

A Degenerate Extreme Point Strategy for the Classification of Linear Constraints as Redundant or Necessary¹

R. J. CARON,² J. F. McDONALD,³ AND C. M. PONIC⁴

Communicated by D. F. Shanno

Abstract. This paper presents a degenerate extreme point strategy for active set algorithms which classify linear constraints as either redundant or necessary. The strategy makes use of an efficient method for classifying constraints active at degenerate extreme points. Numerical results indicate that significant savings in the computational effort required to classify the constraints can be achieved.

Key Words. Redundancy, degeneracy, linear constraints, linear programming.

1. Introduction

This paper presents a degenerate extreme point strategy for active set algorithms which classify linear constraints as either redundant or necessary. The importance of being able to detect redundant constraints has been well established in Ref. 1. Not only does the elimination of all redundant constraints reduce the computational effort required to solve an associated mathematical programming problem, but it also provides insight into the mathematical model represented by the inequalities. Reference 1 also provides an extensive survey of the literature on redundancy. A brief summary follows.

¹ This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grants A8807 and A4625 and by an Undergraduate Summer Research Award.

² Associate Professor, Department of Mathematics and Statistics, University of Windsor, Windsor, Canada.

³ Professor, Department of Mathematics and Statistics, University of Windsor, Windsor, Canada.

⁴ Research Assistant, Department of Mathematics and Statistics, University of Windsor, Windsor, Canada.

The first paper devoted entirely to redundancy was given by Boot (Ref. 2). Later, Zionts [Ref. 3; see also the closely related paper by Thomson *et al.* (Ref. 4)] improved on the implementation of Boot's method by adding tests which could immediately identify redundant constraints. Gal (Ref. 5) and Telgen (Ref. 6) continued the approach by adding rules for immediately identifying necessary constraints and rules for dealing with degeneracy. The algorithm presented in this paper can be categorized with the algorithms of Rubin (Ref. 7), Telgen (Ref. 6), Gal (Ref. 5), and Zionts and Wallenius (Ref. 8). The algorithms are equivalent in that: they classify all constraints as redundant or necessary; they produce results that are unconditionally correct; they perform iterations of an active set linear programming algorithm [for example, the simplex method (Ref. 9) or the Best-Ritter method (Ref. 10)]; and they use tests which may immediately classify constraints.

The algorithm presented below is an extension of the above algorithms in that it implements a degenerate extreme point strategy (DEPS). The strategy is to immediately eliminate redundancy at degenerate extreme points that are encountered during the classification algorithm.

As noted in Ref. 1, redundant constraints may cause degeneracy, which in turn may cause near-cycling (Ref. 4). The intent of the strategy is to eliminate near-cycling during the classification algorithm. This could result in significant computational savings, since a given extreme point may be encountered more than once during constraint classification.

The strategy makes use of a new procedure for removing redundancy at extreme points. This procedure, which will be referred to as DEPS, results in further computational savings, since it considers simple regions (they have at most one extreme point), defined only by the constraints that are active. Indeed, numerical results show that the strategy, implemented with DEPS, can produce significant savings in the computational effort required to classify the constraints.

2. Problem Formulation, Definitions, and Notation

This paper is concerned with the set of linear constraints that represent the region R , where

$$R = \{x \in R^n \mid a_i^T x \leq b_i, i \in I\},$$

and where $I = \{1, \dots, m\}$.⁵ The region represented by all but the k th

⁵ Equality constraints are omitted from the discussion, since they are handled in a straightforward manner. See Ref. 10, pp. 281–288.

constraint is given by

$$R_k = \{x \in R^n \mid a_i^T x \leq b_i, i \in I/k\},$$

where I/k is the set I with element k removed. Consistent with the definitions given in Ref. 1, constraint k is redundant if and only if $R = R_k$ and is necessary otherwise. If $x^* \in R$, the set of indices of all constraints active at x^* will be denoted by $A(x^*)$. The paper also makes use of the regions given by

$$R(x^*) = \{x \in R^n \mid a_i^T x \leq b_i, i \in A(x^*)\},$$

$$R_k(x^*) = \{x \in R^n \mid a_i^T x \leq b_i, i \in A(x^*)/k\}.$$

It is easily shown that, if $k \in A(x^*)$, then constraint k is redundant if and only if $R(x^*) = R_k(x^*)$.

In the algorithm given below, any constraint which is classified as redundant is immediately removed from the set of constraints defining R , yielding a new representation of R . Thus, the final representation of R (i.e., the representation which contains only necessary constraints) depends on the order in which the constraints are classified. If there are no implicit equalities in the initial representation of R , the final representation is minimal (Ref. 6).

3. Theorems for the Classification of Constraints

The algorithms in Refs. 2, 6, 7, 8, as well as the one presented in this paper are based primarily on the following theorem.

Theorem 3.1. The k th inequality is redundant if and only if problem LP_k has an optimal solution x^* with $a_k^T x^* \leq b_k$, where LP_k is given by

$$\begin{aligned} LP_k: \quad & \text{maximize } a_k^T x, \\ & \text{subject to } x \in R_k. \end{aligned}$$

Proof. See Ref. 1 or 11. □

Theorem 3.1 suggests the following naive classification algorithm which will be referred to as Strategy 1. Solve LP_k , for each $k \in I$, immediately removing from I any constraints found to be redundant. While LP_k is being solved, it is called the controlling LP.

This naive algorithm, and subsequent algorithms, are designed so that all extreme points encountered in the solution of the controlling LP are in R . If a step was to be made outside R , then $R \neq R_k$. In that case, constraint k is classified as necessary and the controlling LP is changed.

The following corollaries improve on Strategy 1 by providing mechanism for immediately classifying other constraints, while LP_k is the controlling LP. The modification of Strategy 1 which makes use of the following corollaries will be referred to below as Strategy 2.

Corollary 3.1. Let $x \in R$. If the gradients of the constraints with indices in the set $A(x)$ are linearly independent, then all such constraints are necessary.

Proof. See Ref. 1 or 11. □

Corollary 3.2. Let $x \in R$, and consider the system of equations

$$\sum_{i \in A(x) \setminus t} u_i a_i = a_t. \quad (1)$$

(a) If there exists a solution to (1) such that $u_i \geq 0$, for all $i \in A(x)/t$, then constraint t is redundant.

(b) If $t \in A(x)$ and if there exists a solution to (1) such that, for some $\tau \in A(x) \setminus t$, $u_\tau > 0$ and $u_i \leq 0$, for all $i \in A(x) \setminus \{t, \tau\}$, then constraint τ is redundant.

Proof. See Ref. 1 or 11. □

Corollary 3.3. Let $x \in R$, and let s be any nonzero vector. For each $i \in I$, define

$$\begin{aligned} \sigma_i &= +\infty, & \text{if } a_i^T s \leq 0, \\ \sigma_i &= (b_i - a_i^T x) / a_i^T s, & \text{otherwise.} \end{aligned}$$

Set

$$\sigma = \min\{\sigma_i \mid i \in I\}.$$

If σ is defined by a unique index r , then constraint r is necessary.

Proof. See Ref. 11 or 12. □

Note that Corollary 3.2(a) is a statement of the optimality conditions for LP_t , while Corollary 3.2(b) is a rearrangement of the terms in Eq. (1). Corollary 3.3 implies that constraint r is necessary if it is the unique constraint defining the boundary of R that intersects the half-line $x + \sigma s$, $\sigma \geq 0$.

4. Classification at a Degenerate Extreme Point

The procedure DEPS for removing redundancy at extreme points is based primarily on the following theorem and corollary.

Theorem 4.1. Let $x^* \in R$. Constraint $\kappa \in A(x^*)$ is redundant if and only if x^* is an optimal solution to the following problem:

$$\text{LP}_\kappa(x^*): \quad \text{maximize } a_\kappa^T x, \\ \text{subject to } x \in R_\kappa(x^*).$$

Proof. Constraint $\kappa \in A(x^*)$ is redundant if and only if $R(x^*) = R_\kappa(x^*)$. By definition, $R(x^*) = R_\kappa(x^*)$ if and only if there is no solution x to

$$\begin{aligned} a_\kappa^T x &> b_\kappa, \\ a_i^T x &\leq b_i, \quad \text{for all } i \in A(x^*) \setminus \kappa. \end{aligned}$$

Since $\kappa \in A(x^*)$, this is equivalent to the system

$$\begin{aligned} a_\kappa^T (x - x^*) &> 0, \\ a_i^T (x - x^*) &\leq 0, \quad \text{for all } i \in A(x^*) \setminus \kappa. \end{aligned}$$

It then follows from Farkas' theorem (Ref. 13) that the system has no solution if and only if x^* is an optimal solution to $\text{LP}_\kappa(x^*)$. Thus, constraint $\kappa \in A(x^*)$ is redundant if and only if x^* is an optimal solution to $\text{LP}_\kappa(x^*)$. \square

It should be noted that Theorem 4.1 is essentially Theorem 3.1 applied to the smaller and simpler $\text{LP}_\kappa(x^*)$. The LP is smaller, since the number of constraints is reduced from m to $|A(x^*)|$, where $|\cdot|$ denotes set cardinality. It is simpler in that it has at most one extreme point. If it does have an extreme point, then x^* is the only extreme point and either x^* is optimal or the LP is unbounded from above.

The following corollary is analogous to Corollary 3.1.

Corollary 4.1. Let $x^* \in R$ be an extreme point with $|A(x^*)| = n + 1$. Let $t \in A(x^*)$, and consider the equation

$$\sum_{i \in A(x^*) \setminus t} u_i a_i = a_t. \quad (2)$$

(a) If the solution to (2) is such that $u_i \geq 0$, for all $i \in A(x^*) \setminus t$, then constraint t is redundant and all constraints $i \in A(x^*) \setminus t$ are necessary.

(b) If the solution to (2) is such that, for some $\tau \in A(x^*) \setminus t$, $u_\tau > 0$ and $u_i \leq 0$, for all $i \in A(x^*) \setminus \{t, \tau\}$, then constraint τ is redundant and constraints $i \in A(x^*) \setminus \tau$ are necessary.

(c) If neither the hypotheses in (a) nor those in (b) are satisfied, then all constraints $i \in A(x^*)$ are necessary.

Proof. Parts (a) and (b) follow from Corollaries 3.1 and 3.2 and the fact that x^* is an extreme point of R . Part (c) follows by contradiction; i.e.,

if there is a redundant constraint, then the hypotheses of either (a) or (b) are satisfied. \square

Note that, if $|A(x^*)| = n + 1$, then Corollary 4.1 implies that all constraints with indices in $A(x^*)$ can be classified by the solution of a single system of linear equations. The modification of Strategy 2 which includes DEPS will be referred to as Strategy 3.

5. Classification Algorithm

The flow chart in Fig. 1 gives a general description of the algorithm. See Ref. 11 for a detailed description. The algorithm is initialized (Box A) with the index k of a controlling LP and with an extreme point x of R_k (and of R). The algorithm moves from extreme point to extreme point of R until LP_k is solved or constraint k is classified.

Box B checks for degeneracy. If x is nondegenerate, then control passes to Box E. If x is degenerate, then control passes to subroutine DEPS (Box C) and returns with all constraints in $A(x)$ classified. Subroutine DEPS is essentially the Strategy 2 algorithm applied to the smaller set of constraints $A(x)$, and the controlling LP's are the smaller and simpler $LP_k(x)$'s. If constraint k was classified (Box D), then either a new controlling LP is determined or the algorithm terminates with all constraints classified (Box F). If the algorithm did not terminate, then control passes to Box E.

Box E tests the optimality of x for LP_k . If x is optimal, constraint k is classified as redundant and control passes to Box F. Otherwise, control passes to Box H, where an ascent direction and stepsize for LP_k is determined. If the stepsize is such that the next extreme point would be in R_k but not in R , constraint k is classified as necessary (Box J) and control returns to Box F. Otherwise, x is tested for optimality with respect to other LP_i 's in an attempt to immediately classify constraints as redundant (Box I). Finally, an update is made to a new extreme point of R_k (Box G) and control returns to Box B.

In the worst case, the algorithm will solve m linear programming problems. Since each LP can be solved in a finite number of steps (using Bland's rules if necessary, Ref. 14), it follows that the classification algorithm has finite termination.

There are two points about the algorithm which should be noted. Let $J(x)$,⁶ a subset of $A(x)$, denote the working set; and let $D(x)$ be a non-singular matrix whose columns contain the gradients of all constraints with indices in $J(x)$. The set $J(x)$ and the matrix $D(x)$ are equivalent to a choice

⁶ If x is not an extreme point, then $J(x)$ will contain less than n elements.

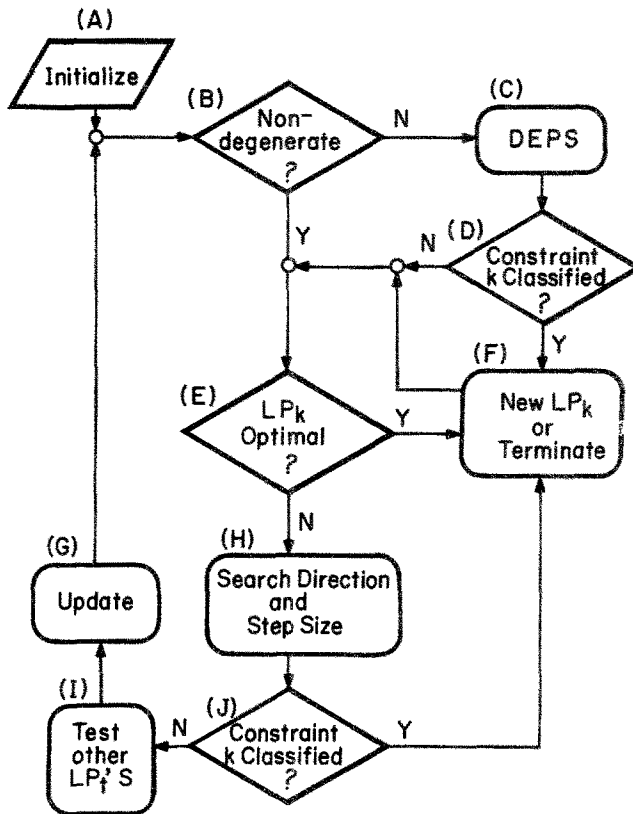


Fig. 1. General description of the classification algorithm.

of basis and basis matrix in the simplex method (Ref. 10). Whenever an index is to be added to $J(x)$, an attempt is made to add an index from $A(x)$ which corresponds to a necessary constraint. This tends to reduce the number of pivots. Also, $J(x)$ must be determined such that the update results in a nonsingular matrix $D(x)$.

The second point concerns Corollary 3.2, which requires the solution of a system of linear equations for each constraint tested. The system is not solved if it can be determined *a priori* that its solution will not result in a classified constraint. The algorithm uses the iteration formula $x \leftarrow x + \sigma s$, where s is determined by solving, for an appropriate p , $D(x)s = -e_p$ and where e_p denotes the p th column of the identity matrix. If $a_t^T s > 0$ (this inner product is available from the stepsize procedure) and if $\sigma > 0$, it follows easily that constraint t will not be classified. Also, if x is degenerate,

no attempt is made to classify the inactive constraints using Corollary 3.2 until subroutine DEPS has classified all active constraints. This avoids performing tests that fail due to the choice of the working set.

6. Example

Let W be the set of indices of all unclassified constraints. The algorithm starts with $W = I$ and stops when W is empty. During the course of the algorithm, redundant constraints are removed from both W and I , while necessary constraints are removed only from W . Upon termination, all constraints with indices in the set I are necessary. Consider the set of constraints given in Table 1.

The constraints are a representation of the shaded region in Fig. 2. The unique minimal representation is given by the necessary constraints 2, 3, 8, and 9.

The algorithm proceeds as follows. Set $W = I = \{1, \dots, 10\}$. The controlling LP is LP_1 . Let $x_0 = (0, 0)^T$ so that $A(x_0) = \{2, 4, 5, 7, 8\}$. Set $J(x_0) = \{5, 4\}$ and $D(x_0) = [a_5, a_4]$.

Since x_0 is degenerate, enter subroutine DEPS. While inside DEPS, the controlling LP is $LP_2(x_0)$. Solve $D(x_0)v = a_2$ to get $v = (-3, 2)^T$. Corollary 3.2(b) implies that constraint 4 is redundant. Set $J(x_0) = \{5, 7\}$ and $D(x_0) = [a_5, a_7]$. Solve $D(x_0)v = a_2$ to get $v = (-1, 2)^T$. Corollary 3.2(b) then implies that constraint 7 is redundant. Set $J(x_0) = \{5, 8\}$ and $D(x_0) = [a_5, a_8]$. Solve $D(x_0)v = a_2$ to get $v = (1, -2)^T$. Since $|A(x_0)| = |\{2, 5, 8\}| = 3$, Corollary 4.1 implies that constraint 5 is redundant and constraints 2 and 8 are necessary. Thus, $W = \{1, 3, 6, 9, 10\}$, $I = \{1, 2, 3, 6, 8, 9, 10\}$, and $J(x_0) = \{2, 8\}$. Since all constraints in $A(x_0)$ have been classified, exit DEPS.

Table 1. Constraints of the example.

Constraint	Constraint number
$x_1 - x_2 \leq 8$	1
$-x_1 + x_2 \leq 0$	2
$x_1 + x_2 \leq 12$	3
$-2x_1 - x_2 \leq 0$	4
$-x_1 - x_2 \leq 0$	5
$-x_1 - x_2 \leq 4$	6
$-x_1 \leq 0$	7
$-x_2 \leq 0$	8
$x_1 \leq 8$	9
$x_2 \leq 8$	10

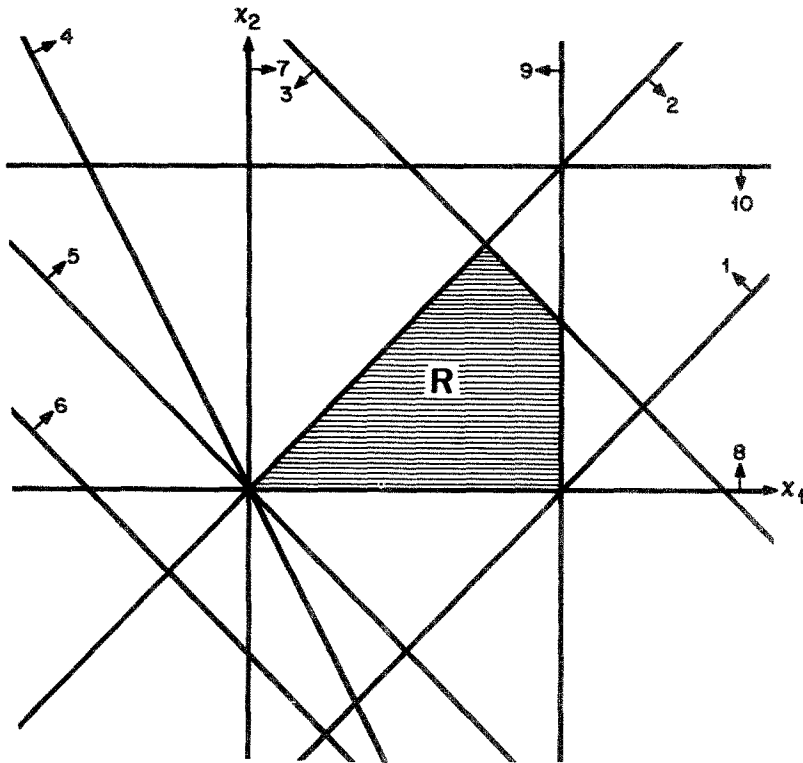


Fig. 2. Example showing redundant and necessary constraints.

The controlling LP is still LP_1 . Solve $D(x_0)v = a_1$ to get $v = (-1, 0)^T$. Thus, x_0 is not optimal for LP_1 . Solve $D(x_0)s = (-1, 0)^T$ to get the search direction $s = (1, 0)^T$ and subsequently the stepsize $\sigma = 8$.

Since $a_i^T s \leq 0$, for $i \in \{3, 9, 10\}$, and since $\sigma > 0$, Corollary 3.2(a) is used only to test the optimality of x_0 for LP_6 . In that case, $v = (1, 2)^T$, so that constraint 6 is redundant.

The update gives $x_1 = (8, 0)^T$, $J(x_1) = \{9, 8\}$, $D(x_1) = [a_9, a_8]$, and $A(x_1) = \{1, 8, 9\}$. Also, $W = \{1, 3, 9, 10\}$ and $I = \{1, 2, 3, 8, 9, 10\}$.

Since x_1 is degenerate, enter subroutine DEPS. Since $|A(x_1)| = 3$, Corollary 4.1 is used. Solve $D(x_1)v = a_1$ to get $v = (1, 1)$. Constraint 1 is classified as redundant and constraint 9 is classified as necessary. Thus, $W = \{3, 10\}$ and $I = \{2, 3, 8, 9, 10\}$. Since all constraints in $A(x_1)$ have been classified, exit DEPS.

Since constraint 1 has been classified in DEPS, the controlling LP is changed to LP_3 . Solving $D(x_1)v = a_3$ gives $v = (1, -1)^T$, so that x_1 is not optimal. Solving $D(x_1)s = -e_2$ yields the search direction $s = (0, 1)^T$ and a

stepsize of 8. The stepsize results in a point violating constraint 3, which implies that $R \neq R_3$, so that constraint 3 is necessary. Thus, $W = \{10\}$, and the controlling LP is now LP_{10} . The algorithm proceeds to $x_2 = (8, 4)^T$ and $x_3 = (6, 6)^T$, which is optimal for LP_{10} . Constraint 10 is then classified as redundant. Since W is now empty, stop with the set of necessary constraints $I = \{2, 3, 8, 9\}$.

7. Implementation

The classification algorithm has been implemented by the double-precision FORTRAN subroutine CLASFY (Ref. 15). The implementation actually includes a modification of the algorithm as given above. A vector is used to store the maximum value of $a_i^T x$, $i \in W$, that has been attained

Table 2. Description and source of test problems.

Example	n	m	r	R	% R	Source
1	5	7	0	0	0	Ref. 1, p. 32
2	2	7	0	4	57	Ref. 1, p. 33
3	4	11	0	3	27	Ref. 1, p. 46
4	2	8	0	2	25	Ref. 1, p. 57
5	2	7	0	2	28	Ref. 1, p. 64
6	2	6	0	2	33	Ref. 1, p. 76
7	5	10	0	2	20	Ref. 1, p. 85
8	2	10	0	6	60	Section 6
9	2	11	0	8	72	*
10	3	23	0	15	65	*
11	2	23	0	16	69	*
12	11	20	8	7	24	Ref. 10, pp. 227-233
13	44	66	12	0	0	*
14	20	29	0	0	0	Ref. 9, pp. 553-555
15	15	20	0	0	0	Ref. 16, p. 110
16	26	130	18	42	28	Ref. 17, p. 594
17	26	130	18	33	22	Ref. 17, p. 594
18	26	130	18	35	23	Ref. 17, p. 594
19	26	130	18	34	23	Ref. 17, p. 594
20	26	130	18	34	23	Ref. 17, p. 594
21	26	130	18	34	23	Ref. 17, p. 594
22	26	130	18	34	23	Ref. 17, p. 594
23	26	130	18	34	23	Ref. 17, p. 594
24	26	130	18	34	23	Ref. 17, p. 594
25	3	8388	0	8335	99	Ref. 18

* Available from the authors.

Table 3. Inner products required for test problems.

Example	Strategy 1	Strategy 2	Strategy 3
1	94	39	39
2	87	85	85
3	370	272	259
4	181	133	125
5	154	155	151
6	87	72	67
7	281	221	220
8	156	136	124
9	159	109	93
10	1,371	600	546
11	1,401	364	305
12	1,641	1,143	1,297
13	21,051	14,574	15,874
14	3,570	2,644	2,476
15	1,097	1,183	1,183
16	189,035	192,411	95,020
17	261,854	210,002	111,986
18	237,255	200,368	109,475
19	243,311	212,597	111,899
20	241,583	206,924	109,393
21	248,739	201,274	106,320
22	248,963	210,188	108,568
23	247,633	190,536	106,471
24	246,932	208,399	113,664
25	*	968,279	587,717

* Not available.

at previously encountered extreme points. If the value of $a_i^T x$ at the current point is less than or equal to the maximum, then LP_i is not tested for optimality.

The subroutine was tested with the 25 examples presented in Table 2. The columns give the number of variables n , inequality constraints m , equality constraints r , redundant constraints R , and percentage of the constraints which are redundant, respectively. Each example was solved using Strategies 1, 2, and 3. The strategies were compared by the number of inner products required. Comparing the number of iterations is misleading, since the computational effort required for each iteration is different for each of the strategies. Execution time was proportional to the number of inner products required. The results are presented in Table 3.

Strategy 3 required the fewest inner products in almost all of the examples. Although Strategy 3 did not outperform the others in Examples

12, 13, and 15, it required only about 10% more inner products. In the larger examples 16 to 25, Strategy 3 required about 50% fewer inner products.

In general, it is expected that Strategy 3 will be no worse than Strategy 2, but will be significantly better in examples with degeneracy.

8. Conclusions

The algorithm presented in this paper improves on previous algorithms by the addition of DEPS, a subroutine which efficiently classifies constraints active at degenerate extreme points. In addition, it implements a degenerate extreme point strategy. The strategy is to immediately classify the active constraints whenever a degenerate extreme point is encountered. The idea is to try to eliminate degeneracy by eliminating redundancy. Numerical results indicate that DEPS can produce significant savings in the computational effort required to classify constraints in degenerate systems. When there is no degeneracy, the algorithm is equivalent to previous methods.

It has been demonstrated (Ref. 1) that the most promising approach to classifying linear constraints involves a hybrid of a random method (Ref. 12) with a deterministic method. The use of the Strategy 3 algorithm with a random method is currently under development.

References

1. KARWAN, M. H., LOTFI, V., TELGEN, J., and ZIONTS, S., Editors, *Redundancy in Mathematical Programming: A State of the Art Survey*, Springer-Verlag, Berlin, Germany, 1983.
2. BOOT, J. C. G., *On Trivial and Binding Constraints in Programming Problems*, Management Science, Vol. 8, pp. 419-441, 1962.
3. ZIONTS, S., *Size Reduction Techniques of Linear Programming and Their Applications*, Carnegie Institute of Technology, PhD Thesis, 1965.
4. THOMPSON, G. L., TONGE, F. M., and ZIONTS, S., *Techniques for Removing Nonbinding Constraints and Extraneous Variables from Linear Programming Problems*, Management Science, Vol. 12, pp. 588-608, 1966.
5. GAL, T., *A Method for Determining Redundant Constraints*, Redundancy in Mathematical Programming: A State of the Art Survey, Edited by M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, Springer-Verlag, Berlin, Germany, pp. 36-52, 1983.
6. TELGEN, J., *Identifying Redundancy in Systems of Linear Constraints*, Redundancy in Mathematical Programming: A State of the Art Survey, Edited by M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, Springer-Verlag, Berlin, Germany, pp. 53-59, 1983.

7. RUBIN, D. S., *Finding Redundant Constraints in Sets of Linear Inequalities*, Redundancy in Mathematical Programming: A State of the Art Survey, Edited by M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, Springer-Verlag, Berlin, Germany, pp. 60-67, 1983.
8. ZIONTS, S., and WALLENIS, J., *A Method for Identifying Redundant Constraints and Extraneous Variables in Linear Programming*, Redundancy in Mathematical Programming: A State of the Art Survey, Edited by M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, Springer-Verlag, Berlin, Germany, pp. 28-35, 1983.
9. DANTZIG, G. B., *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.
10. BEST, M. J., and RITTER, K., *Linear Programming: Active Set Analysis and Computer Programs*, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.
11. CARON, R. J., McDONALD, J. F., and PONIC, C. M., *Classification of Linear Constraints as Redundant or Necessary*, University of Windsor, Windsor Mathematics Report No. WMR-85-09, 1985.
12. BONEH, A., *PREDUCE: A Probabilistic Algorithm for Identifying Redundancy by a Random Feasible Point Generator*, Redundancy in Mathematical Programming: A State of the Art Survey, Edited by M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, Springer-Verlag, Berlin, Germany, pp. 108-134, 1983.
13. MANGASARIAN, O. L., *Nonlinear Programming*, McGraw-Hill, New York, New York, 1969.
14. BLAND, R. G., *New Finite Pivoting Rules for the Simplex Method*, Mathematics of Operations Research, Vol. 2, pp. 103-107, 1977.
15. CARON, R. J., McDONALD, J. F., and PONIC, C. M., *CLASFY: A Fortran IV Subroutine to Classify Linear Constraints—Users' Guide*, University of Windsor, Windsor Mathematics Report No. WMR-85-08, 1985.
16. KOTIAH, T. C. T., and STEINBERG, D. I., *Occurrences of Cycling and Other Phenomena in a Class of Linear Problems*, Communications of the Association of Computing Machinery, Vol. 20, pp. 107-112, 1977.
17. GRIERSON, D. E., and ALY, A. A., *Plastic Design under Combined Stresses*, Journal of the Engineering Mechanics Division, ASCE, Vol. 106, pp. 585-607, 1980.
18. KEERTHI, S. S., and GILBERT, E. G., *Computation of Minimum Time Feedback Control Laws for Discrete-Time Systems with State-Control Constraints*, Paper Presented at the Conference on Information Sciences and Systems, Princeton University, Princeton, New Jersey, 1986.