

# PATTERN SEARCH METHODS FOR LINEARLY CONSTRAINED MINIMIZATION IN THE PRESENCE OF DEGENERACY \*

OLGA A. BREZHNEVA † AND J. E. DENNIS JR. ‡

**Abstract.** This paper deals with generalized pattern search (GPS) algorithms for linearly constrained optimization. At each iteration, the GPS algorithm generates a set of directions that conforms to the geometry of any nearby linear constraints, and this set is used to define the POLL set for that iteration. The contribution of this paper is to provide a detailed algorithm for constructing the set of directions at a current iterate whether or not the constraints are degenerate. The main difficulty in the degenerate case is in classifying constraints as redundant and nonredundant. We give a short survey of the main definitions and methods concerning redundancy and propose an approach, which may be useful for other active set algorithms, to identify the nonredundant constraints.

**Key words.** Pattern search, linearly constrained optimization, derivative-free optimization, degeneracy, redundancy, constraint classification

**AMS subject classifications.** 65K05, 49M30, 90C30, 90C56

**1. Introduction.** This paper continues the development of the generalized pattern search (GPS) algorithms [1, 9] for the linearly constrained optimization problems

$$\min_{x \in \Omega} f(x), \tag{1.1}$$

where  $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$  may be discontinuous, and the feasible region is given by

$$\Omega = \{x \in \mathbb{R}^n : a_i^T x \leq b_i, i \in I\} = \{x \in \mathbb{R}^n : A^T x \leq b\}, \tag{1.2}$$

where  $a_i \in \mathbb{R}^n$ ,  $b_i \in \mathbb{R}$ , and  $A \in \mathbb{Q}^{n \times |I|}$  is a rational matrix. In [1, 9], the feasible region  $\Omega$  is defined as  $\Omega = \{x \in \mathbb{R}^n : l \leq \hat{A}x \leq u\}$  where  $\hat{A} \in \mathbb{Q}^{m \times n}$  is a rational matrix,  $l, u \in \{\mathbb{R} \cup \{\pm\infty\}\}^m$ , and  $l < u$ . As is evident, (1.2) reduces to the definition in [1, 9], where the  $r$ th row  $\hat{a}_r$  of the matrix  $\hat{A}$  is equal to some  $i$ th row  $a_i^T$  of the matrix  $A^T$  with a coefficient of  $+1$  or  $-1$ , and  $b_i = u_r$  or  $b_i = -l_r$ .

We target the case when the function  $f(x)$  can be an expensive black box, or provides few correct digits and may fail to return a value even for feasible points  $x \in \Omega$ . In this situation, the accurate approximation of derivatives is not likely to be practical. The GPS algorithms rely on decrease in  $f(x)$ ; an iterate  $x_{k+1} \in \Omega$  with  $f(x_{k+1}) < f(x_k)$  is considered successful.

Lewis and Torczon [9] introduced and analyzed the generalized pattern search for linearly constrained minimization problems. They proved that if the objective function is continuously differentiable and if the set of directions that defines a local search is chosen properly with respect to the geometry of the boundary of the feasible region, then the GPS converges globally to a Karush-Kuhn-Tucker point. Audet and

\* Date: August 10, 2003

† Institute for Mathematics and its Applications, University of Minnesota, 400 Lind Hall, 207 Church St. SE, Minneapolis, MN 55455 ([olga@ima.umn.edu](mailto:olga@ima.umn.edu)).

‡ Computational and Applied Mathematics Department, Rice University - MS 134, 6100 Main Street, Houston, Texas, 77005-1892 ([dennis@rice.edu](mailto:dennis@rice.edu), <http://www.caam.rice.edu/~dennis>). The research of this author was supported in part by AFOSR F49620-01-1-0013, the Boeing Company, Sandia CSRI, ExxonMobil, the LANL Computer Science (LACSI) contract 03891-99-23, by the Institute for Mathematics and its Applications with funds provided by the National Science Foundation, and by funds from the Ordway Endowment at the University of Minnesota.

Dennis [1] simplified the analysis in [9] and provided new convergence results for a locally Lipschitz objective function by applying the Clarke generalized directional derivative [7] to the pattern search methods.

Generalized pattern search algorithms generate a sequence of iterates  $\{x_k\}$  in  $\mathbb{R}^n$  with non-increasing objective function values. In linearly constrained optimization, a set of directions that defines the so-called POLL step must conform to the geometry of the boundary of the feasible region. The key idea, which was first suggested by May in [10] and applied to the GPS in [9], is to use as search directions the generators of cones that are polar to cones generated by the normals of faces near the current iterate.

Lewis and Torczon [9] showed that in general it is possible to find the set of generators by vertex enumeration techniques [2], but, as they mentioned, an application of this approach to constructing a set of generators can be expensive in the degenerate case when the active constraints are linearly dependent. Lewis and Torczon [9] gave an algorithm, based on the LU factorization, for constructing the set of generators in the nondegenerate case, and left the degenerate case for future work.

Price and Coope [12] gave as an aside a result that can be used for constructing a set of generators in the degenerate case. It follows from their result that, in order to construct a set of generators, it is sufficient to consider maximal linearly independent subsets of the active constraints. However, this approach implies enumeration of all possible linearly independent subsets of maximal rank and does not take into account properties of the problem that can help to reduce this enumeration. Price and Coope [12] outlined an algorithm for constructing frames, but it was not their point to work out details of the numerical implementation in the degenerate case.

The purpose of this paper is to give detailed consideration to the GPS in the degenerate case. Our purpose here is complementary to [1] and [9]. Our main result is a detailed algorithm for constructing the set of generators at a current GPS iterate in both the degenerate and nondegenerate cases. To construct the set of generators in the degenerate case, we identify the redundant and nonredundant active constraints and then use either QR or LU decomposition.

Classification of constraints as redundant or nonredundant is one of the main issues here, because it is sufficient to construct the set of generators only for nonredundant constraints. Several methods to classify constraints exist. For example, there are deterministic algorithms [6, 8], probabilistic hit-and-run methods [3], a probabilistic method based on an equivalence between the constraint classification problem and the problem of finding a feasible solution to a set covering problem [5]. A survey and comparison of strategies for classifying constraints are given in [5, 8]. Any of these approaches can be applied in the framework of the GPS to identify redundant and nonredundant constraints. However, in the paper, we propose a new projection approach to identify nonredundant constraints, that is more suitable for GPS methods.

The projection method is similar to the hit-and-run algorithm [3] in which nonredundant constraints are searched for along random direction vectors from each point in a sequence of random interior points. In contrast to the hit-and-run algorithm, the projection method searches for a nonredundant constraint in a deterministic direction.

The major advantage of the projection method for our application is that the number of direction vectors (in the terminology of the hit-and-run algorithm) is equal to the number of constraints that have to be identified. For us this is generally a small number. In the hit-and-run algorithm, this number is determined by a stop criterion and can be large if many of the random generated directions do not detect a

nonredundant constraint. Moreover, the formulas used in the projection method are simpler than those used for computing the intersection points of a direction vector with the hyperplanes in the hit-and-run algorithm. Let us note that the goal of the hit-and-run algorithm is detecting all nonredundant constraints in a full system of linear inequalities. We use the projection method to detect the nonredundant constraints among only active constraints in the case when they are linearly dependent.

As our numerical tests show, the projection method cheaply detects all, or almost all, nonredundant constraints. To classify constraints that have not been detected by the projection method, we use another approach outlined in [6]. As a result, we guarantee that every active constraint is detected as either redundant or nonredundant. In the worst case, some vertex enumeration techniques [2] mentioned in [9] might be necessary, but our procedure for classification of the constraints seems to eliminate this expense for many cases.

The organization of the paper is as follows: in the next section, we give a brief description of GPS as well as the convergence result for linearly constrained minimization following papers by Audet and Dennis [1], and by Lewis and Torczon [9].

Section 3 is devoted to the topic of redundancy. In the first part of the section, we introduce a definition of the  $\varepsilon$ -active constraints and discuss some scaling issues. The second part of Section 3 contains essential definitions and results concerning redundancy [6, 8, 11, 15] required for our analysis. Then we propose our projection method to determine nonredundant constraints, and we briefly describe a more expensive follow up approach to be applied if some constraints are not identified by the projection method.

In Section 4, we give an algorithm for the set of generators, and we discuss implementation details, including rationality. Section 5 is devoted to some concluding remarks.

**2. Generalized pattern search algorithms.** In this section, we give a brief description with the convergence result for the GPS methods for linearly constrained minimization. We follow papers by Audet and Dennis [1], and by Lewis and Torczon [9], and we refer the reader there for details of managing the mesh size  $\Delta_k$ . Throughout, we will always use the  $\ell_2$  norm.

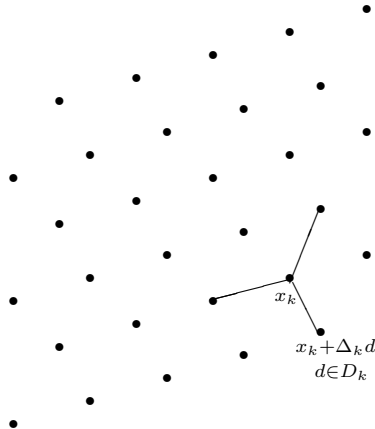
The GPS algorithms can be applied either to the objective function  $f$  or to the barrier function  $f_\Omega = f + \psi_\Omega : \mathbb{R} \rightarrow \mathbb{R} \cup \{+\infty\}$ , where  $\psi_\Omega$  is the indicator function for  $\Omega$ , which is zero on  $\Omega$  and  $\infty$  elsewhere. The value of  $f_\Omega$  is  $+\infty$  on all points that are either infeasible or at which  $f$  is declared to be  $+\infty$ . This barrier approach is probably as old as direct search methods themselves.

A GPS algorithm for linearly constrained optimization generates a sequence of iterates  $\{x_k\}$  in  $\Omega$ . The *current iterate*  $x_k \in \mathbb{R}^n$  is chosen from a finite number of points on a *mesh*, which is a discrete subset of  $\mathbb{R}^n$ . At iteration  $k$ , the mesh is centered around the *current mesh point (current iterate)*  $x_k$  and its fineness is parameterized through the *mesh size parameter*  $\Delta_k > 0$  as follows

$$M_k = \{x_k + \Delta_k Dz : z \in Z_+^{|D|}\}, \quad (2.1)$$

where  $Z_+^{|D|}$  is the set of nonnegative integers, and  $D$  is a *set of positive spanning directions* in  $\mathbb{R}^n$ . At each iteration, some positive spanning matrix  $D_k$  composed of columns of  $D$  is used to construct the *poll set*  $P_k$ , i.e.,

$$P_k = \{x_k + \Delta_k d : d \in D_k\}. \quad (2.2)$$



If  $x_k \in \Omega$  is not near the boundary, then  $D_k$  is a positive spanning set for  $\mathbb{R}^n$  [9]. If  $x_k \in \Omega$  is near the boundary, the matrix  $D_k$  is constructed so its columns  $d_j$  also span the cone of feasible directions at  $x_k$  and conform to the geometry of the boundary of  $\Omega$ . Hence, the set  $D$  must be rich enough to contain generators for the tangent cone  $T_\Omega(x) = \text{cl}\{\mu(\omega - x) : \mu \geq 0, \omega \in \Omega\}$  for every  $x \in \Omega$ , as in:

**DEFINITION 2.1.** *A rule for selecting the positive spanning sets  $D_k \subseteq D$  conforms to  $\Omega$  for some  $\varepsilon > 0$ , if at each iteration  $k$  and for each  $y$  in the boundary of  $\Omega$  for which  $\|y - x_k\| < \varepsilon$ ,  $T_\Omega(y)$  is generated by a nonnegative linear combination of the columns of a subset  $D_k^y$  of  $D_k$ .*

Each GPS iteration is divided into two phases: an optional search and a local poll. In the SEARCH step, the barrier objective function  $f_\Omega$  is evaluated at a finite number of points on a mesh to try to find one that yields a lower objective function value than the incumbent. This step is extremely flexible, and it can be used to incorporate the user's domain knowledge.

When the incumbent is replaced, i.e., when  $f_\Omega(x_{k+1}) < f_\Omega(x_k)$ , or equivalently when  $f(x_{k+1}) < f(x_k)$ , then  $x_{k+1}$  is said to be an *improved mesh point*. When the search step fails in providing an improved mesh point, the POLL step is invoked. This second step consists of evaluating the barrier objective function at the neighboring mesh points to see if a lower function value can be found there. When the POLL step fails in providing an improved mesh point, then the current incumbent solution is said to be a *mesh local optimizer*.

We remind the reader that the normal cone  $N_\Omega(x)$  to  $\Omega$  at  $x$  is the nonnegative span of all the outwardly pointing constraint normals at  $x$  and can be written as the polar of the tangent cone:  $N_\Omega(x) = \{v \in \mathbb{R}^n : \forall \omega \in T_\Omega(x), v^T \omega \leq 0\}$ .

**Assumptions.** We make the following standard assumptions [1]:

- A1:** A function  $f_\Omega$  and  $x_0 \in \mathbb{R}^n$  (with  $f_\Omega(x_0) < \infty$ ) are available.
- A2:** The constraint matrix  $A$  is rational.
- A3:** All iterates  $\{x_k\}$  produced by the Generalized Pattern Search (GPS) algorithm lie in a compact set.

Audet and Dennis [1] proved the following convergence result for the GPS in the linearly constrained case using only these assumptions.

**THEOREM 2.2** (convergence to a Karush-Kuhn-Tucker point). [1] *Under assumptions A1–A3, if  $f$  is strictly differentiable at a limit point  $\hat{x}$  of a subsequence of  $\{x_k\}$  at which  $\Delta_k$  is decreased and for which the corresponding subsequence of  $\{\Delta_k\}$  goes to 0, and if the rule for selecting the positive spanning sets  $D_k \subseteq D$  conforms to*

- **INITIALIZATION:**  
Let  $x_0$  be such that  $f_\Omega(x_0)$  is finite. Let  $D$  be a positive spanning set, and let  $M_0$  be the mesh on  $\mathfrak{R}^n$  defined by  $\Delta_0 > 0$ , and  $D_0$ . Set the iteration counter  $k$  to 0.
- **SEARCH AND POLL STEP:**  
Perform the SEARCH and possibly the POLL steps (or only part of them) until an improved mesh point  $x_{k+1}$  with the lowest  $f_\Omega$  value so far is found on the mesh  $M_k$  defined by equation (2.1).
  - Optional SEARCH: Evaluate  $f_\Omega$  on a finite subset of trial points on the mesh  $M_k$  defined by equation (2.1) (the strategy that gives the set of points is usually provided by the user; it must be finite and the set can be empty).
  - Local POLL: Evaluate  $f_\Omega$  on the poll set defined in equation (2.2).
- **PARAMETER UPDATE:**  
If the SEARCH or the POLL step produced an improved mesh point, *i.e.*, a feasible iterate  $x_{k+1} \in M_k \cap \Omega$  for which  $f_\Omega(x_{k+1}) < f_\Omega(x_k)$ , then update  $\Delta_{k+1} \geq \Delta_k$ .  
Otherwise,  $f_\Omega(x_k) \leq f_\Omega(x_k + \Delta_k d)$  for all  $d \in D_k$  and so  $x_k$  is a mesh local optimizer. Set  $x_{k+1} = x_k$ , update  $\Delta_{k+1} < \Delta_k$ .  
Increase  $k \leftarrow k + 1$  and go back to the SEARCH and POLL step.

FIG. 2.1. A simple GPS algorithm

$\Omega$  for some  $\varepsilon > 0$ , then  $\nabla f(\hat{x})^T \omega \geq 0$  for all  $\omega \in T_\Omega(\hat{x})$  and so  $-\nabla f(\hat{x}) \in N_\Omega(\hat{x})$ . Thus,  $\hat{x}$  is a Karush-Kuhn-Tucker point.

The purpose of this paper is to provide an algorithm for constructing sets  $D_k$  that conform to the boundary of  $\Omega$ . If the active constraints are linearly dependent, we apply strategies for the identification of redundant and nonredundant constraints, which are described in the next section, and then construct sets  $D_k$  taking into account only nonredundant constraints. We now pause to outline the main results concerning redundancy from mathematical programming, and then in Section 4, we continue consideration of the GPS and strategies for constructing the sets  $D_k$ .

**3. Redundancy.** In this section, we give essential definitions and results concerning redundancy [3, 6, 8, 11, 15] that are required for our analysis. Then we propose our approach, the projection method, to determining the nonredundant constraints and briefly describe another approach that is applied if some constraints are not identified by the projection method.

We consider the feasible region  $\Omega$  defined by (1.2), and refer to the inequality  $a_j^T x \leq b_j$  as the *j-th constraint*. The region represented by all but the *j*th constraint is given by

$$\Omega_j = \{x \in \mathfrak{R}^n : a_i^T x \leq b_i, i \in I \setminus j\},$$

where  $I \setminus j$  is the set  $I$  with element  $j$  removed.

**3.1.  $\varepsilon$ -active constraints.** In this subsection, we introduce a definition of the  $\varepsilon$ -active constraints and discuss some scaling issues. First we give the definitions used

by Lewis and Torczon [9] and by Price and Coope [12].

DEFINITION 3.1. (e.g., [12]). Let some scalar  $\varepsilon > 0$  be given and  $x_k \in \Omega$ . The  $j$ th constraint is  $\varepsilon$ -active at  $x_k$  if

$$0 \leq b_j - a_j^T x_k \leq \varepsilon. \quad (3.1)$$

DEFINITION 3.2. (e.g., [9]). Let some scalar  $\varepsilon > 0$  be given and  $x_k \in \Omega$ . The  $j$ th constraint is  $\varepsilon$ -active at  $x_k$  if

$$\text{dist}(x_k, H_j) \leq \varepsilon, \quad (3.2)$$

where  $H_j = \{x \in \mathfrak{R}^n : a_j^T x = b_j\}$ , and  $\text{dist}(x_k, H_j) = \min_{y \in H_j} \|y - x_k\|$  is the distance from  $x_k$  to the hyperplane  $H_j$ .

As is easy to see, the  $j$ th constraint can be made  $\varepsilon$ -active at  $x_k$  in the sense of Definition 3.1 by multiplying the inequality  $b_j - a_j^T x_k \geq 0$  by a sufficiently small number. On the other hand, this multiplication does not change the distance between the point  $x_k$  and any  $H_j$  defined in Definition 3.2. In the paper, we prefer to use Definition 3.1, since it is easier to check than Definition 3.2. However, Definition 3.1 is proper, if we assume preliminary scaling of the constraints so that the following lemma applies.

LEMMA 3.3. Let some scalar  $\varepsilon > 0$  be given,  $x_k \in \Omega$ , and  $\|a_j\| = 1$  for all  $j \in I$  in (1.2). Then, for any  $j \in I$ , Definition 3.1 of the  $\varepsilon$ -active constraint is equivalent to Definition 3.2, and the projection  $P_j(x_k)$  of the point  $x_k$  onto the hyperplane  $H_j = \{x \in \mathfrak{R}^n : a_j^T x = b_j\}$  is defined by

$$P_j(x_k) = x_k + a_j(b_j - a_j^T x_k). \quad (3.3)$$

*Proof.* For any  $j \in I$ , the distance from  $x_k$  to the hyperplane  $H_j = \{x \in \mathfrak{R}^n : a_j^T x = b_j\}$  is given by

$$\text{dist}(x_k, H_j) = \frac{|b_j - a_j^T x_k|}{\|a_j\|}. \quad (3.4)$$

Hence, if  $\|a_j\| = 1$  and  $x_k \in \Omega$ , (3.1) is equivalent to (3.2).

By definition of the projection of  $x_k$  onto  $H_j$ ,

$$\|P_j(x_k) - x_k\| = \text{dist}(x_k, H_j).$$

Since  $x_k \in \Omega$  and  $\|a_j\| = 1$ , it follows from (3.4) that  $\text{dist}(x_k, H_j) = b_j - a_j^T x_k$  and

$$P_j(x_k) = x_k + a_j \text{dist}(x_k, H_j) = x_k + a_j(b_j - a_j^T x_k).$$

Hence, (3.3) holds.  $\square$

To satisfy the conditions of Lemma 3.3, we introduce the matrix  $\bar{A}$  that is an additional scaled copy of the matrix  $A$  given in (1.2) and a scaled vector  $\bar{b}$  such that

$$\bar{a}_i = \frac{a_i}{\|a_i\|}, \quad \bar{b}_i = \frac{b_i}{\|a_i\|}, \quad i \in I. \quad (3.5)$$

Consequently,  $\|\bar{a}_i\| = 1$  for all  $i \in I$  and  $\Omega = \{x \in \mathfrak{R}^n : A^T x \leq b\} = \{x \in \mathfrak{R}^n : \bar{A}^T x \leq \bar{b}\} = \{x \in \mathfrak{R}^n : \bar{a}_i^T x \leq \bar{b}_i, i \in I\}$ .

We use the matrix  $\bar{A}$  and the vector  $\bar{b}$  to define the set of indices of the  $\varepsilon$ -active constraints:

$$I(x_k, \varepsilon) = \{i \in I : 0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon\}, \quad (3.6)$$

as well as to apply the projection method for detection of the nonredundant constraints (see Section 3.3.1 for more details.) We refer to the set  $I(x_k, \varepsilon)$  as the *working index set* at the current iterate  $x_k$ .

The paper also makes use of the regions given by

$$\Omega(x_k, \varepsilon) = \{x \in \mathfrak{R}^n : a_i^T x \leq b_i, i \in I(x_k, \varepsilon)\}, \quad (3.7)$$

$$\Omega_j(x_k, \varepsilon) = \{x \in \mathfrak{R}^n : a_i^T x \leq b_i, i \in I(x_k, \varepsilon) \setminus j\}, \quad j \in I(x_k, \varepsilon).$$

Clearly,  $\Omega \subseteq \Omega(x_k, \varepsilon) \subseteq \Omega_j(x_k, \varepsilon)$ .

**3.2. Redundancy in mathematical programming.** In this subsection, we give definitions and theorems consistent with the mathematical programming literature [3, 6, 8, 11, 15].

The following definitions and results are from [11, 15].

**DEFINITION 3.4** (Polyhedron). *A subset of  $\mathfrak{R}^n$  described by a finite set of linear constraints  $P = \{x \in \mathfrak{R}^n : Cx \leq d\}$  is a polyhedron.*

Obviously,  $\Omega$  given by (1.2) and  $\Omega(x_k, \varepsilon)$  given by (3.7) are polyhedra.

**DEFINITION 3.5.** *The points  $x^1, \dots, x^k \in \mathfrak{R}^n$  are affinely independent if the  $k-1$  directions  $x^2 - x^1, \dots, x^k - x^1$  are linearly independent, or alternatively the  $k$  vectors  $(x^1, 1), \dots, (x^k, 1) \in \mathfrak{R}^{n+1}$  are linearly independent.*

**DEFINITION 3.6.** *The dimension of  $P$ , denoted  $\dim(P)$ , is one less than the maximum number of affinely independent points in  $P$ .*

This means that  $P \subseteq \mathfrak{R}^n$  is *full-dimensional* if and only if  $\dim(P) = n$ . We will assume that  $\Omega$  is full-dimensional. If it were not, then a barrier GPS approach would not be a reasonable way to handle the constraints because it would be difficult to find SEARCH or POLL points in  $\Omega$ . Since we assume  $\Omega$  is full dimensional, this implies that its supersets  $\Omega(x_k, \varepsilon)$  and  $\Omega_j(x_k, \varepsilon)$  are full-dimensional.

**DEFINITION 3.7** (Valid inequality). *An inequality  $c_j x \leq d_j$  is a valid inequality for  $P \subseteq \mathfrak{R}^n$  if  $c_j x \leq d_j$  for all  $x \in P$ .*

**DEFINITION 3.8** (Face and Facet). (i)  *$F$  defines a face of the polyhedron  $P$  if  $F = \{x \in P : c_j x = d_j\}$  for some valid inequality  $c_j x \leq d_j$  of  $P$ .  $F \neq \emptyset$  is said to be a proper face of  $P$  if  $F \neq P$ .*

(ii)  *$F$  is a facet of  $P$  if  $F$  is a face of  $P$  and  $\dim(F) = \dim(P) - 1$ .*

**DEFINITION 3.9** (Dominance of inequalities). *If  $c_i x \leq d_i$  and  $c_j x \leq d_j$  are two valid inequalities for  $P \subset \mathfrak{R}_+^n$ ,  $c_i x \leq d_i$  dominates  $c_j x \leq d_j$  if there exists  $u > 0$  such that  $c_i \geq u c_j$  and  $d_i \leq u d_j$ , and  $(c_i, d_i) \neq (u c_j, u d_j)$ .*

**DEFINITION 3.10** (Redundant inequality). *A valid inequality  $c_j x \leq d_j$  is redundant in the description of  $P$  (in other words, the  $j$ th constraint is redundant) if there exist  $k \geq 1$  valid inequalities  $c_i x \leq d_i$  for  $i = 1, \dots, k$  for  $P$ , and weights  $\alpha_i > 0$  for  $i = 1, \dots, k$  such that  $(\sum_{i=1}^k \alpha_i c_i) x \leq \sum_{i=1}^k \alpha_i d_i$  dominates  $c_j x \leq d_j$ .*

The following example illustrates that we can not replace  $\alpha_i > 0$  with  $\alpha_i \geq 0$  in Definition 3.10.

*Example 1.* Let the following inequalities in  $\mathbb{R}^2$  be valid inequalities for some polyhedron  $P$ :

$$\begin{pmatrix} c_1 x \leq d_1 \\ c_2 x \leq d_2 \end{pmatrix} = \begin{pmatrix} x_1 \leq 1 \\ -x_1 \leq 1 \end{pmatrix}. \quad (3.8)$$

If  $\alpha_1$  were equal to zero in Definition 3.10, we would obtain  $\alpha_1 c_1 = (0, 0)$ ,  $\alpha_1 d_1 = 0$ , and

$$(0, 0) = \alpha_1 c_1 \geq u c_2 = (-1, 0), \quad 0 = \alpha_1 d_1 \leq u d_2 = 1, \quad u = 1.$$

Hence, by Definition 3.9, the first inequality in (3.8) would dominate the second one and we would make the wrong conclusion, by Definition (3.10), that the second inequality is redundant in the description of  $P$ .  $\square$

Let us note that since  $\Omega \subseteq \Omega(x_k, \varepsilon)$ , if the  $j$ th constraint is redundant in the description of  $\Omega(x_k, \varepsilon)$ , it is redundant in the description of  $\Omega$ .

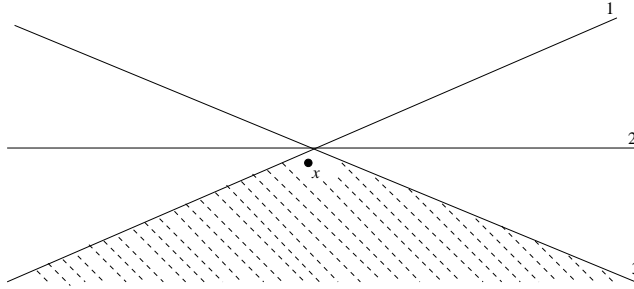


FIG. 3.1. An illustration of  $\varepsilon$ -active and redundant constraints. Constraints 1, 2, and 3 are  $\varepsilon$ -active at the current iterate  $x$  and constraint 2 is redundant.

DEFINITION 3.11 (Interior point). A point  $x \in P$  is called an interior point of  $P$  if  $Cx < d$ .

We need the following results from integer programming [15, pp. 142–144] and [11, pp. 85–92].

PROPOSITION 3.12. [11, Corollary 2.5] A polyhedron is full dimensional if and only if it has an interior point.

THEOREM 3.13. [15, Theorem 9.1] If  $P$  is a full-dimensional polyhedron, it has a unique minimal description

$$P = \{x \in \mathbb{R}^n : c_i x \leq d_i, \quad i = 1, \dots, m\},$$

where each inequality is unique to within a positive multiple.

COROLLARY 3.14. [15, Proposition 9.2] If  $P$  is full-dimensional, a valid inequality  $c_j x \leq d_j$  is necessary in the description of  $P$  if and only if it defines a facet of  $P$ .

Corollary 3.14 means that the following concepts are equivalent for  $\Omega(x_k, \varepsilon)$  defined in (3.7).

- The  $j$ th inequality  $a_j^T x \leq b_j$  defines a facet of  $\Omega(x_k, \varepsilon)$ .
- The  $j$ th inequality  $a_j^T x \leq b_j$  is nonredundant in the description of  $\Omega(x_k, \varepsilon)$ .
- The  $j$ th inequality  $a_j^T x \leq b_j$  is necessary in description of  $\Omega(x_k, \varepsilon)$ , or in other words,

$$\Omega(x_k, \varepsilon) \subsetneq \Omega_j(x_k, \varepsilon). \quad (3.9)$$

Our approach to the identification of nonredundant constraints is based primarily on the following proposition.

PROPOSITION 3.15. Let a working index set  $I(x_k, \varepsilon)$  be given. An inequality  $a_j^T x \leq b_j$ ,  $j \in I(x_k, \varepsilon)$ , is nonredundant in the description of  $\Omega(x_k, \varepsilon)$  if and only if

either  $I(x_k, \varepsilon) = \{j\}$  or there exists  $\bar{x} \in \mathfrak{R}^n$  such that  $a_j^T \bar{x} = b_j$  and  $a_i^T \bar{x} < b_i$  for all  $i \in I(x_k, \varepsilon) \setminus j$ .

*Proof.* Since the case  $I(x_k, \varepsilon) = \{j\}$  is trivial, we give the proof for the case when  $I(x_k, \varepsilon) \setminus j \neq \emptyset$ .

*Necessity.* Since the inequality  $a_j^T x \leq b_j$  is nonredundant, then, by (3.9), there exists  $x^* \in \mathfrak{R}^n$  such that  $a_i^T x^* \leq b_i$  for all  $i \in I(x_k, \varepsilon) \setminus j$ , and  $a_j^T x^* > b_j$ . By Proposition 3.12, there exists an interior point  $\hat{x} \in \Omega(x_k, \varepsilon)$  such that  $a_i^T \hat{x} < b_i$  for all  $i \in I(x_k, \varepsilon)$ . Thus on the line between  $x^*$  and  $\hat{x}$  there is a point  $\bar{x} \in \mathfrak{R}^n$  satisfying  $a_j^T \bar{x} = b_j$  and  $a_i^T \bar{x} < b_i$  for all  $i \in I(x_k, \varepsilon) \setminus j$ .

*Sufficiency.* To obtain a contradiction, we suppose that there exists  $\bar{x} \in \mathfrak{R}^n$  such that  $a_j^T \bar{x} = b_j$  and  $a_i^T \bar{x} < b_i$  for all  $i \in I(x_k, \varepsilon) \setminus j$ , but the inequality  $a_j^T x \leq b_j$  is redundant. Then, using Definition 3.10, we obtain that there exist  $k \geq 1$  valid inequalities  $a_i^T x \leq b_i$  for  $i = 1, \dots, k$  for  $\Omega(x_k, \varepsilon)$ , and weights  $\alpha_i > 0$  for  $i = 1, \dots, k$  such that  $(\sum_{i=1}^k \alpha_i a_i^T) x \leq \sum_{i=1}^k \alpha_i b_i$  dominates  $a_j^T x \leq b_j$ . Thus, by Definition 3.9, there exists  $u > 0$  such that

$$ua_j x \leq \left( \sum_{i=1}^k \alpha_i a_i^T \right) x \leq \sum_{i=1}^k \alpha_i b_i \leq ub_j. \quad (3.10)$$

Since by hypothesis,  $a_i^T \bar{x} < b_i$  for all  $i \neq j$ , it follows from (3.10) that

$$\left( \sum_{i=1}^k \alpha_i a_i^T \right) \bar{x} < \sum_{i=1}^k \alpha_i b_i.$$

Hence, by (3.10),  $a_j \bar{x} < b_j$ . This contradicts the supposition that  $a_j \bar{x} = b_j$ .  $\square$

Proposition 3.15 means that if the  $j$ th constraint,  $j \in I(x_k, \varepsilon)$ , is nonredundant, then there exists a feasible point  $\bar{x} \in \Omega(x_k, \varepsilon)$  such that only this constraint holds with equality at  $\bar{x}$ .

Our approach to the identification of the redundant constraints is based primarily on the following theorem [6].

**THEOREM 3.16.** *The  $j$ th constraint is redundant in system (1.2) if and only if*

$$\text{maximize } a_j^T x, \quad \text{subject to } x \in \Omega_j. \quad (3.11)$$

*has an optimal solution  $x^*$  such that  $a_j^T x^* \leq b_j$ .*

**3.3. Approaches to identification of redundant and nonredundant constraints.** In section 4, to identify the redundant constraints, we use an approach proposed in [6], based on Theorem 3.16 and briefly outlined in Section 3.3.2, and to determine the nonredundant constraints, we apply a method presented in the next subsection.

**3.3.1. A projection method.** In this subsection, we propose the projection method that is intended to identify nonredundant constraints. The main idea of this method is to construct, if it is possible, a point  $\bar{x}$  such that  $a_j^T \bar{x} = b_j$  and  $a_i^T \bar{x} < b_i$  for all  $i \in I(x_k, \varepsilon) \setminus j$ . If such a point  $\bar{x}$  exists, then by Proposition 3.15, the  $j$ th constraint is nonredundant.

We recall that, in (3.5), we defined a scaled copy  $\bar{A}$  of the matrix  $A$  and a scaled vector  $\bar{b}$ . We denote by  $P_j(x_k)$ , the projection of the point  $x_k \in \mathfrak{R}^n$  onto the hyperplane  $H_j = \{x \in \mathfrak{R}^n : \bar{a}_j^T x = \bar{b}_j\}$ . Assume that  $x_k \in \Omega$ . Then by (3.3) and by

$$\|\bar{a}_j\| = 1,$$

$$P_j(x_k) = x_k + \bar{a}_j(\bar{b}_j - \bar{a}_j^T x_k). \quad (3.12)$$

The following proposition is the main one for the projection method.

**PROPOSITION 3.17.** *Let  $x_k \in \Omega$  and let a working index set  $I(x_k, \varepsilon)$  be given. An inequality  $\bar{a}_j^T x \leq \bar{b}_j$ ,  $j \in I(x_k, \varepsilon)$ , is nonredundant in the description of  $\Omega(x_k, \varepsilon)$  if*

$$\bar{a}_i^T P_j(x_k) < \bar{b}_i \quad \text{for all } i \in I(x_k, \varepsilon) \setminus j, \quad (3.13)$$

where  $P_j(x_k)$  is a projection of  $x_k$  onto  $H_j$ .

*Proof.* The proof follows from Proposition 3.15.  $\square$

An application of the projection method follows from Proposition 3.17. Namely, we classify the  $j$ th constraint as nonredundant, where  $j \in I(x_k, \varepsilon)$ , if (3.13) holds for all  $i \in I(x_k, \varepsilon) \setminus j$  with  $P_j(x_k)$  given by (3.12).

**3.3.2. Approach for identifying redundant and nonredundant constraints.** If some constraints have not been identified by the projection method, we apply another approach based on Theorem 3.16 to identify redundant and nonredundant constraints. The book [8] contains description of different methods in the context of this approach. These methods use some very special propositions concerning slack variables and allow the authors to simplify numerical solution of the linear programming (LP) problem (3.11) or to substitute solving the LP problem by some other procedure, which is less expensive from the numerical point of view. We have not included these details of numerical implementation of the solution of the LP problem into the paper, since the reader can find them in [8].

**4. Construction of the set of generators.** The purpose of this section is to provide a detailed algorithm for constructing the set  $D_k$  introduced in Section 2.

Let some scalar  $\varepsilon > 0$  be given, and let  $\bar{a}_i^T$  be the  $i$ th row of the matrix  $\bar{A}^T$  in (3.5). At the current iterate  $x_k$ , we construct the working index set  $I(x_k, \varepsilon)$  as follows:

$$0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon \iff i \in I(x_k, \varepsilon).$$

The last inequality means that every constraint that is active at  $x_k$  or at some point near  $x_k$  appears in  $I(x_k, \varepsilon)$ . In [1], the authors suggest not setting  $\varepsilon$  so small that  $\Delta_k$  is made small by approaching the boundary too closely before including conforming directions that allow the iterates to move along the boundary of  $\Omega$ .

Without loss of generality, we assume that  $I(x_k, \varepsilon) = \{1, \dots, m\}$ , for  $m \geq 2$ . This avoids more cumbersome notation, like  $I(x_k, \varepsilon) = \{i_1(x_k, \varepsilon), \dots, i_m(x_k, \varepsilon)\}$ .

We denote by  $B_k$ , the matrix whose columns are the columns of  $A$  corresponding to the indices  $I(x_k, \varepsilon) = \{1, \dots, m\}$ ; i.e.,

$$B_k = [a_1 \dots a_m]. \quad (4.1)$$

**4.1. Classification of degeneracy at the current iterate.** Let the matrix  $B_k$  be defined by (4.1). At the current iterate  $x_k$ , one of the following cases holds:

- a *nondegenerate case* when the matrix  $B_k$  has full rank;
- a *degenerate case* when  $B_k$  does not have full rank and all nonredundant constraints are linearly independent;
- a *degenerate nonredundant case* when  $B_k$  does not have full rank and all nonredundant constraints are linearly dependent. This case can be illustrated by the following example.

*Example 2.* (Charles Audet sent this example to us.) Suppose that the feasible region  $\Omega$  (1.2) is defined by the following system shown in Figure 4.1.

$$\begin{aligned} x_1 - 2x_2 - 2x_3 &\leq 0 \\ -2x_1 + x_2 - 2x_3 &\leq 0 \\ -2x_1 - 2x_2 + x_3 &\leq 0 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \end{aligned} \tag{4.2}$$

If  $x_k \in \mathbb{R}^3$  is near the origin, all six constraints are active, linearly dependent, and nonredundant. The matrix  $B_k$  is given as

$$B_k = \begin{pmatrix} 1 & -2 & -2 & -1 & 0 & 0 \\ -2 & 1 & -2 & 0 & -1 & 0 \\ -2 & -2 & 1 & 0 & 0 & -1 \end{pmatrix}.$$

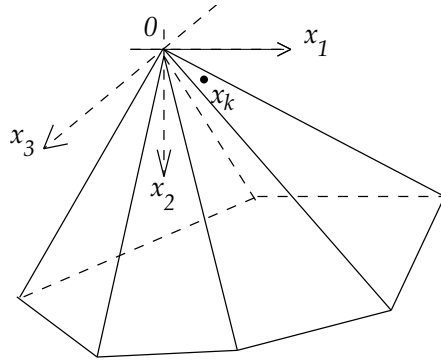


FIG. 4.1. *Example 2. An illustration of the degenerate nonredundant case.*

**4.2. Set of generators.** Following [9], we define the cone  $K(x_k, \varepsilon)$  as the cone generated by the normals to the  $\varepsilon$ -active constraints, and  $K^0(x_k, \varepsilon)$  as its polar:

$$K^0(x_k, \varepsilon) = \{w \in \mathbb{R}^n : a_i^T w \leq 0 \quad \forall i \in I(x_k, \varepsilon)\}. \tag{4.3}$$

The cone  $K^0(x_k, \varepsilon)$  defined by (4.3) can be rewritten as a finitely generated cone [14]:

$$K^0(x_k, \varepsilon) = \{w : w = \sum_{j=1}^r \lambda_j v_j, \quad \lambda_j \geq 0, \quad j = 1, \dots, r\}, \tag{4.4}$$

where the vectors  $v_1, \dots, v_r$  are a set of generators for the cone  $K^0(x_k, \varepsilon)$  defined as follows.

**DEFINITION 4.1** (Set of generators). *A set  $V = \{v_1, \dots, v_r\}$  is called a set of generators of the cone  $K^0(x_k, \varepsilon)$  defined by (4.3) if*

- (i) *any vector  $v \in K^0(x_k, \varepsilon)$  can be represented as a nonnegative linear combination of vectors  $v_i$  from  $V$ , i.e. (4.4) holds;*
- (ii) *no subset of  $\{v_1, \dots, v_r\}$  satisfies (4.4).*

The key idea, which was first suggested by May in [10] and applied to the GPS in [9], is to include in  $D_k$  the generators of the cone  $K^0(x_k, \varepsilon)$ . Hence, the problem of construction of the set  $D_k$  reduces to the problem of constructing the generators  $\{v_1, \dots, v_r\}$  of the cone  $K^0(x_k, \varepsilon)$  and then completing them to a positive spanning set for  $\mathfrak{R}^n$ .

The following proposition means that it is sufficient to construct the set of generators only for nonredundant constraints.

**PROPOSITION 4.2.** *Let  $I(x_k, \varepsilon)$  be a set of indices of constraints that are  $\varepsilon$ -active at the point  $x_k$ . Let  $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$  be the subset of indices of the nonredundant constraints that define  $\Omega(x_k, \varepsilon)$ . Let the cone  $K^0(x_k, \varepsilon)$  be defined by (4.3) and let the cone  $K_N^0(x_k, \varepsilon)$  be given by*

$$K_N^0(x_k, \varepsilon) = \{w \in \mathfrak{R}^n : a_i^T w \leq 0 \quad \forall i \in I_N(x_k, \varepsilon)\}.$$

*If  $\{v_1, \dots, v_p\}$  is a set of generators for  $K_N^0(x_k, \varepsilon)$ , then the set of vectors  $\{v_1, \dots, v_p\}$  is a set of generators for  $K^0(x_k, \varepsilon)$ .*

*Proof.* The proof of this proposition follows from Corollary 3.14.  $\square$

As is mentioned in [9], pattern search methods require their iterates to lie on a rational lattice. To arrange this, Lewis and Torczon [9] placed an additional requirement that the matrix of constraints  $A^T$  in (1.2) is rational. Under this requirement, Lewis and Torczon [9] showed, in the following theorem, that it is always possible to find rational generators for the cones  $K^0(x_k, \varepsilon)$ , which, with the rational mesh size parameter  $\Delta_k$ , makes the GPS points lie on a rational lattice.

**THEOREM 4.3.** *Suppose  $K$  is a cone with rational generators  $V$ . Then there exists a set of rational generators for  $K^0$ .*

Moreover, for the case of linearly independent active constraints, Lewis and Torczon [9] proposed constructing the set of generators for all the cones  $K(x_k, \varepsilon)$ ,  $0 \leq \varepsilon \leq \delta$ , as follows:

**THEOREM 4.4.** *Suppose that for some  $\delta$ ,  $K(x, \delta)$  has a linearly independent set of rational generators  $V$ . Let  $N$  be a rational positive basis for the nullspace of  $V^T$ .*

*Then, for any  $\varepsilon$ ,  $0 \leq \varepsilon \leq \delta$ , a set of rational generators for  $K^0(x, \varepsilon)$  can be found among the columns of  $N$ ,  $V(V^T V)^{-1}$ , and  $-V(V^T V)^{-1}$ .*

As is shown in [9], the matrix  $N$  can be constructed by taking columns of the matrices  $\pm(I - V(V^T V)^{-1} V^T)$ .

Let us recall that we use the scaled matrix  $\bar{A}$  defined in (3.5) to determine  $\varepsilon$ -active, redundant, and nonredundant constraints. Then we use the result stated in Theorem 4.4 together with rational columns of  $A$ , which correspond to the nonredundant and  $\varepsilon$ -active constraints, to obtain a set of rational generators.

A set of generators, which may be irrational in exact arithmetic, can also be found by using the QR factorization of the matrix  $V$ . The following corollary shows how to use the QR factorization of  $V$  to construct the generators for all the cones  $K^0(x_k, \varepsilon)$ ,  $0 \leq \varepsilon \leq \delta$ . We recall that the full QR factorization of  $V$  can be represented as

$$V = [Q_1 \ Q_2] \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix},$$

where  $R_1$  is upper triangular and  $\text{rank } R_1 = \text{rank } V$ , and the columns of  $Q_1$  form an orthonormal basis for the space spanned by the columns of  $V$ , while the columns of  $Q_2$  constitute an orthonormal basis for nullspace of  $V^T$ .

**COROLLARY 4.5.** *Suppose that for some  $\delta$ ,  $K(x, \delta)$  has a linearly independent set of rational generators  $V$ . Then, for any  $\varepsilon$ ,  $0 \leq \varepsilon \leq \delta$ , a set of generators for  $K^0(x, \varepsilon)$  can be found among the columns of  $Q_2$ ,  $Q_1 R_1 (R_1^T R_1)^{-1}$ , and  $-Q_1 R_1 (R_1^T R_1)^{-1}$ .*

*Proof.* By substituting  $V = QR$  and using the properties of the matrices in the QR factorization, we obtain

$$V(V^T V)^{-1} = QR((QR)^T(QR))^{-1} = QR(R^T Q^T Q R)^{-1} = QR(R^T R)^{-1}.$$

By applying Theorem 4.4 and by taking into account that columns of  $Q_2$  span the nullspace of  $V^T$ , we obtain the statement of the corollary.  $\square$

From the theoretical point of view, a set of generators obtained by using Corollary 4.5 may be irrational since an implementation of the QR decomposition involves calculation of square roots. Of course, we use floating point arithmetic, and so these generators are rational, but they probably generate a different cone due to numerical errors. Still the same is true of generators found by using the LU factorization.

**4.3. An algorithm for constructing the set of generators.** This section presents an algorithm for constructing a set of generators for the cone  $K^0(x_k, \varepsilon)$  at the current iterate  $x_k$  for a given parameter  $\varepsilon$ .

**4.3.1. Comments on the algorithm.** The algorithm consists of two main parts. In the first part, we determine the set of indices of the nonredundant  $\varepsilon$ -active constraints  $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$  and form the matrix  $B_N$  whose columns are the columns of  $A$  corresponding to the indices in  $I_N(x_k, \varepsilon)$ . We use information about the set  $I_N(x_k, \varepsilon)$  from the previous iterations of the GPS algorithm. Namely, we put into the set  $I_N(x_k, \varepsilon)$  all indices that correspond to the  $\varepsilon$ -active constraints at the current iterate and that were detected as indices of the nonredundant constraints at the previous iterations of the algorithm. In the second part of the algorithm, we construct the set of generators  $D_k$  required by the GPS and by Theorem 2.2.

First of all, we try to identify the nonredundant active constraints. If the matrix  $B_k$  defined by (4.1) has full rank, then all  $\varepsilon$ -active constraints are nonredundant,  $I_N(x_k, \varepsilon) = I(x_k, \varepsilon)$ , and  $B_N = B_k$ .

If the matrix  $B_k$  does not have full rank and we have indices that have not been classified at the previous iterations of the algorithm, we suggest using two steps in succession.

The first strategy is intended to determine nonredundant constraints cheaply by applying the projection method described in section 3.3.1. By Proposition 3.17, if the projection  $P_j(x_k)$  of the current iterate  $x_k$  onto the hyperplane  $H_j = \{x \in \mathbb{R}^n : \bar{a}_j^T x = \bar{b}_j\}$  is feasible, and only the  $j$ th constraint holds with equality at  $P_j(x_k)$ , then the  $j$ th constraint is nonredundant, and we can put index  $j$  into the set  $I_N(x_k, \varepsilon)$ . If some constraints have not been identified by the projection method, we can either apply the projection method with some other point  $\tilde{x} \neq x_k$  or apply the second strategy.

The second strategy is intended to classify redundant and nonredundant constraints among those constraints that have not been determined as nonredundant by the projection method. To identify the constraint, an approach outlined in [6] is applied. The basis of this second strategy is provided by Theorem 3.16. If the number of constraints we have to identify is too big, we can skip an application of the second strategy to the algorithm and construct a set of generators using the set  $I_N(x_k, \varepsilon)$  defined by application of the first strategy. Then, while doing the poll step, if we find some point  $\bar{x} = x_k + \Delta \bar{d}$ , where  $\bar{d}$  is some column of  $D_k$ , such that  $a_j^T \bar{x} > b_j$  and  $a_i^T \bar{x} \leq b_i$  for all  $i \in I(x_k, \varepsilon) \setminus j$ , we can conclude that  $\Omega(x_k, \varepsilon) \subsetneq \Omega_j(x_k, \varepsilon)$ . Hence, by Corollary 3.14, the  $j$ th constraint is nonredundant and we include  $j$  into set  $I_N(x_k, \varepsilon)$ .

When we have specified all redundant and nonredundant constraints, we compose the matrix  $B_N$  of those columns of the rational matrix  $A$  that correspond to nonredundant constraints. To determine the rank of  $B_N$ , the QR factorization can be used. If  $B_N$  has full rank, we can apply the function *Set\_QR* or *Set\_LU*, which uses the QR or LU decomposition of  $B_N$ , to construct the set of generators. Both functions are constructed on the basis of results proved in Theorem 4.4 and in Corollary 4.5, where  $V = B_N$ .

If the matrix  $B_N$  does not have full rank, we apply the result proved in [12]. Specifically, to construct the set  $D_k$ , it is sufficient to apply the function *Set\_QR* or *Set\_LU* to all maximal linearly independent subsets of the columns of  $B_N$ . We can estimate the number  $S$  of these subsets. If  $B_N$  has  $m_N$  columns and rank  $r$ , then

$$S = \frac{m_N!}{r!(m_N - r)!}. \quad (4.5)$$

In this worst case, some vertex enumeration techniques [2] mentioned in [9] might be necessary, but our procedure for classification of the constraints should eliminate this expense for many cases.

**4.3.2. Algorithm.** We denote the set of indices of the nonredundant active constraints by  $I_N(x_k, \varepsilon)$ . Thus, for  $j \in I(x_k, \varepsilon)$ ,

- (i) if  $j \in I_N(x_k, \varepsilon)$ , the inequality  $a_j^T x \leq b_j$  is nonredundant; and
- (ii) if  $j \in I(x_k, \varepsilon) \setminus I_N(x_k, \varepsilon)$ , the inequality  $a_j^T x \leq b_j$  is redundant.

We use  $I_N \subseteq I$  to denote the set of indices that are detected as nonredundant at some iteration of the algorithm. Thus  $I_N = \emptyset$  in the beginning of the algorithm.

We denote the rational matrix in (1.2) by  $A^T$  and the scaled matrix defined in (3.5) by  $\bar{A}^T$ . The matrix  $B_k$  is defined by (4.1) and is composed of columns  $a_j$  of  $A$ , where  $j \in I(x_k, \varepsilon)$ , while the matrix  $B_N$  is composed of those columns of  $A$  whose indices are in the set  $I_N(x_k, \varepsilon)$ . Thus  $B_N$  consists of the normal vectors to the nonredundant constraints.

**Algorithm for constructing the set of generators  $D_k$ .**

Let the current iterate  $x_k$  and a parameter  $\varepsilon > 0$  be given.

*% Part I. Constructing the set  $I_N(x_k, \varepsilon)$*

*% Constructing the working index set  $I(x_k, \varepsilon)$*

**for**  $i = 1$  **to**  $|I|$

**if**  $0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon$

$I(x_k, \varepsilon) \leftarrow i$ ;  $B_k \leftarrow a_i$ ;

**endif**

**endfor**

**if**  $\text{rank}(B_k) = |I(x_k, \varepsilon)|$

*%  $B_k$  has full rank, hence, all constraints are nonredundant*

$I_N(x_k, \varepsilon) \leftarrow I(x_k, \varepsilon)$ ;  $B_N \leftarrow B_k$ ;

**else**

*% using information from the previous iterations of the algorithm*

$I_N(x_k, \varepsilon) \leftarrow \{I(x_k, \varepsilon) \cap I_N\}$ ;

**if**  $\{I(x_k, \varepsilon) \setminus I_N(x_k, \varepsilon)\} \neq \emptyset$

*% there are active constraints that have not been identified*

*% at the previous iterations*

```

% Identification of the nonredundant and redundant constraints
for  $j = 1$  to  $|I(x_k, \varepsilon)|$ 
    % the first strategy
    %  $P_j(x_k)$  is the projection of  $x_k$  onto  $\{x \in \mathbb{R}^n : \bar{a}_j^T x = \bar{b}_j\}$ 
     $P_j(x_k) = x_k + \bar{a}_j(\bar{b}_j - \bar{a}_j^T x_k)$ ;
    if  $\bar{a}_i^T P_j(x_k) < \bar{b}_i$  for all  $i \in I \setminus j$ 
        % equality at  $P_j(x_k)$  holds for only the  $j$ th constraint
         $I_N(x_k, \varepsilon) \leftarrow j$ ;  $B_N \leftarrow a_j$ ;
    else % at least two constraints hold with equality at  $P_j(x_k)$ 
        % or the point  $P_j(x_k)$  is infeasible;
        % the second strategy
        solve LP problem (3.16); let  $x^*$  be a solution to (3.16);
        if  $a_j^T x^* \leq b_j$  % the  $j$ th constraint is redundant
            take  $a_j x \leq b_j$  out of  $\Omega$  and
            take  $j$  out of the sets  $I$  and  $I(x_k, \varepsilon)$ ;
        else % the  $j$ th constraint is nonredundant
             $I_N(x_k, \varepsilon) \leftarrow j$ ;  $B_N \leftarrow a_j$ ;
        endif
    endif
endfor
endif
    % saving information for the next iterations
     $I_N \leftarrow I_N(x_k, \varepsilon)$ ;

    % Part II. Constructing the set of generators  $D_k$ 
    if  $\text{rank}(B_N) = |(\mathbf{I}_N)(\mathbf{x}_k, \varepsilon)|$ 
        % nondegenerate case
         $[D_k] = \text{Set\_QR}(B_N)$  or  $[D_k] = \text{Set\_LU}(B_N)$ ;
    else
        % degenerate nonredundant case
         $[D_k] = \text{Set}(B_N)$ ;
    endif

```

Function *Set\_QR* constructs a set of generators by using the QR decomposition. The procedure is defined as follows.

```

function  $[D] = \text{Set\_QR}(B)$ 
     $[Q, R] = qr(B)$ ;
     $r = \text{rank}(R)$ ;
     $[Q1, Q2, R1] = \text{decomposition}(Q, R, r)$ ;
     $D1 \leftarrow \pm(Q1 * R1 * \text{inv}(R1' * R1))$ ;
     $D2 \leftarrow \pm Q2$ ;
     $D = [D1 \ D2]$ ;
end

```

Function *Set\_LU* is described next and constructs a set of generators by using the LU factorization.

```

function [D] = Set_LU (B)
    D1 ← ±(V * inv(V' * V));
    D2 ← ±(I - V * inv(V' * V) * V');
    D = [D1 D2];
end

```

Function *Set* constructs a set of generators in the case when the matrix  $B_N$  does not have full rank. If  $B_N$  has  $m_N$  columns and rank  $r$ , then the number  $S$  of the maximal linearly independent subsets of the columns of  $B_N$  is given by (4.5).

```

function [D] = Set (B)
    for i = 1 to S
        Bi is composed of r linearly independent columns of B;
        [Di] = Set_QR(Bi) or [Di] = Set_LU(Bi);
        D ← D ∩ Di;
        %D ∩ Di eliminates the identical vectors from the set D %
    endfor
end

```

To reduce the number of operations in the function *Set*, we can use properties of QR factorizations. Namely, at the first step, a QR decomposition of  $B$  is computed. Then at the  $i$ th step,  $i = 2, \dots, S$ , only one of the first  $r$  columns of  $B$  is replaced with some  $j$ th column of  $B$ , where  $j > r$ , in such a way that  $B_{i_1} \neq B_{i_2}$  for any steps  $i_1$  and  $i_2$ , and this is used to simplify computing another QR factorization of  $B$ . The QR factorization of  $B$  is used to construct a set of generators for  $B_i$ , since  $B_i$  can be defined as the first  $r$  columns of  $B$  at the  $i$ th step.

In the next two tables, we present some numerical results. In Table 4.1, we report results regarding identification of the nonredundant and redundant constraints at the current iterate  $x_k$  and regarding constructing the set of the nonredundant constraints  $I_N(x_k, \varepsilon)$ . In Table 4.2, we report results concerning the number of iterations in the GPS algorithm that is described in Fig.2.1 and is applied to a problem with an increasing number of redundant constraints.

TABLE 4.1  
Constructing the set  $I_N(x_k, \varepsilon)$  at the current iterate  $x_k$

variables	$ I(x_k, \varepsilon) $	$ I_N(x_k, \varepsilon) $	detected as nonredundant	
	$\varepsilon$ -active constraints	nonredundant constraints	by the projection method	by the LP program
3	6	6	6	
5	6	5	4	1
5	7	7	6	1
5	7	7	5	2

In Table 4.2, we report results for the same problem with no redundant constraints in the first row, with one additional redundant constraint in the second row, and with two additional redundant constraints in the third row. Since the algorithm detects the redundant constraints and takes them out, the number of iterations is the same in all three tests.

TABLE 4.2  
GPS algorithm in the degenerate case

variables	constraints	redundant constraints	iterations
5	9	0	18
5	10	1	18
5	11	2	18

**5. Concluding remarks.** It is interesting to compare our results with those of Lewis and Torczon [9] and of Price and Coope [12]. Lewis and Torczon [9] proved that it is possible to find a set of generators for the cones  $K^0(x_k, \varepsilon)$  in both the degenerate and nondegenerate case, but left all details of numerical implementation for future work.

Price and Coope [12] presented a new result that can be used for constructing a set of generators in the degenerate case. It follows from their result that, in order to construct a set of generators, it is sufficient to consider maximal linearly independent subsets of the active constraints. However, this approach implies enumeration of all possible linearly independent subsets of maximal rank and does not take into account properties of the problem that can help to reduce this enumeration. Price and Coope [12] outlined an algorithm for constructing frames, but did not consider details of the numerical implementation in the degenerate case.

To construct the set of generators, we first classify constraints as redundant and nonredundant by applying some results concerning redundancy from the mathematical programming literature and by using our approach presented in the paper. We give a detailed algorithm for constructing the set of generators. However, the degenerate nonredundant case, when all constraints are nonredundant but linearly dependent, still implies enumeration of all maximal linearly independent subsets of the constraints. Therefore, the issue left for future analysis is whether it is possible to reduce the enumeration of the subsets in the degenerate nonredundant case.

This work was begun at the IMA while the first author was a postdoctoral fellow and the second was a longterm visitor. We thank the IMA for providing such a fine atmosphere for collaboration.

#### REFERENCES

- [1] C. AUDET AND J. E. DENNIS JR., *Analysis of generalized pattern searches*, SIAM J. Optim., 13 (2003), pp. 889–903.
- [2] D. M. AVIS AND K. FUKUDA, *A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra*, Discrete Comput. Geom., 8 (1992), pp. 295–313.
- [3] H. C. P. BERBEE, C. G. E. BOENDER, A. H. G. R. KAN, C. L. SCHEFFER, R. L. SMITH, AND J. TELGEN, *Hit-and-run algorithms for the identification of nonredundant linear inequalities*, Math. program., 37 (1987), pp. 184–207.
- [4] D. P. BERTSEKAS, *Nonlinear programming*, Athena Scientific, Belmont, MA, 1999.
- [5] A. BONEH, S. BONEH, AND R. J. CARON, *Constraint classification in mathematical programming*, Math. program., 61 (1993), pp. 61–73.
- [6] R. J. CARON, J. F. McDONALD, AND C. M. PONIC, *A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary*, J. Optim. Theory Appl., 62 (1989), pp. 225–237.
- [7] F. H. CLARKE, *Optimization and nonsmooth analysis*, SIAM Classics in applied mathematics Vol.5 (1990), Philadelphia.

- [8] M. H. KARWAN, V. LOTFI, J. TELGEN, AND S. ZIONTS, *Redundancy in Mathematical programming*, Springer-Verlag, Berlin, 1983.
- [9] R. M. LEWIS AND V. TORCZON, *Pattern search methods for linearly constrained minimization*, SIAM J. Optim., 10 (2000), pp. 917–941.
- [10] J. H. MAY, *Linearly constrained nonlinear programming: a solution method that does not require analytic derivatives*, PhD thesis, Yale University, December 1974.
- [11] G. L. NEMHAUSER AND L. A. WOLSEY, *Integer and combinatorial optimization*, John Wiley & Sons, New York, 1988.
- [12] C. J. PRICE AND I. D. COOPE, *Frames and grids in unconstrained and linearly constrained optimization: a non-smooth approach*, Optimization and Engineering, to appear.
- [13] L. N. TREFETHEN AND D. BAU, III, *Numerical linear algebra*, SIAM, Philadelphia, 1997.
- [14] J. VAN TIEL, *Convex Analysis*, John Wiley & Sons, New York, 1984.
- [15] L. A. WOLSEY, *Integer programming*, John Wiley & Sons, New York, 1998.