JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics
Vol. 71B, No. 1, January–March 1967

Algorithms for Frames and Lineality Spaces of Cones*

Roger J.-B. Wets and Christoph Witzgall****

Institute for Basic Standards, National Bureau of Standards, Washington, D.C. 20234

(November 3, 1966)

A frame of a cone C is a minimal set of generators, and the lineality space L of C is the greatest linear subspace contained in C. Algorithms are described for determining a frame and the lineality space of a cone $C(S)$ spanned by a finite set S. These algorithms can be used for determining the vertices, edges, and other faces of low dimension of the convex hull of a finite set $H(S)$. All algorithms are based on the simplex method of linear programming. The problem of finding the lineality space can be successively reduced to problems in spaces of lower dimensions.

Key Words: Algorithm, cone, convex hull, face, frame, lineality space, linear programming.

Introduction

Let

$$S=\{A_1, \ldots, A_n\}$$

be a family of points in R^m. We denote by $L(S)$ its *linear hull*, by $H(S)$ its *convex hull*, and by $C(S)$ its *conical* or *positive hull*, i.e., the convex polyhedral cone expressible as the set of all nonnegative weighted sums of elements of S. We use the conventions $L(\phi)=C(\phi)=\{0\}$.

A subfamily $T\subseteq S$ is called a *frame* [4][1] of $C(S)$ if $C(T)=C(S)$ but $C(T-\{A_j\})\neq C(T)$ for each $A_j\in T$. The greatest linear subspace contained in $C(S)$ is called its *lineality space* [5]. Two main problems are, given S, to find a frame of $C(S)$ and to determine the lineality space of $C(S)$. These two problems are closely related.

Several important problems are equivalent to or included in these two main problems. Consider, for instance, the system of linear inequalities

$$A_j^TX\leq b_j, \qquad j=1, \ldots, n,$$

for $X\in R^m$. Removing redundant constraints amounts to finding a frame for the cone spanned by the vectors $\hat{A}_j=\begin{pmatrix}A_j\\ b_j\end{pmatrix}$, $j=1, \ldots, n$, and $\begin{pmatrix}0\\ 1\end{pmatrix}$. Determining the dimension of a polyhedron given by linear inequalities, can be reduced to determining the dimension of a polyhedral cone $C=\{U|A^TU\leq 0\}$ (see [10]). As to the latter problem, it suffices to note that the linear hull $L(C)$ is the orthogonal complement of the lineality space of the polar cone $C^p=C\{A_1, \ldots, A_n\}$, where $A_1, \ldots, A_n$ are the columns of A. This problem also arises if the uniqueness of an optimal solution to a linear program is to be established in the presence of degeneracy.

Clearly, the problem of finding the vertices of a convex hull $H(S)$ can be solved by finding the frame of a suitable cone. It should not be confused with the problem of finding the vertices of a polyhedron defined by linear inequalities. The latter problem corresponds to finding the *facets*, that is, the proper faces of highest dimension, of some $H(S)$. Even a moderate number of inequalities is apt to generate a huge number of vertices [6, 9], and by the same token, the number of facets of $H(S)$ may be extremely large compared to the cardinality of S. Thus finding the facets of $H(S)$ is inherently more difficult than finding the vertices of $H(S)$.

The problem of finding the *edges* of $H(S)$, and other facets of low dimension, may still be expected to be essentially easier than to find the facets of $H(S)$. Indeed, one is tempted to conjecture that an efficient determination of the facets requires prior determination of the lower dimensional faces.

Note that finding the edges of $H(S)$ could be accomplished by finding the vertices of $H(S_2)$, where

$$S_2:=\{A_{(ij)}=\tfrac{1}{2}(A_i+A_j)\,|\,i<j\}.^2$$

*This paper was in part stimulated by and profited from discussions with A. J. Goldman of the National Bureau of Standards. Thanks for assistance in coding some of the algorithms for automatic computers go to T. Bray and M. Geier.

**Mathematics Research Laboratory, Boeing Scientific Research Laboratories, Seattle, Wash. 98110.

[1] Figures in brackets indicate the literature references at the end of this paper.

[2] The symbol ":=" stands for "is defined by."

However, this procedure is not recommended since the cardinality of S_2 tends to become quite large. We shall prefer a more compact technique, which works with S rather than S_2 or its analogs for higher dimensional faces, and which is closely related to an algorithm for solving the second main problem of determining the lineality space of a cone. The faces of $H(S)$ will be characterized in terms of sign patterns of matrices representing $H(S)$. This aspect is examined more closely in [13].

Charnes [2], Motzkin, Raiffa, Thompson, and Thrall [11], and Farrell and Fieldhouse [14] have proposed methods for finding the facets *and* the vertices of a convex polyhedron P determined by a system of linear inequalities. On the other hand, Goldstein [7] suggests an algorithm for solving the same problem where P is given as the convex hull of a finite set of points. Recently, Thompson, Tonge, and Zionts [12] published a method for removing redundant constraints. One of our algorithms for determining a frame of the cone $C(S)$ essentially coincides with theirs.

We shall give algorithms both for finding a frame for $C(S)$, and for determining the lineality space of $C(S)$. As will be pointed out, the latter algorithm can be modified to decide whether a given subset of S characterizes a face of $C(S)$. Applied to single elements of S, such an algorithm can also be used to find the frame of a pointed cone $C(S)$ and, therefore, the vertices of a convex hull $H(S)$.

All algorithms in this paper are based on the simplex method for linear programming (see, for instance, [3]). Not surprisingly, there are "primal" and "dual" algorithms in each case. We shall describe a primal algorithm for finding a frame and a dual algorithm for determining the lineality space of $C(S)$. The remaining two algorithms are readily constructed by the reader.

It has been our experience with the above algorithms that in the presence of degeneracies round-off errors may cause cycling if zero tolerances are not chosen and handled correctly. This danger is diminished if degeneracy provisions are made, but we shall not include such provisions in the descriptions of our algorithms.

1. General Remarks

A set S may contain more than one frame for $C(S)$, and the cardinalities of these frames may differ. For instance, if $C(S)$ is a linear subspace of dimension l, then the cardinality of a frame T may range from $l+1$ to $2l$. We shall only concern ourselves with finding *some* frame for $C(S)$, and not, for instance, a frame of minimum cardinality.

A point $A_c \in S$ is *redundant in* S if $C(S-\{A_c\}) = C(S)$; otherwise, A_c is *necessary in* S. The algorithms for finding a frame will consist of repeated applications of a subalgorithm which determines for some $A_c \in \bar{S} \subseteq S$ whether it is redundant or necessary in $\bar{S}$. The subset $\bar{S}$ arises from S by deleting those columns that were already found to be redundant.

Now let L be the lineality space of $C(S)$. Clearly,

$$L = C(S) \cap C(-S).$$

We partition the set S into two classes, its *lineal part* $S_L := S \cap L$ and its *conical part* $S_C := S - S_L$. Then

$$L = C(S_L). \tag{1.1}$$

PROOF. $S_L \subseteq L$, hence $L \supseteq C(S_L)$. To prove the other inclusion, suppose $X \in L$. Then $X \in C(S)$ and $X \in C(-S)$. Hence $X = \Sigma A_i u_i$, $u_i \geq 0$, and $X = \Sigma(-A_i)v_i$, $v_i \geq 0$. It follows that

$$\Sigma A_i(u_i + v_i) = 0, \qquad u_i + v_i \geq 0.$$

If $u_i + v_i > 0$, and in particular if $u_i > 0$, then clearly $-A_i \in C(S)$ and, therefore, $A_i \in S_L$. Thus $X \in C(S_L)$—[3].

Let $\bar{S}$ be a subset of S such that $\bar{S} \supseteq S_L$. Let $\bar{L}$ denote the lineality space of $C(\bar{S})$. Then (1.1) gives

$$L = C(S_L) \cap C(-S_L) \subseteq C(\bar{S}) \cap C(-\bar{S}) \subseteq C(S) \cap C(-S) = L,$$

whence $L = \bar{L}$. As a consequence

$$\bar{S}_L = S_L \text{ for } S_L \subseteq \bar{S} \subseteq S. \tag{1.2}$$

We shall describe an algorithm for finding the lineal part S_L of a given set S. Again, this algorithm will be based on a subalgorithm, which decides whether a specified $A_c \in \bar{S} \subseteq S$ is in $\bar{S}_L$. The set $\bar{S}$ arises from S by deleting those points A_i which were already found to be in S_C. This deletion is justified by (1.2).

We proceed to show how the problem of finding faces of $C(S)$ relates to the problem of determining lineality spaces. The faces of a cone C are *extreme subsets*, that is, if the sum of points in C lies on a face F of C, then so do all the summands. More precisely,

(1.3) *$X \in F$, and $X = \Sigma Y_i u_i$ with $Y_i \in C$ and $u_i \geq 0$, imply $u_i = 0$ for $Y_i \notin F$.*

This property characterizes faces. If F is a face of $C(S)$, and if $S_F := F \cap S$, then

(1.4) *$L(S_F)$ is the lineality space of $C(S \cup (-S_F))$.*

PROOF. $C(S \cup (-S_F)) \subseteq C(S_F \cup (-S_F)) = L(S_F)$. Now suppose

$$X \notin C(S \cup (-S_F)) \cap C((-S) \cup S_F).$$

Then there exist $u_i, w_i, v_i, z_i \geq 0$ such that

$$X = \sum_{A_i \in S} A_i u_i - \sum_{A_i \in S_F} A_i w_i = -\sum_{A_i \in S} A_i v_i + \sum_{A_i \in S_F} A_i z_i.$$

[3] The symbol "—" marks the end of a proof.

Hence

$$\sum_{A_i \in S} A_i(u_i + v_i) = \sum_{A_i \in S_F} A_i(w_i + z_i).$$

By (1.3), $u_i + v_i = 0$ if $A_i \notin S_F$. In particular, $u_i = 0$ if $A_i \notin S_F$. Hence

$$X = \sum_{A_i \in S_F} A_i u_i - \sum_{A_i \in S_F} A_i w_i,$$

which implies $X \in L(S_F)$. —

The points $A_1, \ldots, A_k$ (all in S) are said to *subdetermine* a face F of $C(S)$ if

$$F = L\{A_1, \ldots, A_k\} \cap C(S).$$

We then have

(1.5) *The points* $A_1, \ldots, A_k$ *in* S *subdetermine a face of* $C(S)$ *if and only if* $L\{A_1, \ldots, A_k\}$ *is the lineality space of* $C(S \cup \{-A_1, \ldots, -A_k\})$.

PROOF. Suppose $L\{A_1, \ldots, A_k\}$ is the lineality space of $C(S \cup \{-A_1, \ldots, -A_k\})$. In order to prove that $F := L\{A_1, \ldots, A_k\} \cap C(S)$ is a face, we have to show according to (1.3) that if $X \in F$, $X = \Sigma A_i u_i$ with $u_i \geq 0$, and $u_h > 0$, then $A_h \in F$. Suppose, therefore, that X and u_h are as above. Then

$$A_h = \frac{1}{u_h}(X + Z),$$

where

$$Z := -\sum_{i \neq h} A_i u_i \in C((-S) \cup \{A_1, \ldots, A_k\}).$$

Now X is in $L\{A_1, \ldots, A_k\}$, which is by hypothesis the lineality space of $C(S \cup \{-A_1, \ldots, -A_k\})$. Therefore, $X \in C((-S) \cup \{A_1, \ldots, A_k\})$, whence $A_h \in C((-S) \cup \{A_1, \ldots, A_k\})$. It follows that A_h is in the lineality space of $C(S \cup \{-A_1, \ldots, -A_k\})$. Thus $A_h \in L\{A_1, \ldots, A_k\} \cap C(S) = F$.

On the other hand, if $F := L\{A_1, \ldots, A_k\} \cap C(S)$ is a face, then $L(S_F) = L\{A_1, \ldots, A_k\}$ is the lineality space of $C(S \cup (-S_F)) = C(S \cup \{-A_1, \ldots, -A_k\})$ by virtue of (1.4). —

Note that if $A_1, \ldots, A_k$ subdetermine F, then $L(F) = L\{A_1, \ldots, A_k\}$. Hence

$$\dim(F) = k$$

if $A_1, \ldots, A_k$ are linearly independent. Thus any algorithm for determining lineal parts can be used for deciding whether a given set of points $\{A_1, \ldots, A_k\} \subseteq S$ subdetermines a k-face (= face of dimension k) of $C(S)$.

2. Cones and Equivalent Matrices

By definition, the point A_c is redundant in S if and only if $C(S - \{A_c\}) = C(S)$. This in turn is equivalent to $A_c \in C(S - \{A_c\})$, or

$$A_c = \sum_{j \neq c} A_j u_j, \qquad u_j \geq 0. \tag{2.1}$$

The point A_c is in the lineal part S_L of S, if and only if $A_c \in L = C(S) \cap C(-S)$. This is equivalent to $-A_c \in C(S - \{A_c\})$, or

$$-A_c = \sum_{j \neq c} A_j v_j, \qquad v_j \geq 0. \tag{2.2}$$

The standard "phase I" simplex procedures of linear programming are available for deciding the solvability of (2.1) or (2.2) (see, for instance, [3]). All that remains is to integrate such a procedure with the overall algorithm in a way that minimizes the rearrangement of data.

The points $A_1, \ldots, A_n$ of S can be represented by an $m \times n$ matrix

$$A := (A_1, \ldots, A_n).$$

The matrix A is in *canonical form* if it contains an $m \times m$ permutation matrix, called *basis*. Its columns are the *basic* columns; all other columns are *nonbasic*. If A is in canonical form, and if the column A_j is nonbasic, then the elements of A_j constitute the representation of the point A_j in terms of the basis of A. Criteria for redundancy and lineality will be therefore particularly simple if the matrix A is in canonical form.

Two matrices A and $\overline{A}$ are called *equivalent* if there exist (possibly rectangular) matrices T and $\overline{T}$ such that $TA = \overline{A}$ and $\overline{T}\overline{A} = A$. If and only if A and $\overline{A}$ are equivalent in this sense, then for all $X \in R^n$:

$$\mathrm{AX} = 0 \text{ } \textit{if and only if} \text{ } \overline{\mathrm{A}}\mathrm{X} = 0. \tag{2.3}$$

Both (2.1) and (2.2) express the existence of a suitable *linear dependence*

$$AX = 0, \qquad X \neq 0$$

of the columns of A. By (2.3) equivalent matrices admit the same linear dependences. If, therefore, A is replaced by any equivalent matrix $\overline{A}$ — preferably in canonical form — then the same columns (= columns with the same indices) will be redundant in the respective column sets, and the same columns will be in the lineal parts.

3. Determining a Frame of $C(S)$

For matrices A in canonical form the following criteria are immediate:

(3.1) *If* A_j *is nonbasic and* $A_j \geqq 0$, *then* A_j *is redundant.*

(3.2) *If* A_j *is nonbasic and has exactly one positive entry* a_{rj}, *then the basic column* A_c *for which* $a_{rc} = 1$ *is redundant.*

The following two criteria hold whether A is canonical or not:

(3.3) *If the* i*th row of* A, *henceforth denoted* ${}_iA$, *contains exactly one negative entry* a_{is}, *then* A_s *is necessary.*

(3.4) *If* ${}_iA$ *contains exactly one positive entry* a_{is}, *then* A_s *is necessary.*

As was pointed out before, any "phase I" simplex procedure can be used for deciding whether a given column is redundant. We prefer a variant (described in [8]) without artificial variables which works on each infeasible row separately, treating it to some extent as an objective function while conserving the feasibilities already achieved. While this variant may not be the most efficient one for finding a first feasible solution—using a positive combination of the infeasible rows is in general better—the terminal situations of this variant are precisely the ones to which criteria (3.1) and (3.3) apply; it is therefore particularly easy to implement.

The matrix A will be repeatedly transformed and some of its columns may be deleted. In the following description of the algorithm, the symbol "A" will always refer to the particular matrix at hand. It will also be convenient, not to change the indexing of columns which remain after others have been deleted.

(3.5) *Primal algorithm for determining a frame:*

(i) (*Canonical form*) Use Jordan elimination for bringing A into canonical form. Every column is labeled "undecided."

(ii) (*Constant column*) Select a nonbasic undecided column A_c as "constant column." If there are no such columns, go to (viii); else proceed to (iii).

(iii) (*Pilot row*) If $A_c \geqq 0$, delete A_c and return to (ii). Else select ${}_pA$ such that $a_{pc} < 0$; call it the "pilot row."

(iv) (*Pivot column*) If a_{pc} is the only negative entry in the pilot row, change the label of A_c from "undecided" to "necessary" and go to (ii). Else select a "pivot column A_l such that $a_{pl} < 0$ and $l \neq c$.

(v) (*Pivot row*) Select a "pivot row" ${}_rA$ such that

$$0 \leqq \frac{a_{rc}}{a_{rl}} \leqq \min \left\{ \frac{a_{ic}}{a_{il}} \,\middle|\, a_{ic} \geqq 0, \quad a_{il} > 0 \right\}.$$

There exists always an index r of this kind, since either the minimum on the right-hand side is finite and assumed—we use the convention $\min(\phi) = +\infty$ —, or $r = p$ is a permissible choice.

(vi) (*Pivoting*) Execute a simplex step (= Jordan transformation) with a_{rl} as pivot. In the absence of degeneracies, this will increase the old entry a_{pc} while keeping nonnegative entries of A_c nonnegative.

(vii) (*Return*) If the new entry a_{pc} is still negative, keep the pth row as pilot row and go to (iv). Else go to (iii).

(viii) (*Termination*) If all basic columns are decided, terminate the procedure. Else select an undecided basic column A_c.

(ix) (*Clear basis*) Suppose $a_{rc} = 1$. If this is the only positive entry in ${}_rA$, then change the label of the column A_c from "undecided" to "necessary" and go to (viii). Else pivot so as to remove the undecided column A_c from the basis. Go to (iii) with A_c as constant column. (*End of the algorithm.*)

In the absence of degeneracies, this algorithm increases at every step the entry a_{pc} until it becomes nonnegative or remains the only negative entry in the pilot row. In the latter case, the column A_c has been decided. In the former case, a_{pc} stays nonnegative during all subsequent transformations, while some other negative element of A_c is being increased. Hence a decision on A_c will be reached after finitely many simplex steps.

The algorithm is sped up by checking after each iteration whether some of the columns or rows satisfy criteria (3.1), (3.2), (3.3), or (3.4).

A dual algorithm results if the subalgorithm which decides whether A_c is redundant or necessary is replaced by its dual. The primal decision algorithm tries to make a given column of A nonnegative (criterion (3.1)), and encounters criterion (3.3) if this is not possible. The dual decision algorithm aims at criterion (3.3), that is, it attempts to make all entries but one in some row of A nonnegative. If this attempt fails, then criterion (3.2) is encountered. This algorithm is a "phase I" procedure for the dual simplex method.

4. Determining the Lineality Space of $C(S)$

The following criterion is immediate for matrices in canonical form:

(4.1) *If* $A_j \leqq 0$, *then* $A_j \in S_L$.

For any matrix A one has

(4.2) *If* ${}_iA \geqq 0$, *then all columns* A_j *with* $a_{ij} > 0$ *belong to* S_C.

An algorithm analogous to the algorithm (3.5) can be based on these criteria. The pivot rules arise from those of the algorithm (3.5) by reversing the sign of the constant column. Whenever a nonnegative row ${}_iA$ is found, all columns A_j with $a_{ij} > 0$ are deleted.

Thus, ith row of the remaining matrix vanishes and is therefore deleted. Whenever a nonpositive column A_c is found, it is labeled "lineal," and this process is repeated until an empty matrix results or until all columns are labeled "lineal." There are primal and dual strategies available, the former aiming at nonpositive columns, the latter at nonnegative rows.

The algorithms are sped up if the following refinement of criterion (4.1) is used (for matrices in canonical form). (See [13]):

(4.3) *Suppose* $\mathrm{A_c} \leqq 0$, *and let* $\hat{\mathrm{A}}$ *be the matrix that arises from* A *if all rows* ${}_i\mathrm{A}$ *with* $\mathrm{a_{ic}} < 0$ *are deleted. Then* $\hat{\mathrm{A}}_j$ *is in the lineal (conical) part of* $\hat{\mathrm{A}}$ *if and only if* $\mathrm{A_j}$ *is in the lineal (conical) part of* A. *If* $\mathrm{A_c} < 0$, *then* $\mathrm{S_C} = \emptyset$.

PROOF. Not only A_c but also all basic columns A_h such that $a_{ih} = 1$ for $a_{ic} < 0$ are in S_L. These basic columns A_h span the linear subspace $E := \{X | x_i = 0 \text{ if } a_{ic} = 0\}$, which is therefore contained in the lineality space L of $C(S)$. We define $\tilde{A}_j$ by putting

$$\tilde{a}_{ij} := \begin{cases} a_{ij} \text{ if } a_{ic} = 0 \\ 0 \quad \text{if } a_{ic} < 0. \end{cases}$$

Then $\tilde{A}_j - A_j \in E \subseteq L$. Hence $\tilde{A}_j \in L$ if and only if $A_j \in L$.

Denote by $\tilde{S}$ the set of all $\tilde{A}_j$. It clearly suffices to show that $\tilde{A}_j \in \tilde{S}_L$ if and only if $A_j \in S_L$. Suppose $A_j \in S_L$. Then $\tilde{A}_j \in L$ and therefore $-\tilde{A}_j = \Sigma A_i u_i$, $u_i \geqq 0$. This relation remains true if we replace A_i by $\tilde{A}_i$: $-\tilde{A}_j = \Sigma \tilde{A}_i u_i$, whence $\tilde{A}_j \in \tilde{S}_L$. In the other direction, if $-\tilde{A}_j = \Sigma \tilde{A}_i u_i$, $u_i \geqq 0$, then

$$-A_j = \tilde{A}_j - A_j + \Sigma(\tilde{A}_i - A_i)u_i + \Sigma A_i u_i$$

but $\tilde{A}_j - A_j$, $\tilde{A}_i - A_i \in L \subseteq C(S)$. Hence $-A_j \in C(S)$ and, therefore, $A_j \in S_L$.—

We proceed to describe a dual algorithm. The corresponding primal algorithm is readily constructed by the reader. Again it will be convenient to denote by A the particular matrix at hand, and not to change the indices of rows and columns if other rows and columns are deleted.

(4.4) *Dual algorithm for determining the lineality space of* C(S):

(i) (*Canonical form*) Use Jordan elimination for bringing A into canonical form. Label all zero columns of A "lineal."

(ii) (*Termination*) If A is empty or all columns of A are labeled "lineal," then terminate the procedure. Else proceed to (iii).

(iii) (*Objective row*) Select any row ${}_cA$ as "objective row."

(iv) (*Pilot column*) If ${}_cA \geqq 0$, then delete all columns A_j with $a_{cj} > 0$, delete ${}_cA$, and return to (ii). Else select a "pilot column" A_p with $a_{cp} < 0$.

(v) (*Pivot row*) If $A_p \leqq 0$, delete all rows ${}_iA$ with $a_{ip} < 0$, label all zero columns generated by this deletion (all columns, if $A_p < 0$) "lineal," and return to (ii). Else select a "pivot row" ${}_rA$ with $a_{rp} > 0$ and $r \neq c$.

(vi) (*Pivot column*) Select a "pivot column" A_l such that

$$0 \geqq \frac{a_{cl}}{a_{rl}} \geqq \max \left\{ \left| \frac{a_{cj}}{a_{rj}} \right| \; a_{cj} \geqq 0, \quad a_{rj} < 0 \right\}$$

(compare algorithm (3.5) step (v)).

(vii) (*Pivoting*) Pivot on a_{rl}. In the absence of degeneracies, this will increase the entry a_{cp} while keeping nonnegative entries of ${}_cA$ *nonnegative*.

(viii) (*Return*) If the new entry a_{cp} is still negative, keep the pth column as pilot column and go to (v). Else return to (iv). (*End of algorithm*)

In the absence of degeneracies, this algorithm will terminate in a finite number of steps (compare (3.5)). The algorithm is sped up by checking after each iteration whether some of the rows or columns satisfy criteria (4.1) or (4.2).

5. Determining k-Faces of $C(S)$

For finding all k-faces of $C(S)$ we employ a subalgorithm which decides for each set of k linearly independent elements of S whether or not they subdetermine a face. We do not raise the question of the most efficient arrangement of this subalgorithm within the overall search algorithm. Instead, we try to formulate the decision algorithm in a manner that does not preclude its implementation as a subalgorithm. To be more precise, in order to decide whether $A_1, \ldots, A_k$ subdetermine a k-face, it suffices by (1.5) to determine the lineality space of $C(S \cup \{-A_1, \ldots, -A_k\})$. If, however, algorithm (4.4) is used for this purpose, after the columns $-A_1, \ldots, -A_k$ are adjoined to the matrix A, then a matrix results, which is no longer equivalent to A, and each decision must thus start from scratch. This procedure is therefore not suitable for implementation as a subalgorithm.

We proceed to sketch an algorithm which does not require new columns to be adjoined, and which terminates with a matrix that is equivalent to the original one.

A column A_j is *lexico-nonnegative* if $a_{1j} > 0$, or $a_{1j} = 0$ and $a_{2j} > 0$, or $a_{1j} = a_{2j} = 0$ and $a_{3j} > 0$, or . . ., $a_{1j} = \cdots = a_{nj} = 0$. If A_j is lexico-nonnegative and does not vanish, then A_j is *lexico-positive*. We call a matrix *lexico-nonnegative* (*lexico-positive*) if this holds for each column. We then have the following generalization of criterion (4.2):

(5.1) *Suppose* $\hat{\mathrm{A}}$ *arises from* A *by deleting some rows. If* $\hat{\mathrm{A}}$ *is lexico-nonnegative and* $\hat{\mathrm{A}}_j \neq 0$, *then* $\mathrm{A_j} \in \mathrm{S_C}$.

PROOF. Let B be the row that results from combining the rows of $\hat{A}$ with weights $\epsilon, \epsilon^2, \ldots$. For suitable $\epsilon > 0$ we have $B \geq 0$ with $b_j > 0$ if and only if $\hat{A}_j \neq 0$. Then criterion (4.2) applies to the matrix $\begin{pmatrix} A \\ B \end{pmatrix}$, which is equivalent to A.

We proceed to formulate two criteria on which an algorithm for finding the k-faces of $C(S)$ can be based. Suppose $A_1, \ldots, A_k$ are basic columns. We call the submatrix A of A formed by all those rows in which $A_1, \ldots, A_k$ simultaneously vanish the *complement* of $A_1, \ldots, A_k$. The complement of the entire basis of A is the empty matrix. Schematically, we have

$$A = \left\{ \begin{array}{|c|c|c|} \hline I^{(k)} & 0 & \\ \hline 0 & I^{(m-k)} & \\ \hline \end{array} \right\} = \text{complement}$$

Criterion (5.1) then leads to (see [13]):

(5.2) *If the complement of the basic columns* $A_1, \ldots, A_k$ *is lexico-nonnegative, then* $A_1, \ldots, A_k$ *subdetermine a* k*-face.*

PROOF. It follows from (5.1) that $A_1, \ldots, A_k$, $-A_1, \ldots, -A_k$ and those elements of S which are linear combinations of the former are the only elements in the lineal part of $S \cup \{-A_1, \ldots, -A_k\}$, and (1.5) applies. —

(5.3) *The basic columns* $A_1, \ldots, A_k$ *do not subdetermine a face if their complement contains a nonpositive nonzero column.*

PROOF. Assume for simplicity that $A_{k+1}, \ldots, A_m$ are the remaining basic columns, and that $a_{ii} = 1$ for $i \leq m$. The complement of $A_1, \ldots, A_k$ then consists of the rows ${}_iA$ with $i > k$. Let now A_c, $c > m$, be a column whose portion in the complement is nonpositive and nonzero. Then

$$A_c = \sum_{i=1}^{k} A_i u_i + \sum_{i=k+1}^{m} A_i w_i,$$

where $w_i = a_{ic} \leq 0$, and, therefore, $A_c \in C((-S) \cup \{A_1, \ldots, A_k\})$. Hence A_c belongs to the lineality space of $C(S \cup \{-A_1, \ldots, -A_k\})$. Now if $A_1, \ldots, A_k$ subdetermine a face, then $L\{A_1, \ldots, A_k\}$ is the lineality space of $C(S \cup \{-A_1, \ldots, -A_k\})$ by (1.5). Thus $A_c \in L\{A_1, \ldots, A_k\}$, which contradicts the fact that not all w_i vanish in the above unique representation of A_c in terms of the basis. —

It is easy to see that algorithm (4.4) and its corresponding primal algorithm can be modified to yield an algorithm whose terminal situations are (5.2) and (5.3). No columns and no nonnegative rows are deleted. A hierarchy of nonegative rows leads eventually to the nonnegative submatrix required by (5.2). The pivot columns vanish in these rows. Hence pivoting does not spoil the nonnegativities already achieved.

6. The 1-Skeleton of $H(\hat{S})$

The problem of finding the vertices of a convex hull $H(\hat{S})$ can be reduced to finding the frame of the cone $C(S)$ where $S := \{A_i = \begin{pmatrix} \hat{A}_i \\ 1 \end{pmatrix} | \hat{A}_i \in \hat{S}\}$. Similarly, the edges of $H(\hat{S})$ correspond to the 2-faces of $C(S)$. There are, however, special features of the problem of finding the vertices and edges of $H(\hat{S})$. First,

(6.1) *If* $\bar{\mathrm{A}}$ *is equivalent to* A, *then* $\bar{\mathrm{A}}_j \in \mathrm{C}(\bar{\mathrm{S}} - \{\bar{\mathrm{A}}_j\})$ *if and only if* $\bar{\mathrm{A}}_j \in \mathrm{H}(\bar{\mathrm{S}} - \{\bar{\mathrm{A}}_j\})$.

PROOF. If $\bar{A}_j = \sum_{i \neq j} \bar{A}_i u_i$, then by (2.3) $A_j = \sum_{i \neq j} A_i u_i$, which gives $\Sigma u_i = 1$ by considering the last component.

As a consequence of (6.1), a criterion for necessity can be based on the relative magnitudes rather than the signs of the elements of A (Goldstein [7]):

(6.2) *If* $\bar{\mathrm{A}}$ *is equivalent to* A, *and if* $\bar{\mathrm{a}}_{ij}$ *is the unique maximum or minimum of the entries in a row* ${}_i\bar{\mathrm{A}}$ *of* $\bar{\mathrm{A}}$, *then* $\hat{\mathrm{A}}_j$ *is a vertex of* $\mathrm{H}(\hat{\mathrm{S}})$.

Indeed, $\bar{A}_j$ is a vertex of $H(\bar{S})$, and by (6.1) necessary in $\bar{S}$. Criterion (6.2) is valuable in practice since it enables one in general to find some vertices right away as well as during the algorithm. It can be generalized to the case in which the maximum or minimum is not unique. Suppose the maximum of the first row is assumed for the subset $M \subseteq S$. Then a column $A_i \in M$ is necessary in S if it is necessary in M. Hence the examination can be temporarily restricted to M. In particular, criterion (6.2) can be applied to M. For instance, if a_{2s} is the unique maximum or minimum of the entries a_{2j} with $A_j \in M$, then A_s is necessary in S.

Suppose that $\bar{A}$ is in canonical form and equivalent to A. Then the same argument that established (6.2) gives

$$\sum_{i=1}^{m} \bar{a}_{ij} = 1$$

for all j. Thus each column has at least one positive entry. This follows also from the fact that $C(S)$ is pointed i.e., has lineality dimension = 0 (4.2) and that S contains no zero columns.

Suppose S is a frame. Then each nonbasic column has at least one negative and two positive entries by (3.1) and (3.2). If a column has precisely two positive entries, then by (5.3) the corresponding basic columns do not determine an edge.

7. References

[1] Balinsky, M., An algorithm for finding all vertices of convex polyhedral sets, J. Soc. Indust. Appl. Math., **9**, 72–88 (1961).

[2] Charnes, A., Optimality and degeneracy in linear programming, Econometrica **20,** 160–170 (1952).

[3] Dantzig, G. B., Linear Programming and Extensions (Princeton University Press, Princeton, 1963).

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

[5] Fenchel, W., Convex cones, sets and functions, Lecture Notes, Department of Mathematics, Princeton University, Princeton (1953).

[6] Gale, D., On the number of faces of a convex polytope, Can. J. Math. **16,** 12–17 (1964).

[7] Goldstein, A. J., A procedure for determining the convex hull of a set of points of hyperplanes, Comm. ACM, **6,** 356 (1963). Abstract of a paper at 18th Annual ACM Conference 1963.

[8] Gomory, R. E., and H. D. Mills, unpublished manuscript, Princeton (1958).

[9] Klee, V., On the number of vertices of a convex polytope, Can. J. Math. **16,** 701–720 (1964).

[10] Kuhn, H. W., Solvability and consistency for linear equations and inequalities, Amer. Math. Monthly **63,** 217–232 (1956).

[11] Motzkin, T. S., Raiffa, H., Thompson, G., and Thrall, R., The double description method in Contributions to the Theory of Games II, 51–73, Ann. Math. Studies, No. 28 (Princeton University Press, 1953).

[12] Thompson, G. L., Tonge, F. M., and Zionts, S., Techniques for removing nonbinding constraints and extraneous variables from linear programming problems, Management Science **12,** 588–608 (1965).

[13] Wets, R. J.-B. and Witzgall, C., Towards an algebraic characterization of convex polyhedral cones, submitted to a technical Journal.

[14] Farrell, M. J., and M. Fieldhouse, Estimating efficient production functions under increasing returns to scale, J. Roy. Statist. Soc., Ser. A, **125** (II), 252–267 (1962).

(Paper 71B1–189)