
`order,mesh,search (store-all)`	-30.43%	33.33%	66.67%	88.89%

TABLE 8.7

Variation in the number of function evaluations by comparison to the basic pattern search method (second column) and cumulative optimal gaps for final iterates (third to fifth columns). Case `shessian` *and $B = [\,-I\ \ I\,]$.*

diagonal simplex Hessians. It is important to remark that the incorporation of these strategies in pattern search is done at no further expense in function evaluations (except when a search step is tried).

The introduction of simplex derivatives in pattern search methods can lead to an improvement in the quality of the final iterates and, more importantly, a significant reduction in the number of function evaluations.

The impact of the ordering of the polling vectors according to a descent indicator is more visible, as expected, when the number of polling vectors is higher.

As a descent indicator, we recommend the use of the simplex gradient, in detriment of the simplex Newton direction, especially when used in the `store-all` variant. In fact, most of the iterations of a pattern search run are performed for small values of the mesh size parameter. In such cases, the negative gradient is better than the Newton direction as an indicator for descent, and the same argument applies to their simplex counterparts.

A mesh update based on a sufficient decrease condition could be considered if the main goal is the decrease of the number of functions evaluations, as long as medium quality final iterates are acceptable. When the number of polling vectors is small, the use of a search step is a suitable strategy, both in terms of the quality of the final iterate and in terms of the number of function evaluations.

Although we focused on unconstrained optimization, most of the extension of the use of these strategies to constrained optimization when constraint derivatives are known is straightforward. Whatever the technique used to compute positive generators to the tangent cones is chosen, one can always order them according to a descent indicator for the objective function (a simplex gradient or an approximated simplex Newton step). The update of the mesh size parameter at successful iterates is also possible using simplex derivatives. The search step, however, might have to be redefined to accommodate the presence of the constraints. In the case where the constraint derivatives are absent our strategies could be of use, for instance, in the mesh adaptive direct search methods developed recently in [4].

REFERENCES

[1] M. A. Abramson, C. Audet, and J. E. Dennis, *Generalized pattern searches with derivative information*, Math. Program., 100 (2004), pp. 3–25.

[2] P. Alberto, F. Nogueira, H. Rocha, and L. N. Vicente, *Pattern search methods for user-provided points: Application to molecular geometry problems*, SIAM J. Optim., 14 (2004), pp. 1216–1236.

[3] C. Audet and J. E. Dennis, *Analysis of generalized pattern searches*, SIAM J. Optim., 13 (2003), pp. 889–903.

[4] ———, *Mesh adaptive direct search algorithms for constrained optimization*, Tech. Report CAAM TR04-02, Dept. of Computational and Applied Mathematics, Rice University, 2004.

[5] D. M. Bortz and C. T. Kelley, *The simplex gradient and noisy optimization problems*, in Computational Methods in Optimal Design and Control, Progress in Systems and Control Theory, edited by J. T. Borggaard, J. Burns, E. Cliff, and S. Schreck, vol. 24, Birkhäuser, Boston, 1998, pp. 77–90.

[6] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of sample sets in derivative free optimization. Part II: under and over determined models.* In preparation.

[7] ———, *Geometry of sample sets in derivative free optimization. Part I: polynomial interpolation*, Tech. Report 03-09, Departamento de Matemática, Universidade de Coimbra, Portugal, 2003. Revised September 2004.

[8] C. Davis, *Theory of positive linear dependence*, Amer. J. Math., 76 (1954), pp. 733–746.

[9] E. D. Dolan, R. M. Lewis, and V. Torczon, *On the local convergence of pattern search*, SIAM J. Optim., 14 (2003), pp. 567–583.

[10] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr, a Constrained and Unconstrained Testing Environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[11] P. Hough, T. G. Kolda, and V. Torczon, *Asynchronous parallel pattern search for nonlinear optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 134–156.

[12] C. T. Kelley, *Iterative Methods for Optimization*, SIAM, Philadelphia, 1999.

[13] T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Rev., 45 (2003), pp. 385–482.

[14] ———, *Stationarity results for generating set search for linearly constrained optimization*, Tech. Report SAND 2003–8550, Sandia National Laboratories, 2003.

[15] C. Price and Ph. L. Toint, *Exploiting problem structure in pattern-search methods for unconstrained optimization*, Tech. Report 04/01, Dept. of Mathematics, FUNDP, 2004.

[16] C. J. Price and I. D. Coope, *Frames and grids in unconstrained and linearly constrained optimization: a non-smooth approach*, SIAM J. Optim., 14 (2003), pp. 415–438.

[17] V. Torczon, *On the convergence of pattern search algorithms*, SIAM J. Optim., 7 (1997), pp. 1–25.