USING SAMPLING AND SIMPLEX DERIVATIVES IN PATTERN SEARCH METHODS

A. L. CUSTÓDIO * AND L. N. VICENTE †

Abstract. In this paper, we introduce ways of making pattern search more efficient by reusing previous evaluations of the objective function, based on the computation of simplex derivatives (*e.g.*, simplex gradients).

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

The simplex gradient can then be used to reorder the evaluations of the objective function associated with the directions used in the poll step or to update the mesh size parameter according to a sufficient decrease criterion, neither of which requires new function evaluations.

We present these procedures in detail and apply them to a set of problems from the CUTEr collection. Numerical results show that these procedures can enhance significantly the practical performance of pattern search methods.

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

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

1. Introduction.

We are interested in this paper in designing efficient (derivative-free) pattern search methods for nonlinear optimization problems. We focus our attention on unconstrained optimization problems of the form $\min_{x \in \mathbb{R}^n} f(x)$.

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 at quickly decreasing the objective function from its initial value. However, they can be slow thereafter and especially towards stationarity, when the frequency of unsuccessful iterations tends to increase.

There has not been 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 parallelization (see Hough, Kolda, and Torczon [14]). In the context of generating set search methods, Frimannslund and Steihaug [12] rotate the generating sets based on curvature information extracted from function values. Other authors have considered particular instances where the problem structure can be exploited efficiently. Price and Toint [20] 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 strategies for improving the efficiency of the current pattern search iteration, based on function evaluations obtained 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

[*] 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 da Universidade de Coimbra, Centro de Matemática e Aplicações da Universidade Nova de Lisboa, Fundação Calouste Gulbenkian, and by FCT under grant POCI/MAT/59442/2004.

[†] 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 POCI/MAT/59442/2004.

resulting from running black-box codes or performing physical experiments). More importantly, these strategies (i) require no extra function evaluation and (ii) do not interfere with existing requirements for global convergence.

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 directions used for polling. In fact, 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 one way of updating the mesh size parameter according to a sufficient decrease condition.

These ideas were tested in a set of CUTEr [13] unconstrained problems, collected from papers on derivative-free optimization. The corresponding numerical results are reported in Section 8 and show the effectiveness of using sampling-based simplex derivatives in pattern search. Section 9 states some concluding remarks and ideas for future work. The default norms used in this paper are Euclidean.

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, 10, 17]. If the objective function possesses certain smoothness properties and the number of positive bases used remains finite, then pattern search is known to exhibit global convergence to stationary points in the $\liminf$ sense (see [3, 17]).

We present pattern search methods in the generalized format introduced by Audet and Dennis [3]. The positive spanning set used is represented by D, and its cardinality by $|D|$. It is convenient to regard D as an $n \times |D|$ matrix whose columns are the elements of D. 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

$$M_k = \{x_k + \alpha_k D z : \; z \in \mathbb{Z}_+^{|D|}\},$$

where $\mathbb{Z}_+$ is the set of nonnegative integers. This mesh is centered at the current iterate x_k, and its fineness is defined by the mesh size (or step size) parameter $\alpha_k > 0$. 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 rationality 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 lying on the mesh M_k. The choice of points in M_k is totally flexible 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 the parameter α_k and 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 or polling directions.

The purpose of the poll step is to ensure decrease in the objective function for sufficiently small mesh sizes. Provided that the function retains some differentiability properties, 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 $w^\top b \; > 0$ (see [10]). For instance, if the function f is continuously differentiable and one selects $w = -\nabla f(x_k)$, then 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. The report [18] presents two distinct classes of pattern search algorithms, namely the rank ordering and the positive bases pattern search methods. In the context of rank ordering pattern search, it is suggested ordering the simplex vertices, but with the single purpose of identifying the vertices with the best and the worst objective function values in order to compute a crude estimate of the direction of steepest descent. The authors explicitly state in [18] that their intention was not reordering the remaining vertices. Most papers do not address the issue of poll ordering at all and, as a result, numerical testing is typically done using the ordering in which the vectors are originally stored. Another ordering we discuss later consists of bringing into the first column (in B_{k+1}) the polling vector b_k associated with the most recent successful polling iterate $(f(x_k + \alpha_k b_k) < f(x_k))$. This ordering procedure has been called *dynamic polling* (see Audet and Dennis [4]). Our presentation of pattern search assumes that poll ordering is specified before polling starts.

If the poll step also fails to produce a point with a lower objective function value $f(x_k)$, then both the poll step and the iteration are declared unsuccessful. In this situation the mesh size parameter is decreased. On the other hand, the mesh size is held constant or increased if, in either the search or 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 `order` procedure that determines the order of the polling directions, and the `mesh` procedure 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

Pattern Search Method

Initialization
Choose x_0 and $\alpha_0 > 0$. Choose a positive spanning set D. Select 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`

The constant τ must satisfy $\tau \in \mathbb{Q}$ and $\tau > 1$, and should be initialized at iteration $k = 0$ together with $j_{max} \in \mathbb{Z}$, $j_{max} \geq 0$, and $j_{min} \in \mathbb{Z}$, $j_{min} \leq -1$.

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

FIG. 2.2. *Updating the mesh size parameter (for rational lattice requirements).*

convergence purposes (except for finiteness of the number of mesh points considered in `search`).

The `mesh` procedure, however, must update the mesh size parameter as described in Figure 2.2. The most common choice is to divide the parameter in half at unsuccessful iterations and to keep it or double it at successful ones. As noted by Hough, Kolda, and Torczon [14], increasing the mesh size parameter for all successful iterations can result into an excessive number of later contractions, each one requiring a complete polling, thus leading to an increase in the total number of function evaluations required. A possible strategy to avoid this behavior (fitting the procedure of Figure 2.2) has been suggested in [14] and consists of expanding the mesh only if two consecutive successful iterates have been computed using the same direction.

The global convergence analysis for this class of pattern search methods is divided into two parts. The first part establishes that a subsequence of mesh size parameters goes to zero. This result was first proved by Torczon in [21] and it is stated here as Theorem 2.1.

THEOREM 2.1. *Consider a sequence $\{x_k\}$ of pattern search iterates. If $L(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is compact, then the sequence of the mesh size parameters* *satisfies* $\liminf_{k \to +\infty} \alpha_k = 0.$

The second part of the analysis requires some differentiability properties of the objective function, and can be found, for instance, in [3, 17]. We formalize it here for unconstrained minimization.

THEOREM 2.2. *Consider a sequence* $\{x_k\}$ *of pattern search iterates. If* $L(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ *is compact, 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. If* f *is strictly differentiable near* x_**, then* $\nabla f(x_*) = 0$*. If* f *is continuously differentiable in an open set containing* $L(x_0)$*, then* $\liminf_{k \longrightarrow +\infty} \|\nabla f(x_k)\| = 0.$

Pattern search and direct search methods for unconstrained optimization are surveyed in the comprehensive *SIAM Review* paper of Kolda, Lewis, and Torczon [17].

3. Simplex derivatives. A simplex derivative of order one is known as a *simplex gradient.* Simplex gradients were used by Bortz and Kelley [5] in their implicit filtering method, which can be viewed as a line search method based on simplex gradients. Tseng [22] developed a class of simplex-based direct search methods imposing sufficient decrease conditions. He suggested the use of the norm of a simplex gradient in a stopping criterion for his class of methods. No numerical results were reported with this criterion, and no other use of the simplex gradient was suggested. In the context of the Nelder-Mead simplex-based direct search algorithm, Kelley [15] used the simplex gradient norm in a sufficient decrease type condition to detect stagnation, and the simplex gradient signs to orient the simplex restarts.

Calculation of a simplex gradient first requires selection of 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 (or regression) simplex gradients.

In the determined case, a simplex gradient is computed by first sampling the objective function at $n{+}1$ points. The convex hull of a set of $n{+}1$ affinely independent points $\{y^0, y^1, \ldots, y^n\}$ is called a simplex. The $n{+}1$ points are called the vertices of the simplex. Since the points are affinely independent, the matrix $S = [\, y^1{-}y^0 \ \cdots \ y^n{-}y^0 \,]$ is nonsingular. Given a 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 $\delta(f; S) = [\, f(y^1) - f(y^0), \ldots, f(y^n) - f(y^0) \,]^\top$.

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 $q{+}1 \neq n{+}1$ points from which to compute a simplex gradient. We say that a sample set is *poised* for a simplex gradient calculation if S is full rank, *i.e.*, if $\operatorname{rank}(S) = \min\{n, q\}$. (The notions of poisedness and affine independence coincide for $q \leq n$, but affine independence is not defined when $q > n$.) 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' g of the system

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

where $S = [\, y^1 - y^0 \ \cdots \ y^q - y^0 \,]$ and $\delta(f; S) = [\, f(y^1) - f(y^0), \ldots, f(y^q) - f(y^0) \,]^\top$. 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 nondetermined simplex gradients can be expressed using the reduced singular value decomposition (SVD) of $S^\top$. 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 appropriate to take the reduced SVD of a scaled form of $S^\top$. For this purpose, let

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

which is the radius of the smallest enclosing ball of $\{y^0, y^1, \ldots, y^q\}$ centered at y^0. Now we write the reduced 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 [16]. The extension of the analysis to the nondetermined cases is developed in Conn, Scheinberg, and Vicente [6].

Theorem 3.1. *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 $\mathcal{B}(y^0;\Delta)$ of this sample set, centered at y^0, where $\Delta = \max_{1\leq i\leq q}\|y^i - y^0\|$. Let $S = [\, y^1 - y^0 \ \cdots \ y^q - y^0 \,]$ and let $U\Sigma V^\top$ be the reduced SVD of $S^\top/\Delta$.*

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

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

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

where $\hat{V} = I$ if $q \geq n$ and $\hat{V} = V$ if $q < n$.

Notice that the error difference is projected over the null space of $S^\top/\Delta$. Unless we have enough points ($q + 1 \geq n + 1$), 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 3.1 depends on the size of the constant $\sqrt{q}\gamma\|\Sigma^{-1}\|/2$ which multiplies Δ. This constant, in turn, depends essentially on an unknown Lipschitz constant γ and on $\|\Sigma^{-1}\|$, which is associated to the geometry of 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 [7, 6]. If a sample set $\{y^0, y^1, \ldots, y^q\}$ is Λ–poised in the sense of [7, 6], 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 therefore say that a poised sample set is Λ*–poised* if $\|\Sigma^{-1}\| \leq \Lambda$, for some positive constant Λ.

In pattern search, we do not necessarily need an algorithm to build or maintain Λ–poised sets. Rather, we are given a sample set at each iteration, and our goal is just to identify a Λ–poised subset. 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 the following system in the variables $g \in \mathbb{R}^n$ and $H \in \mathbb{R}^{n\times n}$ with $H = H^\top$

$$(y^i - y^0)^\top g + \frac{1}{2}(y^i - y^0)^\top H (y^i - y^0) \; = \; f(y^i) - f(y^0), \quad i = 1, \ldots, p. \tag{3.1}$$

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. Similar to the linear case, the simplex gradient $g = \nabla_s f(y^0)$ and the simplex Hessian $H = \nabla_s^2 f(y^0)$, computed from system (3.1) with $p+1 = (n+1)(n+2)/2$ points, coincide with the coefficients of the quadratic multivariate polynomial interpolation model associated 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 [7, 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 in the model and to compute solutions in the minimum norm sense. Another option 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 $\delta(f;S) = [\, f(y^1) - f(y^0), \ldots, f(y^{2n}) - f(y^0) \,]^\top$ 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, one can try to identify a subset of these points with desirable geometrical properties (Λ–poisedness in our context).

If successful in such an attempt, we compute some form of simplex derivatives, such as a simplex gradient. We can then compute, at no additional cost, a direction of potential descent or of potential steepest descent (a negative simplex gradient, for example). We call such direction a *descent indicator*. There may be iterations (especially at the beginning) in which we fail to compute a descent indicator, but such failures cost no extra function evaluations either.

Our main goal is to use descent indicators based on simplex derivatives to order the poll vectors efficiently in the poll step. We can also explore the use of simplex derivatives in other components of a pattern search method such as the search step or the mesh size parameter update.

We adapt the description of pattern search to follow the approach described above. The class of pattern search methods remains essentially the same and is spelled out in Figure 4.1. All modifications to the algorithm reported in Figure 2.1 are marked in italics in Figure 4.1 for better identification.

The algorithm maintains a list X_k of evaluated points with maximum size p_{max}. Each time a new point is evaluated, the algorithm calls a new procedure `store`, which controls the adding (and deleting) of points to X_k.

Pattern Search Method — Using Sampling and Simplex Derivatives

Initialization
Choose x_0 and $\alpha_0 > 0$. Choose a positive spanning set D. Select 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 number p_{max} 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 ($2 \leq s_{min} \leq s_{max}$). Choose $\Lambda > 0$ and $\sigma_{max} \geq 1$.*

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 $\Delta_k = \sigma_k \, \alpha_{k-1} \max_{b \in B_{k-1}} \|b\|$, where $\sigma_k \in [1, \sigma_{max}]$. Try to identify a set of points Y_k in $X_k \cap \mathcal{B}(x_k; \Delta_k)$, 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.*

A new step is included at the beginning of each iteration for computing simplex derivatives. 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 certain distance Δ_k to the current iterate:

$$\Delta_k \;=\; \sigma_k \, \alpha_{k-1} \max_{b \in B_{k-1}} \|b\|,$$

where $\sigma_k \in [1, \sigma_{max}]$ and $\sigma_{max} \geq 1$ is fixed *a priori* for all iterations. Note that Δ_k is chosen such that $\mathcal{B}(x_k; \Delta_k)$ contains all the points in $P_{k-1} = \{x_{k-1} + \alpha_{k-1} b : \; b \in B_{k-1}\}$ when $k - 1$ is an unsuccessful iteration. The dependence of Δ_k on α_{k-1} guarantees the asymptotic quality of the simplex derivatives computed at a subsequence of unsuccessful iterates (see Theorems 3.1 and 5.1).

We consider two simple strategies for deciding whether or not to store a point, once the function has been evaluated there:

```
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.*

- `store-succ`: keeps only the successful iterates x_{k+1} (for which $f(x_{k+1}) < f(x_k)$).
- `store-all`: keeps every evaluated point.

In both cases, points are added sequentially to X_k at the top of the list. In `store-succ`, the points in the list X_k are ordered by increasing objective function values. When (and if) X_k has reached its predetermined size p_{max}, we must first remove a point before adding a new one. We assume that the points are removed from the end of the list. Note that both variants store successful iterates x_{k+1} (for which $f(x_{k+1}) < f(x_k)$). Clearly, the current iterate x_k is always in X_k, when `store-succ` is chosen. However, for `store-all`, x_k could be removed from the list if a number of consecutive unsuccessful iterates occur. We must therefore add a safeguard to prevent this from happening.

Having a descent indicator d_k at hand, we can order the polling vectors according to increasing magnitudes of the angles between d_k and the polling directions. So, the first polling point to be evaluated is the one corresponding to the polling vector making the smallest angle with d_k. We describe this procedure `order` in Figure 4.2 and illustrate it in Figure 4.3.

The descent indicator could be a negative simplex gradient $d_k = -\nabla_s f(x_k)$, where $S_k = [\, y_k^1 - x_k \;\cdots\; y_k^{q_k} - x_k \,]$ is formed from the sample set $Y_k = \{y_k^0, y_k^1, \ldots, y_k^{q_k}\}$, with $q_k + 1 = |Y_k|$ and $y_k^0 = x_k$. We designate this approach 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 Section 8, we test numerically the diagonal simplex Hessians described at the end of Section 3. This approach is designated by `shessian`.

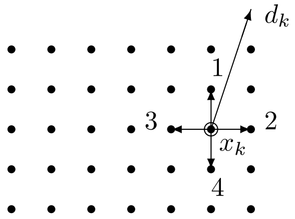

Fig. 4.3. *Ordering the polling vectors using a descent indicator. The positive basis considered is $B_k = [\, I \; -I \,]$.*

5. Geometry of the sample sets. If evaluated points 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. We define

$$s_{pb} \;=\; \min\{|B| : \; B \subset D, \; B \text{ positive basis}\}.$$

First we assume that $s_{min} \leq s_{pb}$, *i.e.*, that simplex gradients can be computed from s_{pb} points of X_k with appropriate geometry. If iteration $k-1$ was unsuccessful, then at least $|B_{k-1}|$ points were added to X_{k-1} (the polling points $x_{k-1} + \alpha_{k-1} b$, for all $b \in B_{k-1}$). Such points are part of X_k as well as the current iterate $x_k = x_{k-1}$. It is shown in the next theorem that the sample set $Y_k = \{x_k\} \cup \{x_{k-1} + \alpha_{k-1} b : \; b \in B_{k-1}\} \subset X_k$ is poised for a simplex gradient calculation.

It is also shown that the sample set $Y_k \subset X_k$ formed by x_k and by only $|B_{k-1}|-1$ of the points $x_{k-1}+\alpha_{k-1}b$, $b \in B_{k-1}$, is also poised for a simplex gradient calculation. In this case, we set $s_{min} \leq s_{pb} - 1$.

Theorem 5.1. *Let $k-1$ be any unsuccessful iteration of the pattern search method of Figure 4.1 using the* `store-all` *strategy.*

- *Suppose $s_{min} \leq s_{pb}$. There exists a positive constant Λ_1 (independent of k) such that the sample set $Y_k \subset X_k$ formed by $x_k = x_{k-1}$ and the points $x_{k-1}+\alpha_{k-1}b$, $b \in B_{k-1}$, is Λ_1–poised for a (overdetermined) simplex gradient calculation.*
- *Suppose $s_{min} \leq s_{pb} - 1$. There exist a positive constant Λ_2 (independent of k) such that the sample set $Y_k \subset X_k$ formed by $x_k = x_{k-1}$ and by only $|B_{k-1}|-1$ of the points $x_{k-1}+\alpha_{k-1}b$, $b \in B_{k-1}$, is Λ_2–poised for a (determined or overdetermined) simplex gradient calculation.*

Proof. To simplify the notation we write $B = B_{k-1}$. To prove the first statement, let $Y_k = \{y_k^0, y_k^1, \ldots, y_k^{q_k}\}$ 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 it linearly spans $\mathbb{R}^n$ by definition. Thus,

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

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

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

Since $\sigma_k \geq 1$, if we choose the poisedness constant such that

$$\Lambda_1 \;\geq\; \max \left\{ \|\Sigma^{-1}\| : \; \begin{array}{l} \frac{1}{\max_{b \in B} \|b\|} B^\top \;=\; U \Sigma V^\top, \\ \forall \, \text{positive bases} \, B \subset D \right\}, \end{array}$$

then we are guaranteed to identify a Λ_1–poised sample set after any unsuccessful iteration.

In the second case, we have $q_k + 1 = |Y_k| = |B|$ 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, $B_{|B|-1}$ linearly spans $\mathbb{R}^n$ (see [10, Theorem 3.7]), and therefore it has rank n. The difference now is that we must consider all submatrices $B_{|B|-1}$ of B. Thus, if we choose the poisedness constant such that

$$\begin{aligned}\Lambda_2 \;\geq\; \max \Big\{ \|\Sigma^{-1}\| : \quad & \tfrac{1}{\max_{b \in B} \|b\|} B_{|B|-1}^\top \;=\; U \Sigma V^\top, \\ & \forall\, B_{|B|-1} \subset B,\ \forall \,\text{positive bases}\, B \subset D \Big\}\,,\end{aligned}$$

we are guaranteed to identify a Λ_2–poised sample set after any unsuccessful iteration. □

We point out that a result of this type is not necessarily restricted to unsuccessful iterations. Other geometry scenarios can be explored at successful iterations.

6. Pruning the polling directions. Abramson, Audet, and Dennis [1] show that, for a special choice of the positive spanning set D, rough approximations to the gradient of the objective function can be used to 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

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

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

$$d^\epsilon(g)_i \;=\; \left\{ \begin{array}{cl} \mathrm{sign}(g_i) & \text{if } \; i \in J^\epsilon(g), \\ 0 & \text{otherwise.} \end{array} \right. \tag{6.1}$$

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 $\mathrm{sign}(g_i) = \mathrm{sign}(v_i)$ 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 $-\nabla_s f(x_k)$, 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 readdress 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 $\mathcal{B}(x_k; \Delta_k)$ centered at x_k with radius $\Delta_k = \sigma_k \, \alpha_{k-1} \max_{b \in B_{k-1}} \|b\|$, where B_{k-1} is the positive basis used for polling at the previous iteration.

THEOREM 6.1. *Let Y_k be a Λ–poised sample set (for simplex gradients) computed at iteration k of a pattern search method, with $q_k + 1 \geq n+1$ points.*

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

Then, if

$$\alpha_k \;\leq\; \frac{\|\nabla f(x_k)\|_\infty}{\sqrt{q_k} \gamma \Lambda \sigma_{max} \max_{b \in B_{k-1}} \|b\|}, \tag{6.2}$$

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

$$\epsilon_k \;=\; \left(q_k^{\frac{1}{2}} \gamma \Lambda \sigma_{max} \max_{b \in B_{k-1}} \|b\| \right) \alpha_k.$$

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 3.1 that

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

From Theorem 3.1 we also know that

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

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

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

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

Theorem 4 in Abramson, Audet, and Dennis [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 $-\nabla f(x_k)$, $d^\epsilon(\cdot)$ is defined in (6.1), and

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

represents the set of the ascent 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 $-\nabla f(x_k)^\top d \ge 0$.

So, under the hypotheses of Theorem 6.1, it follows that the negative simplex gradient $-\nabla_s f(x_k)$ prunes the positive spanning set,

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

to a singleton, namely $\{d^{\epsilon_k}(-\nabla_s f(x_k))\}$, where ϵ_k is given in Theorem 6.1.

Now we analyze in more detail the role of condition (6.2). There is no guarantee that this condition on α_k can be satisfied asymptotically. Condition (6.2) 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:

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

where

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

is the cosine measure of the positive basis B_k (see [17, 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 [11] that, for some problems, α_k goes to zero faster than $\|\nabla f(x_k)\|$. Our numerical experience with pattern search has also pointed us in this direction. It is more difficult, however, to sharply verify condition (6.2), since it depends on the Lipschitz constant of ∇f. A detailed numerical study of these asymptotic behaviors is beyond 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 other than ordering the polling vectors. In this section, we suggest two other uses for simplex derivatives in pattern search: the update of the mesh size parameter and the computation of a search step.

When a simplex gradient $\nabla_s f(x_k)$ is computed, a linear model $m_k(y) = f(x_k) + \nabla_s f(x_k)^\top (y - x_k)$ can be used to update the mesh size parameter α_k by imposing a sufficient decrease condition. In this case, we set

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

If x_{k+1} is computed in a successful poll step, then $x_{k+1} - x_k = \alpha_k b_k$ for some $b_k \in B_k$. In the quadratic case, the model is replaced by $m_k(y) = f(x_k) + g_k^\top (y - x_k) + (1/2)(y - x_k)^\top H_k (y - x_k)$. We call this procedure `mesh-sd` and describe it in Figure 7.1, where the sufficient decrease is only applied to successful iterations.

Since the expansion and contraction parameters are restricted to integer powers of τ and since the contraction rules match what was given in the `mesh` procedure of Figure 2.2, the modification introduced in `mesh-sd` has no influence on the global convergence properties of the underlying pattern search method.

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 $\mathcal{B}(x_k; \Delta_k)$, after which we would project the minimizer onto the mesh M_k. We described above two examples of such a model $m_k(y)$, but many others could be considered. The use of surrogate models in the search step is the topic of a separate research.

8. Implementation and numerical results. To serve as a baseline for numerical comparisons, we have implemented a basic pattern search algorithm of the form given in Figure 2.1. Specifically, no search step is used, the mesh size parameter is left unchanged at successful iterations, and points in the poll step are always evaluated in the same consecutive order as originally stored. We refer to this version of pattern search as `basic`.

We have tested a number of pattern search methods of the form described in Figure 4.1. The strategies `order` (Figure 4.2) and `mesh-sd` (Figure 7.1) were run in four different modes according to the way of storing points (`store-succ` or `store-all`) and to the way of computing simplex derivatives and descent indicators (`sgradient`

procedure mesh-sd

The constants τ and ξ must satisfy $\tau \in \mathbb{Q}$, $\tau > 1$, and $\xi > 0$, and should be initialized at iteration $k = 0$ together with $j_{max} \in \mathbb{Z}$, $j_{max} \geq 0$, and $j_{min} \in \mathbb{Z}$, $j_{min} \leq -1$. The exponents satisfy $j_k^+ \in \{0, 1, 2, \ldots, j_{max}\}$ and $j_k^- \in \{j_{min}, \ldots, -1\}$.

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 $\rho_k > \xi$ then $\alpha_{k+1} = \tau^{j_k^+} \alpha_k$,
If $\rho_k \leq \xi$ then $\alpha_{k+1} = \alpha_k$.

If the iteration was unsuccessful, then contract mesh by decreasing the mesh size parameter $\alpha_{k+1} = \tau^{j_k^-} \alpha_k$.

FIG. 7.1. *Updating the mesh size parameter (using sufficient decrease but meeting rational lattice requirements).*

or `shessian`). Moreover, we implemented the strategy suggested in [14] and described in Section 2 for updating the mesh size parameter (here named as `mesh-HKT`), and the `dynamic polling` strategy suggested in [4] for changing the order of the polling directions (see Section 2). We tested a very crude search step based on taking a step along the descent indicator with a step size of the order of α_k (see [9] for the details).

The algorithms were coded in MATLAB and ran on 27 unconstrained problems belonging to the CUTEr collection [13], gathered mainly from papers on derivative-free optimization. The objective functions of these problems are twice continuously differentiable. Their dimensions are given in Table 8.1. The starting points used were those reported in CUTEr. Problems bdvalue, integreq, and broydn3d were posed as unconstrained optimization problems like originally in [19]. The stopping criterion consisted of the mesh size parameter becoming lower than 10^{-5} or a maximum number of 100000 iterations being reached.

The simplex derivatives were computed based on Λ-poised sets Y_k, where $\Lambda = 100$. The factor σ_k was chosen as: 1 ($k-1$ unsuccessful); 2 ($k-1$ successful and $\alpha_k = \alpha_{k-1}$); 4 ($k-1$ successful and $\alpha_k > \alpha_{k-1}$). The values for the parameters s_{min}, s_{max}, and p_{max} are given in Table 8.2. We started all runs with the mesh size parameter $\alpha_0 = 1$. In all versions, the contraction factor was set to $\tau^{j_k^-} = 0.5$ and the expansion factor (when used) was set to $\tau^{j_k^+} = 2$. In the `mesh-sd` strategy of Figure 7.1, we set ξ equal to 0.75.

We draw conclusions based on two positive bases: $[\,I\ -I\,]$ and $[\,-e\ I\,]$. The maximal positive basis $[\,I\ -I\,]$ corresponds to coordinate search and it provided the best results for the `basic` version among a few positive bases stored in different orders (which included $[\,I\ -I\,]$, $[\,-I\ I\,]$, $[\,-e\ I\,]$, $[\,e\ -I\,]$, $[\,I\ -e\,]$, $[\,-I\ e\,]$, and a minimal basis with angles between vectors of uniform amplitude). The positive basis stored as $[\,-e\ I\,]$ was the minimal positive basis which behaved the best. In Table 8.1 we report the results obtained by the `basic` version for these two positive bases.

By combining all possibilities, we tested a total of 120 versions, 112 involving simplex derivatives. A summary of the complete numerical results is reported in [9].

8.1. Discussion based on complete results. First, we point out that 91% of the versions involving simplex derivatives lead to an average decrease in the number

		positive basis			
		$D = [-e\ I]$		$D = [I\ -I]$	
problem	**dimension**	**fevals**	**fvalue**	**fevals**	**fvalue**
arwhead	10	1068	4.19e-09	361	0.00e+00
arwhead	20	3718	8.85e-09	721	0.00e+00
bdqrtic	10	2561	1.19e+01	948	1.19e+01
bdqrtic	20	19038	3.54e+01	4120	3.54e+01
bdvalue	10	36820	4.39e-07	33077	4.39e-07
bdvalue	20	255857	1.30e-05	245305	1.29e-05
biggs6	6	339840	6.50e-03	467886	9.58e-06
brownal	10	468150	1.84e+00	74922	2.02e-06
brownal	20	1073871	1.55e+01	284734	1.04e-05
broydn3d	10	2281	3.26e-08	1743	4.52e-09
broydn3d	20	17759	2.91e-07	6868	2.47e-08
integreq	10	2595	4.42e-09	1034	2.35e-10
integreq	20	20941	3.20e-08	4244	4.86e-10
penalty1	10	552357	7.33e-05	234274	7.09e-05
penalty1	20	999305	1.66e-04	535100	1.58e-04
penalty2	10	46696	4.09e-04	496275	4.04e-04
penalty2	20	366131	8.32e-03	1494751	8.30e-03
powellsg	12	192270	1.85e-04	58987	9.85e-07
powellsg	20	480158	3.08e-04	158591	1.64e-06
srosenbr	10	401321	6.83e-05	171061	6.83e-05
srosenbr	20	1076983	2.68e-02	649621	1.37e-04
tridia	10	1000805	5.95e-01	901720	5.85e-01
tridia	20	20483	6.24e-01	6635	6.24e-01
vardim	10	251599	2.23e-05	86316	6.64e-07
vardim	20	961697	1.76e+04	1230761	8.71e-04
woods	12	164675	1.02e-04	110662	3.78e-05
woods	20	435786	3.53e-04	300296	6.29e-05

TABLE 8.1
Test set and results for the `basic` *version.*

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

TABLE 8.2
Sizes of the list X_k and of the set Y_k.

of function evaluations [9]. Moreover, 61 out of the 112 strategies tested provided a negative 75% percentile for the variation in the number of function evaluations. This means that for each of these 61 strategies, a reduction in the number of function evaluations was achieved for 75% of the problems tested.

The overall results [9] showed a superiority of `sgradient` over `shessian`, which is not surprising because the number of points required to identify Λ–poised sets in `sgradient` is lower than in `shessian`. Also, another reason for `sgradient` being possibly better than `shessian` is that, if the simplex gradient is sufficiently close to the true gradient, then directions making a small angle with the negative simplex gradient will be descent directions, while the same is not guaranteed when we use simplex Newton directions. Some `shessian` versions, however, have behaved relatively well [9].

For the positive basis $[\, I \ -I \,]$ there is a clear gain when using `store-all` compared to `store-succ` [9]. However, for the positive basis $[\, -e \ I \,]$, the advantage of `store-all` over `store-succ` is not as clear [9]. In general, the advantage of `store-all` may be explained by the frequent number of unsuccessful iterations that tend to occur in the last iterations of a pattern search run. The effect of the poll ordering is also more visible when using the positive basis $[\, I \ -I \,]$, due to the larger number of polling vectors.

Strategies `mesh-sd`, and `mesh-HKT` made a clear positive impact when using the smaller positive basis $[\, -e \ I \,]$ (see [9]). This effect was lost in the larger positive basis $[\, I \ -I \,]$, where the `order` procedure seems to perform well on its own for this test set.

8.2. Discussion based on best results. We report in Table 8.3 a summary of the results for a number of versions based on $[\, I \ -I \,]$. Included in this restricted set of versions are the ones that lead to the best results among all the 120 versions tested. (The results for the remaining versions are summarized in [9].)

An explanation about Table 8.3 is in order. For each strategy and for each problem, we calculated the percentage of iterations that used simplex descent indicators as well as the variation in the number of function evaluations required relatively to the `basic` version. These percentages were grouped by strategy and their average values are reported in the second and third columns of Table 8.3. The last three columns of the table represent the cumulative percentages for the optimal gaps of the final iterates.

		number of	**optimal gap**		
strategy	**% poised**	**evaluations**	10^{-7}	10^{-4}	10^{-1}
`basic`	—	—	33.33%	81.48%	92.59%
`mesh-HKT`	—	+4.02%	40.74%	81.48%	92.59%
`dynamic polling`	—	-10.99%	33.33%	81.48%	92.59%
`mesh-HKT,dynamic polling`	—	-15.17%	48.15%	81.48%	92.59%
`mesh-sd (store-succ)`	14.40%	-3.07%	33.33%	81.48%	92.59%
`mesh-sd (store-all)`	73.33%	+0.45%	33.33%	81.48%	92.59%
`order (store-all)`	27.26%	-51.16%	37.04%	85.19%	92.59%
`mesh-sd,order (store-all)`	28.56%	-51.47%	37.04%	85.19%	92.59%
`mesh-HKT,order (store-all)`	58.51%	-54.22%	51.85%	81.48%	88.89%

TABLE 8.3

Average percentage of iterations that used simplex descent indicators (second column), average variation of function evaluations by comparison to the `basic` *version (third column), and cumulative percentages for the optimal gaps of the final iterates (fourth to sixth columns). Case* `sgradient` *and* $D = [\, I \ -I \,]$.

The quality of the final objective function values obtained for the versions included in Table 8.3 is comparable to the `basic` version, as one see from the final cumulative optimal gaps reported.

It is clear that none of the strategies for updating the step size parameter (`mesh-sd` and `mesh-HKT`) made improvements on their one, the former being slightly better than the latter.

The best result without using simplex derivatives was obtained by combining `dynamic polling` and `mesh-HKT` (15% less function evaluations than the `basic` version).

Three versions that incorporated `order` reached a reduction of around 50% in number of function evaluations. The `order` procedure in the `store-all` mode lead, on its own, to a 51% improvement, compared to 11% of `dynamic polling`.

The best version achieved a reduction of 54% in number of evaluations by combining `order` and `mesh-HKT` in the `store-all` mode. In Table 8.4 we report the results obtained by this version as well as by the version that only applies the `order` procedure in the `store-all` mode.

		strategy			
		`order`		`order,mesh-HKT`	
problem	**dimension**	**fevals**	**fvalue**	**fevals**	**fvalue**
arwhead	10	361	0.00e+00	361	0.00e+00
arwhead	20	721	0.00e+00	721	0.00e+00
bdqrtic	10	696	1.19e+01	696	1.19e+01
bdqrtic	20	2138	3.54e+01	2138	3.54e+01
bdvalue	10	34922	6.89e-07	28411	6.52e-07
bdvalue	20	255989	1.66e-05	213297	1.65e-05
biggs6	6	105592	4.50e-07	164168	4.53e-07
brownal	10	21045	1.88e-06	38398	2.44e-06
brownal	20	67152	6.16e-06	4227	1.00e+00
broydn3d	10	917	4.73e-09	917	4.73e-09
broydn3d	20	2940	2.35e-08	2940	2.35e-08
integreq	10	597	2.35e-10	597	2.35e-10
integreq	20	1573	4.86e-10	1573	4.86e-10
penalty1	10	126307	7.09e-05	177360	7.09e-05
penalty1	20	229825	1.58e-04	279491	1.58e-04
penalty2	10	55087	4.04e-04	93192	4.05e-04
penalty2	20	189446	8.29e-03	355154	8.29e-03
powellsg	12	594	0.00e+00	614	0.00e+00
powellsg	20	45258	1.31e-06	8702	2.81e-11
srosenbr	10	136327	6.83e-05	119830	6.83e-05
srosenbr	20	567937	1.37e-04	358656	1.36e-04
tridia	10	539119	5.85e-01	908097	5.89e-01
tridia	20	2724	6.24e-01	2828	6.24e-01
vardim	10	5382	2.29e-07	6550	9.15e-08
vardim	20	67487	9.27e-06	71692	1.68e-06
woods	12	59565	3.94e-05	577	0.00e+00
woods	20	106339	6.55e-05	1064	0.00e+00

TABLE 8.4

Results for the best versions, using the positive basis $D = [\, I \ -I \,]$.

8.3. Additional tests. 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 or with the increase in its cosine measure (both of which happen, for instance, when going from $[-e\ I]$ to $[I\ -I]$).

We also tried to investigate how sensitive the different algorithmic versions are to the choice of the parameter ξ used in the `mesh-sd` strategy. We tried other values (for instance 0.5 and 0.95), but the results did not improve.

We repeated these computational tests on a different set of test problems, consisting of seven randomly generated quadratic functions, each one of dimension 10. The quadratic functions were defined by $f(x) = x^\top Ax$, where $A = B^\top B$ and B is a matrix with random normal entries of mean 0 and standard deviation 1. We also randomly generated the starting point for the algorithm, using the same normal distribution. Once again, the conclusions for the different strategies remained essentially the same (with improvement of the results for the minimal positive basis $[-e\ I]$). We used these examples to study the descent properties of the negative simplex gradient. In our experiments, the simplex gradient made an acute angle with the true gradient in average in 77% of the cases where it was computed. These occurrences tend to happen more towards the end of the runs when the mesh size parameter gets smaller.

8.4. Pruning. To better understand the theoretical results derived in Section 6, we implemented a computational variant of the pruning strategy. We did not consider the generating set $D = \{-1, 0, 1\}^n$, as suggested by Abramson, Audet, and Dennis [1], nor did we verify condition (6.2) (or some approximated form of it by estimating the Lipschitz constant involved) before pruning the polling vectors. As result, we are violating the conditions required for the analysis of the pruning strategy. We tested two different variants for pruning the positive bases $[I\ -I]$ and $[-e\ I]$: (i) pruning to a single direction, namely the one that makes the angle of smallest amplitude with the descent indicator; (ii) pruning to all the directions that make an acute angle with the descent indicator.

To reach a final iterate of quality nearly similar to the one obtained by the `basic` version, we had to use the positive basis $[I\ -I]$ and prune with more than one direction. In this case, pruning achieved an average reduction in the number of function evaluations of 10% and 42%, for the `store-succ` and `store-all` variants, respectively. Pruning tends to generate less polling points, which in turn decreases the chances of building well-poised sets.

More research is needed in order to evaluate the potential of the negative simplex gradient as an ϵ-approximation to the large components of the negative gradient vector and its use for pruning the polling directions. The use of the generating set $D = \{-1, 0, 1\}^n$ and the implementation of some form of the condition (6.2) might have a positive impact.

9. Concluding remarks and future work. We have proposed the use of simplex derivatives in pattern search methods in two ways: ordering the polling vectors and updating the mesh size parameter. 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 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.

The introduction of simplex derivatives in pattern search methods can lead to a significant reduction in the number of function evaluations, for the same quality of the final iterates.

As a descent indicator, we recommend the use of the negative simplex gradient over the simplex Newton direction. 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.

For coordinate search ($D = [\, I \ -I \,]$), ordering the polling directions according to a simplex descent indicator (negative simplex gradient) made a significant impact in the reduction of the number of function evaluations. For this type of positive basis, storing all recent points where the objective function was evaluated seems to be the best approach.

Our numerical findings showed that updating the mesh size parameter based on a sufficient decrease condition can be worthwhile applying when using minimal positive bases (like $D = [\, -e \ I \,]$). In such cases, storing only all recent successful iterates may also be advantageous.

There are at least two natural generalizations of the ideas presented in this paper. One is to apply simplex derivatives based strategies to improve parallel versions of pattern search. Another generalization consists of analyzing the properties of simplex gradients when direct search methods are applied to nonsmooth functions [8]. The use of simplex derivatives in the design of an efficient search step is also subject of future research.

REFERENCES

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

[4] ———, *Mesh adaptive direct search algorithms for constrained optimization*, SIAM J. Optim., 17 (2006), pp. 188–217.

[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: Polynomial regression and underdetermined interpolation*, Tech. Report 05-15, Departamento de Matemática, Universidade de Coimbra, Portugal, 2005.

[7] ———, *Geometry of interpolation sets in derivative free optimization*, Math. Program., (2006, to appear).

[8] A. L. Custódio, J. E. Dennis Jr., and L. N. Vicente, *Using simplex gradients of nonsmooth functions in direct search methods*, Tech. Report 06-48, Departamento de Matemática, Universidade de Coimbra, Portugal, 2006.

[9] A. L. Custódio and L. N. Vicente, *Using sampling and simplex derivatives in pattern search methods (complete numerical results)*. See `http://www.mat.uc.pt/∼lnv/papers/sid-psm-complete.pdf`, 2006.

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

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

[12] L. Frimannslund and T. Steihaug, *A generating set search method using curvature information*, Comput. Optim. and Appl., (2006, to appear).

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

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

[15] C. T. Kelley, *Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition*, SIAM J. Optim., 10 (1999), pp. 43–55.

[16] ———, *Iterative Methods for Optimization*, SIAM, Philadelphia, 1999.

[17] 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.

[18] R. M. Lewis and V. Torczon, *Rank ordering and positive bases in pattern search algorithms*, Tech. Report 96-71, ICASE, NASA Langley Research Center, USA, 1999.

[19] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.

[20] C. Price and P. L. Toint, *Exploiting problem structure in pattern-search methods for unconstrained optimization*, Optim. Methods Softw., 21 (2006), pp. 479–491.

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

[22] P. Tseng, *Fortified-descent simplicial search method: a general approach*, SIAM J. Optim., 10 (1999), pp. 269–288.

Acknowledgments.

During the course of the revision of this paper, the first author visited College of William & Mary. The authors are grateful to Robert Michael Lewis, Virginia Torczon, and Michael Trosset for many interesting and stimulating comments, which have contributed to improvements in the manuscript. We also thank the anonymous referees for their suggestions.