INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

University Microfilms International
300 N. Zeeb Road
Ann Arbor, MI 48106

OKO, SELINA OMAGHA

SURROGATE METHODS FOR LINEAR INEQUALITIES AND LINEAR PROGRAMMING PROBLEMS

City University of New York PH.D. 1983

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __✓__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Other Dissertation contains pages with print at a slant, filmed as received.

SURROGATE METHODS FOR LINEAR INEQUALITIES AND LINEAR PROGRAMMING PROBLEMS

By

Selina Omagha Oko

A dissertation submitted to the
Graduate Faculty in Computer Science,
School of Engineering, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy,
The City University of New York.

1983

This manuscript has been read and accepted for the Graduate Faculty in Computer Science, School of Engineering, in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/19/83 — Donald Goldfarb
date — Chairman of Examining Committee

9/19/83 — Paul P. Kanwal
date — Executive Officer

Prof. M. Anshel
Prof. S. Burr
Prof. D. Goldfarb
Prof. F. Pickel
Prof. P. Sterbenz
Supervisory Committee

The City University of New York

DEDICATED TO

G. U. Oko, my father, in memoriam

L. Omagha Oko, my mother

and

all my sisters and brothers.

Abstract

SURROGATE METHODS FOR LINEAR INEQUALITIES

AND

LINEAR PROGRAMMING PROBLEMS

by

Selina Omagha Oko

Adviser: Professor Donald Goldfarb

We consider the problem of finding a point which satisfies a given system of linear inequalities. We present 4 algorithms for solving this problem. All are iterative schemes, and are based upon the orthogonal projection of an infeasible point onto the manifold of the bounding hyperplanes of some of the given constraints. The choice of the constraints and the actual projection are accomplished through the use of 'surrogate' constraints or hyperplanes. Proof of convergence for one of the algorithms that closely resembles a method due to Agmon is given. A general proof is done by means of a monotonic decreasing sequence of continuous functions of the iterates. The algorithms are all adapted to handle equations without replacing them with pairs of inequalities and they can also be used to solve linear programming problems. Three of the algorithms are faster than Agmon's and all 4 can detect inconsistency. The results of computational tests carried out on a variety of problems are reported. Implementations for large sparse matrices are also given.

ACKNOWLEDGEMENTS

My gratitude goes to my supervisor, Prof. D. Goldfarb. He did not only give me the topic, but also made available the relevant references as he directed me in the right course. I also wish to thank the other members of my thesis committee, Prof. M. Anshel, Prof. S. Burr, Prof. F. Pickel and Prof. P. Sterbenz. Their hint, during the proposal, on a continuous function approach for proof of convergence was very helpful.

S. Omagha Oko

NOTATION

$\mathcal{A}$	m x n constraint matrix of the given problem.
$\mathcal{B}$	m-vector, right hand side of the given problem.
α^i	the ith row of $\mathcal{A}$.
$\mathcal{B}_i$	the ith element of $\mathcal{B}$.
a^i	the outer normal of the ith constraint.
A	m x n matrix whose ith row is a^i.
A^T	transpose of A.
b_i	the right hand side associated with a^i of constraint i.
b	m-vector with b_i as the ith element.
a_j	the jth column of A.
a_{ij}	the jth element of a^i.
C	AA^T.
I	the index set for the constraints.
x	the n-vector to be found.
$x^{(k)}$	the kth iterate of x.
$x_j^{(k)}$	the jth element of $x^{(k)}$.
$e^{(k)}$	error in $x^{(k)}$.
e	m-vector of ones.
e_j	n-vector of zeros everywhere and 1 at the jth position.
$R^{m \times n}$	the set of real m x n matrices.
R^n	the set of real n-vectors.
S	the solution set.
∂S	the boundary of S.
S_δ	set of tolerant solutions for a given tolerant value $\delta \geq 0$.

$d(x,S)$	distance of x from S.
S^c	complement of S.
r_k	distance of $x^{(0)}$ from the kth surrogate hyperplane.
$\|\|v\|\|$	Euclidean norm of v.
$\|w\|$	absolute value of w.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Genesis of the Algorithms

Since the advent of electronic computers, much interest has been generated in the use of iterative methods to solve problems. For systems of linear equations, the Jacobi, Gauss-Seidel, SOR and SSOR methods(14) were developed. In the case of linear inequalities, equivalent schemes called relaxation methods were independently developed by Agmon(2), and Motzkin and Schoenberg(18).

The relaxation method as given by Agmon (and others) consists of a sequence $\{x^{(k)}\}$ of iterates. Each iterate is determined from the previous one by moving in the direction opposite to that of the outer normal(for the case of $Ax \leq b$) to the most violated half-space. The step-length moved is a multiple of the distance of $x^{(k)}$ to the hyperplane of this half-space and the multiplier, $0 < \lambda \leq 2$, is called the relaxation parameter. The terms used in conjunction with different values of the parameter are

$0 < \lambda < 1$	under-relaxation
$\lambda = 1$	projection
$1 < \lambda < 2$	over-relaxation
$\lambda = 2$	reflection.

Some research work already done on the relaxation method include those of Goffin(6,7) where he showed that the convergence of the algorithm depends on 2 condition number, the outer measure and the inner measure of a cone

formed by some of the constraints. He also in (8) gave a class of linear programming problems which Agmon's method cannot solve in polynomial time for any value of the relaxation parameter. Hoffman(11) approached the convergence proof differently. Jeroslow(13) in his modification of the algorithm gave a bound on the number of itrations k, required to get a solution. This he gave as $k \leq D^2/\delta^2$, where D is the distance of the initial point $x^{(0)}$ to any solution and δ is a tolerable error. He treated the relaxation parameter as a term not a factor. That is, he used

$$x^{(k+1)} = x^{(k)} - (d_p + \lambda)a^p \quad \text{for some} \quad \lambda \geq 0,$$

where p is the index of the most violated constraint and

$$d_p = a^p x^{(k)} - b_p > -(\lambda - \delta).$$

Jeroslow also handled an infinite number of constraints by choosing, at the kth iteration, the most violated of the first (k+1) constraints. Maurras, Truemper and Akgul(17) used Jeroslow's modification to give 7 algorithms that can solve a special class of linear programming problem in polynomial time.

Todd(21) gave a formula for a condition number for convergence that is different from that of Agmon. He also suggested how to use some constraints for simultaneous projection (over- or under-) so that finite convergence can be obtained where Agmon's fails to do so. Goldfarb and Todd(9), and Krol and Mirman(15) independently introduced and proved the validity of the idea of projecting onto multiple constraints. In Goldfarb and Todd(9) a nonnegative

linear combination of some violated constraints that make acute angles with one another (i.e. their constraint normals are mutually obtuse) is formed. They called these surrogate cuts. They also showed that not all the constraints used to form the surrogate had to be violated. Krol and Mirman(15) formed a positive linear combination of the most violated constraint and n-1 others (not necessarily all different) one at a time, with some specified formulae. These they called additional inequalities. Both parties used the idea not as a method by itself, but to modify the ellipsoid method.

1.2 What Is a Surrogate Method?

The term surrogate came from Goldfarb and Todd(9). The surrogate algorithms we have established are iterative schemes in the class of the relaxation methods. They are similar to those of Agmon(2), Motzkin and Schoenberg(18), and Krol and Mirman(15). But unlike Agmon's, and Motzkin and Schoenberg's the most violated is not simply one of the given hyperplanes. Rather it is a substitute hyperplane containing the manifold of the most violated and some other given hyperplanes. Also unlike Krol and Mirman we are not using these in an already proven algorithm. We have proven that they will terminate with or without a solution. Although the original method can be extremely slow, the modified versions are moderately fast.

1.3 Organisation

The following gives a brief cutline of how the chapters are organized. Chapter 2 contains the unnormalised (original) problem, the normalised form of it, and the auxilliary problem of finding surrogate constraints from the given ones. In chapter 3, we give additional definitions and notations, a brief sketch of Agmon's approach, a description of the first surrogate algorithm, Surrogate-I, lemmas for valid surrogate constraints, recursive definitions and a formal specification of the algorithm. There is one theorem for termination of the algorithm, three propositions on inconsistency and one theorem for detecting it. The chapter ends by exposing a situation where convergence is but only in the limit.

Chapter 4 presents faster algorithms. In stead of fixing the initial guess at one point, we re-initialize it at certain stages of the iterations. The first one, Surrogate-R, re-initializes x after each time a constraint has been chosen twice for the construction of a surrogate. The second, Surrogate-II, re-initializes at each iteration using two original constraints (in stead of one original and the current surrogate) to construct a new surrogate. The third one, Surrogate-III, is similar to Surrogate-II except that it uses three instead of two original constraints for the construction of a new one.

Chapter 5 is mainly devoted to convergence. The first of

the 3 theorems in this chapter is a prcof of ccnvergence of Surrogate-II which is based upon one given by Agmon for his method. The second theorem proves convergence of surrcgate for the narrow cone problem. The third proves convergence for Surrogate-I in the general case. The work done by each cf the 4 algorithms and Agmon's in terms of the number of constraints and variables are tabulated in this chapter.

Chapter 6 describes how the surrcgate methods can be adapted for solving equations and LP problems. How tc use the nonnegaity restriction on x as constraints without increasing the storage demand for the coefficient matrix is shown. Implementation for large sparse matrices is gives in chapter 7 together with manipulation codes. Chapter 8 has the computational results and analysis of a variety cf problems. In the appendix we give the flowchart and the code used for the test problems.

In the conclusion, we have expressed our belief that some good initial point and relaxation parameter for faster convergence can be found.

1.4 Equation Numbering

Formulae proved in lemmas and theorems and which are globally referenced are assigned decimal numbers using their section number as the first part. E.g. (3.4.1) refers to the first equation in sectien 3.4. Those only locally referenced are assigned integral numbers. E.g. (2) is the second equation in whatever section it appears.

2. LINEAR INEQUALITIES

2.1 The Initial Problem

Given i) m, n

ii) $\alpha^i \in R^n$, $\forall\, i \in I = \{1, 2, \ldots, m\}$, the ith row of the matrix $\mathcal{A} \in R^{m \times n}$,

iii) $\beta \in R^m$.

We want to find $x \in R^n$ such that

$$\alpha^i x \leq \beta_i \qquad \forall\, i \in I \tag{2.1.1}$$

i.e. $$\mathcal{A}x \leq \beta . \tag{2.1.2}$$

To simplify proofs of lemmas and theorems that follow, we shall normalize all α^i.

Let $$a^i = \frac{\alpha^i}{\| \alpha^i \|} \quad \text{and} \quad b_i = \frac{\beta_i}{\| \alpha^i \|} . \tag{2.1.3}$$

Then $$\frac{\alpha^i}{\| \alpha^i \|} x \leq \frac{\beta_i}{\| \alpha^i \|} \quad <==> \quad a^i x \leq b_i . \tag{2.1.4}$$

Therefore, $$Ax \leq b, \tag{2.1.5}$$

$$\text{where} \quad A = \begin{pmatrix} a^1 \\ a^2 \\ \cdot \\ \cdot \\ \cdot \\ a^m \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{pmatrix} . \tag{2.1.6}$$

From now on we shall use (2.1.3) to (2.1.6) instead of (2.1.1) and (2.1.2).

2.2 The Auxilliary Problem

In surrogate methods, a surrogate constraint

$$\hat{a}x \leq \hat{b} \tag{2.2.1}$$

is constructed at each iteration. The construction of

(2.2.1) gives rise to a new problem of the form: given a current $x^{(0)} \in R^n$,

determine $h \in R^m$ such that h solves the following problem,

$$\text{maximize} \qquad r = \sum_{i=1}^{m} h_i (a^i x^{(0)} - b_i) \qquad (2.2.2)$$

$$\text{subject to} \quad ||\hat{a}|| = 1 \qquad (2.2.3)$$

$$\text{and} \qquad h_i \geq 0 \quad \forall\, i \in I, \text{ with } h \neq \bar{0}, \qquad (2.2.4)$$

$$\text{where} \qquad \hat{a} = \sum_{i=1}^{m} h_i a^i. \qquad (2.2.5)$$

Note: the $\hat{b}$ in (2.2.1) is given by $\hat{b} = \sum_{i=1}^{m} h_i b_i$.

Since we are dealing with inequalities, care must be taken not to cut off any feasible region when we carry out scalar multiplications on the a^i's, hence the nonnegativity constraint on h. Constraint (2.2.3) guarantees that $\hat{a}$ is normalized.

2.3 Tolerant Value

In practice it is not always easy to get an exact solution. Therefore a small value $\delta \geq 0$ is often given so that any $x \in R^n$ resulting in

$$Ax \leq b + \delta e, \qquad (2.3.1)$$

where $e \in R^m$ is a vector of ones,

is accepted as a solution.

3. SURROGATE-I

Before describing the algorithm for finding a solution to the given problem, we would like to set down some other notation and definitions that will be basic throughout.

3.1 Notations and Definitions

1) The half-space defined by the ith inequality is given as

$$H_i = \{x \in R^n \mid a^i x - b_i \leq 0\}.$$

2) The bounding hyperplane to H_i is given as

$$\partial H_i = \{x \in R^n \mid a^i x - b_i = 0\}.$$

3) (a) The set of exact solutions is given as

$$S = \bigcap_{i=1}^{m} H_i = \{x \in R^n \mid Ax - b \leq 0\}.$$

(b) The set of tolerant solutions for a given tolerant value $\delta \geq 0$ is

$$S_\delta = \{x \in R^n \mid Ax - b \leq \delta e\},$$

where e is a vector of ones.

4) The error in $x \notin S$ is

$$d(x,S) = \min_{y \in S} ||x - y||.$$

5) If $x \in H_i$, we say that x is on the right side of H_i, otherwise it is on the wrong side.

6) The cosine of the angle between ∂H_i and ∂H_j is

$$c_{ij} = a^i . a^j,$$

and the sine is

$$\lambda_{ij} = \sqrt{1 - c_{ij}^2}.$$

With reference to figure 3.1, we alsc have the following:

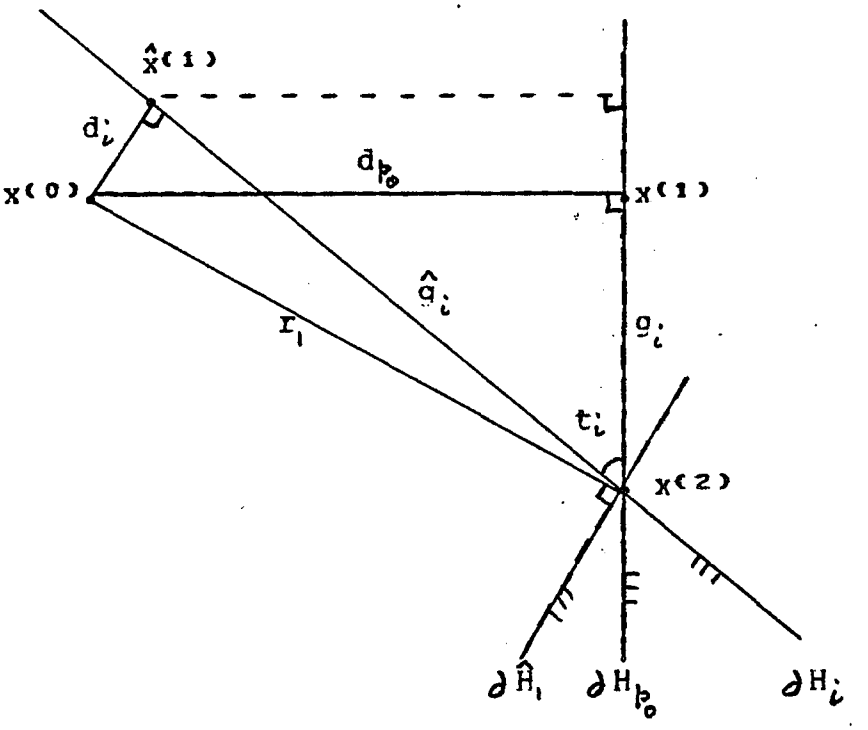


Fig. 3.1 Surrogate Half-space

7) The distance of $x^{(0)} \in R^n$ from ∂H_i is

$$d_i = a^i x^{(0)} - b_i \quad \forall\ i \in I.$$

Note: If $x^{(0)} \in H_i$, $d_i \leq 0$.

8) $p_o \in I$ is such that ∂H_{p_o} is the farthest hyperplane from $x^{(0)}$. In other words

$$d_{p_o} \geq d_i \quad \forall\ i \in I.$$

9) The orthogonal projection of $x^{(0)}$ onto ∂H_{p_o} is

$$x^{(1)} = x^{(0)} - d_{p_o} a^{p_o}.$$

10) The orthogonal projection of $x^{(0)}$ cnto ∂H_i, $i \neq p_o$, is

$$\hat{x}^{(1)} = x^{(0)} - d_i a^i.$$

11) $x^{(2)}$ is a point on the manifold $\partial H_{p_o} \cap \partial H_i$ closest to $x^{(1)}$.

Note that $(x^{(0)} - x^{(1)})^T (x^{(1)} - x^{(2)}) = 0$.

12) t_i is the angle between ∂H_{p_0} and ∂H_i.

13) The distance between $x^{(1)}$ and $x^{(2)}$ is

$$g_i = \frac{a^i x^{(1)} - b_i}{\sin(t_i)} .$$

14) The distance between $x^{(1)}$ and $x^{(2)}$ is

$$\hat{g}_i = \frac{a^{p_0} \hat{x}^{(1)} - b_{p_0}}{\sin(t_i)} .$$

15) The distance between $x^{(0)}$ and $x^{(2)}$ is

$$r_1 = \sqrt{d_{p_0}^2 + g_i^2} = \sqrt{d_i^2 + \hat{g}_i^2} .$$

16) $p_1 \in I$ is such that ∂H_{p_1} is the farthest viclated hyperplane intersecting ∂H_{p_0} from the point $x^{(1)}$,

i.e. $g_{p_1} \geq g_i \quad \forall\, i \in I$.

17) (a) A surrogate half-space $\hat{H}_1$ for H_{p_0} and H_{p_1} is a half-space containing the intersection $H_{p_0} \cap H_{p_1}$, and we write this as

$$\hat{H}_1 = \mathcal{L}(H_{p_0}, H_{p_1}) .$$

(b) A surrogate hyperplane $\partial \hat{H}_1$ for H_{p_0} and H_{p_1} is the hyperplane containing the manifold $\partial H_{p_0} \cap \partial H_{p_1}$ and we express this as

$$\mathcal{L}(\partial H_{p_0}, \partial H_{p_1}) .$$

3.2 Agmon's Approach

To bring out the similarities and dissimilarities in Agmon's method and the surrogate method, we shall briefly present his approach first. In his method only one constraint is involved at each iteration. The following are the three lemmas that support his convergence theorem.

Lemma 3.2.1 (Agmon's lemma 2.1)

Consider the following 3 points, x, y, z, and a half-space H with $x \notin H$, $y \in H$ and such that $z \in \partial H$ is the orthogonal projection of x onto ∂H.

Then a) $||z - y|| < ||x - y||$,

b) $||z - y||^2 \leq ||x - y||^2 - ||z - x||^2$.

Proofs

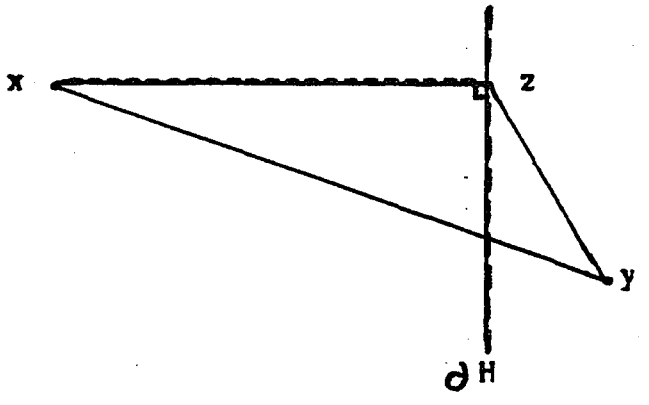


Fig. 3.2 Points Separated by a Half-space

The proofs of the lemma are trivial. We know that any 3 points in space lie on a 2-dimensional plane. Since z is the orthogonal projection of x onto ∂H, and $y \in H$, then $\angle xzy \geq 90°$. Therefore $\angle yxz < 90°$. From elementary geo-

metry, we know that in any triangle, the side opposite the greater angle is the greater side.

$\therefore$ a) $||z - y|| < ||x - y||$.

For the (b) part, we apply the cosine rule.

$$||x - y||^2$$
$$= ||x - z||^2 + ||z - y||^2 - 2||x - z||.||z - y||\cos(xzy)$$
$$\geq ||x - z||^2 + ||z - y||^2 , \quad \text{since } \cos(xzy) \leq 0.$$

$\therefore$ b) $||z - y||^2 \leq ||x - y||^2 - ||z - x||^2$.

Lemma 3.2.2 (Agmon's lemma 2.2)

Let i) no superfluity exist in the inequalities

$$a^i x \leq b_i \qquad \forall\, i \in I$$

that defines the polytope S,

ii) $x \notin S$ and $y^* \in \partial S$ such that

$$||x - y^*|| < ||x - y|| \quad \forall\, y \in S \text{ and } y \neq y^*, \quad \ldots(1)$$

iii) $I_{y^*} = \{i \in I | a^i y^* - b_i = 0\}$,

iv) S_{y^*} be the polyhedral cone defined by

$$a^j x \leq b_j \qquad \forall\, j \in I_{y^*}.$$

Then $x \notin S_{y^*}$ and $y^* \in \partial S_{y^*}$ and is such that

$$||x - y^*|| < ||x - z|| \quad \forall\, z \in S \quad \text{and } z \neq y^*.$$

Proof

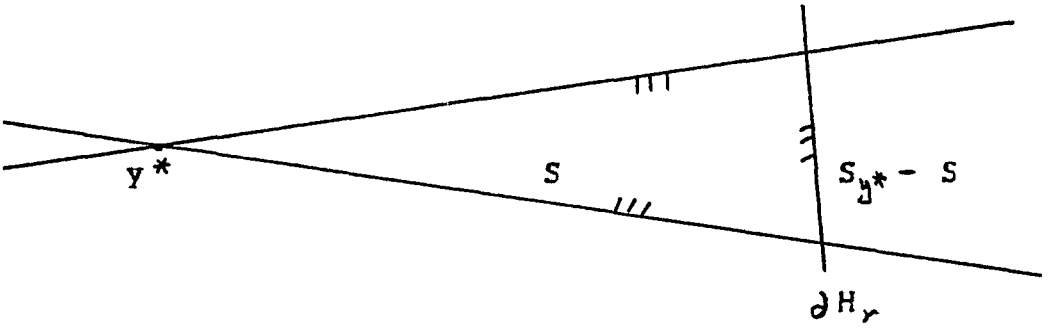


Fig. 3.3 Cone Containing S

Suppose $x \in S_{y^*}$. Then since $x \notin S$, $\exists r \in (I - I_{y^*})$ such that $x \in H_r^c \bigcap (S_{y^*} - S)$. But $y^* \in H_r$ since $y^* \in S$. Therefore there exists $\hat{y} \in \partial H_r$ and on the line segment xy^* such that

$$||x - \hat{y}|| < ||x - y^*|| \qquad \ldots\ldots\ldots (2)$$

But $\hat{y} \in S$. Therefore (2) contradicts (1).

$\therefore \quad x \notin S_{y^*}$.

For the second part, let $\bar{y} \in \partial S_{y^*}$ with $\bar{y} \neq y^*$. Then there is a spherical neighborhood N_ϵ in R^n around y^* such that its points belong to H_r. Therefore the intersection of N_ϵ and the line segment $\bar{y}y^*$ is in ∂S. Let this point of intersection be y'. Now for some $\alpha_1 > 0$ and $\alpha_1 > 0$ with $\alpha_1 + \alpha_2 = 1$, we have

$$||x - y'|| < \alpha_1 ||x - y^*|| + \alpha_2 ||x - \bar{y}||.$$

Since $y' \in S$, $||x - y^*|| < ||x - y'||$ by (1) above.

$\therefore \quad ||x - y^*|| < \alpha_1 ||x - y^*|| + \alpha_2 ||x - \bar{y}||$

$(1 - \alpha_1)||x - y^*|| < \alpha_2 ||x - \bar{y}||$

$\therefore \quad ||x - y^*|| < ||x - \bar{y}||.$

This proves that for $z \in S_{y^*}$ and $z \neq y^*$,

$$||x - y^*|| < ||x - z||,$$

which completes the proof of the lemma.

Lemma 3.2.3 (Agmon's lemma 2.3)

Let i) a polyhedral cone V be defined by the homogeneous inequalities

$$a^i w \leq 0 \qquad \forall\, i \in I,$$

ii) E be the set of pcints $x \notin V$ and such that the origin is the point on ∂V nearest to x,

iii) $J = \{i \in I \mid a^i x > 0\}$,

iv) $f_j(x)$ = distance of $x \in E$ from ∂H_j $\forall\, j \in J$.

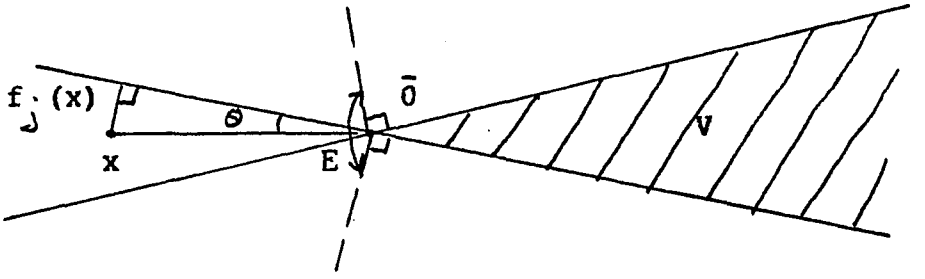


Fig. 3.4 Cone with vertex at the Origin

Then $\exists\ \lambda = \lambda(V) > 0$ such that

$$\inf_{x \in E} \max_{j \in J} \frac{f_j(x)}{\|x\|} = \lambda .$$

(Note: λ is the smallest sine of the angle defined by ∂H_j and the line joining x to the origin.)

Proof

Consider the polyhedral cone S_{y^*} of lemma 3.2.2.

$$z \in S_{y^*} \Rightarrow \|x - y^*\| < \|x - z\|, \quad z \neq y^*$$
$$= \|(x - y^*) - (z - y^*)\| \quad \ldots\ldots (1)$$

Now consider the vectors $(x - y^*)$ and $(z - y^*)$.

$\forall\, i \in I_{y^*}$, $a^i(x - y^*) = a^i x - b_i$ since $a^i y^* = b_i$

> 0 since $x \notin S$.

$\therefore$ $(x - y^*) \notin V$.

$\forall\, i \in I_{y^*}$, $a^i(z - y^*) = a^i z - b_i$ since $a^i y^* = b_i$

≤ 0 since $z \in S_{y^*}$.

$\therefore$ $(z - y^*) \in V$.

Since we can choose z in (1) above to be on ∂S_{y^*}, $(z - y^*)$ is on ∂V. By (1), $(x - y^*)$ is closer to $\bar{0} \in \partial V$ than is $(z - y^*)$.

$\therefore$ $(x - y^*) \in E$ by (ii).

$\forall\, j \in J,\ f_j(x - y^*) = a^j(x - y^*) > 0.$

$$\therefore \quad \frac{f_j(x - y^*)}{\|x - y^*\|} > 0 \qquad \forall\, j \in J.$$

But, $\dfrac{f_j(x - y^*)}{\|x - y^*\|}$ = the sine of the angle defined by ∂H_j and the line joining $(x - y^*)$ to the origin.

Let α_{ij} be the angle between ∂H_i and ∂H_j.

Then $\max_{i,j \in J} \sin(\alpha_{ij}/2) = \lambda(V)$.

$$\inf_{(x - y^*) \in E} \max_{j \in J} \frac{f_j(x - y^*)}{\|x - y^*\|} = \max_{i,j \in J} \sin(\alpha_{ij}/2)$$

$$= \lambda(V).$$

$$\therefore \quad \inf_{(x - y^*) \in E} \max_{j \in J} \frac{f_j(x - y^*)}{\|x - y^*\|} = \lambda(V) > 0.$$

3.2.1 Agmon's Algorithm

1) Choose $x^{(0)} \in R^n$ arbitrarily and set $k = 0$;

2) If $x^{(k)} \in S$, stop;

3) Else set $x^{(k+1)}$ = the orthogonal projection of $x^{(k)}$ onto the most violated hyperplane ∂H_{p_k}.

i.e. $x^{(k+1)} = x^{(k)} - d_{p_k} a^{p_k}$;

4) Set $k = k + 1$ and go to 2;

Theorem 3.2.1.1 (Agmon's theorem 3)

If the system of inequalities of the problem is consistent and $\{x^{(k)}\}$ is the sequence of iterations defined above,

then $x^{(k)} \dashrightarrow x \in S$.

Also $\|x^{(k)} - x\| \leq 2\theta^k \|x^{(0)} - y^{(0)}\|$,

where $0 < \theta < 1$ with $\theta = \theta(A)$.

Proof

Let $\lambda_{min} = \min_{i,j \in I} \sin(\alpha_{ij})$ and $\theta = \sqrt{1 - \lambda_{min}^2}$

$$\|x^{(k+1)} - x^{(k)}\| = \max_{i \in I} d(x^{(k)}, \partial H_i)$$

$$\geq \lambda \|x^{(k)} - y^{(k)}\| \quad \text{by lemma 3.2.3}$$

$$\geq \lambda_{min} \|x^{(k)} - y^{(k)}\| \quad \ldots\ldots\ldots\ldots (1)$$

$$\|x^{(k+1)} - y^{(k)}\|^2 \leq \|x^{(k)} - y^{(k)}\|^2 - \|x^{(k+1)} - x^{(k)}\|^2$$
by lemma 3.2.1(b)

$$\leq \|x^{(k)} - y^{(k)}\|^2 - \lambda_{min}^2 \|x^{(k)} - y^{(k)}\|^2$$
by (1) above

$$= (1 - \lambda_{min}^2) \|x^{(k)} - y^{(k)}\|^2$$

$$= \theta^2 \|x^{(k)} - y^{(k)}\|^2$$

Therefore
$$\|x^{(k+1)} - y^{(k)}\| \leq \theta \|x^{(k)} - y^{(k)}\| \quad \ldots\ldots\ldots\ldots (2)$$
But
$$\|x^{(k+1)} - y^{(k+1)}\| \leq \|x^{(k+1)} - y^{(k)}\| \quad \text{by definition}$$
Therefore
$$\|x^{(k+1)} - y^{(k+1)}\| \leq \theta \|x^{(k)} - y^{(k)}\| \quad \text{by (2) above}$$

$$\leq \theta^2 \|x^{(k-1)} - y^{(k-1)}\|$$

$$\vdots$$

$$\leq \theta^{k+1} \|x^{(0)} - y^{(0)}\|$$

$$\therefore \|x^{(k)} - y^{(k)}\| \leq \theta^k \|x^{(0)} - y^{(0)}\| \qquad (3.2.1)$$

$x, y^{(k)}, x^{(k)}, x^{(k+1)}, \ldots$ are all included in the hypersphere

$$S^{(k)}: \|x - y^{(k)}\| \leq \|x^{(k)} - y^{(k)}\| \leq \theta^k \|x^{(0)} - y^{(0)}\|$$

$$\therefore \quad S^{(k)} \supset S^{(k+1)} \supset \ldots$$

i.e. $\bigcap_{k=0}^{\infty} S^{(k)} = x.$

$\therefore \lim_{k \to \infty} x^{(k)} = x = \lim_{k \to \infty} y^{(k)}.$

The diameter is less than or equal to $2\theta^k ||x^{(0)} - y^{(0)}||.$

$\therefore ||x^{(k)} - x|| \leq 2\theta^k ||x^{(0)} - y^{(0)}||.$

3.3 Description of the Surrogate Algorithm

Until the construction of a surrogate hyperplane is shown, a formal specification of the algorithm cannot be given. Below, the basic idea of the algorithm is presented.

1) Choose $x^{(0)} \in R^n$ arbitrarily;
2) If $x^{(0)} \in S$, stop;
3) Else set $k = 0$ and $\partial \hat{H}_0 = \partial H_{p_0}$, the farthest hyperplane from $x^{(0)}$;
4) Move on $\partial \hat{H}_k$ to $\partial H_{p_{k+1}}$, the farthest hyperplane intersecting $\partial \hat{H}_k$ with $g_{p_{k+1}} > 0$;
5) If there is no such p_{k+1}, then the orthogonal projection of $x^{(0)}$ onto $\partial \hat{H}_k$ is a solution, stop;
6) Else construct $\partial \hat{H}_{k+1}$ from $\partial \hat{H}_k$ and $\partial H_{p_{k+1}}$;
7) Set $k = k + 1$ and go to 4;

3.4 Surrogate Construction

The surrogate for any subset of the given constraints is a linear combination of those constraints. It is dependent not just on those constraints per se, but also on the relationship between each pair of these constraints and

$x^{(0)}$. That is, the multipliers in the linear combination are fuctions of the angle defined by these constraints and $x^{(0)}$. Lemma 1 establishes 5 very useful and frequently used formulae while lemma 2 establishes 2 impcrtant formulae used by the algorithm to construct surrogate constraints. Proof of the validity of this construction is done later in lemma 5.

Lemma 1 For any $i \in I$ and $|c_{p_o i}| \neq 1$,

$$\text{i)}\quad g_i = \frac{d_i - d_{p_o} c_{p_o i}}{\sqrt{1 - c_{p_o i}^2}}, \tag{3.4.1}$$

$$\text{ii)}\quad \hat{g}_i = \frac{d_{p_o} - d_i c_{p_o i}}{\sqrt{1 - c_{p_o i}^2}}, \tag{3.4.2}$$

$$\text{iii)}\quad \hat{g}_i + g_i c_{p_o i} = d_{p_o} \sqrt{1 - c_{p_o i}^2}, \tag{3.4.3}$$

$$\text{iv)}\quad \hat{g}_i c_{p_o i} + g_i = d_i \sqrt{1 - c_{p_o i}^2}, \tag{3.4.4}$$

$$\text{v)}\quad r_i^2 = \frac{\hat{g}_i^2 + g_i^2 + 2\hat{g}_i g_i c_{p_o i}}{1 - c_{p_o i}^2}. \tag{3.4.5}$$

Proofs

i) $$g_i = \frac{a^i x^{(1)} - b_i}{\sin(t_i)} \qquad \text{by definition (13)}$$

$$= \frac{a^i (x^{(0)} - d_{p_o} a^{p_o}) - b_i}{\sin(t_i)} \qquad \text{by definition (9)}$$

$$= \frac{d_i - d_{p_o} a^{p_o} \cdot a^i}{\sin(t_i)} \qquad \text{by definition (7)}$$

$$= \frac{d_i - d_{b_0} c_{b_0 i}}{\sqrt{1 - c_{b_0 i}^2}} \qquad \text{by definitions (6)}$$

ii) Similarly, $\hat{g}_i = \dfrac{d_{b_0} - d_i c_{b_0 i}}{\sqrt{1 - c_{b_0 i}^2}}$

iii) From (i) and (ii) we have,

$$\hat{g}_i + g_i c_{b_0 i} = \frac{d_{b_0} - d_{b_0} c_{b_0 i}^2}{\sqrt{1 - c_{b_0 i}^2}} = d_{b_0} \sqrt{1 - c_{b_0 i}^2}$$

iv) Similarly, $\hat{g}_i c_{b_0 i} + g_i = d_i \sqrt{1 - c_{b_0 i}^2}$

v) $r_1^2 = d_{b_0}^2 + g_i^2$ by definition (15)

$$= \frac{(\hat{g}_i + g_i c_{b_0 i})^2}{1 - c_{b_0 i}^2} + g_i^2 \qquad \text{by (iii)}$$

$$= \frac{\hat{g}_i^2 + g_i^2 + 2\hat{g}_i g_i c_{b_0 i}}{1 - c_{b_0 i}^2}$$

Lemma 2

A hyperplane $\partial \hat{H}_1$ constructed at $\partial H_{b_0} \cap \partial H_i$ such that $x^{(2)}$ is the orthogonal projection of $x^{(0)}$ onto it and such that its normal $\hat{a}_1$ is a linear combination of a^{b_0} and a^i has the following equations.

i) $$\hat{a}_1 = \frac{\hat{g}_i}{r_1 \sqrt{1 - c_{b_0 i}^2}} a^{b_0} + \frac{g_i}{r_1 \sqrt{1 - c_{b_0 i}^2}} a^i, \qquad (3.4.6)$$

ii) $$x^{(2)} = x^{(0)} - r_1 \hat{a}_1 . \qquad (3.4.7)$$

Proof .

i) $$r_1^2 = \frac{\hat{g}_i^2 + g_i^2 + 2\hat{g}_i g_i c_{b_o i}}{1 - c_{b_o i}^2} \qquad \text{by eq. (3.4.5)}$$

$$= \frac{\hat{g}_i^2 + g_i^2 + 2\hat{g}_i g_i a^{b_o} \cdot a^i}{1 - c_{b_o i}^2} \qquad \text{by definition (6)}$$

$$= \left\| \frac{\hat{g}_i}{\sqrt{1 - c_{b_o i}^2}} a^{b_o} + \frac{g_i}{\sqrt{1 - c_{b_o i}^2}} a^i \right\|^2$$

$$\therefore \quad \left\| \frac{\hat{g}_i}{r_1 \sqrt{1 - c_{b_o i}^2}} a^{b_o} + \frac{g_i}{r_1 \sqrt{1 - c_{b_o i}^2}} a^i \right\| = 1$$

Set $$\hat{a}_1 = \frac{\hat{g}_i}{r_1 \sqrt{1 - c_{b_o i}^2}} a^{b_o} + \frac{g_i}{r_1 \sqrt{1 - c_{b_o i}^2}} a^i$$

ii) Taking the dot-product of a^{b_o} and $\hat{a}_1$ from (i), we have

$$a^{b_o} \cdot \hat{a}_1 = \frac{\hat{g}_i + g_i c_{b_o i}}{r_1 \sqrt{1 - c_{b_o i}^2}}$$

$$= \frac{d_{b_o}}{r_1} \qquad \text{by eq. (3.4.3)}$$

$$= \frac{a^{b_o} x^{(0)} - b_{b_o}}{r_1} \qquad \text{by definition (7)}$$

$$\therefore \quad r_1 a^{b_o} \cdot \hat{a}_1 = a^{b_o} x^{(0)} - b_{b_o} \qquad \text{.....(1)}$$

$$x^{(2)} \in \partial H_{b_o} \iff a^{b_o} x^{(2)} - b_{b_o} = 0 \qquad \text{.....(2)}$$

Adding (1) and (2) gives

$$r_1 a^{b_o} \cdot \hat{a}_1 = a^{b_o} x^{(0)} - a^{b_o} x^{(2)}$$

$$a^{b_o} (r_1 \hat{a}_1 - x^{(0)} + x^{(2)}) = 0$$

Since $a^{b_o} \neq \bar{0}$, $r_1 \hat{a}_1 - x^{(0)} + x^{(2)} = 0$

$$\therefore \quad x^{(2)} - x^{(0)} = -r_1 \hat{a}_1$$

$x^{(2)} - x^{(0)}$ being a multiple of $\hat{a}_1$ means that $x^{(2)}$ is the orthogonal projection of $x^{(0)}$ onto $\partial\hat{H}_1$.

$$\therefore \quad x^{(2)} = x^{(0)} - r_1 \hat{a}_1 .$$

This completes the prcof of lemma 2.

3.5 The Significance cf $c_{p_o i} = \pm 1$

In lemma 1, we excluded $c_{p_o i} = \pm 1$ mainly to avoid division by zero. But these values, especially $c_{p_o i} = -1$ under certain conditions, have some important meaning. Since $c_{p_o i}$ is the cosine of the angle between ∂H_{p_o} and ∂H_i, $c_{p_o i} = \pm 1$ implies that ∂H_{p_o} is parallel to ∂H_i. There are 2 cases.

Case 1: $\partial H_{p_o} \cap \partial H_i = \emptyset$ and $H_{p_o} \cap H_i \neq \emptyset$.

Case 2: $\partial H_{p_o} \cap \partial H_i = \emptyset$ and $H_{p_o} \cap H_i = \emptyset$.

Case 1 should not worry us but case 2 should, fcr the obvious reason that there is no soluticn. To establish the condition under which $c_{p_o i} = -1$ is important, we have the following:-

Lemma 3

$$x^{(0)} \notin S_\delta \quad \text{iff} \quad d_{p_o} > \varepsilon .$$

Prcof

a) (==>)

$x^{(0)} \notin S_\delta$ ==> $\exists j \in I$ such that $d_j > \delta$.

But $d_{i_0} \geq d_i$ $\forall\, i \in I$ by definition (8).

$\therefore$ $d_{i_0} > \delta$.

b) (<==)

$d_{i_0} > \delta$ ==> $x^{(0)} \notin H^\delta_{i_0}$, where $H^\delta_{i_0} = \{x \mid a^{i_0} x - b_{i_0} \leq \delta\}$.

==> $x^{(0)} \notin S_\delta$.

Assume that $x^{(0)} \notin S$, then

Lemma 4 (Krol and Mirman)

For any $i \in I$

a) $d_i - d_{i_0} c_{i_0 i} > 0$ and $c_{i_0 i} = -1$ ==> inconsistency,

b) $|c_{i_0 i}| \neq 1$ and $g_i > 0$ ==> $\hat{g}_i > 0$.

Procf of a)

$d_i - d_{i_0} c_{i_0 i} > 0$ ==> $d_i > -d_{i_0}$ since $c_{i_0 i} = -1$. .....(1)

$c_{i_0 i} = -1$ <==> $a^{i_0} = -a^i$. .....(2)

Suppose $v \in S$. Then

$v \in S$ ==> $a^i v - b_i \leq 0$

==> $a^i v - a^i x^{(0)} + d_i \leq 0$ by definition (7)

==> $a^i v - a^i x^{(0)} - d_{i_0} < 0$ by (1) above

==> $-a^{i_0} v + a^{i_0} x^{(0)} - d_{i_0} < 0$ by (2) above

==> $a^{i_0} v - a^{i_0} x^{(0)} + d_{i_0} > 0$

==> $a^{i_0} v - b_{i_0} > 0$ by definition (7)

==> $v \notin S$.

Contradiction, therefcre (a) holds.

Proof of (b)

$|c_{p_0 i}| \neq 1 \iff |c_{p_0 i}| < 1$

$$\hat{g}_i = \frac{d_{p_0} - d_i c_{p_0 i}}{\sqrt{1 - c_{p_0 i}^2}} \qquad \text{by eq. (3.4.2)}$$

$$= \frac{d_{p_0}(1 + c_{p_0 i}) - d_{p_0} c_{p_0 i} - d_i (1 + c_{p_0 i}) + d_i}{\sqrt{1 - c_{p_0 i}^2}}$$

$$= \frac{d_i - d_{p_0} c_{p_0 i} + (d_{p_0} - d_i)(1 + c_{p_0 i})}{\sqrt{1 - c_{p_0}^2}}$$

$$= g_i + (d_{p_0} - d_i)\frac{1 + c_{p_0 i}}{\sqrt{1 - c_{p_0 i}^2}} \qquad \text{by eq. (3.4.1)}$$

$\geq g_i$ since $d_{p_0} \geq d_i$ and $|c_{p_0 i}| < 1$.

$\therefore \quad g_i > 0 \implies \hat{g}_i > 0$

This completes the prcof of lemma 4.

Note that part (b) of the lemma shows that construction of surrogate hyperplane is valid.

3.6 Recursive Definitions

We shall now give the recursive definitions of some of the formulae already established which are necessary for the formal specification of the algorithm. It is obvious from definition 15 that $r_1 > d_{p_0}$. Therefore our surrogate hyperplane of lemma 2 is now the farthest hyperplane from $x^{(0)}$. If we now use r_1 and $\hat{a}_1$ as d_{p_0} and a^{p_0} respectively,

we can compute r_2 and $\hat{a}_2$ and hence $x^{(3)}$ as shown in lemmas 1 and 2. That is, for some $i \in I$ our next set of formulae as derived in lemmas 1 and 2 are

i) $$g_i = \frac{d_i - r_1 \hat{a}_1 \cdot a^i}{\sqrt{1 - (\hat{a}_1 \cdot a^i)^2}},$$

ii) $$\hat{g}_i = \frac{r_1 - d_i \hat{a}_1 \cdot a^i}{\sqrt{1 - (\hat{a}_1 \cdot a^i)^2}},$$

iii) $$\hat{g}_i + g_i \hat{a}_1 \cdot a^i = r_1 \sqrt{1 - (\hat{a}_1 \cdot a^i)^2},$$

iv) $$\hat{g}\, \hat{a}_1 \cdot a^i + g_i = d_i \sqrt{1 - (\hat{a}_1 \cdot a^i)^2},$$

v) $$r_2^2 = r_1^2 + g_i^2 = \frac{\hat{g}_i^2 + g_i^2 + 2\hat{g}_i g_i \hat{a}_1 \cdot a^i}{1 - (\hat{a}_1 \cdot a^i)^2},$$

vi) $$\hat{a}_2 = \frac{\hat{g}_i}{r_2 \sqrt{1 - (\hat{a}_1 \cdot a^i)^2}} \hat{a}_1 + \frac{g_i}{r_2 \sqrt{1 - (\hat{a}_1 \cdot a^i)^2}} a^i,$$

vii) $$x^{(3)} = x^{(0)} - r_2 \hat{a}_2.$$

This process can be repeated as many times as necessary. Therefore, let k be the iteration number. Then for k = 0,1, 2, ... we replace the subscript 1 by k+1 in definitions 15 to 17 and lemmas 1 and 2 making them recursive. These recursive form will be given and used in the formal specification of the algorithm. Future references to such definitions will be directed to the steps of this algorithm.

3.7 Formal Specification of Algorithm I

1) Choose $x^{(0)} \in R^n$ arbitrarily;

2) $d = Ax^{(0)} - b$;

3) Choose $p_0 \in I$ such that $d_{p_0} \geq d_i \quad \forall\, i \in I$;

4) If $d_{p_0} \leq 0$, then $x^{(0)} \in S$, stop;

5) $r_0 = d_{p_0}$, $\hat{a}_0 = a^{p_0}$, $k = 0$;

6) $c_{k+1} = A\hat{a}_k^T$;

7) Choose $p_{k+1} \in I$ such that $g^2_{p_{k+1}} = \max_{i \in I, g_i > 0} g_i^2$

where $g_i^2 = \dfrac{(d_i - r_k c_{k+1,i})^2}{1 - c^2_{k+1,i}} \quad \forall\, i \in I$;

8) [1]If $\exists i \in I$ and k such that $c_{k+1,i} = -1$ and $g_i > 0$, then stop with no solution;

9) [2]If $d_{p_{k+1}} - r_k c_{p_{k+1}} \leq \delta$, stop with $x^{(k+1)} = x^{(0)} - r_k \hat{a}_k$ as a solution;

10) [3]$r_{k+1} = \sqrt{r_k^2 + \lambda^2 c^2_{p_{k+1}}}$;

11) [3]$h_{p_{k+1}} = \lambda \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{r_{k+1}(1 - c^2_{p_{k+1}})}$;

12) [3]$\hat{h}_{p_{k+1}} = \dfrac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{r_{k+1}(1 - c^2_{p_{k+1}})} - (\lambda - 1) c_{p_{k+1}} \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{r\ \ (1 - c^2_{p_{k+1}})}$;

13) $\hat{a}_{k+1} = \hat{h}_{p_{k+1}} \hat{a}_k + h_{p_{k+1}} a^{p_{k+1}}$;

14) $c_{k+2} = \hat{h}_{p_{k+1}} c_{k+1} + h_{p_{k+1}} A (a^{p_{k+1}})^T$;

15) $k = k + 1$ and go to 7;

[1] See section 3.9.1 on inconsistency.

[2] $c_{p_{k+1}}$ is a short form of the scalar $c_{k+1,p_{k+1}}$.

[3] λ is a relaxation parameter. See eq. (8.3.3) for valid values.

3.8 Validity of the Surrogate

Lemmas 1 through 4 are also valid for the recursive forms since we have shown in the recursion section that wherever d_{p_0} and a^{p_0} are used, r_k and $\hat{a}_k$ can be used for all k. In lemma 2 we only proved that $\hat{a}_1$ is a linear combination of a^{p_0} and a^{p_1}. We shall now prove that it is a positive linear combination.

Lemma 5 For all k,

i) $\hat{a}_k$ is a positive linear combination of $\hat{a}_{k-1}$ and a^{p_k}.

ii) $||\hat{a}_k|| = 1$.

Proofs (For projection, i.e. $\lambda = 1$)

i) If $g_i \leq 0 \ \forall \ i \in I$, the algorithm terminates at step 9.

Therefore going beyond step 9 implies that $g_{p_k} > 0$.

$g_{p_k} > 0 \Rightarrow \hat{g}_{p_k} > 0$ by lemma 4(b)

$\therefore$ Both $\hat{h}_{p_k} = \dfrac{g_{p_k}}{r_k\sqrt{1 - c_{p_k}^2}}$ and $h_{p_k} = \dfrac{g_{p_k}}{r_k\sqrt{1 - c_{p_k}^2}}$ are positive.

ii) This part will be proved by induction.

a) k = 0

$$||\hat{a}_0||^2 = \hat{a}_0 \cdot \hat{a}_0$$
$$= a^{p_0} \cdot a^{p_0} \quad \text{by step 5.}$$
$$= 1 \quad \text{by normalization of } a^i \ \forall \ i \in I.$$

b) Assume that the lemma is valid for k = n.

$$\|\hat{a}_{n+1}\|^2 = \left\| \frac{\hat{g}_{p_{n+1}}}{r_{n+1}\sqrt{1 - c^2_{p_{n+1}}}} \hat{a}_n + \frac{g_{p_{n+1}}}{r_{n+1}\sqrt{1 - c^2_{p_{n+1}}}} a^{p_{n+1}} \right\|^2$$

by steps 7,11,12,13

$$= \frac{1}{r^2_{n+1}(1 - c^2_{p_{n+1}})} (\hat{g}^2_{p_{n+1}} + 2\hat{g}_{p_{n+1}} g_{p_{n+1}} c_{p_{n+1}} + g^2_{p_{n+1}})$$

since $\hat{a}_n \cdot \hat{a}_n = 1$ by induction

and $a^{p_{n+1}} \cdot a^{p_{n+1}} = 1$ by normalization

$$= \frac{1}{r^2_{n+1}(1 - c^2_{p_{n+1}})} r^2_{n+1}(1 - c^2_{p_{n+1}}) \text{ by steps 7,10,11,12}$$

$$= 1$$

$$\therefore \quad \|\hat{a}_k\| = 1 \quad \forall\, k$$

This completes the prcof of the lemma.

3.8.1 The Kuhn-Tucker Conditions

We have shown in the last lemma that $\hat{h}_{p_{k+1}}$ and $h_{p_{k+1}}$ are positive. We also have to show that they satisfy the Kuhn-Tucker conditions for the auxilliary problem. Since we use 2 constraints at a time, equations (2.2.2) and (2.2.3) reduce to

$$\text{maximize} \quad f(h) = h_{p_{k+1}} r_k + h_{p_{k+1}} d_{p_{k+1}}$$

$$\text{subject to} \quad \hat{h}^2_{p_{k+1}} + 2\hat{h}_{p_{k+1}} h_{p_{k+1}} c_{p_{k+1}} + h^2_{p_{k+1}} = 1$$

For these above, the Kuhn-Tucker conditions demand that $\exists\, \mu > 0$ such that h satisfies the following 6 conditions:-

1) $r_k - 2\mu(\hat{h}_{p_{k+1}} + h_{p_{k+1}} c_{p_{k+1}}) \leq 0$,

2) $\hat{h}_{p_{k+1}}(r_k - 2\lambda u(\hat{h}_{p_{k+1}} + h_{p_{k+1}} c_{p_{k+1}})) = 0,$

3) $d_{p_{k+1}} - 2\lambda u(\hat{h}_{p_{k+1}} c_{p_{k+1}} + h_{p_{k+1}}) \leq 0,$

4) $h_{p_{k+1}}(d_{p_{k+1}} - 2\lambda u(\hat{h}_{p_{k+1}} c_{p_{k+1}} + h_{p_{k+1}})) = 0,$

5) $\hat{h}^2_{p_{k+1}} + 2\hat{h}_{p_{k+1}} h_{p_{k+1}} c_{p_{k+1}} + h^2_{p_{k+1}} - 1 = 0,$

6) $\lambda u(\hat{h}^2_{p_{k+1}} + 2\hat{h}_{p_{k+1}} h_{p_{k+1}} c_{p_{k+1}} + h^2_{p_{k+1}} - 1) = 0.$

We showed in lemma 5(ii) that condition 5 is satisfied. Therefore any $\lambda u > 0$ that satisfies conditions 1 - 4, satisfies condition 6.

The left hand side (lhs) of condition 1

$$= r_k - \frac{2\lambda u}{r_{k+1}\sqrt{1 - c^2_{p_{k+1}}}}(\hat{g}_{p_{k+1}} + g_{p_{k+1}} c_{p_{k+1}}) \quad \text{by steps 7,11,12}$$

$$= r_k - \frac{2\lambda u}{r_{k+1}\sqrt{1 - c^2_{p_{k+1}}}} r_k \sqrt{1 - c^2_{p_{k+1}}} \quad \text{by steps 7,11,12}$$

$$= r_k \left(1 - \frac{2\lambda u}{r_{k+1}}\right)$$

Set $\lambda u = r_{k+1}/2$ .....(1)

Surely this satisfies conditions 1 and 2. Let us test this on conditions 3 and 4.

The lhs of condition 3

$$= d_{p_{k+1}} - \frac{2\lambda u}{r_{k+1}\sqrt{1 - c^2_{p_{k+1}}}}(\hat{g}_{p_{k+1}} c_{p_{k+1}} + g_{p_{k+1}}) \quad \text{by steps 7,11,12}$$

$$= d_{p_{k+1}} - \frac{1}{\sqrt{1 - c^2_{p_{k+1}}}} (\hat{g}_{p_{k+1}} c_{p_{k+1}} + g_{p_{k+1}}) \quad \text{by (1) above}$$

$$= d_{p_{k+1}} - \frac{1}{\sqrt{1 - c^2_{p_{k+1}}}} d_{p_{k+1}} \sqrt{1 - c^2_{p_{k+1}}} \quad \text{by steps 7,11,12}$$

$$= 0.$$

Similarly, $\mu = r_{k+1}/2$ satisfies condition 4.

Therefore, h satisfies the Kuhn-Tucker conditions for the auxilliary problem.

3.9 Termination of the Algorithm

In all iterative schemes, there has to be at least one exit from the loop. In ours we have given 2 - steps 8 and 9. We want to justify the 2 steps. We have 5 theorems for this. The first one is the proof of step 9 - successful termination of the algorithm. We chose to do this first because it is very short. The second theorem is for aborting the algorithm - proof of step 8. We do not claim, however, that this terminating factor of step 8 will always be found if the system is inconsistent. The other 3 theorems are convergence theorems and are found in chapter 5. In these 3 theorems we show that if the system is consistent, we shall always get the condition of step 9.

Theorem 1

A necessary and sufficient condition for $x^{(k)} \in S_\delta$ is that

$$g_{p_k} \leq \frac{\delta}{\sqrt{1 - c_{p_k}^2}}, \qquad k \in \{1,2,3, \ldots\}$$

Proof

$x^{(k)} \in S_\delta \iff a^i x^{(k)} - b_i \leq \delta \quad \forall\, i \in I$

$\iff a^i (x^{(0)} - r_{k-1} \hat{a}_{k-1}) - b_i \leq \delta \quad \forall\, i \in I$ by step 9

$\iff d_{p_k} - r_{k-1} c_{p_k} \leq \delta$ by definitions (6) and (7)

$$\iff g_{p_k} \leq \frac{\delta}{\sqrt{1 - c_{p_k}^2}}$$ by step 9

3.9.1 Inconsistency

In lemma 4(a), a condition that reveals inconsistency was given. That condition is only good if there are two non-intersecting parallel half-spaces and one of them is the most violated. In such a case, the inconsistency is detected during the very first iteration. If the $x^{(0)}$ we are using does not make one of the parallel half-spaces the most violated, the inconsistency will not be detected using lemma 4(a). Lemma 4(a) will some times fail to detect the case where there are no parallel half-space and yet there is no common intersection of the half-spaces.

In order to prove our claims, let us present the 3 types of inconsistency in the language of propositions.

Type 1:

Two non-intersecting parallel half-spaces and one of them is the most violated.

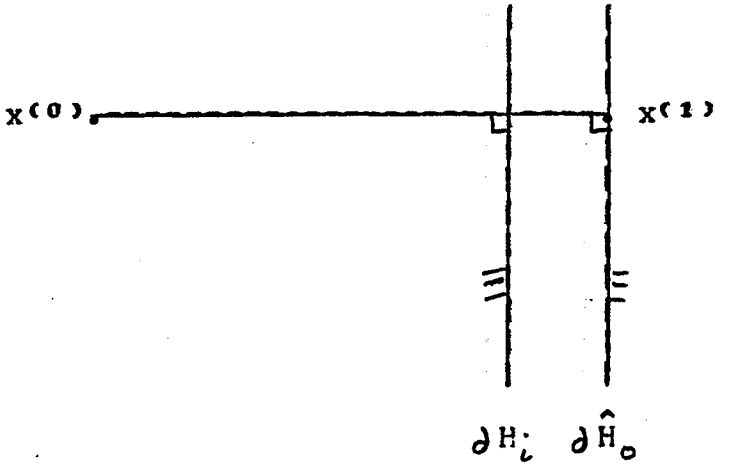


Fig. 3.5

Proposition 1

A type 1 inconsistency will be detected at the very first iteration using lemma 4(a). That is, $\exists i \in I$ such that

$$x^{(1)} \notin H_i \quad \text{and} \quad a^i = -\hat{a}_0$$

i.e. $d_i - r_0 \hat{a}_0 . a^i > 0$ and $\hat{a}_0 . a^i = -1$.

Proof

Type I ==> $\exists i \in I$ such that $\hat{H}_0 \cap H_i = \emptyset$,

==> $a^i = -\hat{a}_0$ and $x^{(1)} \notin H_i$ since $x^{(1)} \in \hat{H}_0$.

To choose the p_1 in order to construct

$$\hat{a}_1 = \hat{h}_{p_1} \hat{a}_0 + h_{p_1} a^{p_1},$$

we must, for all $i \in I$, inspect the distances $d_i - r_0 \hat{a}_0 . a^i$ for a maximum positive value and the dot-products $\hat{a}_0 . a^i$ for

the value of -1. Since these conditions exist for some $i \neq p_o$, they will be detected in the very first iteration.

$\therefore$ $\exists i \in I$ such that $d_i - r_o \hat{a}_o . a^i > 0$ and $\hat{a}_o . a^i = -1$.

Type 2:

Two non-intersecting parallel half-spaces and none of them is the most violated.

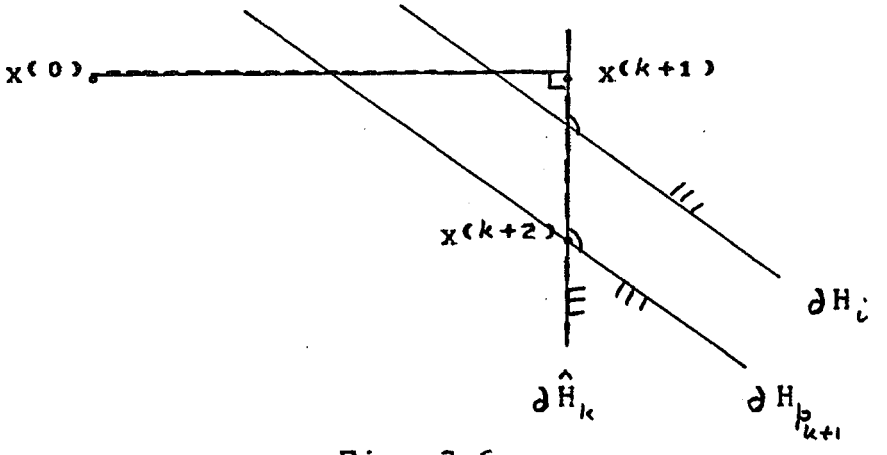


Fig. 3.6

Proposition 2

Once we miss a type 2 inconsistency at the first iteration, we miss it forever. In other words,

$$\hat{a}_o . a^i \neq -1 \Rightarrow \hat{a}_k . a^i \neq -1 \quad \forall k.$$

But $\exists k$ and $i \in I$ such that $x^{(k+2)} \notin H_i$ and $a^i = -a^{p_{k+1}}$.

Proof

$\hat{H}_o$ not a parallel half-spaces $\Rightarrow |\hat{a}_o . a^i| \neq 1 \quad \forall i \neq p_o$

$\Rightarrow \hat{a}_o . a^i \neq -1 \quad \forall i \in I.$

Therefore the construction of the first surrcgate will be done. Since this case is such that $S = \emptyset$, the iteration will continue satifying one constraint and violating

another. Some time in the iteration, one of the parallel hyperplanes will be chosen for the construction of

$$\hat{a}_{k+1} = \hat{h}_{p_{k+1}} \hat{a}_k + h_{p_{k+1}} a^{p_{k+1}}.$$

Since $H_{p_{k+1}}$ is parallel to and non-intersecting with H_i for some $i \in I$, $i \neq p_{k+1}$, we have

$a^{p_{k+1}} \cdot a^i = -1$ and $x^{(k+2)} \notin H_i$ since $x^{(k+2)} \in H_{p_{k+1}}$.

$\therefore a^i = -a^{p_{k+1}}$ by normalization.

$\therefore \hat{a}_{k+1} \cdot a^i = -\hat{a}_{k+1} \cdot a^{p_{k+1}}$

$= -(\hat{h}_{p_{k+1}} c_{p_{k+1}} + h_{p_{k+1}})$ by step 13

$= -\frac{d_{p_{k+1}}}{r_{k+1}}$ by steps 7, 11 and 12

$\neq -1$ since $d_i < r_{k+1}$ $\forall$ $i \in I$ and $\forall$ k.

$\therefore \hat{a}_0 \cdot a^i \neq -1 \Rightarrow \hat{a}_k \cdot a^i \neq -1$ $\forall$ k,

but $x^{(k+2)} \notin H_i$ and $a^i = -a^{p_{k+1}}$ for some k and $i \in I$.

Type 3:

There are no parallel half-spaces and $S = \emptyset$.

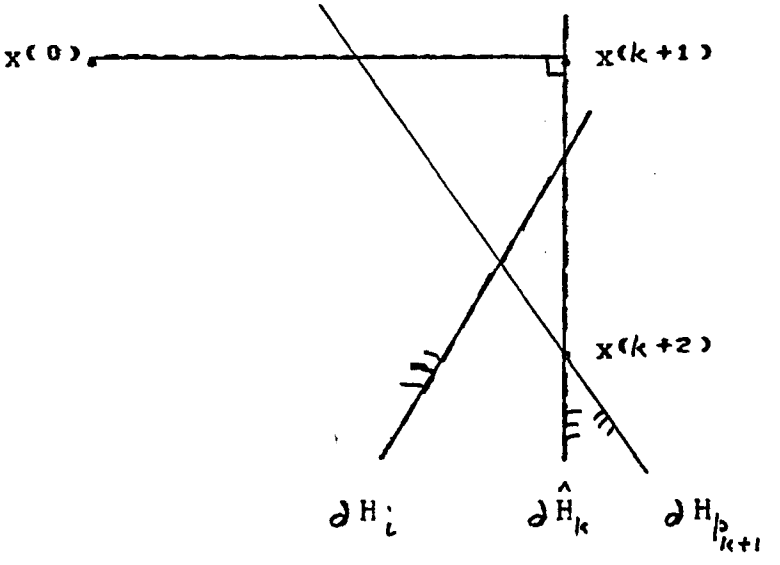


Fig. 3.7

Proposition 3

If a type 3 inconsistency exists, then

$\hat{a}_o \cdot a^i \neq -1$ and $\hat{a}_{k+1} \cdot a^i = -1$ depends on $x^{(0)}$.

Proof

No parallel half-spaces $\Rightarrow |\hat{a} \cdot a^i| \neq 1 \quad \forall i \neq p_o$

$\Rightarrow \hat{a}_o \cdot a^i \neq -1 \quad \forall i \in I.$

Suppose that $\exists k$ and $i \in I$ such that

$d_i - r_{k+1} \hat{a}_{k+1} \cdot a^i > 0$ and $\hat{a}_{k+1} \cdot a^i = -1$,

then $\hat{H}_{k+1}$ is parallel to and non-intersecting with H_i.

$\hat{a}_{k+1} \cdot a^i = -1 \Rightarrow a^i = -\hat{a}_{k+1}$ by normalization

$\Rightarrow a^i \cdot a^{p_{k+1}} = -\hat{a}_{k+1} \cdot a^{p_{k+1}}$

$= -(\hat{h}_{p_{k+1}} c_{p_{k+1}} + h_{p_{k+1}})$ by step 13

$= -\dfrac{d_{p_{k+1}}}{r_{k+1}}$ by steps 7,11 and 12

$= f(d_{p_{k+1}})$

$= f(x^{(0)})$

This shows that $a^i \cdot a^{p_{k+1}}$ is a function of the starting point $x^{(0)}$. But $\forall i,j \in I$, $a^i \cdot a^j$ is independent of $x^{(0)}$.

$\therefore \quad \hat{a}_{k+1} \cdot a^i = -1$ depends on $x^{(0)}$. .....(1)

The condition in (1) means that the surrogate hyperplane, $\partial \hat{H}_{k+1}$, is parallel to hyperplane ∂H_i and inconsistent with it. Whether we will get this condition depends on $x^{(0)}$.

Non-recursive Formulae

We now want to state and prove the general form of

lemma 4(a). To do this we shall need the non-recursive forms of r_k, $\hat{a}_k$ and $\hat{b}_k$. These forms are already defined in section 2.2 as eqs. (2.2.2) and (2.2.5), but we want to re-define them now as functions of k. From the recursive definition in step 13,

$$\hat{a}_{k+1} = \hat{h}_{p_{k+1}}(\hat{h}_{p_k}(\ldots(\hat{h}_{p_2}(\hat{h}_{p_1}a^{p_0} + h_{p_1}a^{p_1}) + h_{p_2}a^{p_2})\ldots) + h_{p_k}a^{p_k}) + h_{p_{k+1}}a^{p_{k+1}}$$

Let us define $u_i^{(k)}$ as the factor at the kth iteration by which any given constraint normal, a^i, has been multiplied to get the surrogate constraint normal, $\hat{a}_k$. We therefore have the following recursive function for the vector, $u^{(k)}$,

$$\begin{aligned} u^{(0)} &= e_{p_0}, \\ u^{(k+1)} &= \hat{h}_{p_{k+1}} u^{(k)} + h_{p_{k+1}} e_{p_{k+1}}. \end{aligned} \tag{3.9.1}$$

With this we can then re-write (1) above as

$$\hat{a}_k = \sum_{i=1}^{m} u_i^{(k)} a^i = A^T u^{(k)} \quad \forall\, k \tag{3.9.2}$$

Also to get the alternative definitions for $\hat{b}_k$ and r_k, we have

$$\begin{aligned} \hat{a}_k x^{(0)} &= \sum_{i=1}^{m} u_i^{(k)} a^i x^{(0)} \\ &= \sum_{i=1}^{m} u_i^{(k)} (d_i + b_i) \\ &= d^T u^{(k)} + b^T u^{(k)} \end{aligned}$$

$$\therefore \quad \hat{b}_k = b^T u^{(k)} \tag{3.9.3}$$

and

$$r_k = \hat{a}_k x^{(0)} - \hat{b}_k = d^T u^{(k)} \quad \forall\, k. \tag{3.9.4}$$

Note that for all k,

1) $u^{(k)} \geq 0$ with $u^{(k)} \neq \bar{0}$

2) $A^T u^{(k)}$ is a normalized vector (3.9.5)

and 3) $d^T u^{(k)} > 0$.

Theorem 2

$S = \emptyset$ if $\exists k$ and $i \in I$ such that $d_i - r_k \hat{a}_k \cdot a^i > 0$

and $\hat{a}_k \cdot a^i = -1$.

Proof

$d_i - r_k \hat{a}_k \cdot a^i > 0 \iff a^i x^{(0)} - b_i - r_k \hat{a}_k \cdot a^i > 0$ by def. 7

$\iff a^i (x^{(0)} - r_k \hat{a}_k) - b_i > 0$

$\iff a^i x^{(k+1)} - b_i > 0$ since $x^{(k+1)} = x^{(0)} - r_k \hat{a}_k$

$\iff x^{(k+1)} \notin S$.

Also $\hat{a}_k \cdot a^i = -1 \iff \hat{a}_k = -a^i$ .....(2) since $a^i \cdot a^i = 1$.

Now consider an arbitrary $v \in H_i$.

$v \in H_i \iff a^i v - b_i \leq 0$

$\iff a^i v + d_i - a^i x^{(0)} \leq 0$ by def. 7

$\implies a^i v + r_k \hat{a}_k \cdot a^i - a^i x^{(0)} < 0$ since $d_i > r_k \hat{a}_k \cdot a^i$

$\iff a^i (v + r_k \hat{a}_k - x^{(0)}) < 0$

$\iff -\hat{a}_k (v + r_k \hat{a}_k - x^{(0)}) < 0$ by (2)

$\iff \hat{a}_k v + r_k - \hat{a}_k x^{(0)} > 0$

$\iff \hat{a}_k v - \hat{b}_k > 0$ by eq. (3.9.4)

$\iff (A^T u^{(k)})^T v - b^T u^{(k)} > 0$ by eq. (3.9.2) and (3.9.3)

$\iff \sum_{j=1}^{m} u_j^{(k)} (a^j v - b_j) > 0$

$\implies \exists t \in I$ such that $a^t v - b_t > 0$

$\iff v \notin H_t$ for some $t \in I$.

$\therefore S = \emptyset$ if $\exists k$ and $i \in I$ such that $d_i - r_k \hat{a}_k \cdot a^i > 0$

and $\hat{a}_k \cdot a^i = -1$.

3.10 Narrow Cone

Let us examine the behaviour of the sequence, $\{x^{(k)}\}$ in a system with a very small angle.

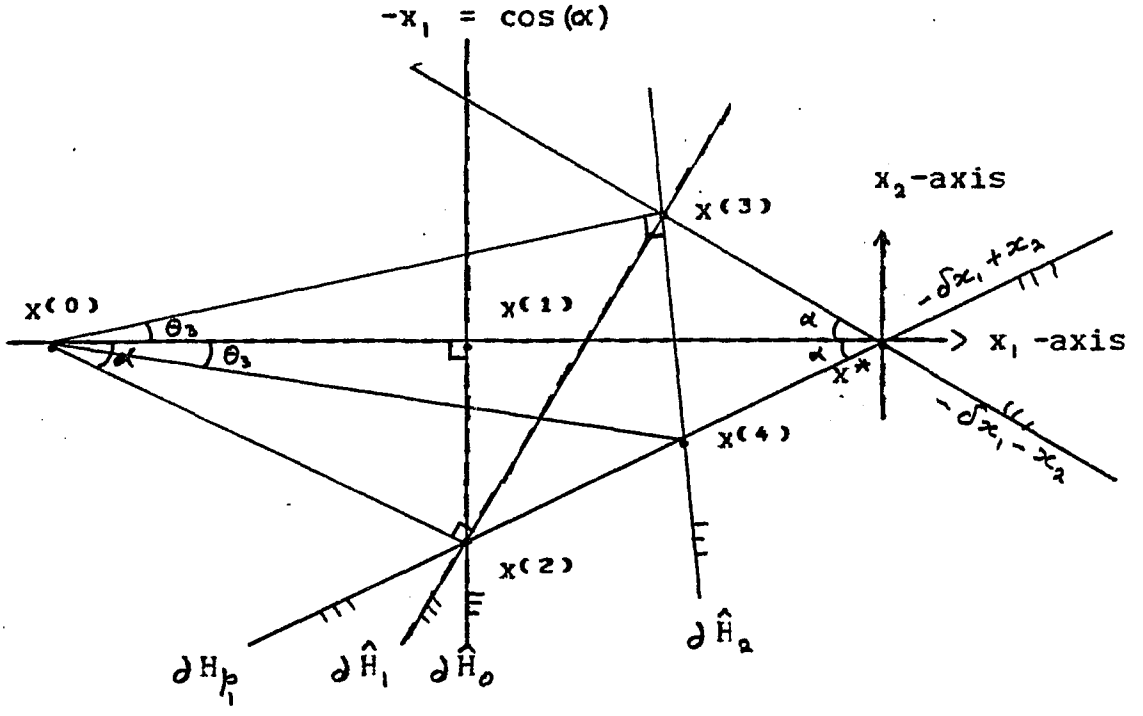


Fig. 3.8

Suppose we have the following 3-constraint system in a 2-D space.

$$\begin{pmatrix} -\delta & 1 \\ -\delta & -1 \\ -1 & 0 \end{pmatrix} x \leq \begin{pmatrix} 0 \\ 0 \\ \cos(\alpha) \end{pmatrix}$$

where $\delta > 0$ is very small such that

$$\tan(\alpha) = \delta < 0.5.$$

$x^* = (0,0)^T$ is the vertex of the feasible cone.

Let the initial guess be

$$x^{(0)} = (-2\cos(\alpha), 0)^T .$$

It is easy to see that

$$x^{(1)} = (-\cos(\alpha), 0)^T$$

and $x^{(2)} = (-\cos(\alpha), -\sin(\alpha))^T$.

Let t_j be the angle defined by the line $x^{(j)}x^{(j+1)}$ and the hyperplane on which $x^{(j+1)}$ lies,

and θ_j be that defined by the line $x^{(0)}x^{(j)}$ and the x_1-axis.

Consider the triangle formed by the points $x^{(2)}, x^{(3)}$ and x^*.

$$\frac{||x^{(3)}||}{\sin(2\alpha + t_2)} = \frac{||x^{(2)}||}{\sin(t_2)} \qquad \text{by the sine rule}$$

$$= \frac{1}{\sin(t_2)}$$

$$\therefore \quad ||x^{(3)}|| = \frac{\sin(2\alpha + t_2)}{\sin(t_2)}$$

But $x^{(3)} = (-||x^{(3)}||\cos(\alpha), ||x^{(3)}||\sin(\alpha))^T$.

$$\therefore \quad x^{(3)} = \frac{\sin(2\alpha + t_2)}{\sin(t_2)}(-\cos(\alpha), \sin(\alpha))^T.$$

Similarly, from the triangle formed by $x^{(3)}$, $x^{(4)}$ and x^* we have

$$||x^{(4)}|| = \frac{\sin(2\alpha + t_3)}{\sin(t_3)}||x^{(3)}||$$

$$= \frac{\sin(2\alpha + t_3)}{\sin(t_3)} \frac{\sin(2\alpha + t_2)}{\sin(t_2)}$$

$$\therefore \quad x^{(4)} = \frac{\sin(2\alpha + t_3)}{\sin(t_3)} \frac{\sin(2\alpha + t_2)}{\sin(t_2)}(-\cos(\alpha), -\sin(\alpha))^T$$

Continuing thus, we have

$$x^{(k+1)} = \prod_{j=2}^{k} \frac{\sin(2\alpha + t_j)}{\sin(t_j)}(-\cos(\alpha), (-1)^k \sin(\alpha)) \qquad \ldots\ldots(1)$$

To examine the factor $\frac{\sin(2\alpha + t_j)}{\sin(t_j)}$ further, we have to give more information about the angles on the diagram.

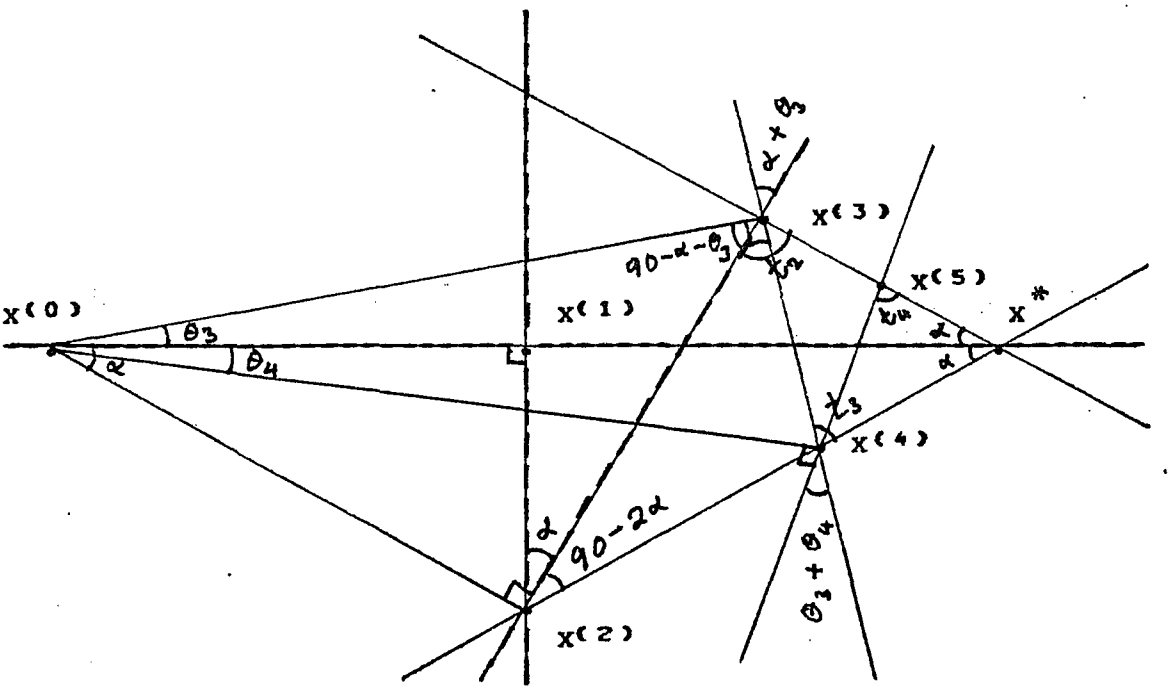


Fig. 3.9

From the triangle formed by the points $x^{(2)}, x^{(3)}, x^*$, we have

$t_2 + (90^\circ - 2\alpha) + 2\alpha = 180^\circ$, sum of interior $\angle$ s of a Δ

$\therefore \quad t_2 = 90^\circ$.

From the triangle formed by $x^{(2)}, x^{(3)}, x^{(4)}$, we have

$t_3 = (90^\circ - 2\alpha) + (\alpha + \theta_3)$, exterior $\angle$. of the triangle.

$\therefore \quad t_3 = 90^\circ - \alpha + \theta_3$. .....(2)

From the triangle formed by $x^{(4)}, x^{(5)}, x^*$, we have

$t_4 + 2x + (t_3 - \theta_3 - \theta_4) = 180^\circ$, sum of int. $\angle$ s of Δ.

i.e. $t_4 + 2\alpha + (90^\circ - \alpha - \theta_4) = 180^\circ$ by (2)

$\therefore \quad t_4 = 90^\circ - \alpha + \theta_4$.

We can now conclude that

$$t_j = 90^\circ - \alpha + \theta_j \qquad \forall\, j \qquad \ldots\ldots\ldots (3)$$

Note that for $j = 2$, $\theta_2 = \alpha$.

$\therefore \quad t_2 = 90^\circ - \alpha + \theta$

$= 90^\circ$ as found above.

Also for $j = 1$, $\theta_1 = 0$.

$\therefore \quad t_1 = 90^\circ - \alpha$ as can be seen in the diagram.

$$\text{Now} \quad \frac{\sin(2\alpha + t_j)}{\sin(t_j)} = \frac{\sin(90^\circ + \alpha + \theta_j)}{\sin(90^\circ - \alpha + \theta_j)} \qquad \text{by (3)}$$

$$= \frac{\cos(\alpha + \theta_j)}{\cos(\alpha - \theta_j)} \qquad \ldots\ldots\ldots (4)$$

Since α is very small and $\alpha \geq \theta_j \quad \forall\, j$, we have

$$90^\circ > 2\alpha \geq \alpha + \theta_j > \alpha - \theta_j \geq 0.$$

$$\therefore \quad 0 < \cos(2\alpha) \leq \cos(\alpha + \theta_j) < \cos(\alpha - \theta_j) \leq 1.$$

$$\therefore \quad 0 < \cos(2\alpha) \leq \frac{\cos(2\alpha)}{\cos(\alpha - \theta_j)} \leq \frac{\cos(\alpha + \theta_j)}{\cos(\alpha - \theta_j)} < 1 \qquad \ldots (5)$$

Equality holds only for $j = 2$. From (1), (4) and (5) we have

$$(\cos(2\alpha))^{k-1} \leq \|x^{(k+1)}\| = \prod_{j=2}^{k} \frac{\cos(\alpha + \theta_j)}{\cos(\alpha - \theta_j)} < 1 \qquad \ldots\ldots (6)$$

where $0 < \dfrac{\cos(\alpha + \theta_j)}{\cos(\alpha - \theta_j)} < 1$ is the error decaying factor.

The lower bound in (6) shows that convergence can be extremely slow if α is very small and we are oscillating between the two hyperplanes with angle $\leq 2\alpha$.

4. RE-INITIALIZING SURROGATES

We saw in the last section that even when convergence seems obvious, the prccess may be very slow because we are trapped in a narrow ccne. The reason for this is that $x^{(0)}$ is fixed. If we can renew $x^{(0)}$ at certain pcints during iterations, we will avoid oscillating tco long in any cone.

Consider fig. 3.8 once more. If $x^{(0)}$ is re-initialized after a repeat use of any constraint, x^* wculd be the immediate successor to $x^{(4)}$. In other words, the feasible vertex would be reached after 4 iterations. On the cther hand, re-initializing at each iteration would have x^* as a successcr to $x^{(2)}$. (Ncte that $x^{(2)}$, not $x^{(1)}$, is the first iterate). A third technique we would like to consider is renewing $x^{(0)}$ at every other iteration. A partial reason for this is that in the seccnd technique, each iteratien is a first-like iteratien and so will not detect non-parallel type of inconsistency.

We shall call the first type Surrcgate-R. 'R' stands for repeat choice of constraint. The second type is named Surrogate-II since it uses two given constraints at each iteration, and the third, Surrogate-III. During any re-initialization, r_{k+1} and $\hat{a}_{k+1}$ are not computed because

$$x^{(k+2)} = x^{(0)} - r_{k+1}\hat{a}_{k+1}$$
$$= x^{(0)} - r_{k+1}(\hat{h}_{l_{k+1}}\hat{a}_k + h_{l_{k+1}}a^{h_{k+1}})$$

$$= x^{(0)} - \left(\frac{\hat{g}_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}}\hat{a}_k + \frac{g_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}}a^{p_{k+1}}\right).$$

This means that after a choice of $H_{p_{k+1}}$ is made and $x^{(0)}$ qualifies for re-initialization, we should then set

$$h_{p_{k+1}} = \lambda \frac{g_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}}, \qquad \ldots\ldots\ldots(4.0.1)$$

$$h = \frac{g_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}} - (\lambda - 1)\, c_{p_{k+1}} \frac{g_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}} \qquad \ldots\ldots\ldots(4.0.2)$$

$$x^{(0)} = x^{(0)} - (\hat{h}_{p_{k+1}} \hat{a}_k + h_{p_{k+1}} a^{p_{k+1}}) \qquad \ldots\ldots\ldots(4.0.3)$$

The new distances are then updated using

$$d = d - (\hat{h}_{p_{k+1}} c_{k+1} + h_{p_{k+1}} A (a^{p_{k+1}})^T) \qquad \ldots\ldots\ldots(4.0.4)$$

With eqs. (4.0.1) to (4.0.4), we now present these modifications of algorithm-I.

Surrogate-R Algorithm

1) Choose $x^{(0)} \in R^n$ arbitrarily, set $k = 0$;

2) $d = Ax^{(0)} - b$;

3) Choose $p_k \in I$ such that $d_{p_k} \geq d_i \quad \forall\, i \in I$;

4) If $d_{p_k} \leq 0$, then $x^{(0)} \in S$, stop;

5) $r_k = d_{p_k}$, $\hat{a}_k = a^{p_k}$, hyp(i) = '0'B $\quad \forall\, i \in I$;

6) $c_{k+1} = A\hat{a}_k^T$;

7) Choose $p_{k+1} \in I$ such that $g^2_{p_{k+1}} = \max_{i \in I, g_i > 0} g_i^2$

where $g_i^2 = \dfrac{(d_i - r_k c_{k+1,i})^2}{1 - c^2_{k+1,i}} \quad \forall\, i \in I$;

8) If $\exists i \in I$ and k such that $c_{k+1,i} = -1$ and $g_i > 0$,

then stop with no solution;

9) If $d_{p_{k+1}} - r_k c_{p_{k+1}} \leq \delta$, stop with $x^{(k+1)} = x^{(0)} - r_k \hat{a}_k$ as a solution;

9b) If (hyp(p_{k+1})), go to 11';

9c) hyp(p_{k+1}) = '1'B;

10) $r_{k+1} = \sqrt{r_k^2 + \lambda^2 g_{p_{k+1}}^2}$;

11) $h_{p_{k+1}} = \lambda \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{r_{k+1}(1 - c_{p_{k+1}}^2)}$;

12) $\hat{h}_{p_{k+1}} = \dfrac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{r_{k+1}(1 - c_{p_{k+1}}^2)} - (\lambda - 1) c_{p_{k+1}} \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{r_{k+1}(1 - c_{p_{k+1}}^2)}$;

13) $\hat{a}_{k+1} = \hat{h}_{p_{k+1}} \hat{a}_k + h_{p_{k+1}} a^{p_{k+1}}$;

14) $c_{k+2} = \hat{h}_{p_{k+1}} c_{k+1} + h_{p_{k+1}} A (a^{p_{k+1}})^T$;

15) k = k + 1 and go to 7;

11') $h_{p_{k+1}} = \lambda \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{1 - c_{p_{k+1}}^2}$;

12') $\hat{h}_{p_{k+1}} = \dfrac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{1 - c_{p_{k+1}}^2} - (\lambda - 1) c_{p_{k+1}} \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{1 - c_{p_{k+1}}^2}$;

13') $x^{(0)} = x^{(0)} - (\hat{h}_{p_{k+1}} \hat{a}_k + h_{p_{k+1}} a^{p_{k+1}})$;

14') $d = d - (\hat{h}_{p_{k+1}} c_{k+1} + h_{p_{k+1}} A (a^{p_{k+1}})^T)$;

15') k = k + 1 and go to 3;

Surrogate-II Algorithm

1) Choose $x^{(0)} \in R^n$ arbitrarily, set k = 0 ;

2) d = $Ax^{(0)} - b$;

3) Choose $p_k \in I$ such that $d_{p_k} \geq d_i \quad \forall\, i \in I$;

4) If $d_{p_k} \leq 0$, then $x^{(0)} \in S$, stop;

5) $c_{k+1} = A(a^{p_k})^T$; $r_k = d_{p_k}$;

6) Choose $p_{k+1} \in I$ such that $g^2_{p_{k+1}} = \max_{i \in I, g_i > 0} g_i^2$

where $g_i^2 = \dfrac{(d_i - r_k c_{k+1,i})^2}{1 - c^2_{k+1,i}}$ $\forall\, i \in I$;

7) If $\exists i \in I$ and k such that $c_{k+1,i} = -1$ and $g_i > 0$, then stop with no solution;

8) If $d_{p_{k+1}} - r_k c_{p_{k+1}} \leq \delta$, stop with $x^{(k+1)} = x^{(0)} - r_k a^{p_k}$ as a solution;

9) $h_{p_{k+1}} = \lambda \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{1 - c^2_{p_{k+1}}}$;

10) $\hat{h}_{p_{k+1}} = \dfrac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{1 - c^2_{p_{k+1}}} - (\lambda - 1) c_{p_{k+1}} \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{1 - c^2_{p_{k+1}}}$;

11) $x^{(0)} = x^{(0)} - (\hat{h}_{p_{k+1}} a^{p_k} + h_{p_{k+1}} a^{p_{k+1}})$;

12) $d = d - (\hat{h}_{p_{k+1}} c_{k+1} + h_{p_{k+1}} A(a^{p_{k+1}})^T)$;

13) $k = k + 1$ and go to 3;

Surrogate-III Algorithm

1) Choose $x^{(0)} \in R^n$ arbitrarily, set $k = 0$;

2) $d = Ax^{(0)} - b$;

3) Choose $p_k \subset I$ such that $d_{p_k} \geq d_i$ $\forall\, i \in I$;

4) If $d_{p_k} \leq 0$, then $x^{(0)} \in S$, stop;

5) $r_k = d_{p_k}$, $\hat{a}_k = a^{p_k}$;

6) $c_{k+1} = A\hat{a}_k^T$;

7) Choose $p_{k+1} \in I$ such that $g^2_{p_{k+1}} = \max_{i \in I, g_i > 0} g_i^2$

where $g_i^2 = \dfrac{(d_i - r_k c_{k+1,i})^2}{1 - c^2_{k+1,i}}$ $\forall\, i \in I$;

8) If $\exists i \in I$ and k such that $c_{k+1,i} = -1$ and $g_i > 0$, then stop with no solution;

9) If $d_{p_{k+1}} - r_k c_{p_{k+1}} \leq \delta$, stop with $x^{(k+1)} = x^{(0)} - r_k \hat{a}_k$ as a solution;

10) $r_{k+1} = \sqrt{r_k^2 + \lambda^2 g_{p_{k+1}}^2}$;

11) $h_{p_{k+1}} = \lambda \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{r_{k+1}(1 - c_{p_{k+1}}^2)}$;

12) $\hat{h}_{p_{k+1}} = \dfrac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{r_{k+1}(1 - c_{p_{k+1}}^2)} - (\lambda - 1) c_{p_{k+1}} \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{r_{k+1}(1 - c_{p_{k+1}}^2)}$;

13) $\hat{a}_{k+1} = \hat{h}_{p_{k+1}} \hat{a}_k + h_{p_{k+1}} a^{p_{k+1}}$;

14) $c_{k+2} = \hat{h}_{p_{k+1}} c_{k+1} + h_{p_{k+1}} A (a^{p_{k+1}})^T$;

15) Choose $p_{k+2} \in I$ such that $g_{p_{k+2}}^2 = \max_{i \in I, g_i > 0} g_i^2$

16) If $\exists i \in I$ and k such that $c_{k+2,i} = -1$ and $g_i > 0$,

17) If $d_{p_{k+2}} - r_{k+1} c_{p_{k+2}} \leq \delta$, stop with $x^{(k+2)} = x^{(0)} - r_{k+1} \hat{a}_{k+1}$ as a solution;

18) $h_{p_{k+2}} = \lambda \dfrac{d_{p_{k+2}} - r_{k+1} c_{p_{k+2}}}{1 - c_{p_{k+2}}^2}$;

19) $\hat{h}_{p_{k+2}} = \dfrac{r_{k+1} - d_{p_{k+2}} c_{p_{k+2}}}{1 - c_{p_{k+2}}^2} - (\lambda - 1) c_{p_{k+2}} \dfrac{d_{p_{k+2}} - r_{k+1} c_{p_{k+2}}}{1 - c_{p_{k+2}}^2}$;

20) $x^{(0)} = x^{(0)} - (\hat{h}_{p_{k+2}} \hat{a}_{k+1} + h_{p_{k+2}} a^{p_{k+2}})$;

21) $d = d - (\hat{h}_{p_{k+2}} c_{k+2} + h_{p_{k+2}} A (a^{p_{k+2}})^T)$;

22) $k = k + 1$ and go to 3;

5. CONVERGENCE

5.1 Proofs of Convergence

Theorem 1 is the condition for successful termination of the algorithm, and Theorem 2 for aborting it if inconsistency exists. The question now is, if the system is consistent, will we always get the condition of Theorem 1 no matter where we start? This question is answered with 3 theorems labelled Theorem 3A, 3B, 3C.

Theorem 3A is the convergence of Surrogate-II which resembles Agmon's method. Theorem 3B is the convergence of the pure surrogate algorithm, Surrogate-I, for the Narrow Cone problem. This is done with an arbitrary $x^{(0)}$ to dispel the fear that being narrow, convergence may occur outside S. Theorem 3C is a general theorem for the convergence of Surrogate-I for any finite m constraints and n variables.

Lemma 6

Let i) i_s = index of a surrogate constraint,

ii) a polyhedral cone V be defined by the homogeneous inequalities

$$a^i w \leq 0 \qquad \forall\, i \in I,$$

iii) E be the set of points $x \notin V$ and such that the origin is the point on ∂V nearest to x,

iv) $J(x) = \{i \in I \mid a^i x > 0\}$,

v) $\hat{J}(x) = \{i \in \hat{I} \mid a^i x > 0\}$, where $\hat{I} = I \cup \{i_s\}$,

vi) $f_j(x)$ = distance of $x \in E$ from ∂H_j $\forall\, j \in \hat{J}$,

vii) $\hat{I}_{y^*} = \{i \in \hat{I} \mid a^i y^* = b_i\}$.

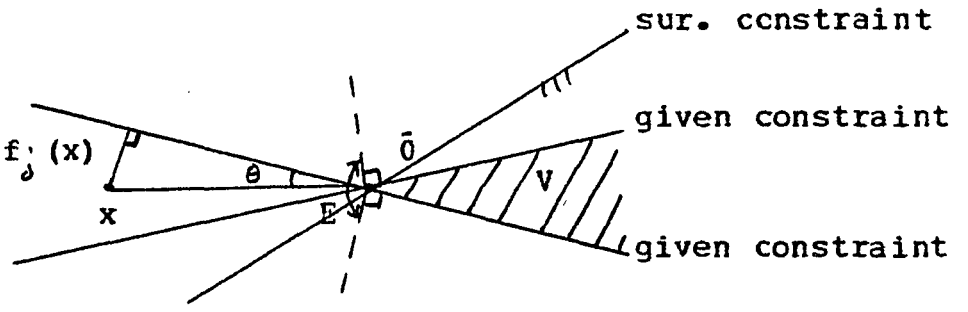


Fig. 5.1 Surrogate Cone at the Origin

Then $\exists\ \lambda_s > 0$ such that

$$\inf_{x \in E} \max_{j \in \hat{J}(x)} \frac{f_j(x)}{\|x\|} = \lambda_s ,$$

and $\lambda_s \geq \lambda_A$, where λ_A is that cf Agmon's.

(Note: λ_s is the smallest sine of the angle defined by ∂H_j and the line joining x to the origin.)

Prcof

(Remark: (1) The proof of the first part is basically the same as that of lemma 3.2.3.

(2) Although the surrogate constraint is superflucus it is an active constraint, and so lemma 3.2.2 still applies.)

Consider the polyhedral cone S_{y^*} of lemma 3.2.2.

$$z \in S_{y^*} \Rightarrow \|x - y^*\| < \|x - z\|, \quad z \neq y^*$$
$$= \|(x - y^*) - (z - y^*)\| \quad \ldots(1)$$

Now consider the vectcrs $(x - y^*)$ and $(z - y^*)$.

$\forall\ i \in \hat{I}_{y^*}$, $a^i(x - y^*) = a^i x - b_i$ since $a^i y^* = b_i$

> 0 since $x \notin S$.

$\therefore$ $(x - y^*) \notin V$.

$\forall\, i \in \hat{I}_{y^*}$, $a^i(z - y^*) = a^i z - b_i$ since $a^i y^* = b_i$

≤ 0 since $z \in S_{y^*}$.

$\therefore$ $(z - y^*) \in V$.

Since we can choose z in (1) above to be on ∂S_{y^*}, $(z - y^*)$ is on ∂V. By (1), $(x - y^*)$ is closer to $\bar{0} \in \partial V$ than is $(z - y^*)$.

$\therefore$ $(x - y^*) \in E$ by (iii).

$\forall\, j \in \hat{J}$, $f_j(x - y^*) = a^j(x - y^*) > 0$.

$$\therefore \quad \frac{f_j(x - y^*)}{\|x - y^*\|} > 0 \qquad \forall\, j \in \hat{J}.$$

But $\dfrac{f_j(x - y^*)}{\|x - y^*\|}$ = the sine of the angle defined by ∂H_j and the line joining $(x - y^*)$ to the origin.

Let α_{ij} be the angle between ∂H_i and ∂H_j.

Then $\max\limits_{i,j \in \hat{J}} \sin(\alpha_{ij}/2) = \lambda_s(V)$.

Since $\hat{J} \supset J$, $\max\limits_{j \in \hat{J}} \dfrac{f_j(x - y^*)}{\|x - y^*\|} \geq \max\limits_{j \in J} \dfrac{f_j(x - y^*)}{\|x - y^*\|}$

for any x.

$\therefore$ $\lambda_s \geq \lambda_A$.

Theorem 3A

Let there be no superfluous constraint in the given problem. If $S \neq \emptyset$ and $\{x^{(k)}\}$ is the sequence of iterations as define by Surrogate-2, then

$$\|x^{(k+1)} - y^{(k+1)}\| < \|x^{(k)} - y^{(k)}\| \qquad \forall\, k.$$

Also, $\exists \theta = \theta(A)$ such that for all k,

$$||x^{(k)} - y^{(k)}|| \leq \theta^k ||x^{(0)} - y^{(0)}||, \quad \text{where} \quad 0 < \theta < 1.$$

Consequently, $x^{(k)} \dashrightarrow y^{(k)} \in S$ as $k \dashrightarrow \infty$.

Proof

By lemma 2

$x^{(k+1)}$ = the orthogonal projection of $x^{(k)}$ onto $\partial \hat{H}_{i_k} \bigcap \partial H_{i_{k+1}}$.

$||x^{(k+1)} - y^{(k+1)}|| \leq ||x^{(k+1)} - y^{(k)}||$ by definition

$||x^{(k+1)} - y^{(k)}|| < ||x^{(k)} - y^{(k)}||$ by lemma 3.2.1a using the sur. half-space

$\therefore ||x^{(k+1)} - y^{(k+1)}|| < ||x^{(k)} - y^{(k)}|| \quad \forall k.$

Let $\hat{H}_{k+1,i}$ = the surrogate at the intersection of $\hat{H}_k$ and H_i,

α_{si} = the angle defined by $\partial \hat{H}_{k+1,i}$ and ∂H_i.

$$\lambda_{smin} = \min_{i \in I} \sin(\alpha_{si}/2) \quad \text{and} \quad \theta = \sqrt{1 - \lambda_{smin}^2}$$

$$||x^{(k+1)} - x^{(k)}|| = \max_{i \in I} d(x^{(k)}, \partial H_{k+1,i})$$

$$\geq \lambda_s ||x^{(k)} - y^{(k)}|| \quad \text{by lemma 6}$$

$$\geq \lambda_{smin} ||x^{(k)} - y^{(k)}|| \quad \ldots\ldots\ldots\ldots (2)$$

$$||x^{(k+1)} - y^{(k)}||^2 \leq ||x^{(k)} - y^{(k)}||^2 - ||x^{(k+1)} - x^{(k)}||^2$$

by lemma 3.2.1(b) using sur. hyperplane

$$\leq ||x^{(k)} - y^{(k)}||^2 - \lambda_{smin}^2 ||x^{(k)} - y^{(k)}||^2$$

by (2) above

$$= \theta^2 ||x^{(k)} - y^{(k)}||^2$$

Therefore

$$||x^{(k+1)} - y^{(k)}|| \leq \theta ||x^{(k)} - y^{(k)}|| \quad \ldots\ldots\ldots\ldots (3)$$

But

$||x^{(k+1)} - y^{(k+1)}|| \leq ||x^{(k+1)} - y^{(k)}||$ by definition

Therefore

$||x^{(k+1)} - y^{(k+1)}|| \leq \theta ||x^{(k)} - y^{(k)}||$ by (3) above

$$\leq \theta^2 ||x^{(k-1)} - y^{(k-1)}||$$

$$\vdots$$

$$\leq \theta^{k+1} ||x^{(0)} - y^{(0)}||$$

$\therefore \; ||x^{(k)} - y^{(k)}|| \leq \theta^k ||x^{(0)} - y^{(0)}||$

and so, $x^{(k)} \dashrightarrow y^{(k)} \in S$ as $k \dashrightarrow \infty$.

Remark: Let us denote the θ in Theorem 3A by θ_{II}, and that of Agmon's by θ_A. We showed in lemma 6 that $\lambda_s \geq \lambda_A$. Therefore $\theta_{II} \leq \theta_A$. This implies that Surrogate-II is at least as fast as Agmon. We note, however that Surrogate-II does more work per iteration than Agmon. We shall later show the work done per iteration by each method.

For the pure surrogate convergence proofs, we need the following three lemmas.

Lemma 7 (Isaacson and Keller)

For any vector, x, the norm, $||x||$, is a continuous function.

Proof

$$||x + \Delta x|| \leq ||x|| + ||\Delta x||$$

$$\therefore \; ||x + \Delta x|| - ||x|| \leq ||\Delta x|| \qquad \ldots\ldots\ldots(4)$$

Also $||x|| = ||x + \Delta x - \Delta x||$

$$\leq ||x + \Delta x|| + ||\Delta x||$$

$$\therefore \; -||\Delta x|| \leq ||x + \Delta x|| - ||x|| \qquad \ldots\ldots\ldots(5)$$

Now for any $\epsilon > 0$ and all Δx, with $||\Delta x||_\infty \leq \epsilon / n$, we have

$| \; ||x + \Delta x|| - ||x|| \; | \leq ||\Delta x||$ by (4) and (5)

$$\leq \sum_{j=1}^{n} (|\Delta x_j|) ||e_j||$$

$$\leq \max_j |\Delta x_j| \sum_{j=1}^{n} ||e_j||$$

$$= n||\Delta x||_\infty$$

$$\leq \epsilon$$

$\therefore$ The norm of any vector is a continuous function.

Lemma 8

If 1) $f(x^{(k)})$ is continuous

2) $f(x^{(k)}) \geq 0 \quad \forall\, k$

3) $\Delta f(x^{(k)}) \equiv f(x^{(k+1)}) - f(x^{(k)}) = 0 \Longrightarrow f(x^{(k)}) = 0$

then if 4) $f(x^{(k+1)}) < f(x^{(k)}) \quad \forall\, k$,

then $f(x^{(k)}) \to 0$ as $k \to \infty$.

Proof

(2) and (4) $\Longrightarrow \lim_{k\to\infty} (f(x^{(k+1)}) - f(x^{(k)})) = 0$

(1) $\Longrightarrow \Delta f(x^{(k)})$ is continuous which together with the above and (3)

$\Longrightarrow \lim_{k\to\infty} f(x^{(k)}) = 0.$

Lemma 9 For all k

a) $x^{(k+1)}, x^{(k+2)} \in \partial \hat{H}_k$. b) $x^{(t)} \notin \hat{H}_k \quad \forall\, t \leq k$.

Proof of a)

By contruction $x^{(k+1)}$ and $x^{(k+2)}$ are the orthogonal projections of $x^{(0)}$ onto $\partial \hat{H}_k$ and $\partial \hat{H}_k \bigcap \partial H_{p_{k+1}} \subset \partial \hat{H}_{k+1}$ respectively.

$$\therefore \quad x^{(k+1)}, x^{(k+2)} \in \partial \hat{H}_k .$$

Proof of (b)

Suppose $\exists t \leq k$ such that $x^{(t)} \in \hat{H}_k$, then let $\bar{x}$ be the point where $\partial \hat{H}_k$ cuts the line joining $x^{(0)}$ to $x^{(t)}$.

$$\therefore \quad ||x^{(0)} - \bar{x}|| \leq ||x^{(0)} - x^{(t)}|| \qquad \ldots\ldots\ldots (6)$$

$\bar{x}, x^{(k+1)} \in \partial \hat{H}_k$ with $x^{(k+1)}$ as the orthogonal projection of $x^{(0)}$ onto $\partial \hat{H}_k$ means that

$$\begin{aligned} ||x^{(0)} - \bar{x}|| &\geq ||x^{(0)} - x^{(k+1)}|| \\ &= r_k \\ &> r_{t-1} \quad \text{since } t \leq k \\ &= ||x^{(0)} - x^{(t)}||. \end{aligned}$$

$$\therefore \quad ||x^{(0)} - \bar{x}|| > ||x^{(0)} - x^{(t)}|. \qquad \ldots\ldots\ldots (7)$$

But (6) and (7) contradict each other.

$$\therefore \quad x^{(t)} \notin \hat{H}_k \qquad \forall\, t \leq k.$$

In lemma 8 we assumed the 4 conditions used there. In the theorem which follows, we shall prove that for all k, the subsequences $\{x^{(2k)}\}$ and $\{x^{(2k+1)}\}$ (of even and odd iterates), satisfy condtions (3) and (4), where $f(x^{(k)})$ is defined as the distance of $x^{(k)}$ to a sclution.

Theorem 3B

If $S \neq \emptyset$ and $\{x^{(k)}\}$ is the sequence of the iterates as defined by the pure surrogate algorithm, Surrogate-I, then for the 3-constraint 2-D prcblem in the Narrow Cone section,

a) $f_{k+2} < f_k \quad \forall\, k$, where $f_k = ||x^{(k)} - x^*||$.

Also b) $\Delta f_k \equiv f_{k+2} - f_k = 0$ iff $x^{(k)} \in S$.

Consequently, c) $x^{(k)} \dashrightarrow x^* \in S$ as $k \dashrightarrow \infty$.

Proof

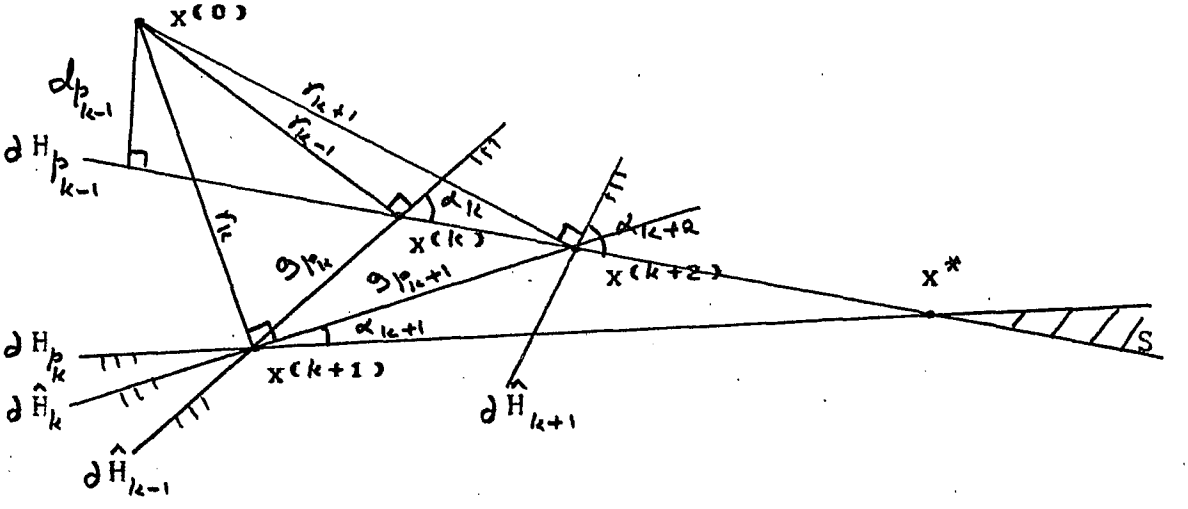


Fig. 5.2. The 3-Constraint 2-D Narrow Cone.

Let i) $x^{(0)}$ be an arbitrary point,

ii) α_k = the acute angle defined by $\partial \hat{H}_{k-1}$ and the given hyperplane on which $x^{(k)}$ lies.

Proof of a)

For all $k \geq 1$, $x^{(k)}, x^{(k+2)}$ and x^* are colinear. .....(8)

For all k, $x^{(k)} \notin \hat{H}_{k+1}$ by lemma 9(b),

$x^{(k+2)} \in \partial H_{k+1}$ by lemma 9(a),

$x^* \in \hat{H}_{k+1}$ since $x^* \in S$.

$\therefore$ $x^{(k+2)}$ lies between $x^{(k)}$ and x^*. .....(9)

$\therefore$ $||x^{(k+2)} - x^*|| < ||x^{(k)} - x^*||$, by (8) and (9).

i.e. $f_{k+2} < f_k \quad \forall\, k$ if $x^{(k)} \notin S$.

Proof of b)

If $f_k = f_{k+2}$ then $x^{(k)} = x^{(k+2)}$.

$x^{(k)} = x^{(k+2)} \Longrightarrow r_{k-1} = r_{k+1}$ and $\alpha_k = \alpha_{k+2}$

$\Longleftrightarrow r_{k-1}^2 = r_{k+1}^2 = g_{p_{k+1}}^2 + g_{p_k}^2 + r_{k-1}^2$

$\Longleftrightarrow g_{p_{k+1}} = g_{p_k} = 0$

$\Longleftrightarrow x^{(k+2)} = x^{(k+1)} = x^{(k)} = x^* \in S$ by Theorem 1

Therefore condition (b) holds.

Proof of c)

By lemma 7, f_k is a continuous function of $x^{(k)}$, and clearly $f_k \geq 0$. Consequently by lemma 8, part (c) of the theorem follows,

i.e. $x^{(k)} \longrightarrow x^* \in S$ as $k \longrightarrow \infty$.

The arguments of Theorem 3B can be extended to a general case for m-constraint n-variable problem. For this type of problem there may be infinite number of iterates. But then in every m+1 iterations, at least 2 iterates must lie on the same hyperplane. Consequently there exists a subsequence of iterates all of which lie on one hyperplane. It is this subsequence that we shall consider in the theorem that follows.

Theorem 3C

Let there be no superfluous constraint in the given problem. If $S \neq \emptyset$ and $\{x^{(k)}\}$ is the sequence of the iterates as defined by the pure surrogate, Surrogate-I,

then for any finite m-constraint n-D problem,

1) there exists a subsequence $\{x^{(k_j)}\}$ of iterates all of which lie on the same hyperplane ∂H_i for some $i \in I$;

2) let $y^{(k_j)}$ be the nearest sclution to $x^{(k_j)}$ and which lies on the same hyperplane as $x^{(k_j)}$, then $f_{k_{j+1}} < f_{k_j} \; \forall \; k_j$, where $f_{k_j} = ||x^{(k_j)} - y^{(k_j)}||$;

3) $\Delta f_{k_j} \equiv f_{k_{j+1}} - f_{k_j} = 0$ iff $f_{k_j} = 0$.

Consequently, 4) $x^{(k_j)} \dashrightarrow y^{(k_j)} \in S$ as $k \dashrightarrow \infty$.

Note: (a) There may be more than one such subsequence as described in (1), and the theorem applies to all.

(b) Since the system is non-superfluous, there is always such $y^{(k_j)}$ as defined in (2) for every $x^{(k_j)}$.

Proof

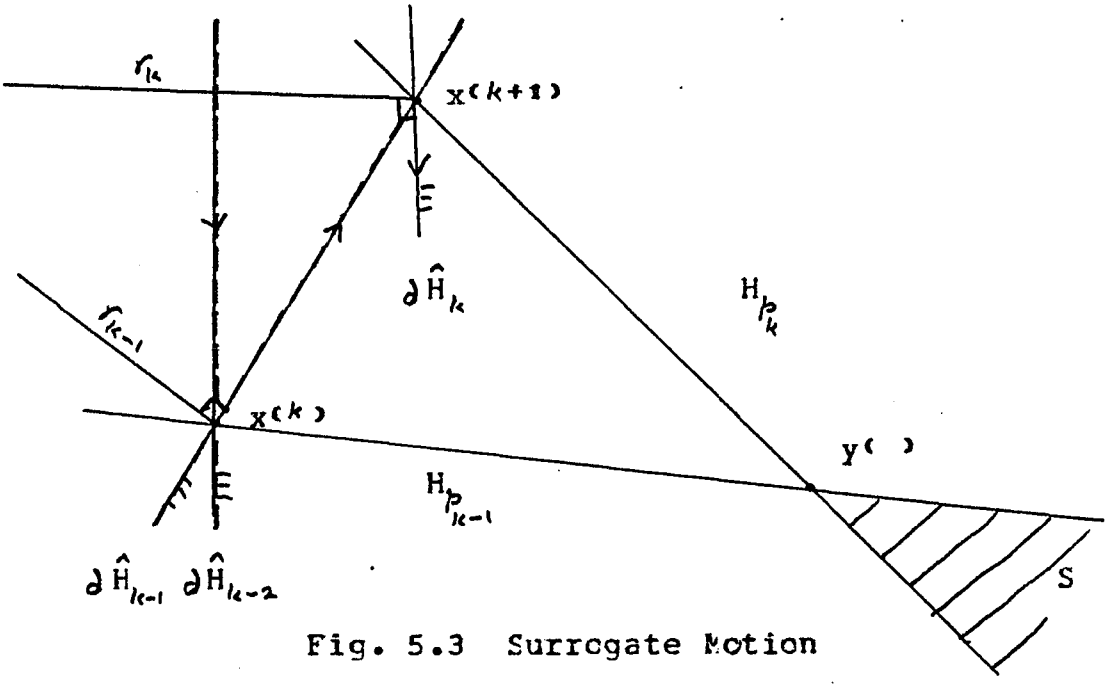


Fig. 5.3 Surrogate Motion

Prcof of 1)

This is trivial. We have infinite points and finite constraints. Therefore there must be at least one hyperplane on which an infinite sequence of the points lie.

Proof of the remaining parts

Let $\mu = k_j$ and $\nu = k_{j+1}$.

i.e. $x(k_j) = x(\mu)$ and $x(k_{j+1}) = x(\nu)$.

2-D System

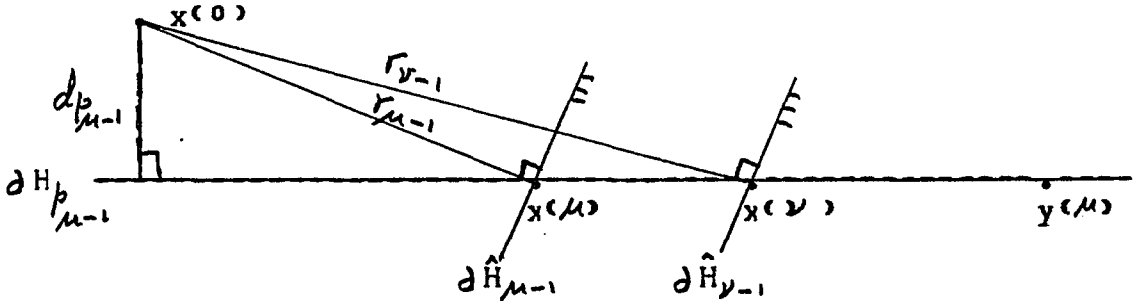


Fig. 5.4. m-Consraint 2-D Problem.

In this system all hyperplanes are straight lines. $\therefore \forall k \geq 1$, $x(\mu), x(\nu), y(\mu) \in \partial H_{p_{\mu-1}} \Rightarrow x(\mu), x(\nu), y(\mu)$ are colinear. It is then obvious that the rest of the argument here will be identical with that of Theorem 3B. The only difference is that in Theorem 3B, the number of iterations between (but excluding) any two successive iterates which lie on the same hyperplane is 1, while in this general case the number varies from 1 to m-1.

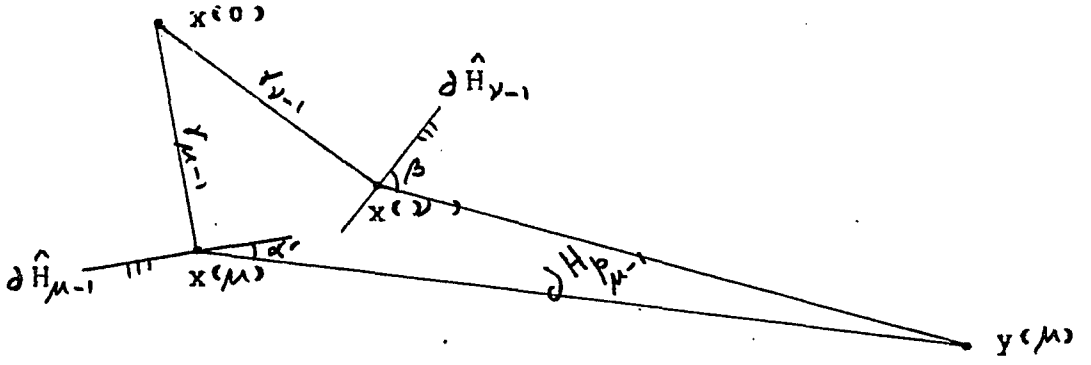


Fig. 5.5. m-Constraint n-D Problem.

Let i) α_μ = the acute angle between $\partial \hat{H}_{\mu-1}$ and $\partial H_{b_{\mu-1}}$,

ii) α' = the acute angle between $\partial \hat{H}_{\mu-1}$ and $(x(\mu) - y(\mu))$,

iii) β = the acute angle between $\partial \hat{H}_{\nu-1}$ and $(x(\nu) - y(\mu))$.

(Note: The angle between 2 hyperplanes is the angle between the 2 infinite lines in 2-D along their normals. The angle between a hyperplane and a line is the angle between that line and its projection onto the hyperplane.)

Since $x(\nu), y(\mu) \in (\partial H_{b_{\mu-1}} \equiv \partial H_{b_{\nu-1}})$, $\beta \geq \alpha_\nu$ . .....(10)

$$\cos(\alpha_\mu) = |\hat{a}_{\mu-1} \cdot a^{b_{\mu-1}}| \qquad \text{by (i) above}$$

$$= \left|\frac{d_{b_{\mu-1}}}{r_{\mu-1}}\right| \qquad \text{by steps 13,12,11 and 7}$$

$$> \left|\frac{d_{b_{\nu-1}}}{r_{\nu-1}}\right| \qquad \text{since } r_{\mu-1} < r_{\nu-1} \text{ and } d_{b_{\mu-1}} \equiv d_{b_{\nu-1}}$$

$$= |\hat{a}_{\nu-1} \cdot a^{b_{\nu-1}}| \qquad \text{by steps 13,12,11 and 7}$$

$$= \cos(\alpha_\nu) \qquad \text{by (i) above}$$

$$\geq \cos(\beta) \qquad \text{by (10)}$$

$\therefore \cos(\alpha_\mu) > \cos(\beta)$ if $r_{\mu-1} < r_{\nu-1}$

i.e. $\alpha_\mu < \beta$ if $x^{(\mu)} \notin S$. .....(11)

α' varies from 0 (for $y^{(\mu)}$ on the surrogate hyperplane $\partial\hat{H}_{\mu-1}$) to α_μ (for $y^{(\mu)}$ on the 2-D plane containing the 3 points $x^{(0)}$, $x^{(\mu)}$ and the projection of $x^{(0)}$ onto the given hyperplane $\partial H_{\mu-1}$). That is,

$$0 \le \alpha' \le \alpha_\mu$$

$$\therefore \quad \alpha' < \beta \qquad \text{.....(12)}$$

Consider the 2 triangles formed by the points $x^{(0)}, x^{(\mu)}, y^{(\mu)}$ and $x^{(0)}, x^{(\nu)}, y^{(\mu)}$. The side $x^{(0)}y^{(\mu)}$ is common to both and so by the cosine rule,

$$r_{\mu-1}^2 + f_\mu^2 - 2r_{\mu-1}f_\mu\cos(90+\alpha') = ||x^{(0)} - y^{(\mu)}||^2$$

$$= r_{\nu-1}^2 + ||x^{(\nu)} - y^{(\mu)}||^2 - 2r_{\nu-1}||x^{(\nu)} - y^{(\mu)}||\cos(90+\beta).$$

$$\therefore \quad r_{\mu-1}^2 + f_\mu^2 + 2r_{\mu-1}f_\mu\sin(\alpha')$$

$$= r_{\nu-1}^2 + ||x^{(\nu)} - y^{(\mu)}||^2 + 2r_{\nu-1}||x^{(\nu)} - y^{(\mu)}||\sin(\beta)... \qquad \text{.....(13)}$$

If $x^{(\mu)} \notin S$, then first using $r_{\nu-1} > r_{\mu-1}$ and then using (12) the right hand side (rhs) of (13) satisfies

$$r_{\nu-1}^2 + ||x^{(\nu)} - y^{(\mu)}||^2 + 2r_{\nu-1}||x^{(\nu)} - y^{(\mu)}||\sin(\beta)$$

$$> r_{\mu-1}^2 + ||x^{(\nu)} - y^{(\mu)}||^2 + 2r_{\mu-1}||x^{(\nu)} - y^{(\mu)}||\sin(\beta)$$

$$> r_{\mu-1}^2 + ||x^{(\nu)} - y^{(\mu)}||^2 + 2r_{\mu-1}||x^{(\nu)} - y^{(\mu)}||\sin(\alpha')$$

Since $f_\nu \le ||x^{(\nu)} - y^{(\mu)}||$ by definition of f_ν, $r_{\mu-1} > 0$ and $\sin(\alpha') > 0$, combining (13) with the above yields

$$r_{\mu-1}^2 + f_\mu^2 + 2r_{\mu-1}f_\mu\sin(\alpha') > r_{\mu-1}^2 + f_\nu^2 + 2r_{\mu-1}f_\nu\sin(\alpha')$$

$$\Rightarrow f_\mu^2 - f_\nu^2 + 2r_{\mu-1}(f_\mu - f_\nu)\sin(\alpha') > 0$$

$$\Rightarrow (f_\mu - f_\nu)(f_\mu + f_\nu + 2r_{\mu-1}\sin(\alpha')) > 0$$

$\Longrightarrow f_\mu - f_\nu > 0.$

$\therefore \quad f_{k_{j+1}} < f_{k_j} \ \forall\ k_j \quad \text{if } x^{(k_j)} \notin S.$

This ends the proof of part (2).

Now for the proof of part (3):

Suppose that $\Delta f_\mu \equiv f_\mu - f_\nu = 0$ but that $f_\mu \neq 0$.

Then since $f_\mu = f_\nu$, (13) implies that

$$r_{\mu-1}^2 + f_\nu^2 + 2r_{\mu-1} f_\nu \sin(\alpha')$$
$$= r_{\nu-1}^2 + ||x^{(\nu)} - y^{(\mu)}||^2 + 2r_{\nu-1}||x^{(\nu)} - y^{(\mu)}||\sin(\beta)$$

I.e. $r_{\mu-1}^2 - r_{\nu-1}^2$

$$= (||x^{(\nu)} - y^{(\mu)}||^2 - f_\nu^2) + 2(r_{\nu-1}||x^{(\nu)} - y^{(\mu)}||\sin(\beta) - r_{\mu-1} f_\nu \sin(\alpha'))$$

Now lhs < 0 since $r_{\mu-1} < r_{\nu-1}$.

rhs $\geq 2(r_{\nu-1}\sin(\beta) - r_{\mu-1}\sin(x'))f_\nu$

since $||x^{(\nu)} - y^{(\mu)}|| \geq f_\nu$

$\therefore$ rhs > 0 since $r_{\nu-1} > r_{\mu-1}$ and $\sin(\beta) > \sin(\alpha')$

Contradiction. Therefore,

$$\Delta f_\mu = 0 \Longrightarrow f_\mu = 0.$$

This ends the proof of part (3).

By lemma 7, f_{k_j} is a continuous function of $x^{(k_j)}$, and clearly $f_{k_j} \geq 0 \ \forall\ k_j$. Consequently by lemma 8 part (4) of the theorem follows; i.e.

$$x^{(k_j)} \dashrightarrow y^{(k_j)} \in S \text{ as } k \dashrightarrow \infty .$$

This completes the proof of convergence.

On Superfluous Constraints

A superfluous constraint is one whose removal does not in any way affect the feasible region. There are 3 types of superfluity as illustrated below. Fig. 5.6(a) is the case 1 we discussed in section 3.5, the significance of $c_{k_k i} = \pm 1$. Although we said in that section that this should not

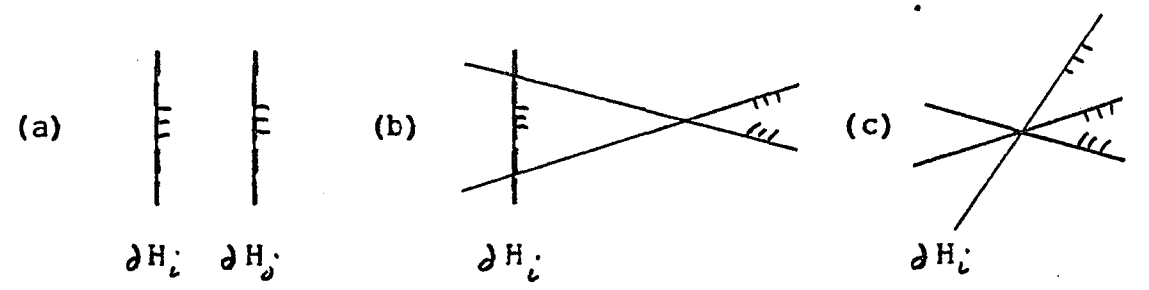


Fig. 5.6 Superfluous Constraint

bother us, the presence of superfluous constraints incr eases the amount of computations to be done. The type (a) case is easy to detect. The condition fcr inequality constraints is

$$d_i - r_k c_{k+1,i} < C \quad \text{and} \quad c_{k+1,i} = 1 \qquad (5.1.1)$$

If this condition holds, then the constraint i in question should be deleted from the system. Types (b) and (c) are not easy to detect but although we have based the general proof of convergence on a non-superfluous system, the existence of a superfluous ccnstraint will not change any of the proofs. This fact is supported by Thecrem 3B which proves convergence fcr a superfluous system. Recall that the narrow cone has a superfluous constraint of type (b).

5.2 Rate of Convergence

The rate of convergence is defined as the error reduction factor per iteration. We showed in the last section that in the pure surrogate, error reduction occurs after q, $2 \leq q \leq m$, iterations for each constraint. In Surrogate-II, it occurs at each iteation, that is, $q = 1$. Let us denote the error reduction factor by ν, and the rate of convergence by R. Then this means that

$$R = \text{mean rate of convergence after q iterations.}$$

$$= \text{qth root of } \lim_{k \to \infty} \frac{d(x^{(k+q)}, S)}{d(x^{(k)}, S)}$$

$$= \text{qth root of } \lim_{k \to \infty} \nu$$

$$= \nu^{\frac{1}{q}}.$$

$$\therefore \quad R = \nu^{\frac{1}{q}} \leq \nu^{\frac{1}{m}}. \qquad (5.2.1)$$

From this and the Narrow Cone section, we can see that R is proportional to

$$\theta = \max_{i,j \in I} |a^i . a^j|.$$

The more constraint we have, the less the chances of being trapped in a narrow cone. Hence there will be a better chance for faster convergence.

5.3 Work Done

It is a generally known fact that the faster we try to make an algorithm, the more computations we introduce into one iteration. We want to give a priori estimate of the number of arithmetic computations per iteration, relative to the size of the problem, m constraints and n variables.

The steps of the algorithms fall into 2 kinds, old-x-steps and new-x-steps. Table A below is a step by step analysis of maximum number of arithmetic operations for each kind, while Table B summarises the oprations for each of the algorithms.

Surrogate-I uses only the old-x-steps and Surrogate-II uses the new-x-steps. Surrogate-R uses both but exclusively, while Surrogate-III uses both, one after the other, for one major iteration. Hence we summarise thus

Table A

Work done at each step

Kind	Step No.	No. of Arithmetic Operations
old-x	7	6m
	9 - 12	10
	13	3n
	14	2mn+2m
		2mn+8m+3n+10
new-x	7	6m
	11' - 12'	4
	13'	4n
	14'	2mn+3m
		2mn+9m+4n+4
Agmon	$r = \lambda\, d_{k}$	1
	$x = x - ra^{k}$	2n
	$d = d - rA(a^{k})^{T}$	2mn+m
		2mn+m+2n+1

Table B

Work done by each algorithm

Algorithm	No. of Arith. Ops. per Iteration
Surrogate-I	2mn+8m+3n+10
Surrogate-R	2mn+8m+3n+10 or 2mn+9m+4n+4
Surrogate-II	2mn+9m+4n+4
Surrogate-III	4mn+17m+7n+14
Agmon	2mn+m+2n+1

On $C = AA^T$

Although A may be sparse, AA^T whose elements are frequently used for updating is dense. This matrix is symmetrical with all its main diagonal elements equal to 1. The elements, $a^i.a^j$, may be computed when needed if we want to minimize storage allocation. If minimizing computations is more important to us, then the strictly upper triangular part of AA^T should be computed once and stored in a vector row by row.

For each i, there are m-i elements. It is easy to verify that this gives a total of $m(m-1)/2$ elements, and that for any pair, (i,j), $i+1 \leq j \leq m$, the position, s(i,j), of $a^i.a^j$ in this vector with size equals $m(m-1)/2$ is

$$s(i,j) = \frac{(i-1)(2m-i)}{2} + j - i. \qquad (5.3.1)$$

Let us now analyse the computations in the 2 updating formulae that use the dot-products.

1) $c_{k+2,i} = \hat{h}_{p_{k+1}} c_{k+1,i} + h_{p_{k+1}} a^i \cdot a^{p_{k+1}}$,

2) $d_i = d_i - \hat{h}_{p_{k+1}} c_{k+1,i} - h_{p_{k+1}} a^i \cdot a^{p_{k+1}}$.

For each i we do n+2 multiplications per iteration if the $a^i \cdot a^j$'s are not pre-computed, and only 2 otherwise. Formula (2) has an alternative,

$$d_i = a^i x^{(k+1)} - b_i .$$

This requires n multiplications. Therefore we recommend its use when the $a^i \cdot a^j$'s are not pre-computed. Observe that from (5.3.1),

$$s(i,p_{k+1}) = \begin{cases} s(i-1,p_{k+1}) + m - i & \forall\ i < p_{k+1} , \\ \dfrac{(p_{k+1}-1)(2m-p_{k+1})}{2} - p_{k+1} + i & \forall\ i > p_{k+1} . \end{cases} \qquad (5.3.2)$$

To use the vector, we have the following algorithm segment:

```
          .
          .
          .
   j1 = p_{k+1} - m;
   j2 = (p_{k+1} -1)*(2*m-p_{k+1} )/2 - p_{k+1} ;
   Do i = 1 to m;
    if i < p_{k+1}  then do;
                         j1 = j1 + m - i;
                         s = j1;
                     end;
                else if i > p_{k+1}  then s = j2 + i;
     .
     .
     .
   end;
          .
          .
          .
```

Note that $a^i \cdot a^{p_{k+1}} = 1$ for $i = p_{k+1}$.

6. LINEAR EQUATIONS AND LINEAR PROGRAMMING

6.1 Linear Equations

In definitions (1) and (2) on page 8, we clearly showed the distinction between an inequality constraint and an equality constraint. An equation is a constraint that insists on hyperplane satisfaction. For a linear system of equations, the problem is to find $x \in R^n$ such that

$$Ax = b,$$

given $A \in R^{m \times n}$ and $b \in R^m$.

Although an equation can be changed into a pair of inequalities, it is not necessary to do so since the surrogate methods work with hyperplanes to obtain a boundary solution. Let us consider for example constraint i which is $a^i x = b_i$.

$$a^i x = b_i \iff a^i x \leq b_i \text{ and } a^i x \geq b_i,$$
$$\iff a^i x \leq b_i \text{ and } -a^i x \leq -b_i.$$

Let H^+ = the half-space for $a^i x \leq b_i$

and H^- = half-space for $-a^i x \leq -b_i$.

Then $\partial H^- = \{x \mid -a^i x + b_i = 0\}$

$= \{x \mid a^i x - b_i = 0\}$

$= \partial H^+$.

Note that $H^- \neq H^+$,

$$x \in (H^+ - \partial H^+) \iff x \notin (H^- - \partial H^-)$$

and $$x \notin (H^+ - \partial H^+) \iff x \in (H^- - \partial H^-).$$

Since the only feasible solution to an equality constraint, is a solution that is on its hyperplane, we have to modify the test steps of the algorithms. The following is a table

of expressions with respect to the 2 half-spaces of an equation.

Table C

Expressions for hyperplanes of equation

H+	H-
a^i	$-a^i$
d_i	$-d_i$
$\hat{a}_k \cdot a^i$	$-\hat{a}_k \cdot a^i$
$d_i - r_k \hat{a}_k \cdot a^i$	$-d_i + r_k \hat{a}_k \cdot a^i$
$r_k - d_i \hat{a}_k \cdot a^i$	$r_k - d_i \hat{a}_k \cdot a^i$

For the satisfaction of a constraint,

$$d_i \leq 0 \quad \text{and} \quad -d_i \leq 0 \iff d_i = 0.$$

Similarly,

$$d_i - r_k \hat{a}_k \cdot a^i \leq 0 \text{ and } -d_i + r_k \hat{a}_k \cdot a^i \leq 0 \iff d_i - r_k \hat{a}_k \cdot a^i = 0$$

Therefore the tests for viclation of an equality constraint

are $d_i = 0$ for $k = 0$

and $d_i - r_k \hat{a}_k \cdot a^i = 0$ for $k \geq 1$.

To initialize/update the surrogate constraint, we have as follows:

For $k = 0$,

$$r_0 = \begin{cases} d_{p_0} & \text{if } d_{p_0} > 0 \\ -d_{p_0} & \text{if } d_{p_0} < 0 \end{cases}$$

$$= |d_{p_0}|.$$

and

$$\hat{a}_0 = \begin{cases} a^{p_0} & \text{if } d_{p_0} > 0 \\ -a^{p_0} & \text{if } d_{p_0} < 0. \end{cases}$$

For all k,

$$\hat{a}_{k+1} = \frac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{1 - c^2_{p_{k+1}}} \hat{a}_k + \begin{cases} \dfrac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{1 - c^2_{p_{k+1}}} a^{p_{k+1}} & \text{if } d_{p_{k+1}} - r_k c_{p_{k+1}} > 0 \\ \dfrac{-d_{p_{k+1}} + r_k c_{p_{k+1}}}{1 - c^2_{p_{k+1}}} (-a^{p_{k+1}}) & \text{if } d_{p_{k+1}} - r_k c_{p_{k+1}} < 0 \end{cases}$$

$$= \frac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{1 - c^2_{p_{k+1}}} \hat{a}_k + \frac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{1 - c^2_{p_{k+1}}} a^{p_{k+1}} .$$

This is exactly the same as the case for inequality constraint. Therefore there is no change in the updating formula even when $d_{p_{k+1}} - r_k c_{p_{k+1}} < 0$.

Finally let us consider the inconsistency test.

For H^+, it is $d_i - r_k \hat{a}_k \cdot a^i > 0$ and $\hat{a}_k \cdot a^i = -1$.

For H^-, it is $-d_i + r_k \hat{a}_k \cdot a^i > 0$ and $-\hat{a}_k \cdot a^i = -1$.

i.e. $d_i - r_k \hat{a}_k \cdot a^i < 0$ and $\hat{a}_k \cdot a^i = 1$.

These 2 tests combine into one,

$$d_i - r_k \hat{a}_k \cdot a^i \neq 0 \text{ and } |\hat{a}_k \cdot a^i| = 1.$$

We shall now modify the algorithm to handle a system of linear equation or a mixture of the 2 types. The whole algorithm will not be listed here, only the affected steps 3,5,7 and 8.

Let $I_e = \{i \in I | \text{constraint } i \text{ is an equation}\}$.

Step 3) Choose $p_0 \in I$ such that

$$r_0 = \max \left(\max_{i \in I - I_e} d_i, \max_{i \in I_e} |d_i| \right);$$

Step 4) If $r_0 \leq 0$, then $x^{(0)} \in S$, stop;

Step 5) Set k = 0; $\hat{a}_0 = \begin{cases} a^{p_0} & \text{if } d_{p_0} > 0 \\ -a^{p_0} & \text{if } d_{p_0} < 0 ; \end{cases}$

Step 7) Choose $p_{k+1} \in I$ such that

$$g^2_{p_{k+1}} = \max\left(\max_{\substack{i \in I - I_e \\ g_i > 0}} g_i^2, \max_{i \in I} g_i^2 \right)$$

where $$g_i^2 = \frac{(d_i - r_k c_{k+1,i})^2}{1 - c^2_{k+1,i}} \quad \forall\, i \in I;$$

Step 8) If $\exists k$ such that $d_i - r_k c_{k+1,i} > 0$ & $c_{k+1,i} = -1$, $i \in I - I_e$

or $d_i - r_k c_{k+1,i} \neq 0$ & $|c_{k+1,i}| = 1$, $i \in I_e$

stop with no solution;

All the other steps remain unchanged. As is seen, these modifications are minor and have neither changed the idea nor the computations.

The best way to handle the constraints is to arrange them so that the equations come after the inequalities. Setting m1 to the number of inequality constraints gives an indicator for when to switch from the inequality to the equality tests.

6.2 Linear Programming

The linear programming, LP, problem differs from the linear inequality, LI, problem and the linear equations, LE, in two ways. The LP has

i) a linear fucntion, z(x), to maximize,

and ii) non-negativity constraint on the variable, x.

The form of the problem, called the primal, is:

Given $u \in R^n$, $A \in R^{m \times n}$ and $b \in R^m$,

find $x \in R^n$ in order to

$$\text{maximize} \quad z = u^T x,$$

$$\text{subject to} \quad Ax \leq b, \qquad \ldots\ldots\ldots (1)$$

and $x \geq 0$.

Another form of the problem known as the dual is to

$$\begin{aligned} \text{find} \quad & \lambda \in R^{m} \text{ such as to} \\ \text{minimize} \quad & z^{*} = b^{T}\lambda \\ \text{subject to} \quad & A^{T}\lambda \geq u, \\ \text{and} \quad & \lambda \geq 0. \end{aligned} \qquad \ldots\ldots\ldots(2)$$

The Duality Theorem

As is obvious, there is a relationship between the primal and the dual. The weak duality theorem states that if x is feasible for the primal and λ is feasible for the dual, then

$$u\,x \leq b^{T}\lambda\,. \qquad \ldots\ldots\ldots(3a)$$

Proof: By (2) and (1), $u^{T}x \leq \lambda^{T}Ax \leq \lambda^{T}b = b^{T}\lambda$.

The strong duality theorem states that if x^{*} is optimal feasible for the primal and λ^{*} is optimal feasible for the dual, then

$$u^{T}x^{*} = b^{T}\lambda^{*}. \qquad \ldots\ldots\ldots(3b)$$

The proof of this, a consequence of the weak part, can be found in any linear programming text. We are simply interested in using it to transform an LP into an LI.

Combining (1), (2) and (3), gives an LI of the form

$$\begin{pmatrix} A & 0 \\ 0 & -A^{T} \\ -I^{n} & 0 \\ 0 & -I^{m} \\ -u^{T} & b^{T} \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} \leq \begin{pmatrix} b \\ -u \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad (6.1.1)$$

where I^{r} is an r by r identity matrix.

Note: the last row of (6.1.1) is the reverse of the weak duality.

Unrestricted Variables

Let us assume that we have arranged the m ccnstraints so that the 1st m1 of these are inequalities, and the last m2, (m = m1+m2) are equations, then we have

$$A1x \leq b1 \quad \text{and} \quad A2x = b2,$$

$$\text{where} \quad A = \begin{pmatrix} A1 \\ A2 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} b1 \\ b2 \end{pmatrix}.$$

The dual to this problem is

$$\min \quad \begin{pmatrix} b1 \\ b2 \end{pmatrix}^T \begin{pmatrix} \lambda 1 \\ \lambda 2 \end{pmatrix}$$

$$\text{subject to} \quad \begin{pmatrix} A1 \\ A2 \end{pmatrix}^T \begin{pmatrix} \lambda 1 \\ \lambda 2 \end{pmatrix} \geq u,$$

$$\text{and} \quad \lambda 1 \geq 0,$$

where the m2 dual variables, $\lambda 2$, are unrestricted.

6.2.1 Storage Requirement

The coefficient matrix, $\tilde{A}$, of (6.1.1) is a (2m+2n+1) by (m+n) matrix. It is made up of block matrices, 4 cf which are zero matrices. Therefore storing $\tilde{A}$ as is above, will be most wasteful both in space and in computaticn. Assuming that the given A is dence (or treated as such), $\tilde{A}$ would generally require $2(m^2+mn+n^2) + 2mn+m+n$ words of the computer memory. $2(m^2+mn+n^2)$ of these are unnecessary and we shall show why.

There are 4 major types of constraints involved here. These are

i) the A-constraints,

ii) the A^T-constraints,

iii) the x-constraints,

iv) the z-constraint.

Since we have to normalize the columns of A for the A^T-constraints, it is imperative that we get a separate space for $-A^T$. For the x-constraints, we need no storage for $-I^{n+m}$ for obvious reasons. Also the same reason of normalization forces us to keep a separate space for $(-u^T, b^T)$ for the z-constraint. All together we need

mn	words for A,
nm	words for $-A^T$,
0	word for $-I^{n+m}$,
n+m	words for $(-u^T, b^T)$.
2mn+m+n	words for $\tilde{A}$.

Let us assume that we have normalized and stored $-A^T$ in an n by m matrix AT and the normalized $(-u^T, b^T)$ in a vector az, then with

$$\tilde{I} = I \cup \{m+1, m+2, \ldots, 2m+2n, 2m+2n+1\},$$

we define the other associated vectors and scalars as follows:

$$\tilde{x} = \begin{pmatrix} x \\ \lambda \end{pmatrix}$$

$$\tilde{b} = \begin{pmatrix} b \\ -\bar{u} \end{pmatrix} \text{ is an } (m+n)\text{-vector with } \bar{u}_j = u_j/\|a_j\|,\ 1 \le j \le n.$$

$$\tilde{a}^i = \begin{cases} a^i & \text{if } 1 \le i \le m, \\ at^{i-m} & \text{if } m+1 \le i \le m+n, \\ -e_{i-m-n} & \text{if } m+n+1 \le i \le 2m+2n, \\ az & \text{if } i = 2m+2n+1. \end{cases}$$

$$\tilde{d}_i = \begin{cases} d_i = a^i x^{(0)} - b_i & \text{if } 1 \le i \le m, \\ \tilde{d}_i = at^{i-m}\lambda^{(0)} - \tilde{b}_i & \text{if } m+1 \le i \le m+n, \\ -\tilde{x}^{(0)} & \text{if } m+n+1 \le i \le 2m+2n, \\ az^T \tilde{x}^{(0)} & \text{if } i = 2m+2n+1. \end{cases}$$

$$\tilde{a}^i \cdot \tilde{a}^j = \begin{cases} 1 & \text{if } i = j, \\ a^i \cdot a^j & \text{if } 1 \le i \ne j \le m, \\ -a_{i,j-m-1} & \text{if } 1 \le i \le m \text{ and } m+n+1 \le j \le m+2n, \\ a^i \cdot az_{first} & \text{if } 1 \le i \le m \text{ and } j = 2m+2n+1, \\ & \text{where } az_{first} = (az_1, \ldots, az_n), \\ at^{i-m} \cdot at^{j-m} & \text{if } m+1 \le i \ne j \le m+n, \\ -at_{i-m,j-m-n} & \text{if } m+1 \le i \le m+n \text{ and } m+2n+1 \le j \le 2m+2n, \\ at^{i-m} \cdot az_{last} & \text{if } m+1 \le i \le m+n \text{ and } j = 2m+2n+1, \\ & \text{where } az_{last} = (az_{n+1}, \ldots, az_{n+m}), \\ -az_{j-m-n} & \text{if } i = 2m+2n+1 \text{ and } m+n+1 \le j \le 2m+2n, \\ 0 & \text{otherwise.} \end{cases}$$

We should observed that 5 out of the 9 definitions for $\tilde{a}^i \cdot \tilde{a}^j$ need no computations.

LP problems sometimes involve mixed constraints and in the last section we showed how to handle equations. With the 4 classifications of the constraints, the recommended arrangements and the above definitions for $\tilde{A}$ and its associated vectors, all that is needed for the algorithm to solve an LP problem is a 'computed-goto' type of statements to select what is involved without using 2m+2n+1 by m+n+1 array.

7. IMPLEMENTATIONS FCR LARGE SPARSE MATRICES

In many real life problems, the ccefficient matrix, A, is always very large (in the order of 10,000 cr more) and sparse. In some cases, the sparseness may have a uniform pattern, while in others it may be irregular. The uniform ones are often diagonally banded or oriented. The band, a set of diagonals, may have no zero diagonals (i.e. diagonals with all its elements zero), or it may have some zero diagonals in between the nonzero ones.

The surrogate methods, unlike the Simplex, do not update A. For this reason we can, with an apprcpriate code, use a smaller size matrix to hold the nonzero elements of A. The pattern of A will be very helpful in designing an implementation matrix.

I) Band Without Zero Diagonals

$$A = \begin{pmatrix} x & x & x & & & & & \\ x & x & x & x & & & & \\ & x & x & x & x & & 0 & \\ & & & \ddots & & & & \\ & & & & \ddots & & & \\ & 0 & & & & x & x & x & x \\ & & & & & & x & x & x \\ & & & & & & & x & x \end{pmatrix}$$

Let w = the maximum number of elements from the first nonzero to the last nonzero in any row.

We can then implement A as an m by w matrix, A', along with an m-vector, FIRST, where

FIRST(i) = column of the first nonzero element in row i.

$a'(i,j) = a(i,j+FIRST(i)-1),\ 1 \le j \le w$ and $i \in I$...(1)

Let us assume that w = 4 for the above A. Then we have

$$A' = \begin{pmatrix} x & x & x & 0 \\ x & x & x & x \\ x & x & x & 0 \\ & & \vdots & \\ x & x & x & x \\ x & x & x & 0 \\ x & x & 0 & 0 \end{pmatrix} \quad \text{and} \quad FIRST = \begin{pmatrix} 1 \\ 1 \\ 2 \\ \vdots \\ n-3 \\ n-2 \\ n-1 \end{pmatrix}$$

Manipulation Code

Equation (1) gives us the mapping cf A' into A. With it we can locate those elements of A that are stored in A'. The main computations in the Surrogate methods are the dot-products of two rows or a row and the vector x. To multiply a'(i1,j1) by a'(i2,j2), f(j1) must be equal to f(j2). From (1) that means that

$$j1+FIRST(i1) = j2+FIRST(i2)$$

$$\therefore \quad j1 = j2 + FIRST(i2) - FIRST(i1) \qquad \text{.....(2)}$$

A piece of FORTRAN code for computing the dot-product $c = a^{i_1}.a^{i_2}$ is as follows:-

```
      .
      .
      .
      C = 0.
      R1 = I1
      R2 = I2
      IF (FIRST(R1) .LE. FIRST(R2)) GC TO 1
      R1 = I2
      R2 = I1
    1 J2 = 1
      K = J2 + FIRST(R2) - FIRST(R1)
      IF (K .GT. W) GC TC 3
      DO 2 J1 = K,W
```

```
      C = C + A(R1,J1) * A(R2,J2)
    2 J2 = J2 + 1
    3 CONTINUE
      .
      .
      .
```

That for computing $c = a^t x$ is as follcws:-

```
      .
      .
      .
      C = 0.
      K = FIRST(I) - 1
      DO 4 J = 1,W
    4 C = C + A(I,J) * X(J+K)
      .
      .
      .
```

Note that the A referenced in the code is the implemented A'.

An example of this type of sparse matrix is the hypercube,

$$0 \le x_1 \le 1,$$

$$2x_{j-1} \le x_j \le 6^{j-1} - 2x_{j-1}, \quad \text{for } j = 2,3, \ldots, n.$$

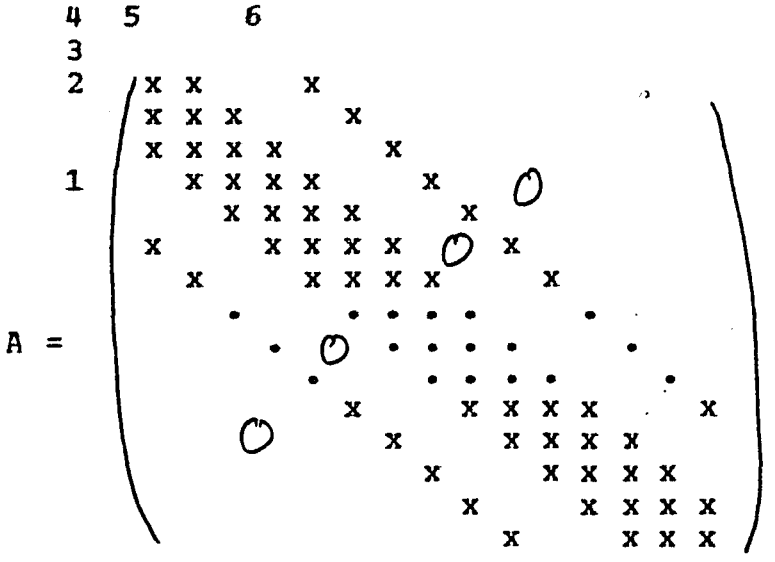
This problem can be expressed as

$$\begin{pmatrix} -1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 2 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 2 & -1 & \cdots & 0 & 0 \\ 0 & 0 & 2 & 1 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \\ 0 & 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 2 & 1 \end{pmatrix} x \le \begin{pmatrix} 0 \\ 1 \\ 0 \\ 6 \\ 0 \\ 36 \\ 0 \\ 216 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 6^{n-1} \end{pmatrix}.$$

Therefore for our implementation, we have

$$w = 2, \quad A' = \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 2 & -1 \\ 2 & 1 \\ 2 & -1 \\ 2 & 1 \\ 2 & -1 \\ 2 & 1 \\ \vdots & \\ 2 & -1 \\ 2 & 1 \end{pmatrix} \quad \text{and} \quad FIRST = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ \vdots \\ n-1 \\ n-1 \end{pmatrix}.$$

II) Band With Zero Diagonals



Let w = the number of nonzero diagonals in the band, numbered consecutively beginning from the lowest as shown above.

We can implement A as an m x w matrix, A', with a w-vector, FIRST, where

FIRST(j) = row of the 1st element of the jth nonzero diagonal

$$= \begin{cases} r, & \text{if the jth nonzero diagonal starts at } a_{r,1}, \\ 2-c, & \text{if the jth nonzero diagonal starts at } a_{1,c} \end{cases}$$

$a'(i,j) = a(i,i-FIRST(j)+1)$, $1 \le j \le w$ and $i \ge FIRST(j)$..(3)

From the A we have above, we shall get the following:-

$$w = 6, \quad A' = \begin{pmatrix} 0 & 0 & 0 & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & x & x & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & x & x & x \\ x & x & x & x & x & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & x & x & 0 \\ x & x & x & x & 0 & 0 \end{pmatrix} \quad \text{and} \quad FIRST = \begin{pmatrix} 6 \\ 3 \\ 2 \\ 1 \\ 0 \\ -2 \end{pmatrix}$$

To have a distinct row for each diagonal, the expression, 2-c, gives a fictitious row above row 1 at which the diagonal originating at column c appears to originate. For example, a diagonal originating at column 2 appears to originate at row 0.

Manipulation Code

Like before, we have to match the columns before doing any multiplication for the dot-products. Equation (3) which is our map for this type gives

$i1-FIRST(j1) = i2-FIRST(j2)$ .....(4)

Using (4), the code for $c = a^{i_1}.a^{i_2}$ is

```
      .
      .
      .
      C = 0.
C
C COMPUTE THE LOCATION J1 AND COLUMN OF THE 1ST NONZERO IN
C ROW I1.
```

```
      J1 = 0
    1 J1 = J1 + 1
      IF (I1 .LT. FIRST(J1)) GO TO 1
      K1 = I1 - FIRST(J1)
C
C COMPUTE THOSE FOR I2.
C
      J2 = 0
    2 J2 = J2 + 1
      IF (I2 .LT. FIRST(J2)) GO TO 2
      K2 = I2 - FIRST(J2)
C
C COMPARE THEIR COLUMNS.
C
    3 IF (K1-K2) 4,6,5
C
C COL. OF 1ST OPERAND IS LESS THAN THAT OF THE 2ND,
C ADVANCE J1.
C
    4 IF (J1 .EQ. W) GO TO 7
      J1 = J1 + 1
      K1 = I1 - FIRST(J1)
      GO TO 3
C
C COL. OF 1ST OPERAND IS GREATER THAN THAT OF THE 2ND,
C ADVANCE J2.
C
    5 IF (J2 .EQ. W) GO TO 7
      J2 = J2 + 1
      K2 = I2 - FIRST(J2)
      GO TO 3
C
C COLUMNS ARE EQUAL.
C
    6 C = C + A(I1,J1) * A(I2,J2)
      IF (J1 .EQ. W .OR. J2 .EQ. W) GO TO 7
      J1 = J1 + 1
      K1 = I1 - FIRST(J1)
      J2 = J2 + 1
      K2 = I2 - FIRST(J2)
      GO TO 3
    7 CONTINUE
      .
      .
      .
```

For computing $c = a^t x$, we have

```
      .
      .
      .
      C = 0.
C
C COMPUTE THE LOCATION OF THE 1ST NONZERO IN ROW I.
C
```

```
      K = 0
    8 K = K + 1
      IF (I .LT. FIRST(K)) GO TO 8
      IPLUS1 = I + 1
      DO 9 J = K,W
    9 C = C + A(I,J) * X(IPLUS1-FIRST(J))
      .
      .
```

An example of this comes from boundary value problems. Consider the Dirichlet problem for Laplace's equation. The problem states that given a function, f(x,y), defined and continuous on the boundary of a region, R. Find a function, u(x,y), continuous both in the interior of R and on ∂R and also satisfying the Laplace's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

and also such that on ∂R

$$u = f \qquad \ldots\ldots\ldots\ldots\ldots\ldots (6)$$

The usual approach is to break the region into small squares of size h, called mesh size. Let us consider a unit square region. Using h = 1/4 gives us the following figure.

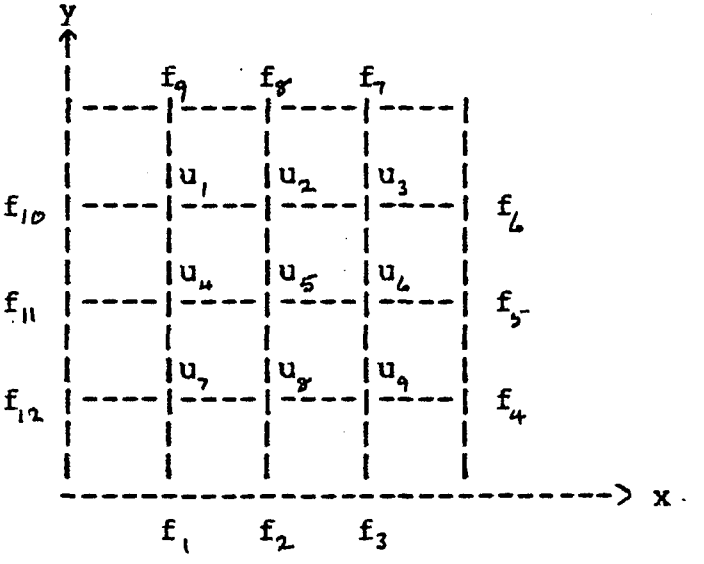


Fig. 7.1 Unit Mesh

From difference equation, we have

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+h,y) + u(x-h,y) - 2u(x,y)}{h^2}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{u(x,y+h) + u(x,y-h) - 2u(x,y)}{h^2}$$

Applying (5) we get

$$4u(x,y) - u(x+h,y) - u(x-h,y)\ u(x,y+h) - u(x,y-h) = 0 \ldots (7)$$

With (6) and (7) we get 9 linear equations,

$$4u_1 - u_2 - f_{10} - f_9 - u_4 = 0$$

$$4u_2 - u_3 - u_1 - f_8 - u_5 = 0$$

$$4u_3 - f_6 - u_2 - f_7 - u_6 = 0$$

$$4u_4 - u_5 - f_{11} - u_1 - u_7 = 0$$

$$4u_5 - u_6 - u_4 - u_2 - u_8 = 0$$

$$4u_6 - f_5 - u_5 - u_3 - u_9 = 0$$

$$4u_7 - u_8 - f_{12} - u_4 - f_1 = 0$$

$$4u_8 - u_9 - u_7 - u_5 - f_2 = 0$$

$$4u_9 - f_4 - u_8 - u_6 - f_3 = 0$$

That is

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix} = \begin{pmatrix} f_9 + f_{10} \\ f_8 \\ f_6 + f_7 \\ f_{11} \\ 0 \\ f_5 \\ f_1 + f_{12} \\ f_2 \\ f_3 + f_4 \end{pmatrix}$$

Generally, for a rectangular region of dimension r by s, this problem always has $(r/h - 1)(s/h - 1)$ interior points and same number of equations. Therefore A frequently comes out as an $(r/h - 1)(s/h - 1)$ by $(r/h - 1)(s/h - 1)$ diagon-

ally banded symmetric matrix, with a bandwidth of 2/h - 1 and always having exactly 5 nonzerc diagonals. These are, the main diagonal, its 2 neighbours, and another 2 originating at $a_{\frac{1}{h},1}$ and $a_{1,\frac{1}{h}}$. If the region is not regular, the edges of the gridwork superimposed on it will not coincide with the boundary. In such a case, A will not be symmetrical but will still have the band structure with 5 ncnzero diagonals. Another time when A comes cut unsymmetrical for this problem is when the polar, instead of the cartesian coordinate is used.

The implementation, therefore, fcr a bcundary value problem reguires

$$w = 5, \quad A' = \begin{pmatrix} 0 & 0 & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & x & x & x & x \\ x & x & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & x & x \\ x & x & x & x & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & x & 0 \\ x & x & x & 0 & 0 \end{pmatrix} \quad \text{and} \quad FIRST = \begin{pmatrix} 1/h \\ 2 \\ 1 \\ 0 \\ 2-1/h \end{pmatrix}$$

To have a diagonally banded structure, the interior points must be numbered in the left-to-right top-to-bcttom manner.

III) Irregular Sparseness

$$A = \begin{pmatrix} 0 & 0 & x & 0 & 0 & x \\ x & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & x & 0 & 0 \\ 0 & x & 0 & 0 & x & x \\ x & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x & 0 \end{pmatrix}$$

It is not always that the large and sparse matrices for our problems come out well patterned. Sometimes the non-zeroes may be irregularly scattered as illustrated above. Examples of this type are found in electrical network problems. In this case using a matrix to represent A is as inadequate as using A itself. We will therefore use a vector, A', to store the nonzeros, row by row, and another vector, KOL, to store the columns of these nonzeros. That is, A'(j) = A(i,KCL(j)) for some i. .....(8)
In addition to A' and KOL is an m-vector, LSTELT, where

LSTELT(i) = the position in A' of the last nonzero in row i of the given matrix A.

Therefore our implementation for the above will be

A' = (x,x,x,x,x,x,x,x,x,x,x)

KOL = (3,6,1,3,4,2,5,6,1,3,5)

LSTELT = (2,4,5,8,10,11)

Manipulation Code

The code for this implementation is similar to the last one. For $c = a^{i_1}.a^{i_2}$, we have

```
      .
      .
      .
      C = 0.
C
C COMPUTE THE LOCATIONS OF THE 1ST NONZERO IN ROWS I1 AND I2
C RESP.
C
      J1 = 1
      IF (I1 .NE. 1) J1 = LSTELT(I1-1) + 1
      J2 = 1
      IF (I2 .NE. 1) J2 = LSTELT(I2-1) + 1
C
C COMPARE THEIR COLUMNS.
C
```

```
    1 IF (KOL(J1)-KOL(J2)) 2,4,3
C
C KOL(J1) .LT. KOL(J2), ADVANCE J1.
C
    2 IF (J1 .EQ. LSTELT(I1)) GO TO 5
      J1 = J1 + 1
      GO TO 1
C
C KOL(J1) .GT. KOL(J2), ADVANCE J2.
C
    3 IF (J2 .EQ. LSTELT(I2)) GO TO 5
      J2 = J2 + 1
      GO TO 1
C
C KOL(J1) .EQ. KOL(J2).
C
    4 C = C + A(J1) * A(j2)
      IF(J1 .EQ. LSTELT(I1) .OR. J2 .EQ. LSTELT(I2)) GO TO 5
      J1 = J1 + 1
      J2 = J2 + 1
      GO TO 1
    5 CONTINUE
      .
      .
      .
```

For c = a'x, we have

```
      .
      .
      .
      C = 0.
      K1 = 1
      IF (I .NE. 1) K1 = LSTELT(I-1) + 1
      K2 = LSTELT(I)
      DO 1 J = K1,K2
    1 C = C + A(J) * X(KOL(J))
      .
      .
      .
```

8. SOLVED PROBLEMS

To test and support the theory we have developed, we ran several problems with the 4 algorithms and with Agmon's (projection and reflection). With varying sizes, we ran most of them starting at 4 different points. The averages of the number of iterations and multiplications/divisions it took each algorithm to get a solution are recorded. Before taking up each individual problem, we shall explain some of the notation in the accompanying tables.

Surrg1 = Surrogate-I, the pure surrogate

Surrgr = Surrogate-R, the repeat surrogate

Surrg2 = Surrogate-II

Surrg3 = Surrogate-III

Agmon1 = Agmon's projection, relaxation parameter $\lambda = 1$

Agmos2 = Agmon's reflection, relaxation parameter $\lambda = 2$

In most of the blocks of the tables, there are 3 entries

a
b
c/d

The first number, a, is the average of the number of iterations. The second, b, is the average of the multiplications/divisions done. In the third entry, c is the number of times that particular algorithm got a solution without exhausting the maximum iterations allowed, while d gives the number of starting points used. E.g.

1/4

means that 4 starting poits were used, out of which the algorithm got to a sclution only once without reaching the limit set for the iteration. The maximum iteration was arbitrarily set at 500. If a = 500, then c = 0. Alsc the tolerant value was set at 0.00015.

8.1 Inequalities

a) Todd's

The 1st test case is example 3 in Todd (21). The problem is that of finding an $x \in R^3$ such that, for a small positive δ,

$$\begin{pmatrix} \delta & 1 & 2 \\ \delta & 1 & -2 \\ \delta & -1 & 2 \\ \delta & -1 & -2 \end{pmatrix} x \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Todd recommended the starting point

$$x^{(0)} = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{pmatrix}, \quad \text{where} \quad \theta = \tan^{-1}\delta.$$

For δ, we used the value 0.1. We also went further to expand the problem for bigger values of n, $x \in R^4$, $x \in R^5$ and $x \in R^6$, with

$$A = \begin{pmatrix} \delta & 1 & 2 & 3 \\ \delta & 1 & 2 & -3 \\ \delta & 1 & -2 & 3 \\ \delta & 1 & -2 & -3 \\ \delta & -1 & 2 & 3 \\ \delta & -1 & 2 & -3 \\ \delta & -1 & -2 & 3 \\ \delta & -1 & -2 & -3 \end{pmatrix}, \qquad A = \begin{pmatrix} \delta & 1 & 2 & 3 & 4 \\ \delta & 1 & 2 & 3 & -4 \\ \delta & 1 & 2 & -3 & 4 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ \delta & -1 & -2 & 3 & -4 \\ \delta & -1 & -2 & -3 & 4 \\ \delta & -1 & -2 & -3 & -4 \end{pmatrix} \text{ and}$$

$$A = \begin{pmatrix} \varepsilon & 1 & 2 & 3 & 4 & 5 \\ \varepsilon & 1 & 2 & 3 & 4 & -5 \\ \varepsilon & 1 & 2 & 3 & -4 & 5 \\ & & & \vdots & & \\ \varepsilon & -1 & -2 & -3 & 4 & -5 \\ \varepsilon & -1 & -2 & -3 & -4 & 5 \\ \varepsilon & -1 & -2 & -3 & -4 & -5 \end{pmatrix} \quad \text{resp.}$$

Note that $m = 2^{n-1}$.

Table 1

Results of Todd's Problem and Extensions

M	N	SURRG1	SURRGR	SURRG2	SURRG3	AGMON1	AGMON2
4	3	15	5	2	2	500	34
		344	114	61	91	3515	250
		1/1	1/1	1/1	1/1	0/1	1/1
8	4	41	5	2	2	500	58
		1737	222	124	179	6036	728
		/1	1/1	1/1	1/1	0/1	1/1
16	5	83	5	2	2	500	84
		6533	437	254	356	10585	1844
		1/1	1/1	1/1	1/1	0/1	1/1
32	6	145	5	2	2	500	114
		21560	874	526	719	19198	4524
		1/1	1/1	1/1	1/1	0/1	1/1

Todd gave the example to show that finite termination cannot be guaranteed for Agmon's projection method. As table 1 shows, the surrogates, especially the re-initializing ones, converge much faster than Agmon's in this problem. Surrogate-2 is the best, followed by Surrogate-3. The performance of Surrogate-R is quite acceptable. Surrogate-1 is like Agmon2.

b) Trapezoidal Hypercube

The 2nd test case is a trapezoidal hypercube in R^n. We are to find an $x \in R^n$ such that

$$0 \le x_1 \le 1,$$

$$2x_{j-1} \le x_j \le 6^{j-1} - 2x_{j-1}, \quad j = 2,3, \ldots, n.$$

In matrix form, this can be expressed as

$$\begin{pmatrix} -1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 2 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 2 & 1 & 0 & \cdots & 0 & 0 \\ & & \vdots & & & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 2 & 1 \end{pmatrix} x \le \begin{pmatrix} 0 \\ 1 \\ 0 \\ 6 \\ 0 \\ 36 \\ \vdots \\ 0 \\ 6^{n-1} \end{pmatrix}$$

This problem has m = 2n constraints.

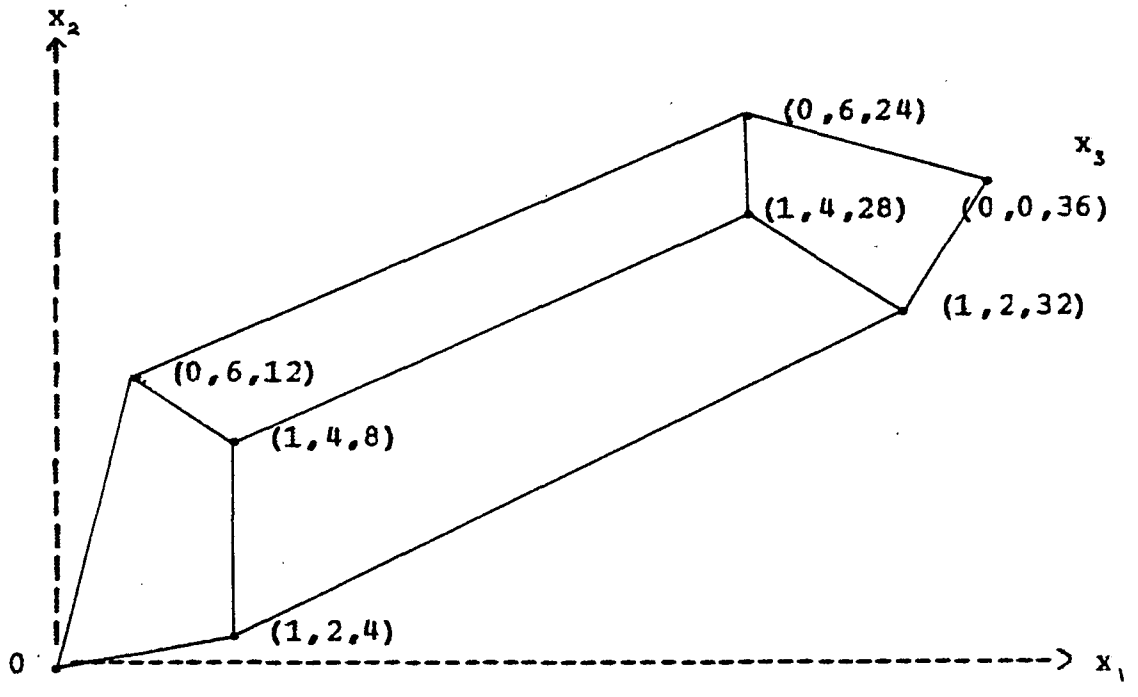


Fig. 8.1 The trapezoidal hypercube in R^3.

For n = 3, 4, 5 and 6, we used the following starting points

1) $x^{(0)} = (6^{n-1}, 6^{n-1}, \ldots, 6^{n-1})^T$,

2) $x^{(0)} = (0, 0, \ldots, -6^{n-1})^T$,

3) $x^{(0)} = (0, 0, \ldots, 2(6^{n-1}))^T$,

4) $x^{(0)} = (-1, -6, -36, \ldots, -6^{n-1})^T$.

The starting points were made exponential functions of the number of the variables so as to be far from the hypercube and thereby make approach to a solution as difficult as possible.

Table 2

Results of Trapezoidal Hypercube Problem

M	N	SURRG1	SURRGR	SURRG2	SURRG3	AGMON1	AGMON2
6	3	122	8	61	2	375	54
		3537	252	1775	123	3402	508
		4/4	4/4	4/4	4/4	1/4	4/4
8	4	500	59	375	101	376	409
		19860	2308	15058	7689	4550	4940
		0/4	4/4	1/4	4/4	1/4	4/4
10	5	500	383	394	362	500	500
		25676	18374	19711	34464	7555	7555
		0/4	1/4	1/4	4/4	0/4	0/4
12	6	500	378	376	376	392	500
		31225	22119	22750	44798	7141	9078
		0/4	1/4	1/4	1/4	1/4	0/4

Surrogate-3 is the best of all. In 3 out of the 4 sizes used, it got 4/4. That is, Surrogate-3 got 4 solutions out of the 4 starting points for each size except in one, where it got 1/4. However, we must point out that the amount of computations done by Surrogate-3 can be a liability if it

takes longer to get a solution.

Starting point (1) was the best. All 4 surrogate algorithms converged very fast for all m and n. Starting point (4) was the worst. This was not surprising because the feasible region is in the positive sectant of the n-space. At the 500th iteration and for m = 12, n = 6, only Surrogate-R had considerably reduced the distance. Starting points (2),(3) and (4) were bad for Surrogate-2.

c) Randomly Generated Problem

Problems (a) and (b) above are specialized problems. To test an arbitrary problem, we did the following:

i) A is on the average 10% dense,

(5-15% of the m entries in each column are nonzero.)

ii) $-1 < a_{ij} < 1$ was uniformly generated for $i = 1,2, .., m$ and $j = 1,2, .., n$

iii) $b = A(1,1, ..., 1)^T + r$, r_i uniformly generated in (0,1)

iv) $100 < |x_j^{(0)}| < 200$ and x_j uniformly generated for all j.

The reason for defining the vector b as we did is to guarantee that the solution set is not empty. We made 6 tests with various m and n.

Table 3

Results of Random Problem

M	N	SURRG1	SURRGR	SURRG2	SURRG3	AGMON1	AGMON2
30	20	500	21	28	8	82	370
		6352	3563	4538	2804	4700	19100
		0/1	1/1	1/1	1/1	1/1	1/1
60	20	500	500	500	500	500	500
		144803	129467	131942	260168	41220	41220
		0/1	0/1	0/1	0/1	0/1	0/1
90	20	500	117	203	61	500	500
		195455	44930	73065	44438	56820	56820
		0/1	1/1	1/1	1/1	0/1	0/1
30	30	500	36	20	12	206	23
		91192	6913	4237	4786	13260	2280
		0/1	1/1	1/1	1/1	1/1	1/1
60	30	500	500	500	500	500	322
		152026	141013	143713	284336	46830	30780
		0/1	0/1	0/1	0/1	0/1	1/1
90	30	500	265	292	208	500	500
		216109	104592	113183	158886	62730	62730
		0/1	1/1	1/1	1/1	0/1	0/1

For some strange reasons, 60 constraints seem tc be a bad case for all the methcds. Since they converge for 90 constraints, the trouble cannot be the size, but rather the structure. Again surrogates R, 2 and 3 outperforms Agmon.

d) Epsilon-Cube

Another inequality prcblem tested was one with specialty and randomness combined. The constraints are defined by a small cube.

$$-\epsilon \leq x_j \leq \epsilon \quad \text{for } j = 1,2, \ldots, n.$$

To this cube we added more constraints which are some combinations of the original ones. These additional constraints are defined as

$$\tilde{a} = \sum_{j \in J} \alpha_j (\pm e_j) \quad \text{and} \quad \tilde{b} = \epsilon \sum_{j \in J} \alpha_j ,$$

where the α^j's are randomly generated numbers uniformly distributed between 0 and 1. J was chosen as a random subset of indices $\{1, 2, \ldots, n\}$. For each pair of m and n, the following initial points were used:

i) $x_j^{(0)} = 1000 r_j$ for $-1 < r_j < 1$
and r_j uniformly generated,

ii) $x^{(0)} = (1000, 1000, \ldots, 1000)$,

iii) $x^{(0)} = -(1000, 1000, \ldots, 1000)$,

iv) $|x_j^{(0)}| = 1000$ for all j
and the sign randomly chosen.

The main idea was simply to start from a distant point to the solution set. With $\epsilon = 0.1$, the results are

Table 4

Results of Epsilon-Cube

M	N	SURRG1	SURRGR	SURRG2	SURRG3	AGMON1	AGMON2
20	10	10	10	5	4	10	500
		1103	1103	685	821	500	15210
		4/4	4/4	4/4	4/4	4/4	0/4
40	10	500	24	8	7	21	500
		89593	4810	1867	2829	1450	25410
		0/4	4/4	4/4	4/4	4/4	0/4
60	10	382	27	11	6	19	500
		94447	7478	3451	4019	1947	35610
		1/4	4/4	4/4	4/4	4/4	0/4
80	10	500	31	8	6	20	500
		156712	10904	3594	5109	2600	45810
		0/4	4/4	4/4	4/4	4/4	0/4
30	15	15	15	8	5	15	500
		2518	2518	1517	1895	1125	22965
		4/4	4/4	4/4	4/4	4/4	0/4
50	15	387	35	12	8	17	500
		89602	8902	3408	4402	1855	33265
		1/4	4/4	4/4	4/4	4/4	0/4
70	15	500	45	13	10	28	500
		152016	15026	4968	7116	3493	43565
		0/4	4/4	4/4	4/4	4/4	0/4
90	15	500	49	11	8	32	500
		187981	20313	5799	7898	4710	53865
		0/4	4/4	4/4	4/4	4/4	0/4

Surrogate R, 2 and 3 were very fast with this problem, with Surrogate-3 as the best, closely followed by Surrogate-2. The three algorithms converged very fast for all starting points and all sizes. Surrogate-1 was extremely slow in all except in starting point 1. Agmon1 was as good as the others while Agmon2 was worse than Surrogate-1.

8.2 Equations and LP Problems

Starting Points

For the tests of equations and linear programming problems, we used 4 starting points we considered meaningful. The first one is

$$x^{(0)} = \bar{0} \qquad \text{.....(1)}$$

Since zero is the minimum acceptable value for the variables of an LP problem, it made sense to include it as a starting point. (We do not claim that the variable will remain feasible during the entire iterations).

The second starting point we considered was the right hand side of the constraints. We assigned the right hand side to the variables in this manner

$$\begin{aligned} x_j^{(0)} &= b_j \qquad j = 1,2, \ldots, \min(m,n) \\ x_j^{(0)} &= 0 \qquad j = m+1, \ldots, n \quad \text{if } m < n. \end{aligned} \qquad \text{.....(2a)}$$

For the dual variable of LP, (see section 6.3 on storage)

$$\begin{aligned} x_{n+j}^{(0)} &= b_{m+j} \qquad j = 1,2, \ldots, \min(m,n) \\ x_{n+j}^{(0)} &= 0 \qquad j = n+1, \ldots, m \quad \text{if } n < m. \end{aligned} \qquad \text{.....(2b)}$$

In the third initialization, we distributed equally the right hand side to the variables. By 'equally', we mean assigning to each of the n variables, (1/n)th of the average of the right hand side. That is,

$$x_j^{(0)} = \frac{\sum_{i=1}^{m} b_i}{mn} \qquad j = 1,2, \ldots, n \qquad \text{.....(3a)}$$

Like the previous case, the dual variables got the dual right hand side. That is

$$x_j^{(0)} = \frac{\sum_{i=m+1}^{m+n} b_i}{mn} \qquad j = 1,2, \ldots, m \qquad \ldots\ldots\ldots(3b)$$

The final starting point we used was

$$x^{(0)} = (1,1, \ldots, 1)^T \qquad \ldots\ldots\ldots(4)$$

The idea behind this was to have

$$d_i = \sum_{j=1}^{\tilde{n}} a_{ij} - b_i \qquad i = 1,2, \ldots, \tilde{m}$$

where $\tilde{n} = \begin{cases} n & \text{if not LP} \\ n + m & \text{if LP} \end{cases}$

and $\tilde{m} = \begin{cases} m & \text{if not LP} \\ m + n & \text{if LP.} \end{cases}$

e) Linear Equation

We ran 4 sets of small equaticns to ascertain that the algorithms can handle equations as discussed in section 6.1.

Table 5

Results of Systems of Equations

M	N	SURRG1	SURRGR	SURRG2	SURRG3	AGMON1	AGMON2
2	3	2	2	1	1	14	500
		28	28	23	28	77	2509
		4/4	4/4	4/4	4/4	4/4	0/4
5	3	2	2	1	1	5	500
		58	58	50	58	57	4818
		4/4	4/4	4/4	4/4	4/4	0/4
3	3	9	9	29	4	92	500
		206	202	655	166	565	3612
		4/4	4/4	4/4	4/4	4/4	0/4
4	4	30	14	8	8	58	500
		903	432	272	435	486	4820
		4/4	4/4	4/4	4/4	4/4	0/4

f) Linear Programming

Eight linear programming problems were tested, the last 2 of which are the nutrition problem (short and long) in Dantzig(4). For each LP problem we ran, we verified our solution by using the simplex routine provided in the IMSL package. The results which follow used a scaled version of

Table 6

Results of Linear Programming Problems

M	N	SURRG1	SURRGR	SURRG2	SURRG3	AGMON1	AGMON2
7	3	178	12	30	2	220	500
		4074	290	752	106	1068	2127
		3/4	4/4	4/4	4/4	4/4	0/4
9	4	114	14	27	7	230	500
		2837	372	786	384	1209	2419
		4/4	4/4	4/4	4/4	3/4	0/4
13	6	445	214	497	244	500	500
		16602	8194	20548	19195	4007	4073
		1/4	4/4	1/4	4/4	0/4	0/4
15	7	432	329	389	354	450	500
		20122	15027	19892	33577	4651	5133
		1/4	2/4	1/4	2/4	1/4	0/4
15	7	375	191	500	38	500	500
		17354	8903	24711	3330	5498	5369
		1/4	3/4	0/4	4/4	0/4	0/4
17	8	500	355	394	197	500	500
		25351	18241	22120	20556	5641	5499
		0/4	3/4	2/4	4/4	0/4	0/4
29	14		767	867	591	1000	
		*	64026	78680	104073	19121	*
			3/4	1/4	4/4	0/4	
59	29		979	1000	763	1000	
		*	164872	190418	275079	30776	*
			1/4	0/4	4/4	0/4	

* = not run

the data given in Dantzig(4). The reason for this is discussed in the next subsection.

Although some of the algorithms did not get a solution with some starting points, we noticed that the x got at the 500th iteration was close to a solution. Therefore for a large problem like the nutrition case, we increased the iteration cut-off from 500 to 1000 and ran surrogate R, 2, 3 and Agmon1. As seen from the table Surrogate-3 got 4/4 for the short and long lists. Favorite starting points were not regular with any of the algorithms. It varied with the problems. Because of space, we have not shown the results got with random starting points, but we must point out that they were not better than starting points 1 through 4 listed above.

Phases for LP Problems

We tried to vary the style of choice of constraints and found out that using the most violated manifold gives the best speed. In the LP problems, there was a small variation that improved the speed a little. We shall give the variation here.

I. 1-Phase Algorithm

1) Get a feasible solution to the problem as given as given in (6.1.1);
2) Stop;

II. 2-Phase Algcrithm

1) Get a feasible solution to the primal and dual of (6.1.1);
2) If z-feasible, stop;
3) Project onto the z-hyperplane;
4) Gc to 1);

In the 2-phase algorithm, feasibility got priority over optimality. We got a faster convergence than with the 1-phase case. However, the differece was nct much. The results in the tables are those of 2-phase algcrithm.

8.3 Scaling

8.3.1 Data scaling

Definition: An overdominant element in A is an element whose absolute value is much greater than the sum of the absolute values of the other elements in its row or column. That is, if for some r and k,

$$|a_{rk}| >> \sum_{\substack{i=1 \\ i\neq r}}^{m} |a_{ik}| \qquad \text{or} \qquad |a_{rk}| >> \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{rj}|$$

then a_{rk} is (row or column) overdominant.

A row overdominant element can cause error. After normalizing the row, the small elements become too small. As a result the distance may fall within the tolerant region, thereby causing the program to terminate with inacurate results. This was our observation in the nutrition problem. Let us examine the data for that problem.

Nutrition (i) / Commodity (j)	Calories (1000)	Calcium (grams)	Vitamin A (10 I.U)	Riboflanin (mg.)	Ascorbic Acid (mg.)	Cost/Unit (dollars)
Wheat flour (enriched)	44.7	2.0	0	33.3	0	1.00
Evaporated milk (can)	8.4	15.1	26.0	23.5	60	1.00
Cheese (Cheddar)	7.4	16.4	28.1	10.3	0	1.00
Liver (beef)	2.2	0.2	169.2	50.8	525	1.00
Cabbage	2.6	4.0	7.2	4.5	5369	1.00
Spinach	1.1	0	918.4	13.8	2755	1.00
Sweet potatoes	9.6	2.7	290.7	5.4	1912	1.00
Lima beans (dried)	17.4	3.7	5.1	38.2	0	1.00
Navy beans (dried)	26.9	11.4	0	24.6	0	1.00
Daily allowace	3.0	0.8	5.0	2.7	75	

The commodity entries are column elemnts fcr the primal but rcw elements for the dual. The element, 5369, in the cabbage entries is column cverdominant. This may not have been a problem if this was not an LP problem. As it is, that element is row overdominant with respect to the dual. When we ran the program with the data as given, we had very fast convergence, with feasible primal but inacurate dual solution. We ran it again with riboflavin and ascorbic acid in grams, and the resulting effect was slow convergence. Finally, we scaled down some of the data so that the maximum number of digits before any decimal point was 2. We

then got acceptable results. Therefcre, if there exists a column overdominant element in the data and the problem is an LP, the row in which it occurs should be scaled down. The choice of scale factors depends cn the entire data. Our choice of 0.01 and 0.1 for ascorbic acid and vitamin A respectively, was guided by the fact that 3 out of the 5 nutrients have their entries in a maximum of 2 digits before decimal point. The scale factors and the tolerant value, delta, must balance each other well. The smaller the scale factor, the bigger the delta may have tc be.

8.3.2 Over-relaxation

Another thing that can be varied is the distance tc the projection. This is called relaxation. Consider the following.

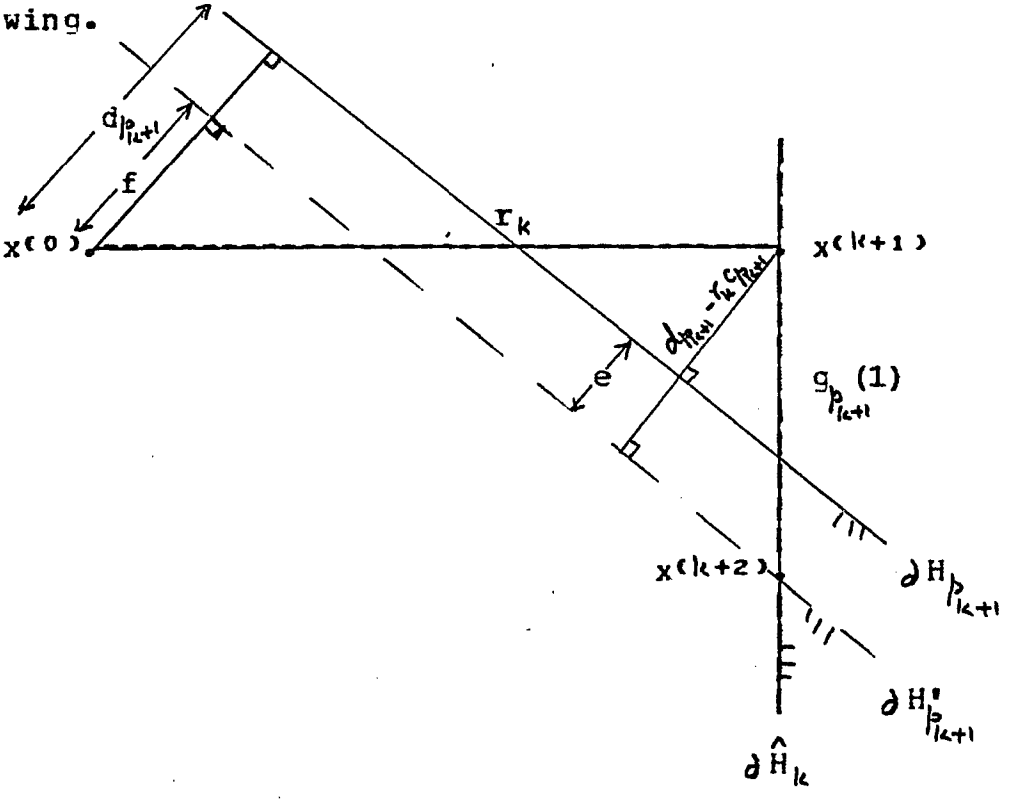


Fig. 8.2 Translate of $H_{p_{k+1}}$

Let i) $H^{*}_{p_{k+1}}$ = a translate of $H_{p_{k+1}}$

ii) e = displacement resulting from the translation

$= w(d_{p_{k+1}} - r_k c_{p_{k+1}})$ for some w

iii) f = distance of $x^{(0)}$ from $H^{*}_{p_{k+1}}$

Then $f = d_{p_{k+1}} + e$

$= d_{p_{k+1}} + w(d_{p_{k+1}} - r_k c_{p_{k+1}})$ by def. (ii) .....(1)

With respect to the translate, $H^{*}_{p_{k+1}}$

$$g_{p_{k+1}}(\lambda) = \frac{f - r_k c_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}} \quad \text{by step 7}$$

$$= \frac{d_{p_{k+1}} + w(d_{p_{k+1}} - r_k c_{p_{k+1}}) - r_k c_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}} \quad \text{by (1)}$$

$$= (1+w)\frac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}}$$

Set $\lambda = 1+w$

$$\therefore\ g_{p_{k+1}}(\lambda) = \lambda\, g_{p_{k+1}}(1) \qquad (8.3.1)$$

To meet the positivity requirement, λ has to be greater than 0. That is,

$$\lambda > 0 \qquad .....(2)$$

Also $$\hat{g}_{p_{k+1}}(\lambda) = \frac{r_k - f c_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}}$$

$$= \frac{r_k - d_{p_{k+1}} c_{p_{k+1}} - w(d_{p_{k+1}} - r_k c_{p_{k+1}}) c_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}} \quad \text{by (1)}$$

$$= \frac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}} - w c_{p_{k+1}} \frac{d_{p_{k+1}} - r_k c_{p_{k+1}}}{\sqrt{1 - c^2_{p_{k+1}}}}$$

$$\therefore\ \hat{g}_{p_{k+1}}(\lambda) = \hat{g}_{p_{k+1}}(1) - (\lambda-1)\, c_{p_{k+1}} g_{p_{k+1}}(1) \qquad (8.3.2)$$

where $(\lambda-1)\, c_{p_{k+1}} < \hat{g}_{p_{k+1}}(1) / g_{p_{k+1}}(1)$ .....(3)

From (2) and (3), we have

$$\text{for } c_{p_{k+1}} > 0, \quad 0 < \lambda < 1 + \frac{\hat{g}_{p_{k+1}}(1)}{c_{p_{k+1}} g_{p_{k+1}}(1)}$$

$$\text{for } c_{p_{k+1}} = 0, \quad 0 < \lambda$$

$$\text{for } c_{p_{k+1}} < 0, \quad \max\left(0,\ 1 + \frac{\hat{g}_{p_{k+1}}(1)}{c_{p_{k+1}} g_{p_{k+1}}(1)}\right) < \lambda.$$

But
$$\frac{\hat{g}_{p_{k+1}}(1)}{c_{p_{k+1}} g_{p_{k+1}}(1)} = \frac{1}{c_{p_{k+1}}} \frac{r_k - d_{p_{k+1}} c_{p_{k+1}}}{d_{p_{k+1}} - r_k c_{p_{k+1}}}$$

$$= \frac{1}{c_{p_{k+1}}}\left(1 + \frac{(r_k - d_{p_{k+1}})(1+c_{p_{k+1}})}{d_{p_{k+1}} - r_k c_{p_{k+1}}}\right)$$

$$\begin{cases} > 1 & \text{if } c_{p_{k+1}} > 0 \\ < -1 & \text{if } c_{p_{k+1}} < 0 \end{cases}$$

$$\therefore\ \frac{\hat{g}_{p_{k+1}}(1)}{c_{p_{k+1}} g_{p_{k+1}}(1)} + 1 \begin{cases} > 2 & \text{if } c_{p_{k+1}} > 0 \\ < 0 & \text{if } c_{p_{k+1}} < 0 \end{cases} \qquad .....(4)$$

Therefore from (2), (3) and (4) the valid range is

$$\begin{cases} 0 < \lambda < 1 + \dfrac{\hat{g}_{p_{k+1}}(1)}{c_{p_{k+1}} g_{p_{k+1}}(1)} & \text{if } c_{p_{k+1}} > 0 \\ 0 < \lambda & \text{if } c_{p_{k+1}} \le 0. \end{cases} \qquad (8.3.3)$$

The different values for different types of relaxation are

$0 < \lambda < 1$	under-relaxation
$\lambda = 1$	projection
$1 < \lambda < 2$	over-projection
$\lambda = 2$	reflection
$2 < \lambda$	over-reflection.

The last 3 are called over-relaxation. We tried over-

relaxation ($1 \leq \lambda \leq 2$) on our test problems with a variable parameter that is a function of $c_{p_{k+1}}$,

$$\lambda = \begin{cases} 1+c_{p_{k+1}} & \text{if } c_{p_{k+1}} > 0 \\ 1 & \text{if } c_{p_{k+1}} \leq 0 \end{cases}$$

The resulting effect was a mixed one. Convergence improved for some and got worse for others. There were some that were not affected. The results in the tables above are for pure projection, $\lambda = 1$.

Warning

In doing over-relaxation, extra care must be taken to ensure a valid and 'reasonable' parameter value. There are two problems that could be encountered.

1) Inconsistency condition could be falsely raised for some $\lambda \neq 1$.
2) If the feasible set is a bounded polytope and we over-relax beyond this set, convergence may no longer be possible for Surrogate-1 since r_k is increasing without being reset.

The 1st problem may arise if there are 2 parallel but intersecting half-spaces. The translate of any one of them will still be parallel to, but not necessarily intersecting with, the other half-space. Consequently, inconsistent test will not be used for $\lambda > 1$. The 2nd problem will not happen with the re-initializing surrogates because r_k is reset at certain stages as iteration progresses.

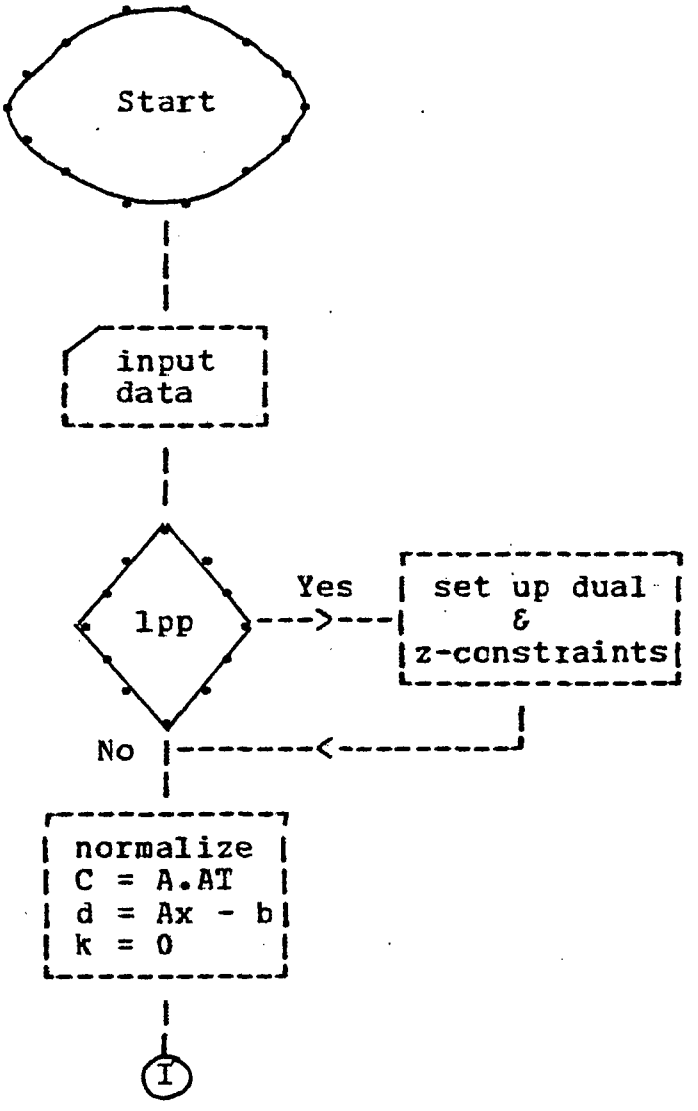
CONCLUDING REMARKS

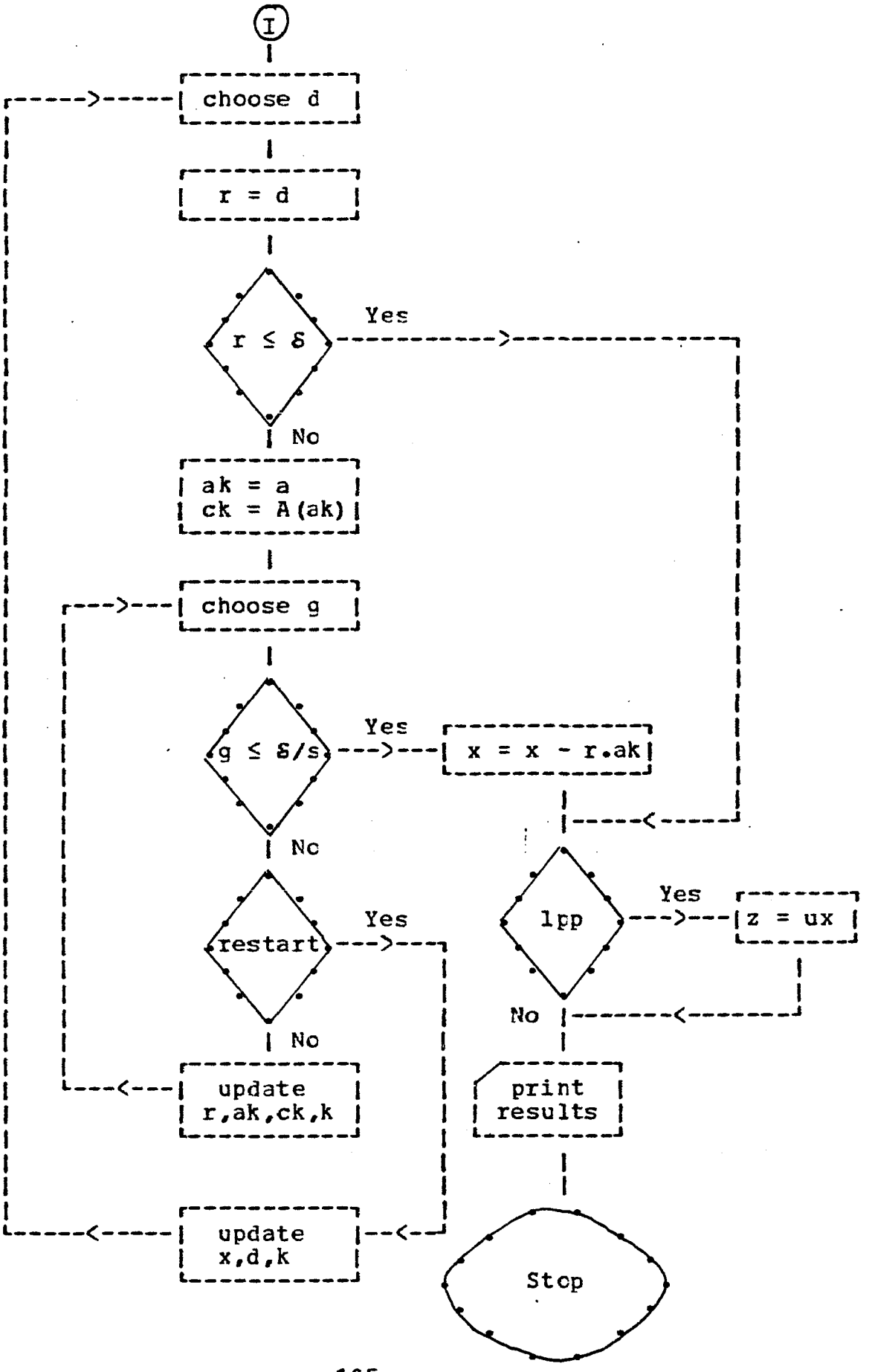
On the algorithms

Although Surrogate-3 does more work than the rest, we get more solutions with it than with the others. The amount of work it does is a liability only when it takes long to reach a solution. We observed that in cases where it found a solution early, the amount of work it did was smaller than that done by any of the others. Surrogate-R is our second best. In fact, from the amount of work it does, it may be considered over Surrogate-3. It varies the number of constraints it uses before re-initialization. The 3rd best is Surrogate-2. Surrogate-1 is very slow but it has served as a means of getting better algorithms.

One advantage of the surrogate methods is that the coefficient matrix A is not updated. For this reason, we can economize space by storing only the nonzero elements of A. The dot-products, $a^i.a^j$, may be computed when needed. Another advantage of the algorithms is that they are easy to restart. Although they are not competitive with the simplex method, they are quite reliable.

APPENDIX


Start
input data
lpp
Yes
set up dual & z-constraints
No
normalize
C = A.AT
d = Ax - b
k = 0
I


I
choose d
r = d
r ≤ δ
Yes
No
ak = a
ck = A(ak)
choose g
g ≤ δ/s
Yes
x = x - r.ak
No
restart
Yes
No
update
r,ak,ck,k
update
x,d,k
lpp
Yes
z = ux
No
print
results
Stop

LIST OF SUBRCUTINES

```
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
      INTEGER NAME(7,2), TAB(42,8)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
C M       NO. OF CONSTRAINTS                                        C
C N       NO. OF VARIABLES                                          C
C M1      NO. INEQUALITIES                                          C
C LSTROW=M OR M+N  IF LP PROBLEM                                    C
C LSTCOL=N OR M+N  IF LP PROBLEM                                    C
C A       M BY N COEFFICIENT MATRIX OF THE PROBLEM                  C
C B       THE R.H.S. OF THE PROBLEM, SIZE = LSTROW                  C
C X,X0    SOLUT. VECTOR, STARTING POINT,  SIZE = LSTCOL             C
C D       DIST. VECTOR FROM X0 TO HYPS., SIZE=LSTROW                C
C AT      THE TRANSPOSE OF A                                        C
C U       N-VECTOR OPTIMAL FUCTION COEFFICIENTS                     C
C C       VECTOR HOLDING THE UPPER TRIANGLE OF A.AT                 C
C             SIZE=M(M-1)/2 OR M(M+1)/2 + N(N+1)/2 IF LPP           C
C AK      SURROGATE NORMAL, SIZE = LSTCOL                           C
C CK      DOT-PROD VEC. FOR AK AND GIVEN NORMALS, SIZE=LSTROW
C AZ      THE Z-COSTRAINT NORMAL, SIZE = M+N                        C
C DEL     TOLERANT ERROR                                            C
C R       DISTANCE FROM X0 TO THE SURROGATE HYPERPLANE              C
C H0,H1   MULTIPLIERS FOR SURROG. NORMAL AND THE CHOSEN, RESP.
C Z,DZ    OPTIMAL FUNCTION, DIST. FROM X0 TO Z-HYPERPLANE           C
C CZ      DOT-PRODUCT OF AK AND AZ                                  C
C LIM     MAXIMUM ITERATION ALLOWED                                 C
C LPP     SIGNAL THAT THE PROBLEM IS LINEAR PROGRAMMING             C
C OVRLX   SIGNAL THAT OVER-RELAXATION IS REQUIRED                   C
C IZ      TOTAL NO. OF CONSTRAINTS IN LP PROBLEM                    C
C EPS     WORKING SMALL NO.                                         C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      READ(5,5) ((NAME(I,J),J = 1,2),I = 1,7)
    5 FORMAT(7(A4,A3))
      DO 15 I = 1,42
       DO 10 J = 1,8
   10  TAB(I,J) = 0
   15 CONTINUE
      ISEED = 7359
      LIM = 500
      DEL = 1.5D-04
      EPS = 0.1D 00
      LPP = .FALSE.
      OVRLX = .TRUE.
```

```
C        IF (.TRUE.) GO TO 41
C
      WRITE(6,20)
   20 FORMAT(35X,'TABLE  A'//30X,'LINEAR INEQUALITIES.'/
     1       30X,'-------------------'/
     2       25X,'TODD''S EXAMPLE AND ITS EXPANSION.'/
     3       25X,'--------------------------------'/
     4    5X,'MATRIX METHOD ITERATIONS MUL/DIV',15X,'X'/
     5    5X,'------ ------ ---------- -------',15X,'-')
C
      IROW = -2
      DO 25 N = 3,6
       CALL TODHYP(1)
       LSTROW = M
       LSTCOL = N
       M1 = M
       CALL NRMROW
       CALL DOTPRD
       IROW = IROW + 3
       TAB(IROW,1) = M
       TAB(IROW,2) = N
C
       CALL PRINT(.TRUE.,NAME1,NAME2,0)
       CALL SURRG1(K)
       CALL PRINT(.FALSE.,NAME(1,1),NAME(1,2),K)
       TAB(IROW,3) = K
       TAB(IROW+1,3) = MULT
       IF (K .LT. LIM) TAB(IROW+2,3) = 1
C
       CALL SURRGR(K)
       CALL PRINT(.FALSE.,NAME(2,1),NAME(2,2),K)
       TAB(IROW,4) = K
       TAB(IROW+1,4) = MULT
       IF (K .LT. LIM) TAB(IROW+2,4) = 1
C
       CALL SURRG2(K)
       CALL PRINT(.FALSE.,NAME(3,1),NAME(3,2),K)
       TAB(IROW,5) = K
       TAB(IROW+1,5) = MULT
       IF (K .LT. LIM) TAB(IROW+2,5) = 1
C
       CALL SURRG3(K)
       CALL PRINT(.FALSE.,NAME(4,1),NAME(4,2),K)
       TAB(IROW,6) = K
       TAB(IROW+1,6) = MULT
       IF (K .LT. LIM) TAB(IROW+2,6) = 1
C
       CALL AGMON(K,1.0D 00)
       CALL PRINT(.FALSE.,NAME(5,1),NAME(5,2),K)
       TAB(IROW,7) = K
       TAB(IROW+1,7) = MULT
       IF (K .LT. LIM) TAB(IROW+2,7) = 1
C
       CALL AGMON(K,2.0D 00)
       CALL PRINT(.FALSE.,NAME(6,1),NAME(6,2),K)
```

```
            TAB(IROW,8) = K
            TAB(IROW+1,8) = MULT
            IF (K .LT. LIM) TAB(IROW+2,8) = 1
            WRITE(6,360)
         25 CONTINUE
      C
            WRITE(6,30)
         30 FORMAT(1H1/35X,'TABLE  B'//30X,'LINEAR INEQUALITIES.'/
           1           30X,'-------------------'/
           2           27X,'THE TRAPEZOIDAL HYPERCUBE.'/
           3           27X,'-------------------------'/
           4    5X,'MATRIX METHOD ITERATIONS MUL/DIV',15X,'X'/
           5    5X,'------ ------ ---------- -------',15X,'-')
      C
            IROW = 10
            DO 40 N = 3,6
             CALL TODHYP(2)
   LSTROW = M
            LSTCOL = N
            M1 = M
            CALL NRMROW
            CALL DOTPRD
            IROW = IROW + 3
            TAB(IROW,1) = M
            TAB(IROW,2) = N
      C
      C USE 4 DIFFERENT X0.
      C
            DO 35 KASE = 1,4
             CALL SETX0(KASE)
             CALL PRINT(.TRUE.,NAME1,NAME2,0)
             CALL SURRG1(K)
             CALL PRINT(.FALSE.,NAME(1,1),NAME(1,2),K)
             TAB(IROW,3) = TAB(IROW,3) + K
             TAB(IROW+1,3) = TAB(IROW+1,3) + MULT
             IF (K .LT. LIM) TAB(IROW+2,3) = TAB(IROW+2,3) + 1
      C
             CALL SURRGR(K)
             CALL PRINT(.FALSE.,NAME(2,1),NAME(2,2),K)
             TAB(IROW,4) = TAB(IROW,4) + K
             TAB(IROW+1,4) = TAB(IROW+1,4) + MULT
             IF (K .LT. LIM) TAB(IROW+2,4) = TAB(IROW+2,4) + 1
      C
             CALL SURRG2(K)
             CALL PRINT(.FALSE.,NAME(3,1),NAME(3,2),K)
             TAB(IROW,5) = TAB(IROW,5) + K
             TAB(IROW+1,5) = TAB(IROW+1,5) + MULT
             IF (K .LT. LIM) TAB(IROW+2,5) = TAB(IROW+2,5) + 1
      C
             CALL SURRG3(K)
             CALL PRINT(.FALSE.,NAME(4,1),NAME(4,2),K)
             TAB(IROW,6) = TAB(IROW,6) + K
             TAB(IROW+1,6) = TAB(IROW+1,6) + MULT
             IF (K .LT. LIM) TAB(IROW+2,6) = TAB(IROW+2,6) + 1
      C
```

```
      CALL AGMON(K,1.0D 00)
      CALL PRINT(.FALSE.,NAME(5,1),NAME(5,2),K)
      TAB(IROW,7) = TAB(IROW,7) + K
      TAB(IROW+1,7) = TAB(IROW+1,7) + MULT
      IF (K .LT. LIM) TAB(IROW+2,7) = TAB(IRCW+2,7) + 1
C
      CALL AGMON(K,2.0D 00)
      CALL PRINT(.FALSE.,NAME(6,1),NAME(6,2),K)
      TAB(IROW,8) = TAB(IRCW,8) + K
      TAB(IROW+1,8) = TAB(IROW+1,8) + MULT
      IF (K .LT. LIM) TAB(IROW+2,8) = TAB(IRCW+2,8) + 1
      WRITE(6,360)
   35 CONTINUE
   40 CONTINUE
C
C COMPUTE AVERAGE OF THE WORK DONE
C
      DO 50 I = 13,22,3
       DO 45 J = 3,8
        TAB(I,J) = TAB(I,J) / 4
   45  TAB(I+1,J) = TAB(I+1,J) / 4
   50 CONTINUE
C
      WRITE(6,55)
   55 FORMAT(1H1/////45X,'TABLE 1'/45X,'-------'//)
      WRITE(6,60)
   60 FORMAT(10X,'.....................................
     1.....................')
      WRITE(6,65) ((NAME(I,1),NAME(I,2)),I = 1,6)
   65 FORMAT(10X,'. PROBLEM .  M .  N .',6(1X,A4,A3,'.'))
      WRITE(6,60)
      DO 70 I = 1,22,3
       WRITE(6,75) (TAB(I,J),J = 1,8)
       WRITE(6,80) (TAB(I+1,J),J = 3,8)
       IF (I .LT. 13) WRITE(6,85) (TAB(I+2,J),J = 3,8)
       IF (I .GE. 13) WRITE(6,90) (TAB(I+2,J),J = 3,8)
       IF (I .EQ. 4) WRITE(6,95)
       IF (I .EQ. 16) WRITE(6,100)
       IF (I .EQ. 10 .OR. I .EQ. 22) WRITE(6,60)
       IF (I.NE.4 .AND. I.NE.10 .AND. I.NE.16 .AND. I.NE.22)
     1 WRITE(6,105)
   70 CONTINUE
   75 FORMAT(10X,'.',9X,'.',2(I3,' .'),6(I7,' .'))
   80 FORMAT(10X,'.',9X,'.',2(3X,' .'),6(I7,' .'))
   85 FORMAT(10X,'.',9X,'.',2(3X,' .'),6(I5,'/1 .'))
   90 FORMAT(10X,'.',9X,'.',2(3X,' .'),6(I5,'/4 .'))
   95 FORMAT(10X,'.    A    .....................................
     1.....................')
  100 FORMAT(10X,'.    B    .....................................
     1.....................')
  105 FORMAT(10X,'.         .....................................
     1.....................')
      WRITE(6,110)
  110 FORMAT(/30X,'A = TODD''S PROBLEM'/
     1        30X,'B = TRAPEZOIDAL HYPERCUBE PROBLEM')
```

```
C                    IF (.TRUE.) STOP
C
C TEST ON RANDOMLY GENERATED PROBLEM.
C
  411 DO 120 I = 1,24
       DO 115 J = 1,8
  115  TAB(I,J) = 0
  120 CONTINUE
      IROW = -2
      DO 130 N = 20,30,10
       LSTCOL = N
       DO 125 M = 30,90,30
        LSTROW = M
        M1 = M
        CALL GENAB(ISEED)
        CALL NRMROW
        CALL DOTPRD
        IROW = IROW + 3
        TAB(IROW,1) = M
        TAB(IROW,2) = N
C
        CALL PRINT(.TRUE.,NAME1,NAME2,C)
        CALL SURRG1(K)
        CALL PRINT(.FALSE.,NAME(1,1),NAME(1,2),K)
        TAB(IROW,3) = K
        TAB(IROW+1,3) = MULT
        IF (K .LT. LIM) TAB(IROW+2,3) = 1
C
        CALL SURRGR(K)
        CALL PRINT(.FALSE.,NAME(2,1),NAME(2,2),K)
        TAB(IROW,4) = K
        TAB(IROW+1,4) =  MULT
        IF (K .LT. LIM) TAB(IROW+2,4) = 1
C
        CALL SURRG2(K)
        CALL PRINT(.FALSE.,NAME(3,1),NAME(3,2),K)
        TAB(IROW,5) = K
        TAB(IROW+1,5) = MULT
        IF (K .LT. LIM) TAB(IROW+2,5) = 1
C
        CALL SURRG3(K)
        CALL PRINT(.FALSE.,NAME(4,1),NAME(4,2),K)
        TAB(IROW,6) = K
        TAB(IROW+1,6) = MULT
        IF (K .LT. LIM) TAB(IROW+2,6) = 1
C
        CALL AGMON(K,1.0D 00)
        CALL PRINT(.FALSE.,NAME(5,1),NAME(5,2),K)
        TAB(IROW,7) = K
        TAB(IROW+1,7) = MULT
        IF (K .LT. LIM) TAB(IROW+2,7) = 1
C
        CALL AGMON(K,2.0D 00)
        CALL PRINT(.FALSE.,NAME(6,1),NAME(6,2),K)
        TAB(IROW,8) = K
```

```
        TAB(IROW+1,8) = MULT
        IF (K .LT. LIM) TAB(IROW+2,8) = 1
  125  CONTINUE
  130 CONTINUE
C
C EPSILON-CUBE.
C
      IROW = 16
      DO 210 N = 10,15,5
       LSTCOL = N
       INITM = 2 * N
       DO 205 I = INITM,90,20
        IF (I .GT. INITM) CALL EPCUBE(ISEED,INITM,20)
        IF (I .GT. INITM) GO TO 150
        M = I
        DO 140 I2 = 1,M
         DO 135 J = 1,N
  135    A(I2,J) = 0.0D 00
  140   B(I2) = EPS
        DO 145 J = 1,N
         I2 = 2 * J
         A(I2-1,J) = 1.0D 00
  145   A(I2,J) = -1.0D 00
  150   LSTROW = M
        M1 = M
        CALL DOTPRD
        IROW = IROW + 3
        TAB(IROW,1) = M
        TAB(IROW,2) = N
C
C USE 4 DIFFERENT X0 FOR EACH M.
C
        DO 200 KASE = 1,4
         GO TO (155,165,175,185), KASE
  155    DO 160 J = 1,N
          CALL RANDU(ISEED,ISEED,X0(J))
          X0(J) = 4.0D 00 * X0(J) - 2.0D 00
          X0(J) = DMAX1(X0(J),-1.0D 00)
          X0(J) = DMIN1(X0(J),1.0D 00)
  160    X0(J) = 1.0D 03 * X0(J)
         GO TO 195
  165    DO 170 J = 1,N
  170    X0(J) = 1.0D 03
         GO TO 195
  175    DO 180 J = 1,N
  180    X0(J) = -X0(J)
         GO TO 195
  185    DO 190 J = 1,N
          CALL RANDU(ISEED,ISEED,R)
          I1 = MOD(IDINT(1.0D 03 * R),2)
          IF (I1 .EQ. 1) X0(J) = -X0(J)
  190    CONTINUE
C
  195    CALL PRINT(.TRUE.,NAME1,NAME2,0)
         CALL SURRG1(K)
```

```
          CALL PRINT(.FALSE.,NAME(1,1),NAME(1,2),K)
          TAB(IROW,3) = TAB(IROW,3) + K
          TAB(IROW+1,3) = TAB(IROW+1,3) + MULT
          IF (K .LT. LIM) TAB(IROW+2,3) = TAB(IROW+2,3) + 1
C
          CALL SURRGR(K)
          CALL PRINT(.FALSE.,NAME(2,1),NAME(2,2),K)
          TAB(IROW,4) = TAB(IROW,4) + K
          TAB(IROW+1,4) = TAB(IROW+1,4) + MULT
          IF (K .LT. LIM) TAB(IROW+2,4) = TAB(IROW+2,4) + 1
C
          CALL SURRG2(K)
          CALL PRINT(.FALSE.,NAME(3,1),NAME(3,2),K)
          TAB(IROW,5) = TAB(IROW,5) + K
          TAB(IROW+1,5) = TAB(IROW+1,5) + MULT
          IF (K .LT. LIM) TAB(IROW+2,5) = TAB(IROW+2,5) + 1
C
          CALL SURRG3(K)
          CALL PRINT(.FALSE.,NAME(4,1),NAME(4,2),K)
          TAB(IROW,6) = TAB(IROW,6) + K
          TAB(IROW+1,6) = TAB(IROW+1,6) + MULT
          IF (K .LT. LIM) TAB(IROW+2,6) = TAB(IROW+2,6) + 1
C
          CALL AGMON(K,1.0D 00)
          CALL PRINT(.FALSE.,NAME(5,1),NAME(5,2),K)
          TAB(IROW,7) = TAB(IROW,7) + K
          TAB(IROW+1,7) = TAB(IROW+1,7) + MULT
          IF (K .LT. LIM) TAB(IROW+2,7) = TAB(IROW+2,7) + 1
C
          CALL AGMON(K,2.0D 00)
          CALL PRINT(.FALSE.,NAME(6,1),NAME(6,2),K)
          TAB(IROW,8) = TAB(IROW,8) + K
          TAB(IROW+1,8) = TAB(IROW+1,8) + MULT
          IF (K .LT. LIM) TAB(IROW+2,8) = TAB(IROW+2,8) + 1
  200    CONTINUE
  205   CONTINUE
  210  CONTINUE
C
C COMPLETE AVERAGING.
C
       DO 220 I = 19,40,3
        DO 215 J = 3,8
         TAB(I,J) = TAB(I,J) / 4
  215   TAB(I+1,J) = TAB(I+1,J) / 4
  220  CONTINUE
C
       WRITE(6,225)
  225  FORMAT(1H1//45X,'TABLE 2'/45X,'-------'//)
       WRITE(6,60)
       WRITE(6,65)  ((NAME(I,1),NAME(I,2)),I = 1,6)
       WRITE(6,60)
       DO 230 I = 1,40,3
        WRITE(6,75)  (TAB(I,J),J = 1,8)
        WRITE(6,80)  (TAB(I+1,J),J = 3,8)
        IF (I .LT. 19) WRITE(6,85)  (TAB(I+2,J),J = 3,8)
```

```
      IF (I .GE. 19) WRITE(6,90) (TAB(I+2,J),J = 3,8)
      IF (I .EQ. 7) WRITE(6,235)
      IF (I .EQ. 28) WRITE(6,240)
      IF (I .EQ. 16 .OR. I .EQ. 40) WRITE(6,60)
      IF (I.NE.7 .AND. I.NE.16 .AND. I.NE.28 .AND. I.NE.40)
     1 WRITE(6,105)
  230 CONTINUE
  235 FORMAT(10X,'.    C    .....................................
     1.....................')
  240 FORMAT(10X,'.    D    .....................................
     1.....................')
      WRITE(6,245)
  245 FORMAT(/30X,'C = RANDOMLY GENERATED PROBLEM'/
     1        30X,'D = EPSILON-CUBE PROBLEM')
          IF (.TRUE.) STOP
C
C TEST ON LINEAR EQUATIONS.
C
   41  WRITE(6,250)
  250 FORMAT(1H1/30X,'LINEAR EQUATIONS.'/30X,'------------'/
     1   5X,'MATRIX METHOD ITERATIONS MUL/DIV',24X,'X'/
     2   5X,'------ ------ ---------- -------',24X,'-')
C
  414  DO 260 I = 1,42
       DO 255 J = 1,8
  255  TAB(I,J) = 0
  260 CONTINUE
      IROW = -2
C
      DO 275 NO = 1,4
       READ(5,335) M,N,M1
       DO 265 I = 1,M
  265  READ(5,340) (A(I,J),J = 1,N)
       READ(5,340) (B(I),I = 1,M)
       LSTROW = M
       LSTCOL = N
C
       CALL NRMROW
       CALL DOTPRD
       IROW = IROW + 3
       TAB(IROW,1) = M
       TAB(IROW,2) = N
C
C USE 4 DIFFERENT X0.
C
       DO 270 KASE = 1,4
        CALL SETX0(4+KASE)
        CALL PRINT(.TRUE.,NAME1,NAME2,0)
        CALL SURRG1(K)
        CALL PRINT(.FALSE.,NAME(1,1),NAME(1,2),K)
        TAB(IROW,3) = TAB(IROW,3) + K
        TAB(IROW+1,3) = TAB(IROW+1,3) + MULT
        IF (K .LT. LIM) TAB(IROW+2,3) = TAB(IROW+2,3) + 1
C
```

```
      CALL SURRGR(K)
      CALL PRINT(.FALSE.,NAME(2,1),NAME(2,2),K)
      TAB(IROW,4) = TAB(IROW,4) + K
      TAB(IROW+1,4) = TAB(IROW+1,4) + MULT
      IF (K .LT. LIM) TAB(IROW+2,4) = TAB(IROW+2,4) + 1
C
      CALL SURRG2(K)
      CALL PRINT(.FALSE.,NAME(3,1),NAME(3,2),K)
      TAB(IROW,5) = TAB(IROW,5) + K
      TAB(IROW+1,5) = TAB(IROW+1,5) + MULT
C
      IF (K .LT. LIM) TAB(IROW+2,5) = TAB(IROW+2,5) + 1
      CALL SURRG3(K)
      CALL PRINT(.FALSE.,NAME(4,1),NAME(4,2),K)
      TAB(IROW,6) = TAB(IROW,6) + K
      TAB(IROW+1,6) = TAB(IROW+1,6) + MULT
      IF (K .LT. LIM) TAB(IROW+2,6) = TAB(IROW+2,6) + 1
C
      CALL AGMON(K,1.0D 00)
      CALL PRINT(.FALSE.,NAME(5,1),NAME(5,2),K)
      TAB(IROW,7) = TAB(IROW,7) + K
      TAB(IROW+1,7) = TAB(IROW+1,7) + MULT
      IF (K .LT. LIM) TAB(IROW+2,7) = TAB(IROW+2,7) + 1
C
      CALL AGMON(K,2.0D 00)
      CALL PRINT(.FALSE.,NAME(6,1),NAME(6,2),K)
      TAB(IROW,8) = TAB(IROW,8) + K
      TAB(IROW+1,8) = TAB(IROW+1,8) + MULT
      IF (K .LT. LIM) TAB(IROW+2,8) = TAB(IROW+2,8) + 1
      WRITE(6,360)
  270 CONTINUE
  275 CONTINUE
C
C TEST LP PROBLEMS.
C
      LPP = .TRUE.
      WRITE(6,280)
  280 FORMAT(1H1/30X,'LINEAR PROGRAMMING PROBLEMS.'/
     1           30X,'--------------------------'/
     2 5X,'MATRIX METHOD ITERATIONS MUL/DIV',5X,'Z',24X,'X'/
     3 5X,'------ ------ ---------- -------',5X,'-',24X,'-')
      IROW = 10
      DO 295 NO = 1,8
       IF (NO .EQ. 7) LIM = 1000
       READ(5,335) M,N,M1
       LAST = MINO(N,6)
       READ(5,340) (U(J),J = 1,LAST)
       WRITE(6,345) (U(J),J = 1,LAST)
       IF (LAST .LT. N) READ(5,340) (U(J),J = 7,N)
       IF (LAST .LT. N) WRITE(6,350) (U(J),J = 7,N)
       DO 285 I = 1,M
        READ(5,340) (A(I,J),J = 1,LAST)
        IF (LAST .LT. N) READ(5,340) (A(I,J),J = 7,N)
  285  CONTINUE
       READ(5,340) (B(I),I = 1,M)
```

```
      MPLUS1 = M + 1
      LSTROW = M + N
      LSTCOL = LSTROW
      IZ = 2 * LSTROW + 1
      CALL NRMROW
      CALL DOTPRD
      IROW = IROW + 3
      TAB(IROW,1) = IZ
      TAB(IROW,2) = LSTROW
C
C USE 4 DIFFERENT XO.
C
      DO 290 KASE = 1,4
       CALL SETXO(4+KASE)
       CALL PRINT(.TRUE.,NAME1,NAME2,0)
         IF (LIM .GT. 500) GO TO 286
       CALL SURRG1(K)
       CALL PRINT(.FALSE.,NAME(1,1),NAME(1,2),K)
       TAB(IROW,3) = TAB(IROW,3) + K
       TAB(IROW+1,3) = TAB(IROW+1,3) + MULT
       IF (K .LT. LIM) TAB(IROW+2,3) = TAB(IROW+2,3) + 1
C
  286     CONTINUE
       CALL SURRGR(K)
       CALL PRINT(.FALSE.,NAME(2,1),NAME(2,2),K)
       TAB(IROW,4) = TAB(IROW,4) + K
       TAB(IROW+1,4) = TAB(IROW+1,4) + MULT
       IF (K .LT. LIM) TAB(IROW+2,4) = TAB(IRCW+2,4) + 1
C
       CALL SURRG2(K)
       CALL PRINT(.FALSE.,NAME(3,1),NAME(3,2),K)
       TAB(IROW,5) = TAB(IROW,5) + K
       TAB(IROW+1,5) = TAB(IROW+1,5) + MULT
       IF (K .LT. LIM) TAB(IROW+2,5) = TAB(IRCW+2,5) + 1
C
       CALL SURRG3(K)
       CALL PRINT(.FALSE.,NAME(4,1),NAME(4,2),K)
       TAB(IROW,6) = TAB(IROW,6) + K
       TAB(IROW+1,6) = TAB(IROW+1,6) + MULT
       IF (K .LT. LIM) TAB(IROW+2,6) = TAB(IRCW+2,6) + 1
C
       CALL AGMON(K,1.0D 00)
       CALL PRINT(.FALSE.,NAME(5,1),NAME(5,2),K)
       TAB(IROW,7) = TAB(IROW,7) + K
       TAB(IROW+1,7) = TAB(IROW+1,7) + MULT
       IF (K .LT. LIM) TAB(IROW+2,7) = TAB(IRCW+2,7) + 1
         IF (LIM .GT. 500) GO TO 290
C
       CALL AGMON(K,2.0D 00)
       CALL PRINT(.FALSE.,NAME(6,1),NAME(6,2),K)
       TAB(IROW,8) = TAB(IROW,8) + K
       TAB(IROW+1,8) = TAB(IROW+1,8) + MULT
       IF (K .LT. LIM) TAB(IROW+2,8) = TAB(IRCW+2,8) + 1
  290  CONTINUE
C
```

```
      IPHASE = 1
      IF (M1 .EQ. M) IPHASE = 2
      CALL SIMPLX(IPHASE)
      WRITE(6,355) Z,(X(J),J = 1,LAST)
      IF (LAST .LT. N) WRITE(6,350) (X(J),J = 7,N)
      WRITE(6,360)
  295 CONTINUE
C
C TAKE AVERAGE.
C
      DO 305 I = 1,34,3
       DO 300 J = 3,8
       TAB(I,J) = TAB(I,J) / 4
  300  TAB(I+1,J) = TAB(I+1,J) / 4
  305 CONTINUE
      WRITE(6,310)
  310 FORMAT(1H1/////45X,'TABLE 3'/45X,'-------'//)
      WRITE(6,60)
      WRITE(6,65) ((NAME(I,1),NAME(I,2)),I = 1,6)
      WRITE(6,60)
      DO 315 I = 1,34,3
       WRITE(6,75) (TAB(I,J),J = 1,8)
       WRITE(6,80) (TAB(I+1,J),J = 3,8)
       WRITE(6,90) (TAB(I+2,J),J = 3,8)
       IF (I .EQ. 4) WRITE(6,320)
       IF (I .EQ. 22) WRITE(6,325)
       IF (I .EQ. 10 .OR. I .EQ. 34) WRITE(6,60)
       IF (I.NE.4 .AND. I.NE.10 .AND. I.NE.22 .AND. I.NE.34)
      1 WRITE(6,105)
  315 CONTINUE
  320 FORMAT(10X,'.    E    .....................................
     1.....................')
  325 FORMAT(10X,'.    F    .....................................
     1.....................')
      WRITE(6,330)
  330 FORMAT(/30X,'E = EQUALITY PROBLEMS'/
     1        30X,'F = LINEAR PROGRAMMING PROBLEMS')
      STOP
  335 FORMAT(3I3)
  340 FORMAT(6D10.1)
  345 FORMAT(/12X,'COST COEFT.',22X,6F10.3)
  350 FORMAT(45X,6F10.3)
  355 FORMAT(12X,'IMSL SIMPLEX O. F. SCL.',7F10.3)
  360 FORMAT(//)
      END



      SUBROUTINE PRINT(ZX0,NAME1,NAME2,K)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AI(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX,ZX0
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
```

```
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C  THIS ROUTINE IS CALLED WITH THE NAME OF AN ALGORITHM               C
C TO PRINT EITHER THE STARTING POINT OR THE SOLUTION GOT.            C
C                                                                     C
C  NEW           INDICATES WHETHER INITIAL POINT OR SOLUTION
C  NAME1,NAME2   NAME OF THE ALGORITHM                                C
C  K             NO. OF ITERATIONS                                    C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      LAST = MIN0(N,6)
      IF (LPP) GO TO 10
C
      IF (.NOT.ZX0) GO TO 5
      WRITE(6,20) M,N,K,K,(X0(J),J = 1,LAST)
      IF (LAST .LT. N) WRITE(6,30) (X0(J),J = 7,N)
      RETURN
C
    5 WRITE(6,25) NAME1,NAME2,K,MULT,(X(J),J = 1,LAST)
      IF (LAST .LT. N) WRITE(6,30) (X(J),J = 7,N)
      RETURN
C
   10 IF (.NOT.ZX0) GO TO 15
      WRITE(6,21) IZ,LSTCOL,K,K,Z,(X0(J),J = 1,LAST)
      IF (LAST .LT. N) WRITE(6,31) (X0(J),J = 7,N)
      RETURN
C
   15 WRITE(6,26) NAME1,NAME2,K,MULT,Z,(X(J),J = 1,LAST)
      IF (LAST .LT. N) WRITE(6,31) (X(J),J = 7,N)
      RETURN
C
   20 FORMAT(/I5,' BY',I3,' INIT.VAL',I5,2X,I8,6F10.3)
   21 FORMAT(/I5,' BY',I3,' INIT.VAL',I5,2X,I8,7F10.3)
   25 FORMAT(12X,A4,A3,I6,2X,I8,6F10.3)
   26 FORMAT(12X,A4,A3,I6,2X,I8,7F10.3)
   30 FORMAT(35X,6F10.3)
   31 FORMAT(45X,6F10.3)
      END



      SUBROUTINE TODHYP(KODE)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
      IF (KODE .EQ. 2) GO TO 25
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C     INPUT DATA FOR TODD'S EXAMPLE AND ITS EXPANSION                 C
C                                                                     C
C  M = 2**(N-1),                                                      C
C  A(I,1) = EPSILON,    FOR I = 1,2,...,M,                            C
C  A(1,J) = J-1,        FOR J = 2,3,...,N,                            C
C  FOR 2 .LE. I .LE. M  AND  2 .LE. J .LE. N,                         C
C      IF I .LE. 2**(N-J), THEN A(I,J) = A(1,J)                       C
C                          ELSE A(I,J) = -A(I-2**(N-J),J).            C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      M = 2 ** (N-1)
      A(1,1) = EPS
      A(1,2) = 1.0D 00
      DO 5 J = 3,N
    5 A(1,J) = A(1,J-1) + 1.0D 00
      LAST = M
      DO 15 I = 2,M
       A(I,1) = A(1,1)
       DO 10 J = 2,N
        LAST = LAST / 2
        K = I - LAST
        IF (K .LE. 0) A(I,J) =  A(1,J)
        IF (K .GT. 0) A(I,J) = -A(K,J)
   10  CONTINUE
       LAST = M
   15 B(I) = 0.0D 00
      R = DATAN(A(1,1))
      X0(1) = DCOS(R)
      X0(2) = DSIN(R)
      DO 20 J = 3,N
   20 X0(J) = 0.0D 00
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C          INPUT DATA FOR TRAPEZOIDAL HYPERCUBE                       C
C                                                                     C
C       0 .LE. X(1) .IE. 1,                                           C
C       2*X(J-1) .LE. X(J) .LE. 6**(J-1) - 2*X(J-1),                  C
C          FOR J = 2,3,...,N.                                         C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
   25 M = 2 * N
      DO 35 I = 1,M
       DO 30 J = 1,N
   30  A(I,J) = 0.0D 00
   35 B(I)   = 0.0D 00
      A(1,1) = -1.0D 00
      A(2,1) =  1.0D 00
      B(2)   =  1.0D 00
      DO 40 J = 2,N
       JLESS1 = J - 1
       I = 2 * J
       ILESS1 = I - 1
```

```
      A(ILESS1,JLESS1) =  2.0D 00
      A(ILESS1,J)      = -1.0D 00
      A(I,JLESS1)      =  2.0D 00
      A(I,J)           =  1.0D 00
   40 B(I)             =  6.0D 00 * B(I-2)
      END



      SUBROUTINE SETX0(KASE)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
C        SUBROUTINE INITIALIZING X AS REQUIRED.                   C
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      GO TO (15,5,5,25,5,35,50,75), KASE
C
    5 DO 10 J = 1,LSTCOL
   10 X0(J) = 0.0D 00
      IF (KASE .EQ. 2) X0(N) = -(6**(N-1))
      IF (KASE .EQ. 3) X0(N) = 2.0D 00 * (6**(N-1))
      RETURN
C
   15 DO 20 J = 1,N
   20 X0(J) = 6**(N-1)
      RETURN
C
   25 X0(1) = -1.0D 00
      DO 30 J = 2,N
   30 X0(J) = 6.0D 00 * X0(J-1)
      RETURN
C
   35 LAST = MIN0(M,N)
      DO 40 J = 1,LAST
   40 X0(J) = B(J)
      IF (.NOT.LPP) RETURN
      DO 45 J = 1,LAST
   45 X0(N+J) = B(M+J)
      RETURN
C
   50 R = B(1)
      DO 55 I = 2,M
   55 R = R + B(I)
      R = R / DBLE(FLOAT(M*N))
      DO 60 J = 1,N
   60 X0(J) = R
      IF (.NOT.LPP) RETURN
```

```
      R = 0.0D 00
      DO 65 I = MPLUS1,LSTROW
   65 R = R + B(I)
      R = R / DBLE(FLOAT(M*N))
      DO 70 J = 1,M
   70 X0(N+J) = R
      RETURN
C
   75 DO 80 J = 1,LSTCOL
   80 X0(J) = 1.0D 00
      END



      SUBROUTINE GENAB(ISEED)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
      INTEGER COL(30)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C  THIS ROUTINE GENERATES THE COEFFICIENT MARIX A AND THE          C
C  VECTOR B FOR THE PROBLEM OF FINDING AN X SUCH THAT              C
C                                                                  C
C                     AX .LE. B                                    C
C                                                                  C
C  THE RESTRICTIONS IMPOSED ON A, B AND X0 ARE AS FOLLOWS          C
C                                                                  C
C     1) A SHOULD BE, ON THE AVERAGE, ABOUT 0.1 DENSE.             C
C     2) -1 .LT. A(I,J) .LT. 1   FOR ALL I AND J                   C
C     3) B = AE + R,  WHERE  I) E = (1,1, ..., 1) TRANSPOSE
C                           II) 0 .LT. R(I) .LT. 1                 C
C     4) 100 .LT. ABS(X0(J)) .LT. 200   FOR ALL J                  C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 10 I = 1,M
       DO 5 J = 1,N
    5  A(I,J) = 0.0D 00
   10 CONTINUE
C
C GENERATE THE NO. OF NONZEROS FOR EACH COLUMN.
C
      DO 15 J = 1,N
       CALL RANDU(ISEED,ISEED,R)
       NUM = MOD(IDINT(1.0D 03 * R),11) + 5
   15 COL(J) = (NUM * M) / 100
C
C RANDOMLY PICK THE ROWS WHOSE ELTS. AT COL. J IS TO BE
C NONZERO.
C
```

```
      DO 35 J = 1,N
       NUM = COL(J)
       DO 30 K = 1,NUM
        CALL RANDU(ISEED,ISEED,R)
        I = MOD(IDINT(1.0D 03 * R),M) + 1
   20   IF (A(I,J) .EQ. 0.0D 00) GO TO 25
        I = I + 1
        IF (I .GT. M) I = 1
        GO TO 20
   25   CALL RANDU(ISEED,ISEED,R)
   30  A(I,J) = 2.0D 00 * R - 1.0D 00
   35 CONTINUE
C
C MAKE SURE THAT THERE ARE NO ZERO ROWS.
C
      DO 50 I = 1,M
       DO 40 J = 1,N
        IF (A(I,J) .NE. 0.0D 00) GO TO 50
   40  CONTINUE
C
C THIS IS A ZERO ROW. INCREASE THE LEAST COLUMN.
C
       J = 1
       DO 45 K = 2,N
        IF (COL(K) .LT. COL(J)) J = K
   45  CONTINUE
       CALL RANDU(ISEED,ISEED,R)
       A(I,J) = 2.0D 00 * R - 1.0D 00
       COL(J) = COL(J) + 1
   50 CONTINUE
C
C GENERATE B.
C
      DO 65 I = 1,M
       CALL RANDU(ISEED,ISEED,B(I))
       DO 60 J = 1,N
   60  B(I) = B(I) + A(I,J)
   65 CONTINUE
C
C INITIALIZE X0
C
      DO 75 J = 1,N
       CALL RANDU(ISEED,ISEED,R)
       X0(J) = DBLE(AMOD(AINT(1000.0*SNGL(R)),101.0)+100.0)
       CALL RANDU(ISEED,ISEED,R)
       IF (IDINT(2.0D 00*R) .EQ. 1.0D 00) X0(J) = -X0(J)
   75 CONTINUE
      END



      SUBROUTINE EPCUBE(ISEED,INITM,INCR)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS,S
```

```
      LOGICAL LPP,OVRIX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C  THIS ROUTINE GENERATES ADDITIONAL ROWS FOR THE EPSILON-         C
C  CUBE PROBLEM. THE NO. OF ORIGINAL ROWS IS INDICATED BY          C
C  THE PARAMETER INIT, WHILE THE PARAMETER INCR TELLS HOW          C
C  MANY MORE ROWS TO BE GENERATED. THE NEW ROWS ARE NORMA-         C
C  LIZED AS THE ARE GOT.                                           C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 30 K = 1,INCR
       M = M + 1
       DO 5 J = 1,N
    5  A(M,J) = 0.0D 00
       B(M) = 0.0D 00
       S = 0.0D 00
C
C GENERATE THE NO. OF THE ORIGINAL CONSTRAINTS TO BE
C COMBINED, THEN CHOOSE THEM AT RANDOM.
C
       CALL RANDU(ISEED,ISEED,R)
       NUM = MOD(IDINT(1.0D 03*R),N) + 1
       DO 20 L = 1,NUM
        CALL RANDU(ISEED,ISEED,R)
        I = MOD(IDINT(1.0D 03*R),INITM) + 1
   10   J = (I+1) / 2
        IF (A(M,J) .EQ. 0.0D 00) GO TO 15
        I = I + 1
        IF (I .GT. INITM) I = 1
        GO TO 10
C
C GENERATE ALPHA.
C
   15   CALL RANDU(ISEED,ISEED,R)
        A(M,J) = DMIN1(2.0D 00 * R,1.0D 00)
        IF (A(I,J) .LT. 0.0D 00) A(M,J) = -A(M,J)
        B(M) = B(M) + DABS(A(M,J))
   20   S = S + A(M,J) * A(M,J)
C
C NORMALIZE THIS ROW.
C
       S = DSQRT(S)
       DO 25 J = 1,N
   25  A(M,J) = A(M,J) / S
   30 B(M) = (EPS * B(M)) / S
      END
```

```
      SUBROUTINE NRMRCW
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C          ROUTINE THAT NORMALISES THE ROWS OF A.                  C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF (.NOT.LPP) GO TO 20
C
C SET UP THE DUAL CONSRAINTS.
C
      DO 10 J = 1,N
       DO 5 I = 1,M
    5  AT(J,I) = -A(I,J)
       B(M+J) = -U(J)
   10 AZ(J) = -U(J)
      DO 15 I = 1,M
   15 AZ(N+I) = B(I)
C
C NORMLIZE
C
   20 DO 35 I = 1,M
       R = 0.0D 00
       DO 25 J = 1,N
   25  R = R + A(I,J) * A(I,J)
       R = DSQRT(R)
       DO 30 J = 1,N
   30  A(I,J) = A(I,J) / R
   35 B(I) = B(I) / R
      IF (.NOT.LPP) RETURN
C
      DO 50 I = 1,N
       R = 0.0D 00
       DO 40 J = 1,M
   40  R = R + AT(I,J) * AT(I,J)
       R = DSQRT(R)
       DO 45 J = 1,M
   45  AT(I,J) = AT(I,J) / R
   50 B(M+I) = B(M+I) / R
C
      R = 0.0D 00
      DO 55 J = 1,LSTCOL
   55 R = R + AZ(J) * AZ(J)
      R = DSQRT(R)
      DO 60 J = 1,LSTCOL
   60 AZ(J) = AZ(J) / R
      END
```

```
      SUBROUTINE DOTPRD
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRIX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C   THIS ROUTINE COMPUTES THE DOT-PRODUCTS OF THE NORMALS          C
C   AND STORE THEM IN VECTOR C AS FOLLOWS                          C
C                                                                  C
C        1 TO L1           PRIMAL                                  C
C        L1+1 TO L2        PRIMAL PART OF Z-CONSTRAINT             C
C        L2+1 TO L3        DUAL                                    C
C        L3 TO THE END     DUAL PART OF Z-CONSTRAINT               C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      K = 0
      MLESS1 = M - 1
      DO 15 I = 1,MLESS1
       IPLUS1 = I + 1
       DO 10 I2 = IPLUS1,M
        K = K + 1
        C(K) = 0.0D 00
        DO 5 J = 1,N
    5   C(K) = C(K) + A(I2,J) * A(I,J)
   10  CONTINUE
   15 CONTINUE
      L1 = K
      IF (.NOT.LPP) RETURN
C
      DO 25 I = 1,M
       K = K + 1
       C(K) = 0.0D 00
       DO 20 J = 1,N
   20  C(K) = C(K) + AZ(J) * A(I,J)
   25 CONTINUE
      L2 = K
C
      MLESS1 = N - 1
      DO 40 I = 1,MLESS1
       IPLUS1 = I + 1
       DO 35 I2 = IPLUS1,N
        K = K + 1
        C(K) = 0.0D 00
        DO 30 J = 1,M
   30   C(K) = C(K) + AT(I2,J) * AT(I,J)
   35  CONTINUE
   40 CONTINUE
      L3 = K
C
```

```
      DO 50 I = 1,N
       K = K + 1
       C(K) = 0.0D 00
       DO 45 J = 1,M
   45  C(K) = C(K) + AZ(N+J) * AT(I,J)
   50 CONTINUE
      END



      SUBROUTINE INIT(K)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C   THIS ROUTINE INITIALIZES X, COMPUTES THE DISTANCES AND         C
C   CHOOSES THE MOST VIOLATED CONSTRAINT.  IT THEN INITIA          C
C   LIZES THE SURROGATE NORMAL, AK, WITH THAT OF THE MOST          C
C   VIOLATED.  IF K GREATER THAN 0, THEN X IS ALREADY INI-         C
C   TIALIZED AND D UPDATED.                                        C
C     KP        INDEX OF THE CHOSEN CONSTRAINT                   C
C     KASEP     SIGNAL FOR TYPE OF CONSTRAINT                    C
C                    1 FOR PRIMAL                                C
C                    2 FOR DUAL                                  C
C                    3 FOR X (PRIMAL AND DUAL VARIABLES)         C
C                    4 FOR THE Z-CONSTRAINT                      C
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF (K .GT. 0) GO TO 25
      DO 5 J = 1,LSTCOL
    5 X(J) = X0(J)
C
C COMPUTE THE DISTANCES, D = AX - B
C
      DO 7 I = 1,LSTROW
    7 D(I) = -B(I)
      DO 15 I = 1,M
       DO 10 J = 1,N
   10  D(I) = D(I) + A(I,J) * X(J)
   15 CONTINUE
      MULT = MULT + M * N
      IF (.NOT.LPP) GO TO 25
C
      DO 17 I = MPLUS1,LSTROW
       DO 16 J = 1,M
   16  D(I) = D(I) + AT(I-M,J) * X(N+J)
   17 CONTINUE
      MULT = MULT + N * M
C
```

```
      DZ = 0.0D 00
      Z = 0.0D 00
      DO 20 J = 1,LSTCOL
       IF (J .LE. N) Z = Z + U(J) * X(J)
   20 DZ = DZ + AZ(J) * X(J)
      MULT = MULT + LSTCOL
C
C CHOOSE THE MOST VIOLATED AND SET R TC THE DISTANCE.
C
   25 KASEP = 1
      KP = 1
      R = D(1)
      IF (M1 .EQ. 0) R = DABS(R)
      DO 40 I = 2,LSTROW
       IF (M1 .LT. I .AND. I .LE. M) GC TO 30
       IF (D(I) .LE. R) GO TO 40
       R = D(I)
       GO TO 35
   30  IF (DABS(D(I)) .LE. R) GO TC 40
       R = DABS(D(I))
   35  KP = I
   40 CONTINUE
      IF (.NOT.LPP) GC TO 55
      IF (KP .GT. M) KASEP = 2
C
      I2 = N + M1
      L = 1
      DO 45 J = 2,I2
       IF (X(J) .LT. X(L)) L = J
   45 CONTINUE
      IF (-X(L) .LE. R) GO TO 50
      R = -X(L)
      KP = L
      KASEP = 3
   50 IF (DZ .LE. R) GO TO 55
      R = DZ
      KASEP = 4
   55 IF ((R .LE. 0.0D 00) .OR. (R .LE. DEL .AND. K .GT. 0))
     1 RETURN
C
C THE INITIAL SURROGATE NORMAL, AK, AND THE DOTPRODUCT, CK.
C
      DO 60 J = 1,LSTCOL
   60 AK(J) = 0.0D 00
      DO 65 I = 1,LSTROW
   65 CK(I) = 0.0D 00
      CZ = 0.0D 00
      IF (KASEP .LE. 2) CK(KP) = 1.0D 00
      GO TO (70,110,130,150), KASEP
C
C THE MOST VIOLATED IS AN A-CONSTRAINT.
C
   70 IF (D(KP) .LT. 0.0D 00) GO TC 80
      DO 75 J = 1,N
   75 AK(J) = A(KP,J)
```

```
      GO TO 90
   80 DO 85 J = 1,N
   85 AK(J) = -A(KP,J)
      CK(KP) = -CK(KP)
   90 K1 = KP - M
      K2 = ((KP-1)*(2*M-KP))/2 - KP
      DO 105 I = 1,M
       IF (I .EQ. KP) GO TO 105
       IF (I .GT. KP) GO TO 95
       K1 = K1 + M - I
       CK(I) = C(K1)
       GO TO 100
   95  CK(I) = C(K2+I)
  100  IF (D(KP) .LT. 0.0D 00) CK(I) = -CK(I)
  105 CONTINUE
      IF (LPP) CZ = C(L1+KP)
      IF (LPP .AND. D(KP) .LT. 0.0D 00) CZ = -CZ
      RETURN
C
C THE MOST VIOLATED IS AN A-TRANSPOSE-CONSTRAINT.
C
  110 L = KP - M
      DO 115 J = 1,M
  115 AK(N+J) = AT(L,J)
      K1 = L2 + L - N
      K2 = L2 + ((L-1)*(2*N-L))/2 - KP
      DO 125 I = MPLUS1,LSTROW
       IF (I .EQ. KP) GO TO 125
       IF (I .GT. KP) GO TO 120
       K1 = K1 + LSTROW - I
       CK(I) = C(K1)
       GO TO 125
  120  CK(I) = C(K2+I)
  125 CONTINUE
      CZ = C(L3+L)
      RETURN
C
C THE MOST VIOLATED IS AN X-CONSTRAINT.
C
  130 AK(KP) = -1.0D 00
      CZ = -AZ(KP)
      IF (KP .GT. N) GO TO 140
      DO 135 I = 1,M
  135 CK(I) = -A(I,KP)
      RETURN
  140 L = KP - N
      DO 145 I = MPLUS1,LSTROW
  145 CK(I) = -AT(I-M,L)
      RETURN
C
C THE MOST VIOLATED IS THE Z-CONSTRAINT.
C
  150 DO 155 J = 1,LSICOL
  155 AK(J) = AZ(J)
      DO 175 I = 1,M
```

```
  175 CK(I) = C(L1+I)
      L = L3 - M
      DO 180 I = MPLUS1,LSTROW
  180 CK(I) = C(L+I)
      CZ = 1.0D 00
      END



      SUBROUTINE CHOOSP(G2)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS,G2,W,S,F
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C   THIS ROUTINE CHOOSES THE HYPERPLANE P WHICH, WITH THE          C
C   CURRENT SURROGATE HYPERPLANE, CONSTITUTES THE MOST             C
C   VIOLATED MANIFOLD THE STEPLENGTHS, H0 AND H1, ARE              C
C   PARTIALLY COMPUTED HERE.                                       C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      G2 = 0.0D 00
      KP = 0
      KASEP = 1
C
C THE A- AND A-TRANSPOSE-CONSTRAINTS.
C
      DO 15 I = 1,LSTROW
       F = D(I) - R * CK(I)
       MULT = MULT + 1
       IF (M1 .LT. I .AND. I .LE. M) GO TO 5
       IF (F .LE. DEL) GO TO 15
       IF (CK(I) .LE. -1.0D 00) GO TO 100
       GO TO 10
    5  IF (DABS(F) .LE. DEL) GO TO 15
       IF (DABS(CK(I)) .GE. 1.0D 00) GO TO 100
   10  S = 1.0D 00 - CK(I) * CK(I)
       W = (F * F) / S
       MULT = MULT + 3
       IF (W .LE. G2) GO TO 15
       G2 = W
       KP = I
       H1 = F
       H0 = S
   15 CONTINUE
      IF (.NOT.LPP) GO TO 25
      IF (KP .GT. M) KASEP = 2
C
C THE X-CONSTRAINTS.
C
```

```
      L = N + M1
      DO 20 J = 1,L
       F = X(J) - R * AK(J)
       MULT = MULT + 1
       IF (F .GE. -DEL) GO TO 20
       IF (AK(J) .GE. 1.0D 00) GO TO 100
       S = 1.0D 00 - AK(J) * AK(J)
       W = (F * F) / S
       MULT = MULT + 3
       IF (W .LE. G2) GO TO 20
       G2 = W
       KP = J
       H1 = F
       H0 = S
       KASEP = 3
   20 CONTINUE
      IF (KP .NE. 0) GO TO 25
C
C THE Z-CONSTRAINT.
C
      F = DZ - R * CZ
      MULT = MULT + 1
      IF (F .LE. DEL) GO TO 25
      IF (CZ .LE. -1.0D 00) GO TO 100
      S = 1.0D 00 - CZ * CZ
      W = (F * F) / S
      MULT = MULT + 3
      IF (W .LE. G2) GO TO 25
      G2 = W
      KP = IZ
      H1 = F
      H0 = S
      KASEP = 4
C
   25 IF (KP .EQ. 0) RETURN
      H1 = H1 / H0
      IF (KASEP .LE. 2) H0 = (R - D(KP) * CK(KP)) / H0
      IF (KASEP .EQ. 3) H0 = (R - X(KP) * AK(KP)) / H0
      IF (KASEP .EQ. 4) H0 = (R - DZ * CZ) / H0
      MULT = MULT + 3
C
      IF (.NOT.OVRLX) RETURN
      GO TO (30,30,35,40), KASEP
   30 IF ((CK(KP) .LE. 0.0D 00) .OR. (M1 .LT. KP .AND. KP
     1 .LE. M)) RETURN
      W = CK(KP)
      H0 = H0 - W * CK(KP) * H1
      GO TO 45
C
   35 IF (AK(KP) .GE. 0.0D 00) RETURN
      W = -AK(KP)
      H0 = H0 + W * AK(KP) * H1
      GO TO 45
C
   40 IF (CZ .LE. 0.0D 00) RETURN
```

```
      W = CZ
      H0 = H0 - W * CZ * H1
C
   45 G2 = ((1.0D 00 + W) ** 2) * G2
      H1 = (1.0D 00 + W) * H1
      MULT = MULT + 5
      RETURN
C
  100 WRITE(6,105)
  105 FORMAT(//' THE SYSTEM IS INCONSISTENT.')
      KP = -1
      END


      SUBROUTINE UPDATE(XZERO)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX,XZERO
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C          ROUTINE TO UPDATE AK AND CK, OR X AND D.                C
C                                                                  C
C   THE DOT-PRODUCTS OF THE CONSTRAINT NORMALS ARE IN              C
C   VECTOR C AND WE USE THE FORRMULA GIVEN IN THE TEXT TO          C
C   GET ACCESS TO THE RIGHT ONE.                                   C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF (XZERO) GO TO 1000
C
C UPDATE AK = H0*AK + H1*AKP  AND  CK = H0*CK + H1*CKP.
C
      DO 5 J = 1,LSTCOL
    5 AK(J) = H0 * AK(J)
      DO 10 I = 1,LSTROW
   10 CK(I) = H0 * CK(I)
      MULT = MULT + LSTCOL + LSTROW
      IF (KASEP .LE. 2) CK(KP) = CK(KP) + H1
      IF (LPP) CZ = H0 * CZ
      GO TO (15,35,55,75), KASEP
C
C KP IS AN A-CONSTRAINT.
C
   15 DO 20 J = 1,N
   20 AK(J) = AK(J) + H1 * A(KP,J)
      MULT = MULT + N
      K1 = KP - M
      K2 = ((KP-1)*(2*M-KP))/2 - KP
      DO 30 I = 1,M
       IF (I .EQ. KP) GO TO 30
```

```
          IF (I .GT. KP) GO TO 25
          K1 = K1 + M - I
          CK(I) = CK(I) + H1 * C(K1)
          GO TO 30
       25 CK(I) = CK(I) + H1 * C(K2+I)
       30 CONTINUE
          MULT = MULT + M - 1
          IF (LPP) CZ = CZ + H1 * C(L1+KP)
          RETURN
C
C KP IS AN A-TRANSPOSE-CONSTRAINT.
C
       35 L = KP - M
          DO 40 J = 1,M
       40 AK(N+J) = AK(N+J) + H1 * AT(L,J)
          MULT = MULT + M
          K1 = L2 + L - N
          K2 = L2 + ((L-1)*(2*N-L))/2 - KP
          DO 50 I = MPLUS1,LSTRCW
           IF (I .EQ. KP) GO TO 50
           IF (I .GT. KP) GO TO 45
           K1 = K1 + LSTRCW - I
           CK(I) = CK(I) + H1 * C(K1)
           GO TO 50
       45  CK(I) = CK(I) + H1 * C(K2+I)
       50 CONTINUE
          CZ = CZ + H1 * C(L3+L)
          MULT = MULT + N
          RETURN
C
C  KP IS AN X-CONSTRAINT.
C
       55 AK(KP) = AK(KP) + H1
          CZ = CZ + H1 * AZ(KP)
          IF (KP .GT. N) GO TO 65
          DO 60 I = 1,M
       60 CK(I) = CK(I) + H1 * A(I,KP)
          MULT = MULT + M
          RETURN
       65 J = KP - N
          DO 70 I = MPLUS1,LSTRCW
       70 CK(I) = CK(I) + H1 * AT(I-M,J)
          MULT = MULT + N
          RETURN
C
C KP IS THE Z-CONSTRAINT
C
       75 DO 80 J = 1,LSTCOL
       80 AK(J) = AK(J) + H1 * AZ(J)
          MULT = MULT + LSTCOL
          DO 85 I = 1,M
       85 CK(I) = CK(I) + H1 * C(L1+I)
          MULT = MULT + M
          L = L3 - M
          DO 90 I = MPLUS1,LSTRCW
```

```
   90 CK(I) = CK(I) + H1 * C(L+I)
      MULT = MULT + N
      CZ = CZ + H1
      RETURN
C
C UPDATE X = X - H0*AK - H1*AKP AND D = D - H0*CK - H1*CKP.
C
 1000 DO 1005 J = 1,LSTCOL
 1005 X(J) = X(J) - H0 * AK(J)
      DO 1010 I = 1,LSTROW
 1010 D(I) = D(I) - H0 * CK(I)
      MULT = MULT + LSTCOL + LSTROW
      IF (KASEP .LE. 2) D(KP) = D(KP) - H1
      IF (LPP) DZ = DZ - H0 * CZ
      GO TO (1015,1035,1055,1075), KASEP
C
C KP IS AN A-CONSTRAINT.
C
 1015 DO 1020 J = 1,N
 1020 X(J) = X(J) - H1 * A(KP,J)
      MULT = MULT + N
      K1 = KP - M
      K2 = ((KP-1)*(2*M-KP))/2 - KP
      DO 1030 I = 1,M
       IF (I .EQ. KP) GO TO 1030
       IF (I .GT. KP) GO TO 1025
       K1 = K1 + M - I
       D(I) = D(I) - H1 * C(K1)
       GO TO 1030
 1025  D(I) = D(I) - H1 * C(K2+I)
 1030 CONTINUE
      MULT = MULT + M - 1
      IF (LPP) DZ = DZ - H1 * C(L1+KP)
      RETURN
C
C KP IS AN A-TRANSPOSE-CONSTRAINT.
C
 1035 L = KP - M
      DO 1040 J = 1,M
 1040 X(N+J) = X(N+J) - H1 * AT(L,J)
      MULT = MULT + M
      K1 = L2 + L - N
      K2 = L2 + ((L-1)*(2*N-L))/2 - KP
      DO 1050 I = MPLUS1,LSTROW
       IF (I .EQ. KP) GO TO 1050
       IF (I .GT. KP) GO TO 1045
       K1 = K1 + LSTROW - I
       D(I) = D(I) - H1 * C(K1)
       GO TO 1050
 1045  D(I) = D(I) - H1 * C(K2+I)
 1050 CONTINUE
      DZ = DZ - H1 * C(L3+L)
      MULT = MULT + N
      RETURN
```

```
C
C  KP IS AN X-CONSTRAINT.
C
 1055 X(KP) = X(KP) - H1
      DZ = DZ - H1 * AZ(KP)
      IF (KP .GT. N) GO TO 1065
      DO 1060 I = 1,M
 1060 D(I) = D(I) - H1 * A(I,KP)
      MULT = MULT + M
      RETURN
 1065 J = KP - N
      DO 1070 I = MPLUS1,LSTROW
 1070 D(I) = D(I) - H1 * AT(I-M,J)
      MULT = MULT + N
      RETURN
C
C KP IS THE Z-CONSTRAINT
C
 1075 DO 1080 J = 1,LSTCOL
 1080 X(J) = X(J) - H1 * AZ(J)
      MULT = MULT + LSTCOL
      DO 1085 I = 1,M
 1085 D(I) = D(I) - H1 * C(L1+I)
      MULT = MULT + M
      L = L3 - M
      DO 1090 I = MPLUS1,LSTROW
 1090 D(I) = D(I) - H1 * C(L+I)
      MULT = MULT + N
      DZ = DZ - H1
      END



      SUBROUTINE SURRG1(K),
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS,G2
      LOGICAL LPP,OVRLX,XZERO
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
      DATA XZERC/.TRUE./
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
C            THE MAIN ROUTINE FOR ALGORITHM I                       C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      MULT = 0
      K = 0
      CALL INIT(0)
      IF (R .LE. 0.0D 00) RETURN
C
C X0 IS NOT A SOLUTION.
C
```

```
      DO 100 K = 1,LIM
       CALL CHOOSP(G2)
       IF (KP) 120,105,5
C
C COMPUTE THE KTH SURROGATE CONSTRAINT.
C
    5  R = DSQRT(R**R + G2)
       H0 = H0 / R
       H1 = H1 / R
       CALL UPDATE(.NOT.XZERO)
  100 CONTINUE
C
C TAKING TOO LONG TO CONVERGE. COMPUTE THE X SO FAR GOT.
C
  105 DO 110 J = 1,N
  110 X(J) = X(J) - R * AK(J)
      MULT = MULT + N
      IF (.NOT.LPP) RETURN
      Z = 0.0D 00
      DO 115 J = 1,N
  115 Z = Z + U(J) * X(J)
      MULT = MULT + N
  120 RETURN
      END



      SUBROUTINE SURRGR(K)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS,G2
      LOGICAL LPP,OVRIX,XZERO,USED(101)
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
      DATA XZERO/.TRUE./
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
C            THE MAIN ROUTINE FOR ALGORITHM R                       C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      MULT = 0
      K = 0
      CALL INIT(0)
      IF (R .LE. 0.0D 00) RETURN
C
C X0 IS NOT A SOLUTION.
C
      LAST = M
      IF (LPP) LAST = IZ
      DO 5 I = 1,LAST
    5 USED(I) = .FALSE.
      IF (KASEP .LE. 2) USED(KP) = .TRUE.
      IF (KASEP .EQ. 3) USED(LSTROW+KP) = .TRUE.
```

```
      IF (KASEP .EQ. 4) USED(LAST) = .TRUE.
      DO 100 K = 1,LIM
       CALL CHOOSP(G2)
       IF (KP) 125,105,10
   10  I = KP
       IF (KASEP .EQ. 3) I = LSTROW + KP
       IF (KASEP .EQ. 4) I = LAST
       IF (USED(I)) GO TO 15
C
C COMPUTE THE KTH SURROGATE CONSTRAINT.
C
       R = DSQRT(R**R + G2)
       H0 = H0 / R
       H1 = H1 / R
       CALL UPDATE(.NOT.XZERO)
       USED(I) = .TRUE.
       GO TO 100
C
C COMPUTE A NEW X0 IN STEAD OF THE KTH SURROGATE.
C
   15  CALL UPDATE(XZERO)
       CALL INIT(K)
       IF (R .LE. DEL) GO TO 115
       DO 20 I = 1,LAST
   20  USED(I) = .FALSE.
       IF (KASEP .LE. 2) USED(KP) = .TRUE.
       IF (KASEP .EQ. 3) USED(LSTROW+KP) = .TRUE.
       IF (KASEP .EQ. 4) USED(LAST) = .TRUE.
  100 CONTINUE
C
C TAKING TOO LONG TO CONVERGE.
C
  105 DO 110 J = 1,N
  110 X(J) = X(J) - R * AK(J)
      MULT = MULT + N
  115 IF (.NOT.LPP) RETURN
      Z = 0.0D 00
      DO 120 J = 1,N
  120 Z = Z + U(J) * X(J)
      MULT = MULT + N
  125 RETURN
      END



      SUBROUTINE SURRG2(K)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS
      LOGICAL LPP,OVRLX,XZERO
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
      DATA XZERO/.TRUE./
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C            THE MAIN ROUTINE FOR ALGORITHM II.                       C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      MULT = 0
      K = 0
      CALL INIT(0)
      IF (R .LE. 0.0D 00) RETURN
C
C XO IS NOT A SOLUTION.
C
      DO 100 K = 1,LIM
       CALL CHOOSP(G2)
       IF (KP) 125,105,5
C
C CCMPUTE A NEW XO IN STEAD OF THE KTH SURROGATE.
C
    5  CALL UPDATE(XZERO)
       CALL INIT(K)
       IF (R .LE. DEL) GO TC 115
  100 CONTINUE
C
C TAKING TOO LONG TO CONVERGE.
C
  105 DO 110 J = 1,N
  110 X(J) = X(J) - R * AK(J)
      MULT = MULT + N
  115 IF (.NOT.LPP) RETURN
      Z = 0.0D 00
      DO 120 J = 1,N
  120 Z = Z + U(J) * X(J)
      MULT = MULT + N
  125 RETURN
      END



      SUBROUTINE SURRG3(K)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS,G2
      LOGICAL LPP,OVRLX,XZERO
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
      DATA XZERO/.TRUE./
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C             MAIN ROUTINE FOR ALGORITHM III                          C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      MULT = 0
      K = 0
```

```
      CALL INIT(0)
      IF (R .LE. 0.0D 00) RETURN
C
C X0 IS NOT A SOLUTION.
C
      DO 100 K = 1,LIM
       CALL CHOOSP(G2)
       IF (KP) 125,105,5
C
C COMPUTE THE KTH SURRGATE CONSTRAINT.
C
    5  R = DSQRT(R*R + G2)
       H0 = H0 / R
       H1 = H1 / R
       CALL UPDATE(.NOT.XZERO)
       CALL CHOOSP(G2)
       IF (KP) 125,105,10
C
C COMPUTE A NEW X0 ON THE KTH SURROGATE HYPERPLANE.
C
   10  CALL UPDATE(XZERO)
       CALL INIT(K)
       IF (R .LE. DEL) GO TO 115
  100 CONTINUE
C
C TAKING TOO LONG TO CONVERGE.
C
  105 DO 110 J = 1,N
  110 X(J) = X(J) - R * AK(J)
      MULT = MULT + N
  115 IF (.NOT.LPP) RETURN
      Z = 0.0D 00
      DO 120 J = 1,N
  120 Z = Z + U(J) * X(J)
      MULT = MULT + N
  125 RETURN
      END



      SUBROUTINE AGMON(K,RELAX)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),X0(50),C(4005),AK(50),CK(90),AZ(50),R,H0,
     2       H1,Z,DZ,CZ,DEL,EPS,RELAX
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,H0,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,X0,C,AK,CK,AZ
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                               C
C   WE ARE ONLY DOING PROJECTION AND REFLECTION IN AGMON        C
C     RELAX = 1 FOR PROJECTION AND 2 FOR REFLECTION.            C
C                                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      MULT = 0
      K = 0
      DO 5 J = 1,LSTCOL
    5 X(J) = X0(J)
C
C COMPUTE D = AX - B.
C
      DO 7 I = 1,LSTROW
    7 D(I) = -B(I)
      DO 15 I = 1,M
       DO 10 J = 1,N
   10  D(I) = D(I) + A(I,J) * X(J)
   15 CONTINUE
      MULT = MULT + M * N
      IF (.NOT.LPP) GO TO 25
C
      DO 17 I = MPLUS1,LSTROW
       DO 16 J = 1,M
   16  D(I) = D(I) + AT(I-M,J) * X(N+J)
   17 CONTINUE
      MULT = MULT + N * M
C
      DZ = 0.0D 00
      Z = 0.0D 00
      DO 20 J = 1,LSTCOL
       IF (J .LE. N) Z = Z + U(J) * X(J)
   20 DZ = DZ + AZ(J) * X(J)
      MULT = MULT + LSTCOL
C
C CHOOSE THE MOST VIOLATED.
C
   25  KP = 1
       R = D(1)
       IF (M1 .EQ. 0) R = DABS(R)
       DO 40 I = 2,LSTROW
        IF (M1 .LT. I .AND. I .LE. M) GO TO 30
        IF (D(I) .LE. R) GO TO 40
        R = D(I)
        GO TO 35
   30   IF (DABS(D(I)) .LE. R) GO TO 40
        R = DABS(D(I))
   35   KP = I
   40  CONTINUE
       IF (.NOT.LPP) GO TO 55
C
       LAST = N + M1
       L = 1
       DO 45 J = 2,LAST
        IF (X(J) .LT. X(L)) L = J
   45  CONTINUE
       IF (-X(L) .LE. R) GO TO 50
       R = -X(L)
       KP = LSTROW + L
   50  IF (DZ .LE. R) GO TO 55
       R = DZ
```

```
      KP = IZ
   55 IF (R .LE. 0.0D 00) RETURN
C
C X0 IS NOT A SOLUTION.
C
      DO 100 K = 1,LIM
       IF ((KP .LE. M .AND. D(KP) .LT. 0.0D 00) .OR.
     1     (LSTROW .LT. KP .AND. KP .LT. IZ)) R = -R
       R = RELAX * R
       MULT = MULT + 1
C
C UPDATE.
C
       IF (KP .EQ. IZ) GO TO 85
       IF (KP .GT. LSTROW) GO TO 80
       D(KP) = D(KP) - R
       IF (KP .GT. M) GO TO 70
       DO 60 J = 1,N
   60  X(J) = X(J) - R * A(KP,J)
       MULT = MULT + N
       K1 = KP - M
       K2 = ((KP-1)*(2*M-KP))/2 - KP
       DO 65 I = 1,M
        IF (I .EQ. KP) GO TO 65
        IF (I .GT. KP) GO TO 61
        K1 = K1 + M - I
        D(I) = D(I) - R * C(K1)
        GO TO 65
   61   D(I) = D(I) - R * C(K2+I)
   65  CONTINUE
       MULT = MULT + M - 1
       IF (LPP) DZ = DZ - R * C(L1+KP)
       GO TO 125
C
C MOST VIOLATED IS AN A-TRANSPOSE CONSTRAINT.
C
   70  L = KP - M
       DO 71 J = 1,M
   71  X(N+J) = X(N+J) - R * AT(L,J)
       MULT = MULT + M
       K1 = L2 + L - N
       K2 = L2 + ((L-1)*(2*N-L))/2 - KP
       DO 75 I = MPLUS1,LSTROW
        IF (I .EQ. KP) GO TO 75
        IF (I .GT. KP) GO TO 72
        K1 = K1 + LSTROW - I
        D(I) = D(I) - R * C(K1)
        GO TO 75
   72   D(I) = D(I) - R * C(K2+I)
   75  CONTINUE
       DZ = DZ - R * C(L3+L)
       MULT = MULT + N
       GO TO 125
```

```
C
C MOST VIOLATED IS AN X-CONSTRAINT.
C
   80  KP = KP - LSTROW
       X(KP) = X(KP) - R
       DZ = DZ - R * AZ(KP)
       IF (KP .GT. N) GO TO 82
       DO 81 I = 1,M
   81  D(I) = D(I) - R * A(I,KP)
       MULT = MULT + M
       GO TO 125
   82  J = KP - N
       DO 83 I = MPLUS1,LSTROW
   83  D(I) = D(I) - R * AT(I-M,J)
       MULT = MULT + N
       GO TO 125
C
C MOST VIOLATED IS THE Z-CONSTRAINT.
C
   85  DO 86 J = 1,LSTCOL
   86  X(J) = X(J) - R * AZ(J)
       MULT = MULT + LSTCOL
       DO 87 I = 1,M
   87  D(I) = D(I) - R * C(L1+I)
       MULT = MULT + M
       L = L3 - M
       DO 95 I = MPLUS1,LSTROW
   95  D(I) = D(I) - R * C(L+I)
       MULT = MULT + N
       DZ = DZ - R
C
C CHOOSE THE MOST VIOLATED.
C
  125  KP = 1
       R = D(1)
       IF (M1 .EQ. 0) R = DABS(R)
       DO 140 I = 2,LSTROW
        IF (M1 .LT. I .AND. I .LE. M) GO TO 130
        IF (D(I) .LE. R) GO TO 140
        R = D(I)
        GO TO 135
  130   IF (DABS(D(I)) .LE. R) GO TO 140
        R = DABS(D(I))
  135   KP = I
  140  CONTINUE
       IF (.NOT.LPP) GO TO 155
C
       L = 1
       DO 145 J = 2,LAST
        IF (X(J) .LT. X(L)) L = J
  145  CONTINUE
       IF (-X(L) .LE. R) GO TO 150
       R = -X(L)
       KP = LSTROW + L
  150  IF (DZ .LE. R) GO TO 155
```

```
      R = DZ
      KP = IZ
  155 IF (R .LE. DEL) GO TC 205
  100 CONTINUE
C
C TAKING TOO LONG TO CONVERGE.
C
      IF (KP .EQ. IZ) GO TO 220
      IF (KP .GT. LSTROW) GC TO 215
      IF (KP .GT. M) GO TO 205
      IF (D(KP) .LT. 0.0D 00) R = -R
      DO 200 J = 1,N
  200 X(J) = X(J) - R * A(KP,J)
      MULT = MULT + N
  205 IF (.NOT.LPP) RETURN
      Z = 0.0D 00
      DO 210 J = 1,N
  210 Z = Z + U(J) * X(J)
      MULT = MULT + N
      RETURN
  215 X(KP-LSTROW) = X(KP-LSTROW) - R
      GO TO 205
  220 DO 225 J = 1,N
  225 X(J) = X(J) - R * AZ(J)
      MULT = MULT + N
      GO TO 205
  300 CONTINUE
      END



      SUBROUTINE SIMPLX(IPHASE)
      DOUBLE PRECISION A(90,40),B(90),X(50),D(90),AT(20,20),
     1       U(20),XO(50),C(4005),AK(50),CK(90),AZ(50),R,HO,
     2       H1,Z,DZ,CZ,DEL,EPS,T(25,25),DS(25),ROWK(30),
     3       COPI(25,25),XS(25),WA(25)
      LOGICAL LPP,OVRLX
      COMMON /GLOBL1/M,N,M1,MPLUS1,LSTROW,LSTCOL,LIM,LPP,
     1         OVRLX,KP,R,HO,H1,Z,DZ,CZ,DEL,EPS,MULT,KASEP,
     2         IZ,L1,L2,L3
      COMMON /GLOBL2/A,B,X,D,AT,U,XO,C,AK,CK,AZ
      INTEGER ICOLMS(30),IBV(25)
      DATA K/1/, IR/25/
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
C       MAIN ROUTINE FOR SIMPLEX                                    C
C                                                                   C
C  THIS ROUTINE USES THE SIMPLEX METHOD AS PRCVIDED IN THE IMSL     C
C  PACKAGE. IT FIRST SETS UP THE PARAMETERS TO THEIR RECOMMENDED    C
C  VALUES AND THEN CALLS THE IMSL ZXOLP                             C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      Z = 0.0D 00
      DO 5 J = 1,N
    5 X(J) = 0.0D 00
```

```
      DS(1) = 0.0D 00
      NS = N
      ITMAX = 5 * N
      IF (IPHASE .EQ. 1) GO TO 50
C
C TABLEAU FOR PHASE 2.
C
      DO 15 I = 2,MPLUS1
       I2 = I - 1
       IF (B(I2) .LT. 0.0D 00) GO TO 45
       DO 10 J = 1,N
   10  T(I,J) = A(I2,J)
   15 DS(I) = B(I2)
      DO 20 J = 1,N
   20 T(1,J) = -U(J)
C
      LIC = MPLUS1
      DO 25 J = 2,MPLUS1
       ICOLMS(J) = J
       IBV(J) = N + J
   25 ROWK(J) = 0.0D 00
      ICOLMS(1) = 1
      IBV(1) = N + 1
      ROWK(1) = 1.0D 00
      DO 35 I = 1,MPLUS1
       DO 30 J = 1,MPLUS1
   30  COPI(I,J) = 0.0D 00
   35 COPI(I,I) = 1.0D 00
      MS = M
      CALL ZXOLP(IPHASE,T,DS,ICOLMS,ROWK,K,MS,NS,ITMAX,LIC,
     1IR,COPI,IBV,XS,WA,IER)
      IF (IER .GE. 130) GO TO 145
      Z = XS(1)
      DO 40 I = 2,MPLUS1
       IF (IBV(I) .LE. N) X(IBV(I)) = XS(I)
   40 CONTINUE
      RETURN
C
C TABLEAU FOR PHASE 1.
C
   45 IPHASE = 1
   50 DO 55 J = 1,N
   55 T(2,J) = -U(J)
      DS(2) = 0.0D 00
      NONNEG = 0
      DO 75 I = 1,M
       I2 = I + 2
       IF (B(I) .LT. 0.0D 00) GO TO 65
       IF (I .LE. M1) NONNEG = NONNEG + 1
       DO 60 J = 1,N
   60  T(I2,J) = A(I,J)
       DS(I2) = B(I)
       GO TO 75
   65  DO 70 J = 1,N
   70  T(I2,J) = -A(I,J)
```

```
      DS(I2) = -B(I)
   75 CONTINUE
C
      NEG = M1 - NONNEG
      MPLUS2 = M + 2
      DO 85 J = 1,N
       T(1,J) = 0.0D 00
       DO 80 I = 2,MPLUS2
   80  T(1,J) = T(1,J) - T(I,J)
   85 CONTINUE
C
      LIC = MPLUS2 + NEG
      LAST = NONNEG + 1
      DO 90 J = 1,LAST
       ICOLMS(J) = J + 1
       IBV(J+1) = N + J
   90 ROWK(J) = -1.0D 00
      L = NONNEG + 2
      LAST = M1 + 2
      DO 95 J = L,LAST
       ICOLMS(J) = -(J+1)
   95 ROWK(J) = 1.0D 00
      ICOLMS(LAST) = 1
      IBV(1) = N + LAST
      L = M1 + 3
      DO 100 J = L,LIC
       ICOLMS(J) = J - NEG
       IBV(J-NEG) = N + J
  100 ROWK(J) = 0.0D 00
C
      LAST = NONNEG + 2
      DO 110 I = 1,MPLUS2
       DO 105 J = 1,MPLUS2
  105  COPI(I,J) = 0.0D 00
       IF (I .LE. LAST) COPI(1,I) = 1.0D 00
  110 COPI(I,I) = 1.0D 00
      MS = MPLUS1
      CALL ZX0LP(IPHASE,T,DS,ICOLMS,ROWK,K,MS,NS,ITMAX,LIC,
     1IR,COPI,IBV,XS,WA,IER)
      IF (IER .GE. 130) GO TO 145
      LAST = N + M1 + 1
      DO 115 I = 3,MPLUS2
       IF (IBV(I).GT.LAST .AND. XS(I) .NE. 0.0D 00)GO TO 140
       IF (IBV(I) .GT. LAST) WRITE(6,165)
  115 CONTINUE
      LIC = MPLUS1
      DO 130 I = 1,MPLUS1
       ICOLMS(I) = ICOLMS(I) - ISIGN(1,ICOLMS(I))
  130 ROWK(I) = 0.0D 00
      ROWK(1) = 1.0D 00
      MS = M
      IPHASE = 2
      CALL ZX0LP(IPHASE,T(2,1),DS(2),ICOLMS,ROWK,K,MS,NS,
     1ITMAX,LIC,IR,COPI(2,2),IBV(2),XS,WA,IER)
      IF (IER .GE. 130) GO TO 145
```

```
      Z = XS(1)
      DO 135 I = 3,MPLUS2
       IF (IBV(I) .LE. N) X(IBV(I)) = XS(I-1)
  135 CONTINUE
      RETURN
  140 WRITE(6,150)
      RETURN
  145 IF (IER .EQ. 130) WRITE(6,150)
      IF (IER .EQ. 131) WRITE(6,155)
      IF (IER .EQ. 132) WRITE(6,160)
  150 FORMAT(//'  THE SYSTEM IS INCONSISTENT.')
  155 FORMAT(//'  THE SYSTEM IS UNBOUNDED.')
  160 FORMAT(//'  MAXIMUM NO. OF ITERATION REACHED.')
  165 FORMAT(//'  WARNING, SYSTEM HAS REDUNDANT CONSTRAINT.')
      END


//GO.SYSIN DD *
SURRG1 SURRGR SURRG2 SURRG3 AGMON1 AGMCN2 SIMPLEX
  2  3  0
   2.0D 00  -3.0D 00   4.0D 00
   6.0D 00   5.0D 00  -7.0D 00
   8.0D 00   4.0D 00
  5  3  0
   1.0D 00   2.0D 00   3.0D 00
  -1.0D 00   1.0D 00  -2.0D 00
   1.0D 00   5.0D 00   4.0D 00
   0.0D 00   3.0D 00   1.0D 00
  -1.0D 00   4.0D 00  -1.0D 00
   2.0D 00   1.0D 00   5.0D 00   3.0D 00   4.0D 00
  3  3  0
   1.0D 00   1.0D 00   1.0D 00
   1.0D 00  -1.0D 00   1.0D 00
   1.0D 00   2.0D 00  -1.0D 00
   1.0D 00   3.0D 00   4.0D 00
  4  4  0
   1.0D 00   2.0D 00 -12.0D 00   8.0D C0
   5.0D 00   4.0D 00   7.0D 00  -2.0D 00
  -3.0D 00   7.0D 00   9.0D 00   5.0D 00
   6.0D 00 -12.0D 00  -8.0D 00   3.0D 00
  27.0D 00   4.0D 00  11.0D 00  49.0D 00
  1  2  1
   1.0D 00   2.0D 00
   1.0D 00   1.0D 00
   1.0D 00
  2  2  1
   2.0D 00   3.0D 00
   1.0D 00   2.0D 00
   1.0D 00   1.0D 00
   4.0D 00   3.0D 00
  4  2  4
   1.0D 00   3.0D 00
   1.0D 00   0.0D 00
   0.0D 00   1.0D 00
```

```
 1.0D 00   1.0D 00
-1.0D 00  -1.0D 00
 1.0D 00   1.0D 00   1.5D 00  -0.5D 00
3  4  3
 2.0D 00   4.0D 00   1.0D 00   1.0D 00
 1.0D 00   3.0D 00   0.0D 00   1.0D 00
 2.0D 00   1.0D 00   0.0D 00   0.0D 00
 0.0D 00   1.0D 00   4.0D 00   1.0D 00
 4.0D 00   3.0D 00   3.0D 00
2  5  2
 5.0D 00   8.0D 00   7.0D 00   4.0D 00   6.0D 00
 2.0D 00   3.0D 00   3.0D 00   2.0D 00   2.0D 00
 3.0D 00   5.0D 00   4.0D 00   2.0D 00   4.0D 00
20.0D 00  30.0D 00
5  3  5
 1.0D 00   3.0D 00  -2.0D 00
 0.0D 00   1.0D 00  -2.0D 00
 2.0D 00   1.0D 00   2.0D 00
 1.0D 00   0.0D 00   0.0D 00
 0.0D 00   1.0D 00   0.0D 00
 0.0D 00   0.0D 00   1.0D 00
 1.0D 00  18.0D 00   1.0D 00   3.0D 00   2.0D 00
5  9  5
-1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00
-1.0D 00  -1.0D 00  -1.0D 00
-44.7D 00  -8.4D 00  -7.4D 00  -2.2D 00  -2.6D 00  -1.1D 00
 -9.6D 00 -17.4D 00 -26.9D 00
 -2.0D 00 -15.1D 00 -16.4D 00  -0.2D 00  -4.0D 00   0.0D 00
 -2.7D 00  -3.7D 00 -11.4D 00
  0.0D 00 -26.0D-01 -28.1D-01-169.2D-01  -7.2D-01-918.4D-01
-290.7D-01  -5.1D-01   0.0D 00
-33.3D 00 -23.5D 00 -10.3D 00 -50.8D 00  -4.5D 00 -13.8D 00
 -5.4D 00 -38.2D 00 -24.6D 00
  0.0D 00 -60.0D-02   0.0D 00-525.0D-02-536.9D-01-275.5D-01
-191.2D-01   0.0D 00   0.0D 00
 -3.0D 00  -0.8D 00  -5.0D-01  -2.7D 00 -75.0D-02
9 20  9
-1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00
-1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00
-1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00  -1.0D 00
-1.0D 00  -1.0D 00
-44.7D 00 -36.0D 00  -8.4D 00 -20.6D 00  -7.4D 00 -15.7D 00
-41.7D 00  -2.2D 00  -4.4D 00  -5.8D 00  -2.4D 00  -2.6D 00
 -5.8D 00 -14.3D 00  -1.1D 00  -9.6D 00  -8.5D 00 -12.8D 00
-17.4D 00 -26.9D 00
-141.1D-01-897.0D-02-422.0D-02 -17.0D-02-448.0D-02-661.0D-02
  0.0D 00-333.0D-02-249.0D-02-705.0D-02-138.0D-02-125.0D-02
-166.0D-02-336.0D-02-106.0D-02-138.0D-02 -87.0D-02 -99.0D-02
-105.5D-01-169.1D-01
 -2.0D 00  -1.7D 00 -15.1D 00  -0.6D 00 -16.4D 00  -1.0D 00
  0.0D 00  -0.2D 00  -0.3D 00  -6.8D 00  -3.7D 00  -4.0D 00
 -3.8D 00  -1.8D 00   0.0D 00  -2.7D 00  -1.7D 00  -2.5D 00
 -3.7D 00 -11.4D 00
-365.0D-01 -99.0D-01  -9.0D-01  -6.0D-01 -19.0D-01 -48.0D-01
  0.0D 00-139.0D-01 -37.0D-01 -45.0D-01 -80.0D-01 -36.0D-01
```

```
 -59.0D-01-118.0D-01-138.0D-01 -54.0D-01-173.0D-01-154.0D-01
-459.0D-01-792.0D-01
   0.0D 00 -30.9D-01 -26.0D-01 -55.8D-01 -28.1D-01   0.0D 00
  -0.2D-01-169.2D-01   0.0D 00  -3.5D-01 -69.0D-01  -7.2D-01
 -16.6D-01  -6.7D-01-918.4D-01-290.7D-01 -86.8D-01 -85.7D-01
  -5.1D-01   0.0D 00
 -55.4D 00 -17.4D 00  -3.0D 00  -0.2D 00  -0.8D 00  -9.6D 00
   0.0D 00  -6.4D 00 -18.2D 00  -1.0D 00  -4.3D 00  -9.0D 00
  -4.7D 00 -29.4D 00  -5.7D 00  -8.4D 00  -1.2D 00  -3.9D 00
 -26.9D 00 -38.4D 00
 -33.3D 00  -7.9D 00 -23.5D 00   0.0D 00 -10.3D 00  -8.1D 00
  -0.5D 00 -50.8D 00  -3.6D 00  -4.9D 00  -5.8D 00  -4.5D 00
  -5.9D 00  -7.1D 00 -13.8D 00  -5.4D 00  -4.3D 00  -4.3D 00
 -38.2D 00 -24.6D 00
-441.0D-01-106.0D-01 -11.0D-01   0.0D 00  -4.0D-01-471.0D-01
  -5.0D-01-316.0D-01 -79.0D-01-209.0D-01 -37.0D-01 -26.0D-01
 -21.0D-01-198.0D-01 -33.0D-01 -83.0D-01 -55.0D-01 -65.0D-01
 -93.0D-01-217.0D-01
   0.0D 00   0.0D 00 -60.0D-02   0.0D 00   0.0D 00   0.0D 00
   0.0D 00-525.0D-02   0.0D 00   0.0D 00-862.0D-02-536.9D-01
-118.4D-01-252.2D-01-275.5D-01-191.2D-01 -57.0D-02-257.0D-02
   0.0D 00   0.0D 00
  -3.0D 00 -70.0D-02  -0.8D 00 -12.0D-01  -5.0D-01  -1.8D 00
  -2.7D 00 -18.0D-01 -75.0D-02
/*
```

BIBLIOGRAPHY

1. Abadie, J. (ed.) Nonlinear Programming, John Wiley and Sons, Inc., New York, 1967.

2. Agmon, S. The Relaxation Method for Linear Inequalities, Canadian J. of Math., Vol. 6 (1954), pp. 382 - 392.

3. Dahlquist, G. and Bjorck, A. (Translated by N. Anderson) Numerical Methods, Prentice-Hall, Inc.,Englewood Cliffs, New Jersey, 1974.

4. Dantzig, G.B. Linear Programming and Extentions, Princeton University Press, Princeton, New Jersey, 1963.

5. Gill, P.E. and Murray, W. (ed.) Numerical Methods for Constrained Optimization, Acad. Press, New York, 1974.

6. Goffin, J.L. On the Finite Convergence of the Relaxation Method for Solving Systems of Inequalities Operations Research Center, Univ. of Calif., Berkeley, Dec. 1971

7. Goffin, J.L. The Relaxation Method for Solving Systems of Linear Inequalities, McGill Univ., Montreal, Quebec(1978).

8. Goffin, J.L. On the Non-Polynomiality of the Method for Systems of Linear Inequalities Faculty of Management, McGill University, Montreal, Quebec, Canada, 1979.

9. Goldfarb, D. and Todd, M.J. Modifications and Implementation of the Shor-Khachian Algorithm for Linear Programming, Technical Report no. 446(1980): School of Operations Research and Industrial Engineering, Cornell Univ., Ithaca, New York.

10. Hillier, F.S. and Lieberman, G.J. Operations Research,

2nd ed., Holden-Day, Inc., 1974.

11. Hoffman, A.J. On Approximate Solutions of Systems of Linear Inequalities, J. of Research of the National Bureau of Standards, Vol. 49(1952), pp. 263 - 265.

12. Isaacson, E. and Keller, H.B. Analysis of Numerical Methods, John Wiley and Sons, Inc., New York, 1966.

13. Jeroslow, R.G. Some Relaxation Methods for Linear Inequalities, Cashier du C.E.R.O., Vol. 21(1979) pp. 43-53.

14. Kincaid, D.R. and Young, D.M. Survey of Iterative Methods, Center for Numerical Analysis, The University of Texas at Austin, April 1978.

15. Krol, Y. and Mirman, B. Some Practical Modifications of Ellipsoid Method for LP Problems.

16. Luenberger, D.G. Introduction to Linear and Nonlinear Programming, Addison-Wesley Publishing Company, (1973).

17. Maurras, J.F., Truemper, K. and Akçul, M. Polynomial Algorithms for a Class of Linear Programs, April 1980.

18. Motzkin, T.S. and Schoenberg, I.J. The Relaxation Method for Linear Inequalities, Canadian J. of Maths., Vol. 6 (1954) pp. 393 - 404.

19. Murty, K.G. Linear Algebra and Its Applications, Academic Press, New York, 1956.

20. Oko, S.O. On the Iterative Schemes for Solving Systems of Linear Equations, Postgraduate diploma dissertation, Dept. of Computer Science, Lagos University, Nigeria, 1975.

21. Todd, M.J. Some Remarks On The Relaxation Method for Linear Inequalities, Technical Report no. 419 (1979): School of Operations Research and Industrial Engineering, Cornell Univ., Ithaca, New York.