NEW ITERATIVE METHODS FOR LINEAR INEQUALITIES

Kai Yang
Department of Industrial & Operations Engineering
University of Michigan

Technical Report 90-6

February 1990

NEW ITERATIVE METHODS FOR LINEAR INEQUALITIES

Kai Yang

February, 1990

Abstract

Two classes of new iterative methods for solving systems of linear inequalities are considered in this report. In the first class of methods, a group of violated constraints are identified and a surrogate constraint is derived by taking a convex combination of these violated constraints. Each iterate is obtained by making an orthogonal projection from the current iterative solution onto the surrogate constraint. The second class of methods belongs to modified Newton's method and it can find least squares solution of linear inequalities. These methods can be implemented very efficiently, especially when the systems are large and sparse. Convergence proofs are provided.

Key words: Linear inequalities, Surrogate constraint, Iterative methods, Least squares solution, Modified Newton's Method.

1. INTRODUCTION

We are concerned with a system of linear inequalities:

$$Ax \leqq b \tag{1.1}$$

where A is an m x n matrix and b is an m-vector. We are interested in finding a feasible solution $\bar{x}$ for system (1.1). Solving a system of linear inequalities is a fundamental problem in linear optimization and it has many applications. These applications include linear programs [22], and in particular, the problem of *image reconstruction* from projections [7]. The image reconstruction problem has arisen in a large number of scientific fields. In medical science, *computerized tomography* reconstructs the images of cross-sections of the human body by processing data obtained from measuring the attenuation of X-rays along a large number of lines through the cross-section. Other image reconstruction problems include remote sensing [13], seismic tomography [3] and industrial nondestructive testing.

There are basically two approaches which could be used to solve the system (1.1). The first approach is to transform the linear inequality system (1.1) to a linear programming problem and then use well-established methods such as Karmarkar's method [21] or the simplex method to solve the resulting linear program. Usually this approach is not applicable to the image reconstruction problems due to the following two difficulties.

The first difficulty is the special environment within which the problem has to be solved. This environment is characterized by:
(i) the immense dimension of the system (1.1); for example, $n \geq 10^5$ and m is even greater.
(ii)the sparsity of the A matrix; usually the sparseness of matrix A is less than 1% and the

sparsity structure of A varies greatly from matrix to matrix so it is impossible to take advantage of any structure pattern of sparsity.

(iii)the poor conditioning of matrix A;

(iv)the restrictions on computer power; usually image reconstruction problems are solved by small, dedicated on site computers and the objective is to obtain approximate results in a relative short period of time.

Most linear programming algorithms require matrix manipulations, such as factorization, projection and etc. Since it is difficult to take advantage of the sparsity patterns of the A matrix, sparse factoring techniques can not be used. Using conjugate gradient type methods to do the matrix manipulations is also not an option since it is very difficult to find a preconditioning technique to preprocess this kind of A matrix.

Another difficulty is that system (1.1) may be inconsistent. It is then often desirable to find a least squares solution [16], but most linear programming algorithms can not be used to find least squares solutions.

The second approach involves using iterative methods to solve system (1.1). Most of the iterative methods are derived from the relaxation method for linear inequalities (Agmon Motzkin and Schoenberg [1954])[1][20] and Kaczmarz's method[17] for linear equations. There are numerous special implementations of the relaxation method which are called Algebraic Reconstruction Techniques (ART) [7][9]. Another method is derived from Cimmino's algorithm [10] for linear equations. Y. Censor, T. Elfving [1982] [8] and A.R. De Pierro, A.N. Iusem [1985] [11] developed a Cimmino type algorithm for linear inequalities. Other iterative methods for linear inequalities include Magasarian's SOR type methods [18]. Those iterative methods always work with the original data and most of them do not need matrix manipulations. The basic computation steps in iterative methods are extremely simple and easy to program. Because of those advantages, all linear inequality solvers which are used in image reconstruction are iterative methods. However, the rates of convergence of those iterative algorithm are usually slow [25] and therefore, it often takes hours to solve a single image reconstruction problem.

In this report we propose two classes of new iterative algorithms for solving (1.1) which are sparsity preserving and have finite convergence properties. Therefore, this paper is divided to two parts and each part describes one class of methods. The first class of methods is called 'surrogate constraint methods'. There are three different surrogate constraint methods in this paper designed to tackle image reconstruction problems. The second class of methods is a revised version of S. P. Han's method for finding the least squares solutions of linear inequalities [16] which belongs to the class of modified Newton's method [24].

In section 2 we introduce the surrogate constraint methods and define notations used in PART I. Sections 3, 4 and 5 describe the first, second and third surrogate constraint algorithms, respectively. Section 6 presents the geometric interpretation of the surrogate algorithms and Section 7 presents their extension to linear equations. Section 8 introduces the second class of methods and defines notations used in PART II. Section 9 describes the method. Convergence proofs and geometric interpretations for this method are presented in section 10 and some computational results are presented in section 11. Some theorems and their proofs are summarized in the Appendix.

PART I
THE SURROGATE CONSTRAINT METHODS

2. INTRODUCTION TO SURROGATE CONSTRAINT METHODS

In their classical papers Agmon [1] and Motzkin and Schoeberg [20] introduced the relaxation method to solve the system of linear inequalities. The method is called 'relaxation' method because constraints in the system (1.1) are considered one at a time. At each iteration a violated constraint is identified and an orthogonal projection is made onto this constraint from the current iterative solution. So it is also called the 'successive orthogonal projection' method. Bregman [1965, 1967] [5] [6], Eremin [1965] [12] and Gubin et al. [1967] [15] extended this ' successive orthogonal projection' method to general convex feasibility problems. Making an orthogonal projection onto a single linear constraint is computationally inexpensive. However, when solving a huge system of linear inequalities, which may have hundreds of thousands of constraints, considering only one constraint at a time would lead to slow convergence. Instead, we would like to process a group of constraints at a time. But making an orthogonal projection onto a group of constraints is computationally expensive.

In PART I we will introduce a class of methods which are able to process a group of violated constraints at a time but retain the same computational simplicity as the relaxation method. This class of methods is named as 'surrogate constraint method' since at each iteration, a group of violated constraints is identified and a 'surrogate constraint' is derived from those constraints. An orthogonal projection is then made onto this surrogate constraint from then current iterative solution and the process is repeated until a feasible solution is found.

In order to describe the surrogate constraint methods precisely, it is necessary to define some notations and list the assumptions made for PART I.

Assumptions and Notations

Notations:

A_i: i-th row of matrix A.

a_{ij} : (i,j)-th element of matrix A.

$K = \{x | Ax \leqq b\}$, the feasible solution set of (1.1)

b_i: i-th element of b vector.

Assumptions:

$\|A_i\|_2 = 1$ for all i=1, ..., m.

System (1.1) is feasible, i. e. $\mathbf{K} \neq \emptyset$ in section 3, 4 and 5.

Additional Notations:

$\mathbf{I} \subset \{1,\dots,m\}$, is an index set.

$\mathbf{H}_i = \{x | A_i x = b_i\}$, where $i \in \{1,\dots,m\}$

$\mathbf{K}_i = \{x | A_i x \leqq b_i\}$, where $i \in \{1,\dots,m\}$

$\mathbf{K}_I = \{\cap \mathbf{K}_i\}_{i \in I}$. $\mathbf{K}_I$ is a convex set and $\mathbf{K}_I \neq \emptyset$, since $\mathbf{K}_I \supset \mathbf{K}$.

$d(x, \mathbf{H}_i)$ = minimum Euclidean distance from x to $\mathbf{H}_i$.

$d(x, \mathbf{K}_i)$ = minimum Euclidean distance from x to $\mathbf{K}_i$. Note that

$d(x, \mathbf{K}_i) = 0$, if $x \in \mathbf{K}_i$; otherwise,

$d(x, \mathbf{K}_i) = d(x, \mathbf{H}_i)$.

$$\phi(x) = \sup_{i \in \{1 \,..,m\}} d(x, \mathbf{K}_i)$$

$d(x, \mathbf{K})$ =minimum Euclidean distance from x to $\mathbf{K}$.

Here we define the length of the binary encoding of all problem data in (1.1) as:

$$L = \sum_i \sum_j \log(|a_{ij}| + 1) + \sum_i \log(|b_i| + 1) + \log nm + 2 \tag{2.1}$$

In sections 3, 4 and 5 a_{ij} and b_i are required to be integers. This is not a serious restriction even if we require $\|A_i\|_2 = 1$ for all i, since we can simply determine L first and then rescale the problem.

Two lemmas which will be used in the convergence proofs are as follows:

LEMMA 2.1. *If the system (1.1) is feasible, then there is a feasible solution $\hat{x}$ with $|\hat{x}_j| \leq 2^L/2n$, $j = 1,\dots,n$.*

LEMMA 2.2. *If the system:* $A_i x < b_i + 2^{-L}$ $(i = 1, \dots, m)$ (2.2)
has a solution, then system (1.1) is feasible.

For the proofs of these lemmas, see [14] [22]. It is well known that if we are able to find a solution x for system (2.2), then there exists a procedure with polynomial complexity which can construct a feasible solution for (1.1) from x. So theoretically, if an algorithm is able to find a solution for (2.2) in a finite number of steps, then the algorithm is also a finite algorithm for solving linear inequality (1.1) [22].

3.THE FIRST ALGORITHM (Basic Surrogate Constraint Method)

The first algorithm is called 'basic surrogate constraint method', in which at each iteration all violated constraints are identified and a 'surrogate constraint' is derived by taking a convex combination of these violated constraints. An orthogonal projection is made onto the surrogate constraint from the current iterative solution and this process is repeated until a feasible solution is found.

Additional Notations

$I = I(x) = \{i \mid A_i x - b_i > 2^{-L}\}$

$A_I = (A_i)_{i \in I}$

$b_I = (b_i)_{i \in I}$

$|I(x)|$, the cardinality of the index set $I(x)$, it is assumed that $|I(x)| = q$.

$\pi = (\pi_1, \dots, \pi_q)$, is a weight vector consisting of positive real numbers satisfying:

$$\pi_i > 0 \text{ for all } i=1 \text{ to } q \text{ and } \sum_{i=1}^{q} \pi_i = 1.$$

Surrogate Constraint: A surrogate constraint for the system $Ax \leqq b$ is the constraint of the following form: $\pi A_I x \leqq \pi b_I$.

Surrogate Hyperplane: $H_s = \{x \mid \pi A_I x = \pi b_I\}$.

3.1. Algorithm 1 (Basic Surrogate Constraint Method).

Initialize: Set $x^0 = 0 \in \mathbb{R}^n$, and k=0; start.

Step 1: If $A x^k \leqq b$ is feasible, stop. x^k is the solution of system (1.1).

Step 2: Determine the index set $I = I(x^k)$,

and select a weight vector π. Compute:

$$x^{k+1} = x^k - \frac{\lambda (\pi A_I x^k - \pi b_I)(\pi A_I)^T}{\|\pi A_I\|^2} \tag{3.1}$$

where $0 < \lambda < 2$.

Set $k = k+1$, and go to Step 1.

Remark 3.1. Apparently, in every iteration of this surrogate constraint algorithm, if the relaxation parameter λ is equal to 1, then the new point x^{k+1} is the orthogonal projection of the current point x^k to the surrogate hyperplane $\mathbf{H_s}$. (See FIG. 3.1)

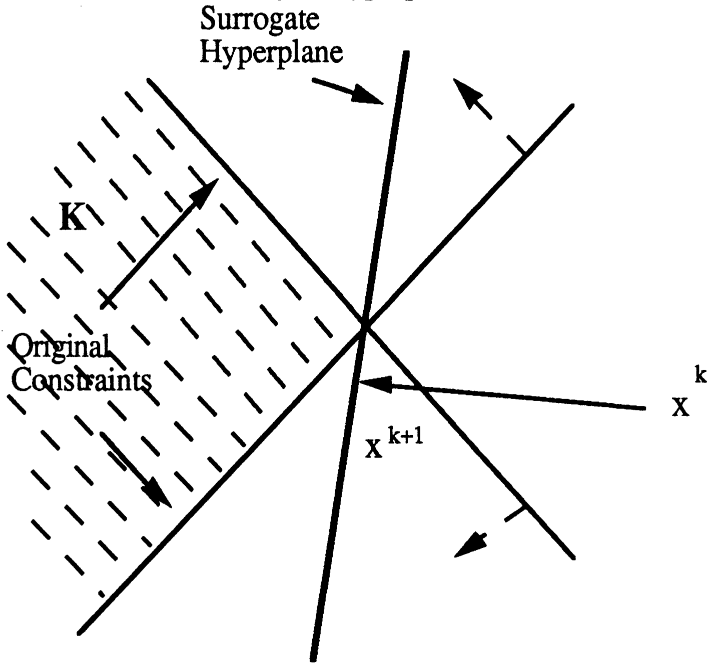


FIG 3.1 Illustration of the Surrogate Constraint Method

Remark 3.2. (Recommended choices of the π vector):

(1) Weight by error.

The quantity $r_i = A_i x^k - b_i$, denotes the Euclidean distance from the current point x^k to $\mathbf{K_i}$,for each $i \in I(x^k)$. Since larger r_i corresponds to greater infeasibility with respect to $\mathbf{K_i}$, it may be desirable to make π_i proportional to r_i. Therefore the following formula for computing the π vector is recommended:

$$\pi_i = \frac{r_i}{\sum_{i\in I} r_i} \qquad \text{clearly,} \quad \pi_i > 0 \quad \text{and} \quad \sum_{i=1}^{q} \pi_i = 1.$$

(2) Weight equally.

$$\pi_i = \frac{1}{q} \ .$$

(3) Convex combination of the two methods:

$$\pi_i = \lambda \frac{r_i}{\sum_{i\in I} r_i} + (1-\lambda)\frac{1}{q} \ , \text{ where } 0 < \lambda < 1.$$

3. 2. Convergence Results.

DEFINITION 3.1. *A sequence $\{x^k\}_{k=1}^{\infty}$ is called strictly Fejer-monotone with respect to the set **K** if for every $x\in$ **K**:*

$$\|x^{k+1} - x\| < \|x^k - x\| \qquad \text{for all } k \geq 0. \tag{3.2}$$

It is easy to check that every Fejer-monotone sequence is bounded.

THEOREM 3.2. *Any sequence $\{x\}_{k=1}^{\infty}$ generated by Algorithm 1 is strictly Fejer-monotone with respect to **K**, provided that $x^k \notin$ **K** for all $k \geq 0$.*

PROOF: $\forall x\in$ **K** we define $e^k = x^k - x$. Then

$$e^{k+1} = e^k - \frac{\lambda\,(\pi A_I x^k - \pi b_I)(\pi A_I)^T}{\|\pi A_I\|^2} \quad \text{and:}$$

$$\|e^{k+1}\|^2 = \|e^k\|^2 + \frac{\lambda^2 \|\pi A_I x^k - \pi b_I\|^2}{\|\pi A_I\|^2} - 2\lambda \frac{(\pi A_I x^k - \pi b_I)}{\|\pi A_I\|^2} (\pi A_I)\, e^k$$

$$= \|e^k\|^2 + \frac{\lambda^2 \|\pi A_I x^k - \pi b_I\|^2}{\|\pi A_I\|^2}$$

$$- 2\lambda \frac{(\pi A_I x^k - \pi b_I)}{\|\pi A_I\|^2} (\pi A_I)(x^k - x)$$

$$= \|e^k\|^2 + \frac{\lambda^2 \|\pi A_I x^k - \pi b_I\|^2}{\|\pi A_I\|^2}$$

$$- 2\lambda \frac{(\pi A_I x^k - \pi b_I)}{\|\pi A_I\|^2} (\pi A_I x^k - \pi b_I - \pi A_I x + \pi b_I)$$

$$= \|e^k\|^2 + \frac{\lambda^2 \|\pi A_I x^k - \pi b_I\|^2}{\|\pi A_I\|^2}$$

$$- \frac{2\lambda \|\pi A_I x^k - \pi b_I\|^2}{\|\pi A_I\|^2} + 2\lambda \frac{(\pi A_I x^k - \pi b_I)(\pi A_I x - \pi b_I)}{\|\pi A_I\|^2}$$

Since $\pi A_I x^k > \pi b_I$ and $\pi A_I x \leq \pi b_I$, we have

$2\lambda \dfrac{(\pi A_I x^k - \pi b_I)(\pi A_I x - \pi b_I)}{\|\pi A_I\|^2} \leq 0$. Therefore it follows that

$$\|e^{k+1}\|^2 \leq \|e^k\|^2 - \lambda(2 - \lambda)\frac{\|\pi A_I x^k - \pi b_I\|^2}{\|\pi A_I\|^2} < \|e^k\|^2 \qquad (3.3)$$

Q.E.D.

Recall that $\phi(x) = \sup_{i \in \{1\ ..,m\}} d(x, \boldsymbol{K}_i)$

THEOREM 3.3. *Any sequence$\{x^k\}_{k=1}^{\infty}$ generated by Algorithm 1 has the property:*

$$\lim_{k=\infty} \phi(x^k) = 0$$

PROOF: Fejer-monotonicity implies that the sequence$\{\|e^k\|\}_{k=1}^{\infty}$ is monotonically decreasing, thus converging. It follows from (3.3) that $\lim_{k=\infty} (\pi A_I x^k - \pi b_I) = 0$
Since $\pi_i > 0$ for all i, which implies that $\lim_{k=\infty} (A_i x^k - b_i) = 0$ for all $i \in I$.
Therefore $\lim_{k=\infty} \phi(x^k) = 0$.

Q.E.D.

LEMMA 3.4: *If $\boldsymbol{K} \neq \varnothing$ an if the sequence$\{x^k\}_{k=1}^{\infty}$ satisfies the conditions*

*(i) $\{x^k\}_{k=1}^{\infty}$ is Fejer-monotone with respect to **K**, and*

(ii) $\lim_{k=\infty} \phi(x^k) = 0$

then $x^k \Rightarrow x \in \boldsymbol{K}$

PROOF: Follows from Lemma 5 and Lemma 6 of Gubin et al.[15]

THEOREM 3.5. *Algorithm 1 converge to a point $x \in \boldsymbol{K}$ in a finite number of iterations.*

PROOF: It follows from Theorem 3.2, Theorem 3.3, and Lemma 3.4. that Algorithm 1 converges to a point $x \in \mathbf{K}$. From Lemma 2.1 it follows that $\|e^0\| \leq 2^{L-1}/\sqrt{n}$, If at iteration k, $A_i x^k - b_i \leq 2^{-L}$ for all $i \in \{1,...,m\}$ then, from Lemma 2.2 the system is feasible and we found a solution for (2.2). A solution for (1.1) can be constructed from x^k by an polynomial procedure. So if we assume that $A_i x^k - b_i \geq 2^{-L}$ for all k, it follows that

$$\pi A_I x^k - \pi b_I \geq \left(\sum_{i=1}^{q} \pi_i \right) \cdot 2^{-L} = 2^{-L}$$

and $\|\pi A_I\| \leq \|\pi\|_2 \|A_I\|_2 \leq \|\pi\|_2 \|A_I\|_F = \left(\sum_{i=1}^{q} \pi_i^2 \right)^{1/2} \left(\sum_{i \in I} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} < q$

where $\|A_I\|_2 = \sup_x \frac{\|A_I x\|_2}{\|x\|_2}$,

$\|A_I\|_F$ is the Frobenius norm of A_I and

$$\|A_I\|_F = \left(\sum_{i \in I} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

It follows that $\|e^{k+1}\|^2 < \|e^k\|^2 - \frac{\lambda(2 - \lambda) \cdot 2^{-L}}{q}$

Therefore, Algorithm 1 converges within k steps where

$$k = \frac{q \cdot 2^{2L-2}/n - 2^{-2L}}{\lambda(2-\lambda) \cdot 2^{-2L}} \leq \frac{q \cdot 2^{4L-2}}{\lambda(2-\lambda) \cdot n}$$

Q.E.D.

4.THE SECOND ALGORITHM
(Sequential Surrogate Constraint Method)

In many applications requiring the solution of linear inequalities, the coefficient matrices of (1.1), that is, the A matrices, are often very large (m and n are of the magnitude 10^5 or more) and sparse (less than 1% of entries are nonzero). If the system

(1.1) is to be solved by the computer on site, working on the whole matrix A is almost impossible. So, it is preferable to work on one small subset of constraints of (1.1) at a time. Specifically, the matrix A can be partitioned into p submatrices, and the right-hand-side vector b can be partitioned into p subvectors, as follows

$$A = \begin{pmatrix} A^1 \\ \cdot \\ A^i \\ \cdot \\ A^p \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} b^1 \\ \cdot \\ b^i \\ \cdot \\ b^p \end{pmatrix} \tag{4.1}$$

where A^i is a matrix with m_i rows and n columns, b^i has m_i rows , for $i = 1$ to p, and

$$\sum_{i=1}^{p} m_i = m \ .$$

Now we will show that the surrogate constraint method can be used to solve the system (1.1) by successively solving the subproblems, $A^i x \leqq b^i$, $i = 1$ to p in cyclic order.

4.1 Notations:

$I^i = I^i (x) = \{i | A^i x - b^i > 2^{-L}\}$ for $i=1$ to p

$q^i = |I^i|$, the cardinality of I^i

$\pi^i = (\pi^i_1, \dots, \pi^i_{m_i})$ for $i= 1$ to p, where $\pi^i_j > 0$ if $j \in I^i$ and $\pi^i_j = 0$ if $j \notin I^i$.

and $\sum_{j \in I^i} \pi^i_j = 1$ for all $i = 1$ to p.

The surrogate constraint of the i-th subproblem is

$$\pi^i A^i x \leqq \pi^i b^i$$

And the surrogate hyperplane of the i-th subproblem is

$$\pi^i A^i x = \pi^i b^i$$

4.2 Algorithm 2 (Sequential Surrogate Constraint Method)

Initialize: Set $x^0 = 0 \in \mathbb{R}^n$, and k=0; start.

Step 1: Do i = 1 to p while $I^i(x^k) \neq \varnothing$ for at least one $i \in \{1,\dots,p\}$.

If the index set $I^i = I^i(x^k) \neq \varnothing$ for subproblem i, then

select a weight vector π^i, and compute

$$x^{k+1} = x^k - \lambda d^k$$

$$\text{where } d^k = \frac{(\pi^i A^i x^k - \pi^i b^i)(\pi^i A^i)^T}{\|\pi^i A^i\|^2} \tag{4.2}$$

where $0 < \lambda < 2$.

Else if the index set $I^i = I^i(x^k) = \varnothing$ for subproblem i, then

$x^{k+1} = x^k$

Endif

Set k = k+1, next i, continue.

Step 2: If $I^i(x^k) = \varnothing$ for all the subproblems $A^i x \leqq b^i$, i=1 to p, then

x^k is a feasible solution, stop.

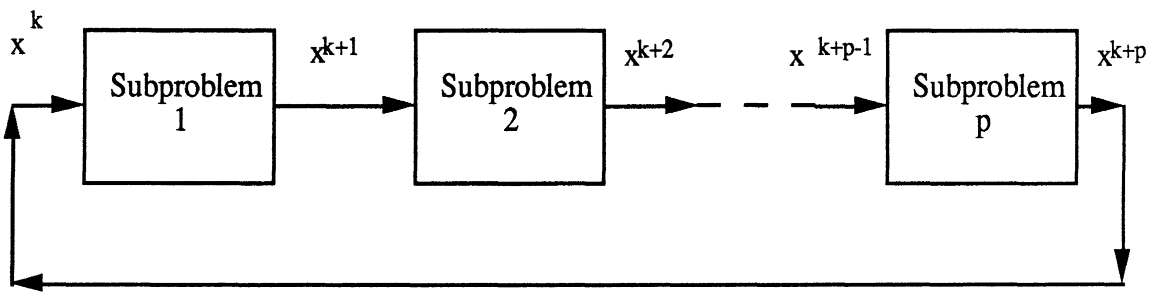


FIG 4.1 Diagram of the Sequential Surrogate Constraint Method

4. 3. Convergence.

LEMMA 4.1. *Let $\Gamma \in \mathbb{R}^n$ be the half space corresponding to the inequality $c^T x \leq c^T c_0$. Let z be an element of $\mathbb{R}^n$ such that $z \notin \Gamma$ and whose projection on Γ coincides with c_0. Then the inequality $//y - z_\lambda// < //y - z//$ is satisfied, where $z_\lambda = z - \lambda(z - c_0)$ for $0<\lambda<2$ and y is an arbitrary element of Γ.*

PROOF: See [1] [12].

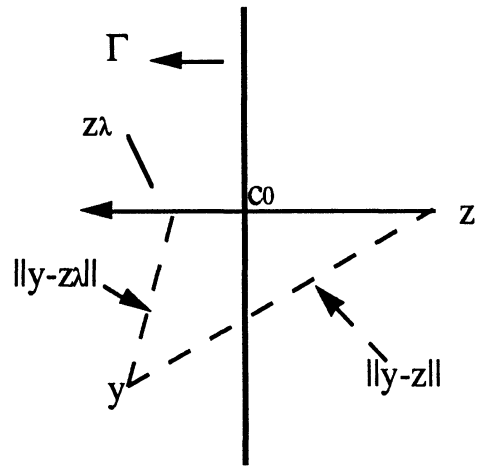


FIG. 4.2 Illustration of Lemma 4.1

COROLLARY 4.2. *Let Γ^i be the half space corresponding to the inequality $\pi^i A^i x \leqq \pi^i b^i$, or $\pi^i A^i x \leqq \pi^i A^i(x^k - d^k)$, for each i=1 to p in Algorithm 2. Let $x_\lambda = x^k - \lambda d^k$ where $0<\lambda<2$. Then, $// y - x_\lambda // < // y - x^k //$, $\forall y \in \Gamma^i$.*

LEMMA 4.3. *If $\Gamma^i = \{x / \pi^i A^i x \leqq \pi^i b^i = \pi^i A^i(x^k - d^k)$) , then $K \subset \Gamma^i$.*

PROOF: Recall that $\mathbf{K} = \{x | Ax \leqq b\}$, clearly $\forall x \in \mathbf{K}$, $\pi^i A^i x \leqq \pi^i b^i$ has to be satisfied for any i = 1 to p. Hence the result follows.

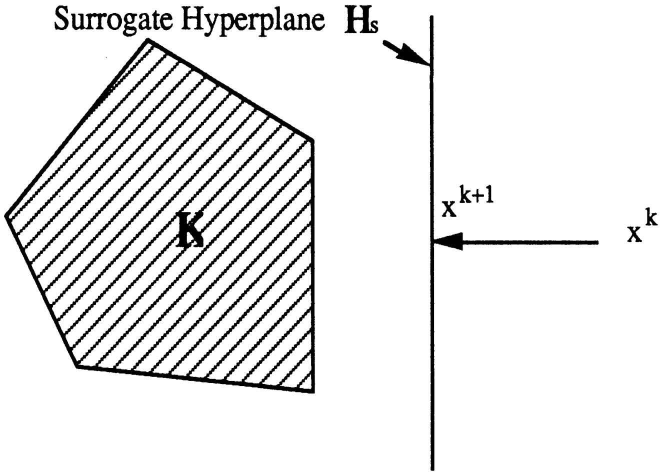


FIG 4.3 Interpretation of Lemma 4.3.

THEOREM 4.4. *$\forall x \in \mathbf{K}$, $\| x - x^{k+1} \| < \| x - x^k \|$ in Algorithm 2.*

PROOF: It follows immediately from Corollary 4.2. and Lemma 4.3.

THEOREM 4.5. *Any sequence $\{x^k\}_{k=1}^{\infty}$ generated by Algorithm 2 has the property:*

$$\lim_{k=\infty} \phi(x^k) = 0$$

PROOF: Theorem 4.4 implies that the sequence $\{\|x^k - x\|\}_{k=1}^{\infty}$ is monotonically decreasing, thus converging. It follows from (4.2) that

$\lim_{k=\infty} (\pi^i A^i x^k - \pi^i b^i) = 0$ for each subproblem $A^i x \leqq b^i$ for i = 1 to p, since for each subproblem π^i_j is strictly positive for all $j \in I^i$ and which implies that

$\lim_{k=\infty} (A^i_j x^k - b^i_j) = 0$ for all $j \in I^i$ and for all subproblem $A^i x \leqq b^i$, i = 1 to p.

Therefore $\lim_{k=\infty} \phi(x^k) = 0$.

Q.E.D.

THEOREM 4.6. *Algorithm 2 converges to a point $x \in \mathbf{K}$ in a finite number of iterations.*

PROOF: It follows from Theorem 4.4, Theorem 4.5 and Lemma 3.4 that Algorithm 2 converges to a point $x \in \mathbf{K}$. From Lemma 2.1 and let's define $e^k = x^k - x$, where $x \in \mathbf{K}$, it follows that:
$\|e^0\| \leq 2^{L-1}/\sqrt{n}$. If $I^i(x^k) \neq \emptyset$ for i-th subproblem , then

$$e^{k+1} = e^k - \lambda \frac{(\pi^i A^i x^k - \pi^i b^i)(\pi^i A^i)^T}{\|\pi^i A^i\|^2} \qquad \text{and}$$

$$\|e^{k+1}\|^2 \leq \|e^k\|^2 - \lambda(2-\lambda)\frac{\|\pi^i A^i x^k - \pi^i b^i\|^2}{\|\pi^i A^i\|^2}$$

After a complete scan of all p subproblems if $I^i(x^k) = \emptyset$ for all i = 1 to p, then x^k is a feasible solution for the problem and we are done. So we assume that there exists at least one subproblem i, $i \in \{1,\dots,p\}$, for each complete scan of all p subproblems, such that $I^i(x^k) \neq \emptyset$, and we assume that $\underset{i}{Max}\, |I^i(x^k)| = q$, It follows that

$$\pi^i A^i x^k - \pi^i b^i \geq \left(\sum_{i=1}^{q} \pi_i\right) \cdot 2^{-L} = 2^{-L}$$

and

$$\|\pi^i A^i\| \leq \|\pi^i\|_2 \|A^i\|_2 \leq \|\pi^i\|_2 \|A_I\|_F = \left(\sum_{i=1}^{q} \pi_i^2\right)^{1/2} \left(\sum_{i \in I} \sum_{j=1}^{n} |a_{ij}|^2\right)^{1/2}$$

$$< q$$

Hence

$$\|e^{k+1}\|^2 < \|e^k\|^2 - \frac{\lambda(2-\lambda)\cdot 2^{-L}}{q} \quad \text{for every p iteration .}$$

Therefore, Algorithm 2 converges within k steps where

$$k = \frac{p\cdot q\cdot 2^{2L-2}/n - 2^{-2L}}{\lambda(2-\lambda)\cdot 2^{-2L}} \leq \frac{p\cdot q\cdot 2^{4L-2}}{\lambda(2-\lambda)\cdot n}$$

Q.E.D.

5. THE THIRD ALGORITHM (Parallel Surrogate Constraint Method)

The surrogate constraint method can also be implemented to work on *ALL* of the subproblems, $A^i x \leqq b^i$, for i =1 to p of (1.1) *SIMULTANEOUSLY* . This is particularly good for the computers with parallel processors.

5.1 Algorithm 3 (Parallel Surrogate Constraint Method)

Initialize: Set $x^0 = 0 \in \mathbb{R}^n$, and k=0; start.

Step 1: Do while $I^i(x^k) \neq \varnothing$ for at least one $i \in \{1,\ldots,p\}$.

For all i=1 to p,

If $I^i(x^k) \neq \varnothing$, then

determine the index set $I^i = I^i(x^k)$, and select a weight vector π^i ,

define: $P_i(x^k) = x^k - d^k$

$$\text{where } d^k = \frac{(\pi^i A^i x^k - \pi^i b^i)(\pi^i A^i)^T}{\|\pi^i A^i\|^2} \tag{5.1}$$

Else, $I^i(x^k) = \varnothing$, define $P_i(x^k) = x^k$

Endif

Compute $P_i(x^k)$ for all $i = 1$ to p simultaneously on parallel processors.

Define $P(x^k) = \sum_{i=1}^{p} \tau_i P_i(x^k)$, and $P_\lambda(x^k) = I + \lambda(P - I)$;

where $\sum_{i=1}^{p} \tau_i = 1, \tau_i > 2^{-L} > 0$ for all i such that $I^i(x^k) \neq \varnothing$

and $0 < \lambda < 2$.

Compute: $x^{k+1} = P_\lambda(x^k)$

Step 2: If $I^i(x^k) = \varnothing$ for all the subproblems $A^i x \leqq b^i$, i=1 to p, then x^k is a feasible solution, stop.

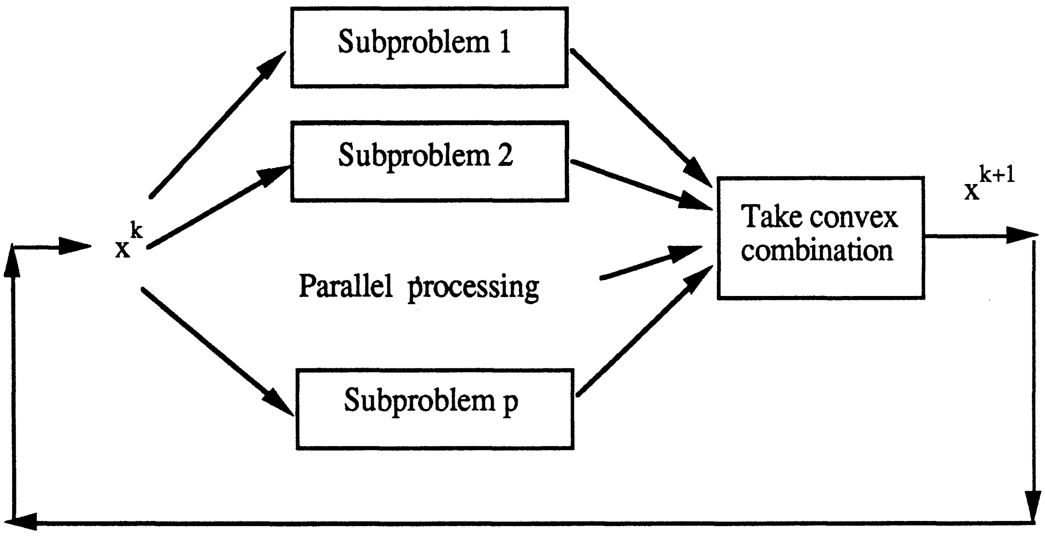


FIG 5.1 Diagram of the Parallel Surrogate Constraint Method

5. 2. Convergence.

THEOREM 5.1. $\forall x \in K \; \| x^{k+1} - x \| < \| x^k - x \|$ *in Algorithm 3.*

PROOF:

$$x^{k+1} = P_\lambda(x^k) = x^k - \lambda \sum_{i=1}^{p} \tau_i \frac{(\pi^i A^i x^k - \pi^i b^i)(\pi^i A^i)^T}{\|\pi^i A^i\|^2}$$

Let: $e^k = x^k - x$, then

$$e^{k+1} = e^k - \lambda \sum_{i=1}^{p} \tau_i \frac{(\pi^i A^i x^k - \pi^i b^i)(\pi^i A^i)^T}{\|\pi^i A^i\|^2} \qquad (5.2)$$

Let $\beta_i = \dfrac{\tau_i (\pi^i A^i x^k - \pi^i b^i)}{\|\pi^i A^i\|^2}$ and it is clear that $\beta_i > 0$, for all i=1 to p

and let: $T = (\beta_1 \pi^1, \ldots, \beta_i \pi^i, \ldots, \beta_p \pi^p) \in \mathbb{R}^{1\times m}$

Then: $(e^{k+1})^T = (e^k)^T - \lambda TA$ and:

$$\|e^{k+1}\|^2 = \|e^k\|^2 + \lambda^2 TAA^T T^T - 2\lambda TAe^k$$

$$= \|e^k\|^2 + \lambda^2 \|TA\|^2 - 2\lambda \sum_{i=1}^{p} \tau_i \frac{(\pi^i A^i x^k - \pi^i b^i)}{\|\pi^i A^i\|^2} [\pi^i A^i (x^k - x)]$$

$$= \|e^k\|^2 + \lambda^2 \|TA\|^2 - 2\lambda \sum_{i=1}^{p} \tau_i \frac{(\pi^i A^i x^k - \pi^i b^i)}{\|\pi^i A^i\|^2} [\pi^i (A^i x^k - b)]$$

$$+ 2\lambda \sum_{i=1}^{p} \tau_i \frac{(\pi^i A^i x^k - \pi^i b^i)}{\|\pi^i A^i\|^2} [\pi^i (A^i x - b)]$$

$$\leq \|e^k\|^2 + (\lambda^2 - 2\lambda) \sum_{i=1}^{p} \tau_i \frac{(\pi^i A^i x^k - \pi^i b^i)}{\|\pi^i A^i\|^2} [\pi^i (A^i x^k - b)]$$

$$= \|e^k\|^2 - \lambda(2-\lambda)\|TA\|^2 \tag{5.3}$$

$$< \|e^k\|^2$$

Q.E.D.

THEOREM 5.2. *Any sequence* $\{x^k\}_{k=1}^{\infty}$ *generated by Algorithm 3 has the property:*

$$\lim_{k=\infty} \phi(x^k) = 0$$

PROOF: Theorem 5.1 implies that the sequence $\{\|x^k - x\|\}_{k=1}^{\infty}$ is monotonically decreasing, thus converging. It follows from (5.2) and since $\tau_i > 0$ for all i such that $I^i(x^k) \neq \emptyset$, then:

$\lim_{k=\infty} (\pi^i A^i x^k - \pi^i b^i) = 0$ for each subproblem $A^i x \leqq b^i$, i = 1 to p, since for each subproblem π^i_j is strictly positive for all $j \in I^i$ and which implies that

$\lim_{k=\infty} (A^i_j x^k - b^i_j) = 0$ for all $j \in I^i$ and for all subproblem $A^i x \leqq b^i$, i = 1 to p.

Therefore $\lim_{k=\infty} \phi(x^k) = 0$.

Q.E.D.

THEOREM 5.3. *Algorithm 3 converges to a point* $x \in K$ *in a finite number of iterations.*

PROOF: It follows from Theorem 5.1., Theorem 5.2. and Lemma 3.4. that Algorithm 3 converge to a point $x \in \mathbf{K}$. From Lemma 2.1 it follows that $\|e^0\| \le 2^{L-1}/\sqrt{n}$. Let $x \in K$ and define $e^k = x^k - x$, if $I^i(x^k) \neq \varnothing$ for at least one subproblem, say, the i-th subproblem, then from (5.2) and (5.3) we get

$$\|e^{k+1}\|^2 \le \|e^k\|^2 - \lambda(2-\lambda)\,\|TA\|^2$$

$$= \|e^k\|^2 - \lambda(2-\lambda) \sum_{i=1}^{p} \tau_i \frac{\|\pi^i A^i x^k - \pi^i b^i\|^2}{\|\pi^i A^i\|^2}$$

$$\le \|e^k\|^2 - \lambda(2-\lambda)\,\tau_i \frac{\|\pi^i A^i x^k - \pi^i b^i\|^2}{\|\pi^i A^i\|^2}$$

If $I^i(x^k) = \varnothing$ for all i subproblems i = 1 to p, then x^k is a feasible solution for the problem and we are done. So we assume that there exist at least one subproblem i, $i \in \{1,\dots,p\}$, such that $I^i(x^k) \neq \varnothing$, again we assume that $A_i x^k - b_i \ge 2 \cdot 2^{-L}$ for all k and assume Max $|I^i(x^k)| = q$ then

$$\pi^i A^i x^k - \pi^i b^i \ge \Big(\sum_{i=1}^{q} \pi_i\Big) \cdot 2^{-L} = 2^{-L} \qquad \text{and}$$

$$\|\pi^i A^i\| \le \|\pi^i\|_2 \|A^i\|_2 \le \|\pi^i\|_2 \|A_I\|_F = \Big(\sum_{i=1}^{q} \pi_i^2\Big)^{1/2} \Big(\sum_{i \in I} \sum_{j=1}^{n} |a_{ij}|^2\Big)^{1/2} < q$$

It follows that

$\|e^{k+1}\|^2 < \|e^k\|^2 - \dfrac{\lambda(2-\lambda)\cdot 2^{-2L}}{q}$ for every p iterations. Therefore, Algorithm 3 converges within k steps where

$$k = \frac{p \cdot q \cdot 2^{2L-2}/n - 2^{-2L}}{\lambda(2-\lambda)\cdot 2^{-4L}} \leq \frac{p \cdot q \cdot 2^{8L-2}}{\lambda(2-\lambda)\cdot n}$$

Q.E.D.

Remark 5.1. It is preferable to choose τ_i in the following way:

If $I^i(x^k) = \emptyset$, then $\tau_i = 0$, Else $\tau_i > 0$ for all $I^i(x^k) \neq \emptyset$ and $\sum_{i=1}^{p} \tau_i = 1.$ and if $I^i(x^k) = \emptyset$ for all i=1 to p , stop.

6. GEOMETRIC INTERPRETATIONS FOR SURROGATE CONSTRAINT METHODS

It is clear from Lemma 4.3 that the surrogate hyperplanes $\mathbf{H}_s$ in Algorithms 1, 2 and 3 are actually separating hyperplanes which separate the current iterative solution x^k from the feasible solution set **K**. So the surrogate constraint methods can also be called 'successive separating hyperplane projection ' methods. It is clear from theorem 4.4 that all surrogate constraint methods possess Fejer-monotone property, which means:
$d(x^{k+1}, \mathbf{K}) < d(x^k, \mathbf{K})$ for every iteration. In Algorithm 1 (basic surrogate constraint method) all violated constraints are identified and the resulting surrogate constraint contains the information on all violated constraints. In Algorithm 2 (sequential surrogate constraint method) only a subset , $I \in \{1,...,m\}$, of constraints are visited, so the resulting surrogate hyperplane $\mathbf{H}_s$ is actually the separating hyperplane which separates $\mathbf{K}_I$ from x^k , and $\mathbf{K}_I \supset \mathbf{K}$. In general, the smaller the subproblem, that is, the smaller the $|I|$, the less information contained in the surrogate hyperplane, and the smaller the improvement at every iteration of the surrogate constraint method and vice versa. (See FIG. 6.1). On the other hand, the smaller the subproblem, the less computational work needed to construct a

surrogate constraint (including scaning and identifying violated constraints), hence the cheaper for every iteration of the surrogate constraint method.

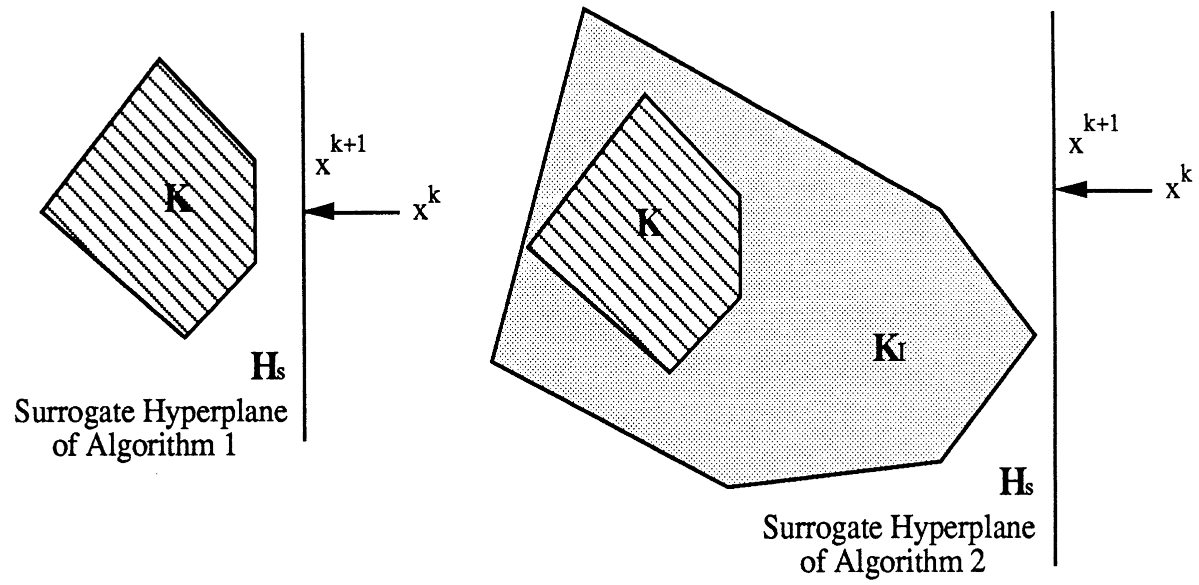
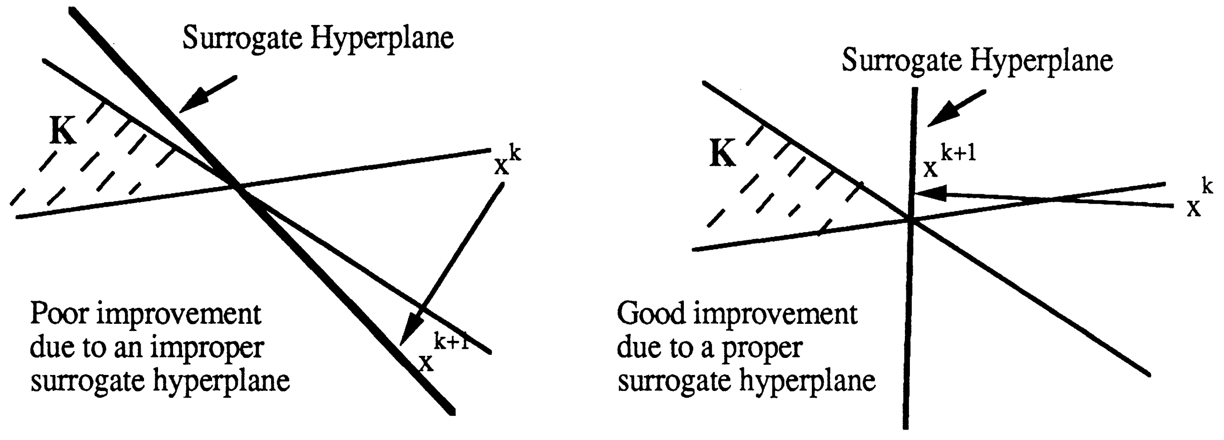

K
x^{k+1}
x^k
H_s
Surrogate Hyperplane of Algorithm 1
K
K_I
x^{k+1}
x^k
H_s
Surrogate Hyperplane of Algorithm 2

FIG. 6.1 Comparison of Surrogate Hyperplanes Generated by Algorithm 1 and Algorithm 2.

The rate of convergence for all surrogate constraint methods depend on the choice of the π vector. A 'better' π vector can make a larger improvement for each iteration of the surrogate constraint method. (FIG. 6.2)


Surrogate Hyperplane
K
x^k
x^{k+1}
Poor improvement due to an improper surrogate hyperplane
Surrogate Hyperplane
K
x^{k+1}
x^k
Good improvement due to a proper surrogate hyperplane

(a) π vector is chosen by 'weight by error' (b) π vector is chosen by 'weight equally'

FIG 6.2 The Effect of a Different Choice of π Vector on the Performance of the Surrogate Constraint Methods

The geometric interpretations of Algorithm 3 (Parallel Surrogate Constraint Method) are summarized in FIG. 6.3. In this example three surrogate hyperplanes are derived from 3 subproblems and orthogonal projections are made simultaneously onto these three surrogate hyperplanes. These three projection points are denoted by the three vertices of the shaded triangle. The new iterative solution x^{k+1} will be somewhere inside the triangle, depending on the choice of the τ vector.

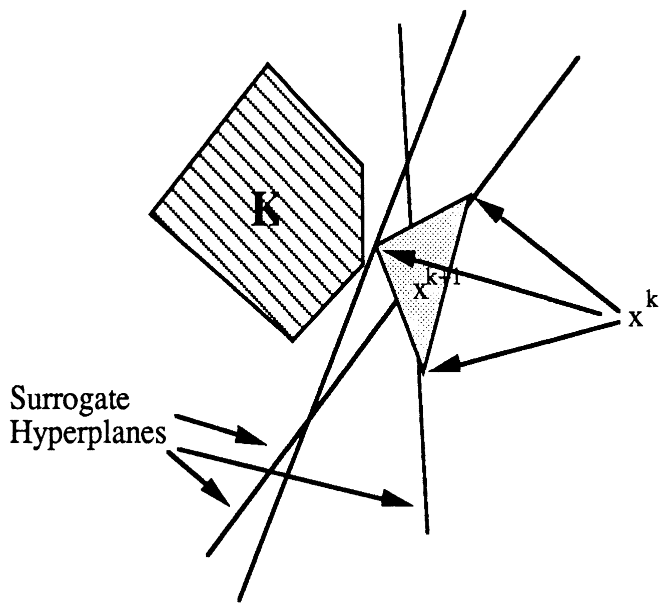


FIG. 6.3. Geometric Interpretation of Algorithm 3

(Parallel Surrogate Constraint Method)

Now let's compare the surrogate constraint methods, that is, Algorithm 1, 2 and 3 in this report, with the relaxation method for solving linear inequalities. The relaxation method is also called the 'successive orthogonal projection method', in which at each iteration an orthogonal projection is made from current iterative solution x^k onto an individual convex set $\mathbf{K}_i$. However, $\mathbf{K}_i$ only contains the information of one constraint. Sometimes the projection on $\mathbf{K}_i$ offers little improvement in reducing the distance from the iteration point x^k to set **K**. On the other hand, the surrogate hyperplane contains the information of more than one violated constraint, so it would generate a better new iterative solution than that of the relaxation method. (See FIG. 6.4)

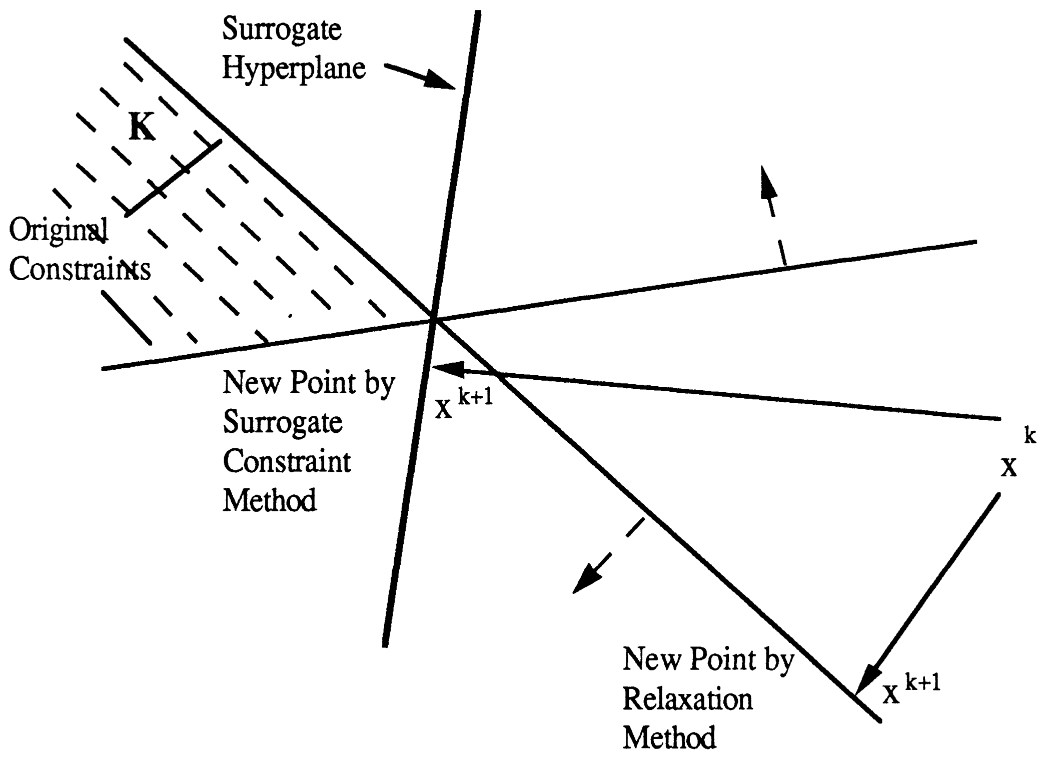

Surrogate Hyperplane
K
Original Constraints
New Point by Surrogate Constraint Method
x^{k+1}
x^{k}
New Point by Relaxation Method
x^{k+1}

FIG. 6.4. Comparison of the Surrogate Constraint Method with the Relaxation Method

We could try to make successive orthogonal projections from current point x^k to set $\mathbf{K}_I$, where I is an index set and $\mathbf{K}_I = \{\cap \mathbf{K}_i\}_{i \in I}$ denotes a group of constraints and $\{\cap \mathbf{K}_I\} = \mathbf{K}$. When $|I| = 1$, $\mathbf{K}_I$ denotes one constraint, and the successive orthogonal projection approach is equivalent to the relaxation method. If $|I| > 1$, then $\mathbf{K}_I$ denotes a polytope. In this case, making an orthogonal projection from current point x^k to set $\mathbf{K}_I$ is simply to find the nearest point in polytope $\mathbf{K}_I$ to current iterative solution x^k , which is very expensive even when $|p| = 2$. In the surrogate constraint methods, instead of making orthogonal projections onto set $\mathbf{K}_I$, the projection is made onto the separating hyperplanes $\mathbf{H}_s$. The projections onto the hyperplanes are easy to calculate and the computational work for constructing those hyperplanes $\mathbf{H}_s$ is very small.

The Cimmino's method for linear inequalities identifies all violated constraint. Othogonal projections are made simultaneously onto all violated constraints from the current iterative solution and then a convex combination of those projection points will be the new iterative solution. (See FIG. 6.5)

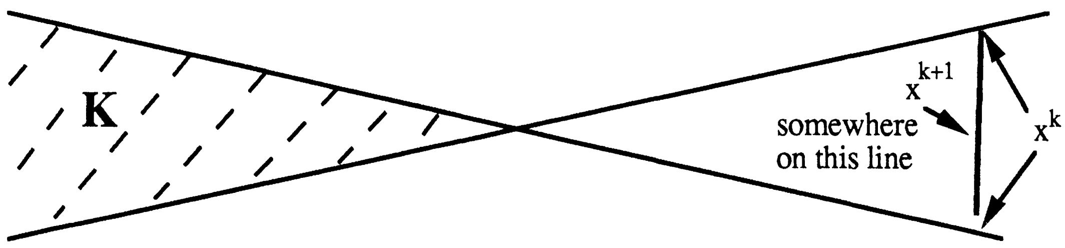


FIG. 6.5. Geometric Interpretation of Cimmino's Method.

Cimmino's method can be implemented on computers with parallel processors. However, when dealing with huge linear inequalities, with many violated constraints, making projections onto *all* violated constraints is expensive and time consuming. On the other hand, by using the parallel surrogate constraint method, the number of subproblems can be chosen by the user, and only one projection is made onto the surrogate hyperplane for each subproblem. Thus, the amount of computational work is much less than that of the Cimmino's method.

7. EXTENSIONS TO LINEAR EQUATIONS

It is very easy to modify Algorithm 1, 2 and 3 to solve a system of linear equations

$$Ax = b \qquad (7.1)$$

by substituting (5.1) with the following equivalent systems of linear inequalities

$$\begin{aligned} Ax &\leqq b \\ -Ax &\leqq -b \qquad (7.2) \end{aligned}$$

and applying Algorithm 1, 2 and 3 to (7.2).

Many of the classical iterative methods, such as the successive approximation method, the Gauss-Seidel method, SOR method, and the steepest descent method, may not always converge for an arbitrary coefficient matrix A. Some methods require A to be positive definite or diagonal dominant, otherwise those methods would have to be applied to the system $A^T A = A^T b$. In the case of successive approximations, convergence requires that the spectral radius of an approximation matrix be less than one.

Whereas the surrogate constraint methods only require that the system (7.1) be feasible. This is one advantage of the surrogate constraint methods over the classical iterative methods.

The surrogate constraint methods for solving linear equations will be presented as follows:

Algorithm 1.1 (Surrogate Constraint Method for Linear Equations)

Initialize: Set $x^0 = 0 \in \mathbb{R}^n$, and $k=0$; start.

Step 1: If $|A x^k - b| \leqq \varepsilon$ is feasible, where ε is a predetermined small positive number, stop. x^k is the solution of system (2.5).

Step 2. Otherwise, partition the rows of the matrix A into 3 sets, A_I, A_{II}, and A_{III}, such that

$$|A_I x^k - b_I| \leqq \varepsilon$$
$$A_{II} x^k < b_{II} - \varepsilon$$
$$A_{III} x^k > b_{III} + \varepsilon$$

Select 2 sets of weight vectors, π^{II} and π^{III} such that:

$$\pi_i^{II} > 0\,,\ \pi_i^{III} > 0 \text{ and } \sum_{i \in II \cup III} \pi_i^{II} + \pi_i^{III} = 1.$$

The surrogate constraint for this problem is

$$\pi_i^{III} A_{III}\, x^k - \pi_i^{II} A_{II}\, x^k = \pi_i^{III} b_{III} - \pi_i^{II} b_{II}$$

Then, compute

$$x^{k+1} = x^k - \frac{\lambda\ (\pi_i^{III} A_{III}\, x^k - \pi_i^{II} A_{II}\, x^k - \pi_i^{III} b_{III} + \pi_i^{II} b_{II})(\pi_i^{III} A_{III} - \pi_i^{II} A_{II})^T}{\|\pi_i^{III} A_{III} - \pi_i^{II} A_{II}\|^2}$$

where $0 < \lambda < 2$. Set k=k+1 and go to step 1.

Algorithm 2.1 (Sequential Surrogate Constraint Method for Linear Equations)

Initialize: Set $x^0 = 0 \in \mathbb{R}^n$, and k=0; start.

Step 1. Partition the rows of the matrix A^i into 3 sets, A_I^i, A_{II}^i and A_{III}^i for all i = 1 to p such that

$$| A_I^i x^k - b_I^i | \leqq \varepsilon$$

$$A^i_{II} x^k < b^i_{II} - \varepsilon$$

$$A^i_{III} x^k > b^i_{III} + \varepsilon$$

Select 2 sets of weight vectors, π^{II} and π^{III} such that:

$$\pi_i^{II} > 0\,,\ \pi_i^{III} > 0 \text{ and } \sum_{i \in II \cup III} \pi_i^{II} + \pi_i^{III} = 1.$$

The surrogate constraint for the i-th subproblem is

$$\pi_i^{III} A^i_{III} x^k - \pi_i^{II} A^i_{II} x^k = \pi_i^{III} b^i_{III} - \pi_i^{II} b^i_{II}$$

Define $I^i (x^k) = II \cup III$ for all $i = 1$ to p.

Step 2: Do $i = 1$ to p while $I^i (x^k) \neq \emptyset$ for at least one $i \in \{1,\ldots,p\}$.

If the index set $I^i = I^i (x^k) \neq \emptyset$ for subproblem i, then

select a weight vector π^i, and compute

$$x^{k+1} = x^k - \lambda d^k$$

where $d^k =$

$$\frac{\left(\pi_i^{III} A^i_{III} x^k - \pi_i^{II} A^i_{II} x^k - \pi_i^{III} b^i_{III} + \pi_i^{II} b^i_{II} \right)\left(\pi_i^{III} A^i_{III} - \pi_i^{II} A^i_{II} \right)^T}{\| \pi_i^{III} A^i_{III} - \pi_i^{II} A^i_{II} \|^2}$$

and $0 < \lambda < 2$.

Else if the index set $I^i = I^i (x^k) = \emptyset$ for subproblem i, then

$$x^{k+1} = x^k$$

Endif

Set $k = k+1$, next i, continue.

Step 3: If $I^i(x^k) = \emptyset$ for all the subproblems $A^i x = b^i$, i=1 to p, then x^k is a feasible solution, stop.

Algorithm 3.1 (Parallel Surrogate Constraint Method for Linear Equations)

Initialize: Set $x^0 = 0 \in \mathbb{R}^n$, and k=0; start.

Step 1. Partition the rows of the matrix A^i into 3 sets, A^i_{I}, A^i_{II} and A^i_{III}

for all i = 1 to p such that

$$| A^i_I x^k - b^i_I | \leqq \varepsilon$$

$$A^i_{II} x^k < b^i_{II} - \varepsilon$$

$$A^i_{III} x^k > b^i_{III} + \varepsilon$$

Select 2 sets of weight vectors, π^{II} and π^{III} such that:

$$\pi_i^{II} > 0 , \pi_i^{III} > 0 \text{ and } \sum_{i \in II \cup III} \pi_i^{II} + \pi_i^{III} = 1.$$

The surrogate constraint for the i-th subproblem is

$$\pi_i^{III} A^i_{III} x^k - \pi_i^{II} A^i_{II} x^k = \pi_i^{III} b^i_{III} - \pi_i^{II} b^i_{II}$$

Define $I^i(x^k) = II \cup III$ for all i = 1 to p.

Step 2: Do while $I^i(x^k) \neq \emptyset$ for at least one $i \in \{1,\ldots,p\}$.

For all i=1 to p,

If $I^i(x^k) \neq \varnothing$, then

determine the index set $I^i = I^i(x^k)$, and select a weight vector π^i,

define: $P_i(x^k) = x^k - d^k$

where $d^k =$

$$\frac{(\pi_i^{III} A_{III}^i x^k - \pi_i^{II} A_{II}^i x^k - \pi_i^{III} b_{III}^i + \pi_i^{II} b_{II}^i)(\pi_i^{III} A_{III}^i - \pi_i^{II} A_{II}^i)^T}{\|\pi_i^{III} A_{III}^i - \pi_i^{II} A_{II}^i\|^2}$$

Else, $I^i(x^k) = \varnothing$, define $P_i(x^k) = x^k$

Endif

Compute $P_i(x^k)$ for all $i = 1$ to p simultaneously on parallel processors.

Define $P(x^k) = \sum_{i=1}^{p} \tau_i P_i(x^k)$, and $P_\lambda(x^k) = I + \lambda(P - I)$;

where $\sum_{i=1}^{p} \tau_i = 1$, $\tau_i > 2^{-L} > 0$ for all i such that $I^i(x^k) \neq \varnothing$

and $0 < \lambda < 2$.

Compute: $x^{k+1} = P_\lambda(x^k)$

Step 3: If $I^i(x^k) = \varnothing$ for all the subproblems $A^i x = b^i$, i=1 to p, then x^k is a feasible solution, stop.

PART II
THE LEAST SQUARES SOLUTION OF LINEAR INEQUALITIES

8. INTRODUCTION AND NOTATIONS

If the feasibility of system $Ax \leqq b$ is unknown, it is often desirable to find a vector $\bar{x}$ that satisfies the system in the least squares sense. In other words, we are interested in a vector $\bar{x}$ that minimizes the quantity $\|(A\bar{x} - b)_+\|^2$, where $(A\bar{x} - b)_+$ is the m-vector whose i-th element is max $\{(A\bar{x} - b)_i, 0\}$. The definition of the least squares solution of linear inequalities and the necessary and sufficient condition for vector $\bar{x}$ to be least squares solution are given as follows:

DEFINITION 8.1. *A least squares solution $\bar{x}$ of $Ax \leqq b$ is a vector which minimizes $f(x) = \frac{1}{2}(Ax-b)_+^T(Ax-b)_+ = \frac{1}{2}\|(Ax - b)_+\|^2$.*

LEMMA 8.2. *x is a least squares solution of the system $Ax \leqq b$ if and only if:*

$$A^T(Ax-b)_+ = 0 \tag{8.1}$$

PROOF: It follows directly from the facts that f(x) is differentiable and convex and its gradient $\nabla f(x)$ is $A^T(Ax-b)_+$.[16]

Q.E.D.

In order to find a least squares solution for system (1.1), we can simply try to use an unconstrained minimization method to minimize the function:

$$f(x) = \frac{1}{2}(Ax-b)_+^T(Ax-b)_+ = \frac{1}{2}\|(Ax - b)_+\|^2$$

However, the function f(x) is not twice differentiable and the powerful Newton's methods are not applicable.

One approach is to solve the above problem iteratively, at any iteration point x, using the following function

$$\bar{f}(x) = \frac{1}{2}(Ax-b)_I^T(Ax-b)_I = \frac{1}{2}\|(Ax - b)_I\|^2$$

as substitute for f(x), where I is an index set and $I=I(x)= \{i|A_i x \geqq b_i\}$. Clearly, $\bar{f}(x)$ is twice differentiable and Newton's method may be applicable.

This approach has been studied by S.P. Han [16]. Though iterative in nature, Han's method is a very fast method for linear inequalities. When applied to linear inequalities with a full dimensional feasible set, that is, *Dim*(**K**) = n, it often produces an solution in a very few number of iterations. The number of iterations required to find a solution is almost independent of the dimension of the problem. Some computational experiment results on Han's method are summarized in Section 11. Han's method is also a finite iterative method; it will find a least squares solution for system (1.1) in a finite number of iterations even if the entries in matrix A or the b vector are real numbers.[16]

The major problem of Han's method is the substitution of f(x) with $\bar{f}(x)$. Han's method finds the new iterative solution by taking a full Newton step from the original iterative solution x in the Newton direction for $\bar{f}(x)$. However, since $\bar{f}(x)$ equals to f(x) only in the neighborhood of the original iterative solution x, many originally satisfied constraints in $A_J x \leqq b_J$ might be violated at the new iterative solution, where J is an index set and $J = \{1,...,m\}\backslash I$. This hampers the performance of Han's method when the feasible set is lower dimensional, that is, when *Dim*(**K**) < n.

One possible alternative is to treat $A_J x \leqq b_J$ as a constraint set for $\bar{f}(x)$. A ' barrier ' function , B(J,x), would be designed to measure the violation of those constraints. Now it is possible to minimize the function $\tilde{f}(x) = \bar{f}(x) + B(J,x)$ by Newton's method. This is the basic idea behind the fourth algorithm (Revised Han's Method).

Notations and Assumptions

$(A_i x - b_i)_+ = \max\{(A_i x - b_i), 0\}$;

$(Ax - b)_+$ = m-vector whose i-th element is $(A_i x - b_i)_+$

$f(x) = \frac{1}{2}(Ax-b)_+^T(Ax-b)_+ = \frac{1}{2}\|(Ax - b)_+\|^2$

$I=I(x)= \{i|A_i x \geqq b_i\}$

$J=J(x)= \{i|A_i x < b_i\} = \{1,...,m)\backslash I$

$|I|$ = cardinality of I

$|J|$ = cardinality of J and we assume that $|J|= q$

$A_I = \{A_i\}_{i\in I}$

$A_J = \{A_i\}_{i\in J}$

$b_I = \{b_i\}_{i\in I}$

$b_J = \{b_i\}_{i\in j}$

$W= \mathrm{diag}(w_i) \in \mathbb{R}^{q\times q}$, where $w_i \geqq 0$ for all i.

$\mathbf{H_I} = \{\cap \mathbf{H_i}\}_{i\in I}$. $\mathbf{H_I}$ is a convex set if the system $A_I y = b_I$ is consistent; otherwise,

$\mathbf{H_I} = \emptyset$.

d_{min} = Min $d(x, \mathbf{H_i})_{i\in I}$.

$d(x, \mathbf{H_I}) = \inf_{\hat{x}} \|\hat{x} - x\|_2$, where $\hat{x}$ is a least-squares solution of the system $A_I y = b_I$.

$d(x, \mathbf{K_I})$ =minimum Euclidean distance from x to $\mathbf{K_I}$.

9. THE REVISED HAN'S METHOD

Algorithm 4 (Revised Han's Method)

Initialize: Set : $x^0 = 0 \in \mathbb{R}^n$, and k=0; start.

Step 1: If $(Ax^k - b)_+ = 0$, then x^k is a feasible solution , or , if $A^T(Ax^k - b)_+ = 0$, then x^k is a least squares solution of the system $Ax \leqq b$, stop. Otherwise, go to step 2.

Step 2: Detect $I = I(x^k)$, and $J = J(x^k)$, and solve the system:

$$A_I \hat{x} = b_I \tag{9.1}$$

$$WA_J \hat{x} = WA_J x^k \tag{9.2}$$

in a least squares sense.

Step 3. Let: $d^k = \hat{x} - x^k$, and $\theta(\lambda) = f(x^k + \lambda d^k)$.

Do a line search to find optimal step size: $\bar{\lambda}$ by minimizing the function $\theta(\bar{\lambda})$.

Step 4: Let $x^{k+1} = x^k + \bar{\lambda} d^k$. Set $k = k+1$, and go to step 1.

Remark 9.1. Since the function $\theta(\lambda) = f(x^k + \lambda d^k)$ is convex, piecewise quadratic and of one variable, the optimal stepsize $\bar{\lambda}$ can be accurately and efficiently computed [19].

Remark 9.2. Algorithm 4 with W=0 is exactly Han's method. To begin the discussion of Algorithm 4 and its convergence proofs, we will analyze some properties of Han's Method as follows.

LEMMA 9.1. $x^{k+1} - x^k = A_I^+ (A_I x^k - b_I)$ (9.3)

in Algorithm 4.[16]

A_I^+ denote the pseudo-inverse of A_I. The pseudo inverse of any matrix B, i.e. $X = B^+$, is the unique matrix X satisfying the following Moore-Penrose conditions:

(a) $BXB = B$; (b) $XBX = X$; (c) $(BX)^T = BX$; (d) $(XB)^T = XB$ (9.4)

There are many efficient methods for solving large, sparse systems of least squares problems, such as system (9.1), (9.2)[23]. An iterative method which is developed by Paige and Saunders is based on the bidiagonalization procedures of Golub and Kahan. It is analytically equivalent to the standard method of conjugate gradients, but possesses more favorable numerical properties. The method, also called algorithm LSQR, is derived by applying the well-known method of Lanczos process. In this method, the matrix A is used only to compute products of the form Av and $A^T u$ for various vectors v and u. Hence, the sparsity of the matrix A can be fully exploited.

Remark 9.3. Newton's Method for solving a system of m equations (nonlinear) in n variables:

$$g_1(x_1,\ldots,x_n) = 0$$

$$g_m(x_1,\ldots,x_n) = 0$$

or

$$g(x) = 0$$

for the case of m=n, is given by:

$$x^{k+1} = x^k - g'(x^k)^{-1} g(x^k) \quad (k=0,1,\ldots)$$

where $g'(x^k)$ is the derivative of g at x^k, represented by the matrix of partial derivatives(the Jacobian matrix). If the nonsingularity of $g'(x^k)$ cannot be assumed for every x^k, and in particular, if the number of equations is different from the number of variables, then it is logical to use the generalized inverse of $g'(x^k)$. This is then called the modified Newton method [24]. We will show that Han's method is a modified Newton's method for solving $g(x^k)=\nabla f(x^k) = 0$, since:

$$\nabla f(x^k)= A^T(Ax^k - b)_+ = A_I^T (A_I x^k - b_I) = A_I^T A_I x^k - A_I^T b_I$$

and

$$(\nabla f(x^k))' = A_I^T A_I$$

$x^{k+1} = x^k - g'(x^k)^{-1} g(x^k)$ with the use of the generalized inverse implies:

$$x^{k+1} = x^k - (A_I^T A_I)^+ A_I^T (A_I x^k - b_I)$$

And it is well known that : $A_I^+ = (A_I^T A_I)^+ A_I^T$

So:
$x^{k+1} = x^k - A_I^+ (A_I x^k - b_I)$ and this is equivalent to solving the system:
$A_I x^{k+1} = b_I$
in a least squares sense. Hence Han's method is equivalent to modified Newton's Method.

Remark 9.4. At each iteration of Algorithm 4, the current iterative solution x^k has the following properties:

$A_I x^k \geqq b_I$
$A_J x^k < b_J$

If we use the above modified Newton method, that is, Han's method, then the new iterative solution x^{k+1} is not the exact new iterative solution desired. While we want minimize f(x)

$= \frac{1}{2}(Ax-b)_+^T(Ax-b)_+$. We would be actually minimizing $\bar{f}(x) = \frac{1}{2}(Ax-b)_I^T(Ax-b)_I$. At the new iterative solution x^{k+1} some of the constraints in $A_J\, x^k \leqq b_J$ could be violated. To remedy this, we could impose some penalty for approaching the boundary of $A_J\, x^k \leqq b_J$ from inside. One way to do this would be to add a barrier function to $\bar{f}(x^k)$, for example, a logarithm barrier function $B(x^k,J)$, where $B(x^k,J) = \sum_{i \in J} \log(z_i)$ and $z_i = b_i - A_i\, x^k$ for all $i \in J$. Thus we now minimize $\bar{f}(x^k) + B(x^k,J)$.

This approach, however, has the following deficiencies:

1). Since $B(x,J)$ is a log function, there are some implementation difficulties.

2). If x approaches the boundary of $A_J\, x \leqq b_J$ from inside, then $B(x,J) \Rightarrow \infty$. This may be too restrictive, since the original system may not be feasible. Adding this logarithm barrier function may prevent us from finding the least squares solution of the linear inequalities.

So, we may want a 'soft' barrier function that imposes only moderate penalties when x approaches the boundary of $A_J\, x \leqq b_J$ from inside. This is exactly why we include:

$$WA_J\, y = WA_J\, x^k \tag{9.2}$$

into system (9.1). It is equivalent to adding a soft barrier function $B(x,J)$ to $\bar{f}(x)$ of the following form:

$$B(x,J) = (A_J\, x - A_J\, x^k)^T W^2 (A_J\, x - A_J\, x^k)$$

This is also called the weighted least squares method. Since (9.1) and (9.2) is usually an overdetermined system, a least squares solution may not satisfy all equations. Also, in general, the smaller the weight for a particular equation, the larger the error. If the current iterative solution is close to the boundary $\mathbf{H_j}$, we assign a larger weight w_j for j-th equation, and if the current iterative solution is far away from the boundary $\mathbf{H_j}$, we assign a smaller or zero weight w_j to the j-th equation. In such a way we can impose the appropriate penalties when x approaches $\mathbf{H_j}$.

10. CONVERGENCE PROOFS AND GEOMETRIC INTERPRETATIONS OF REVISED HAN'S METHOD

10.1. Convergence Results:

LEMMA 10.1. $\nabla f(x^k) = -(A_I^T A_I + W^2 A_J^T A_J)d^k$

PROOF: From Lemma 9.1:
$\nabla f(x^k) = A^T(Ax^k - b)_+ = A_I^T (A_I x^k - b_I) = A_I^T A_I x^k - A_I^T b_I$

The normal equation of (9.1) and (9.2) is the following:

$$(A_I^T A_I + W^2 A_J^T A_J)(x^k + d^k) = A_I^T b_I + W^2 A_J^T A_J x^k \qquad (10.1)$$

But (10.1) is equivalent to:

$$A_I^T A_I x^k - A_I^T b_I = W^2 A_J^T A_J x^k - W^2 A_J^T A_J x^k - (A_I^T A_I + W^2 A_I^T A_J)d^k$$

Hence: $\nabla f(x^k) = -(A_I^T A_I + W^2 A_J^T A_J)d^k$ (10.2)

Q.E.D.

COROLLARY 10.2: $(\nabla f(x^k))^T d^k = -\|A_I d^k\|^2 - \|WA_J d^k\|^2$

LEMMA 10.3.[16] *For any u, v in* $\mathbb{R}^n$, $\|\nabla f(u) - \nabla f(v)\| \leqq \|A\|^2 \|u - v\|$

LEMMA 10.4. *If* x^0 *is the initial guess of the iterative solution, then* $\|\nabla f(x^k)\|^2 \leqq \frac{1}{2}\|A\|^2 f(x^0)$ *for all k in Algorithm 4.*

PROOF: From Corollary 10.2 it is clear that d^k is a descent direction and since x^{k+1} is obtained by performing an optimal line search on f(x), it follows that $f(x^{k+1}) \leqq f(x^k)$ for all k. And from Lemma 8.2 $\nabla f(x^k) = A^T(Ax^k - b)_+$, it follows that

$$\|\nabla f(x^k)\|^2 \leqq \|A^T\|^2\|(Ax^k - b)_+\|^2 = \frac{1}{2}\|A\|^2 f(x^k) \leqq \frac{1}{2}\|A\|^2 f(x^0)$$

THEOREM 10.5: *Let $\{x^k\}$ be generated from any starting point by the Algorithm 4 . Then , either $\nabla f(x^k) = 0$ for some $k<\infty$, or $\lim_{k=\infty} \nabla f(x^k) = 0$*

PROOF: We first note that if $\nabla f(x^k) = 0$ then x^k is a least squares solution of the system and we are done. Hence, we assume that x^k is not a solution and $\nabla f(x^k) \neq 0$. This also implies that the index set $I(x^k) \neq \emptyset$.

Define:

$$c_1 = \|A\|^2$$

$$c_2 = \max_{I \neq \emptyset} \|(A_I^T A_I + W^2 A_J^T A_J x^k)^+\|$$

$$\hat{\lambda} = \frac{-(\nabla f(x^k))^T d^k}{2c_1 \|d^k\|^2}$$

Then it follows that

$$f(x^k + \hat{\lambda} d^k) - f(x^k) = \hat{\lambda} \int_0^1 \nabla f(x^k + t\hat{\lambda} d^k)^T d^k \, dt$$

$$= \hat{\lambda}[\nabla f(x^k)^T d^k + \int_0^1 ((\nabla f(x^k + t\hat{\lambda} d^k) - \nabla f(x^k))^T d^k)dt]$$

From Lemma 10.3. $\nabla f(x^k + t\hat{\lambda} d^k) - \nabla f(x^k) \leqq t\hat{\lambda}\|A\|^2\|d^k\|$

it follows that

$$f(x^k + \hat{\lambda} d^k) - f(x^k) \leqq \hat{\lambda}[\nabla f(x^k)^T d^k + c_1\hat{\lambda}\|d^k\|^2 \int_0^1 t\,dt$$

$$= \frac{\hat{\lambda}}{2}\nabla f(x^k)^T d^k = \frac{-(\nabla f(x^k)^T d^k)^2}{2c_1 \|d^k\|^2}$$

Also from (10.2) it follows that

$$d^k = -(A_I^T A_I + W^2 A_J^T A_J x^k)^+ \nabla f(x^k)$$

Hence

$$\|d^k\|^2 \leqq \|(A_I^T A_I + W^2 A_J^T A_J x^k)^+\|^2 \|\nabla f(x^k)\|^2$$

$$\leqq c_2^2 \|\nabla f(x^k)\|^2$$

Therefore we have

$$f(x^k) - f(x^k + \hat{\lambda} d^k) \geqq \frac{\|\nabla f(x^k)^T d^k\|^2}{2c_1 c_2^2 \|\nabla f(x^k)\|^2}$$

Since $\bar{\lambda}$ is the optimal step size and $x^{k+1} = x^k + \bar{\lambda} d^k$, we have

$$f(x^k) - f(x^{k+1}) \geqq f(x^k) - f(x^k + \hat{\lambda} d^k) \geqq \frac{\|\nabla f(x^k)^T d^k\|^2}{2c_1 c_2^2 \|\nabla f(x^k)\|^2}$$

Since the sequence $\{f(x^k)\}$ is monotone decreasing and bounded below, we have:

$$\infty > \sum_{k=0}^{\infty}(f(x^k) - f(x^{k+1})) \geqq \frac{1}{2c_1 c_2^2} \sum_{k=0}^{\infty} \frac{\|\nabla f(x^k)^T d^k \|^2}{\|\nabla f(x^k)\|^2}$$

Which implies that

$$\lim_{k=\infty} \frac{\|\nabla f(x^k)^T d^k \|^2}{\|\nabla f(x^k)\|^2} = 0$$

From Lemma 10.4 $\|\nabla f(x^k) \|^2 \leqq \frac{1}{2}\|A\|^2 f(x^0)$. Thus it is bounded, which implies:

$$\lim_{k=\infty} \|\nabla f(x^k)^T d^k \|^2 = 0$$

since

$$\|\nabla f(x^k)^T d^k \|^2 = \|A_I d^k \|^2 + \|WA_J d^k \|^2$$

Therefore

$$\lim_{k=\infty} A_I d^k = 0 \quad \text{and} \quad \lim_{k=\infty} WA_J d^k = 0$$

It follows from $\nabla f(x^k) = -(A_I^T A_I + W^2 A_J^T A_J)d^k$ that $\lim_{k=\infty} \nabla f(x^k) = 0$

Q.E.D.

10.2 Geometric Interpretation of Algorithm 4.

10.2.1. Geometric Interpretation of Han's Method

Some preliminaries are needed before we formally give the geometric interpretation of Han's method.

LEMMA 10.6. *In Han's method, let $\hat{x} = x^k + d^k$, then $\hat{x}$ is the closest least squares solution of the system $A_I y = b_I$ to the point x^k in 2-norm.*

PROOF: See theorem 3.2 of [16]

COROLLARY 10.7. $\| d^k \|_2 = d(x^k, H_I)$

THEOREM 10.8. *If the system $A_I y = b_I$ is consistent, and let $\hat{x} = x^k + d^k$, where $\hat{x}$ is the solution of (9.1). Then $\hat{x} \in K_I$. In addition, $\lambda_i = -\left(\frac{A_i x^k - b_i}{A_i d^k}\right) = 1$*

for all $i \in \{i \mid A_i x > b_i\}$

PROOF: If the system $A_I y = b_I$ is consistent, any least squares solution for the system $A_I y = b_I$ is also a feasible solution to $A_I y = b_I$. Since $\hat{x}$ is a least squares solution to $A_I y = b_I$, then it is also a feasible solution to $A_I y = b_I$. Also, if the system $A_I y = b_I$ is consistent, then $H_I \neq \emptyset$ and $\hat{x} \in H_I$. since $H_I \in K_I$. It follows that $\hat{x} \in K_I$, since $H_I \in K_I$. Also substitution of $y = \hat{x} = x^k + d^k$ into $A_i y = b_i$, for all $i \in \{i \mid A_i x > b_i\}$, yields

$A_i\hat{x} = A_i x^k + A_i d^k = b_i$ for all $i \in \{i \mid A_i x > b_i\}$. Hence $\lambda_i = -\left(\frac{A_i x^k - b_i}{A_i d^k}\right) = 1$ for all $i \in \{i \mid A_i x > b_i\}$.

Q.E.D

Remark 10.1. During the process of solving system (9.1), the consistency of the system $A_I y = b_I$ can be automatically determined by checking whether the residual is equal to zero or not.

In Han's method we know that $x^{k+1} = x^k + \lambda d^k$, we may treat:

$$x^{k+1} = x^k + \lambda d^k;$$

$$\lambda \geqq 0 \tag{10.3}$$

as a half line from x^k to x^{k+1}. When the system $A_I y = b_I$ is consistent, then $A_i x^{k+1} > b_i$, $\forall\lambda < 1$, and when $\lambda = 1$, $A_i x^{k+1} = b_i$, for all $i \in \{i \mid A_i x^k > b_i\}$. In other words, when $\lambda = 1$, the end point x^{k+1} hits all the hyperplanes $\{H_i\}_{i \in I}$ simultaneously along the half line (10.3). This is also the geometric interpretation of Han's method when the system $A_I y = b_I$ is consistent. (See FIG. 10.1).

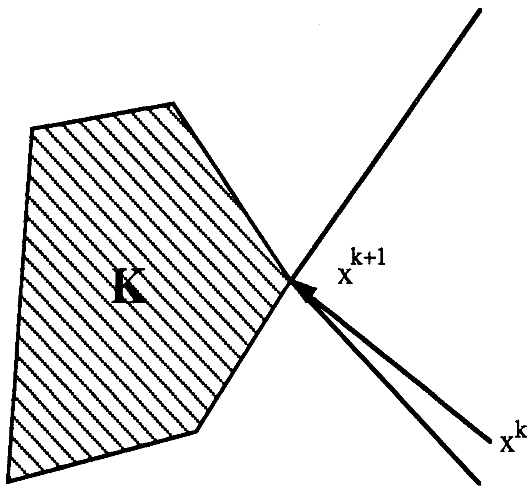


FIG. 10.1 Geometric Interpretation of Han's Method when the System $A_I\, y = b_I$ is Consistent

The results for the case where $A_I\, y = b_I$ is inconsistent are given as follows. The proofs of the Theorems are summarized in the Appendix.

THEOREM 10.9. *If the system $A_I\, y = b_I$ is inconsistent, let $\hat{x} = x^k + d^k$, where $\hat{x}$ is the solution of (9.1). Then $A_I \hat{x} \not> b_I$. In other words, there exists at least one $i \in I$, such that $A_i \hat{x} \leq b_i$.*

Theorem 10.9 indicates that if the step length $\lambda = 1$, then at the new iterative solution x^{k+1}, at least one originally violated constraint is now satisfied.

COROLLARY 10.10. *If the system $A_I\, y = b_I$ is inconsistent, then $\exists \lambda$ such that $0 < \lambda \leqq 1$, and $\lambda = Min\left(\frac{A_i x^k - b_i}{- A_i\, d^k}\right)_{i \in I}$. Let: $x^{k+1} = x^k + \lambda\, d^k$, and for i such that: $Min\left(\frac{A_i x^k - b_i}{- A_i\, d^k}\right)_{i \in I}$ is achieved, there must be: $A_i\, x^{k+1} = b_i$.*

When the system $A_I\, y = b_I$ is inconsistent, we can conclude from the definition of the 'least squares solution' of $A_I\, y = b_I$, $\hat{x}$ is in the *GRAVITATIONAL CENTER* of the set $\{\cup H_i\}_{i \in I}$. (See FIG. 10.2)

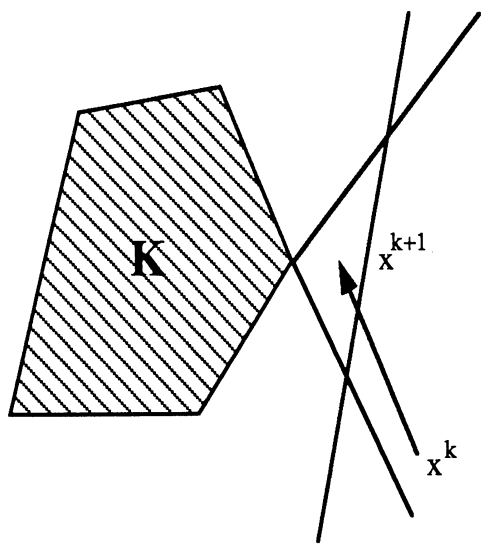


FIG. 10.2. Geometric Interpretation of Han's Method when the System $A_I\, y = b_I$ is Inconsistent

10.2.2. Geometric Interpretation of Algorithm 4

The comparisons of the geometric interpretation between Han's method and Algorithm 4 are illustrated in FIG. 10.3.

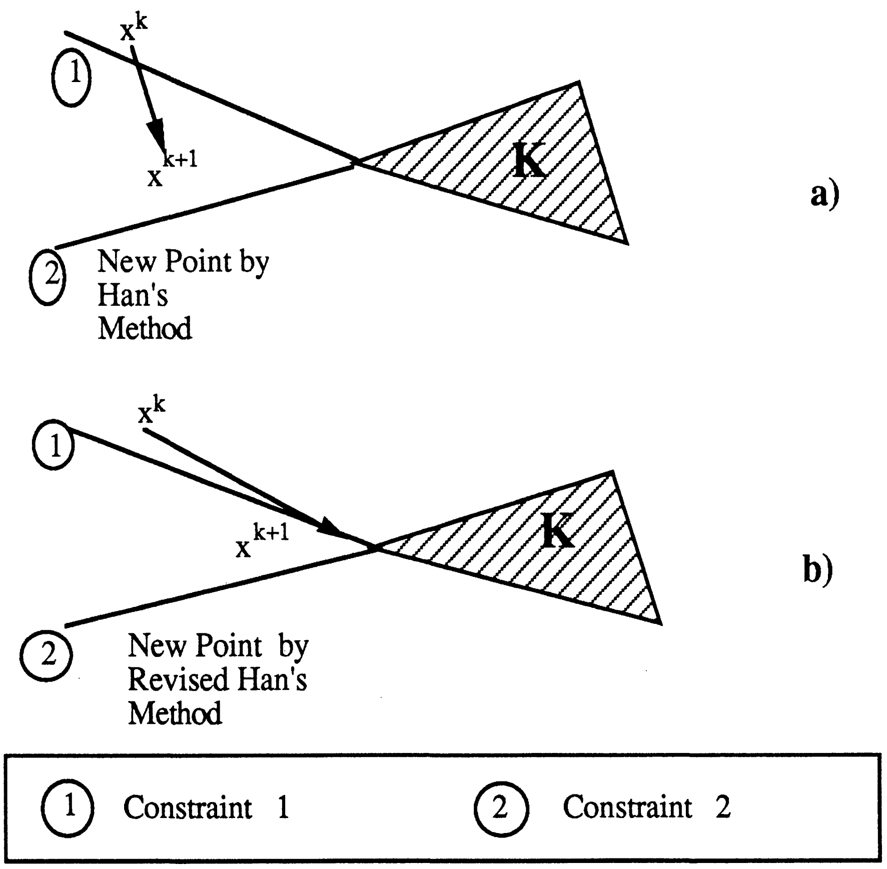


FIG. 10.3. Comparisons Between Han's Method and Algorithm 4.

In FIG. 10.3 a), at current iterative solution x^k, only constraint 2 is violated. For Han's method, the new search direction towards x^{k+1} will be perpendicular to constraint 2. The originally satisfied constraint 1 will soon be violated when moving towards constraint 2. The 'soft barrier' on constraint 1 of Algorithm 4 will make it less likely to for the new iterative solution x^{k+1} to cross constraint 1, as illustrated in FIG. 10.3 b).

11. NUMERICAL RESULTS ON LINEAR INEQUALITIES

The Han's method described in Section 10 has been tested for many randomly generated problems. The search direction d is computed by using algorithm LSQR. The method is very satisfactory in CPU time and number of iterations, where the number of iterations refer to the number of times to compute a new d by LSQR. For each test problem a point with $\|A^T(Ax - b)_+\|^2 \leqq 10^{-20}$ is found. The number of iterations is usually much less than the size of the problem and almost kept constant as problem size grows.

In the following table 11.1, Rows is the number of inequalities and Columns is the number of variables. The computation is done in an IBM 3090 system at the University of Michigan.

TABLE 11.1

Computational Results for Han's Method

Problem Size		**Number of Iterations**	**Total LSQR Steps**	**Average LSQR Steps**
Rows	Columns			
100	100	3	69	23
200	100	7	167	24
200	200	3	94	31
1000	1000	5	243	49
2000	2000	9	416	46
4000	2000	12	457	38
4000	4000	8	277	35

REFERENCES

[1] AGMON,S., The Relaxation Method for linear Inequalities, *Canadian Journal of Mathematics 6 (1954), 382-392*

[2]AL-SULTAN,K.S.; MURTY,K.G., Exterior Point Algorithms for Nearest Point and Convex Quadratic Programs, *Tech. Rept. 89-31, Department of Industrial and Operations Engineering, The University of Michigan, MI*

[3]ANDERSON, D.L.;DZIEWONSKI,A.M., Seismic Tomography, *Sci. Amer. 251 (1984) 58-66*

[4] BEN-ISRAEL,A; GREVILLE,T.N.E., *Generalized Inverses: Theory and Applications, 1974, John Willey & Sons, Inc.*

[5] BREGMAN,L.M.; The Method of Successive Projection for Finding a Common Point of Convex Sets, *Soviet Mathematics Doklady 6, 6 (1965), 688-692*

[6] BREGMAN,L.M., The Relaxation Method of Finding the Common Point of Convex Sets and Its Application to the Solution of Problems in Convex Programming, *U.S.S.R. Computational Mathematics and mathematical Physics 3 (1967), 200-217*

[7] CENSOR,Y. ; HERMAN,G.T., On Some Optimization Techniques in Image Reconstruction From Projections, *Applied Numerical Mathematics 3 (1987) 365-391*

[8] CENSOR,Y. ; ELFVING, T., New Method for Linear Inequalities, *Linear Algebra and Its Applications 42, 199-211 (1982)*

[9] CENSOR,Y. , Row-Action Methods for Huge and Sparse Systems and Their Applications, *SIAM Review, Vol 23, No. 4, Oct. 1981, pp 444-466.*

[10] CIMMINO,G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari, *Ricerca Sci. (Roma), Ser. II, Anno IX, 1 (1938) 326-333*

[11] DE PIERRO,A.R.; IUSEM,A.N., A Simultaneous Projections Method for Linear Inequalities, *Linear Algebra and Its Applications 64, 243-253 (1985)*

[12] EREMIN,I.I., The Relaxation Method of Solving System of Inequalities with Convex Function on the Left Sides, *Soviet Mathematics Doklady 6, (1965), 219-222*

[13] FLEMING,H.E., Satellite Remote Sensing by the Technique of Computerized Tomography, *J. Appl. Meterorology 21 (1982) 1538-1549*

[14] GACS,P.; LOVASZ,L., Khachiyan's Algorithm for Linear Programming, *Report STAN-CS-79-750, Department of Computer Science, Stanford University, (1979)*

[15] GUBIN,L.G.; POLYAK,B.T. AND RAIK,E.V. The Method of Projections for Finding the Common Point of Convex Sets, *U.S.S.R. Computational Mathematics and Mathematical Physics 6 (1967), 1-24*

[16] HAN, S.P., Least-Squares Solution of Linear Inequalities, *Tech. Rept. 2141, Mathematics Research Center, University of Wisconsin, madison, WI, 1980*

[17] KACZMARZ, S., Angenherte Auflosung von Systemn Linearer Gleichungen, *Bull. Internat. Acad. Polon. Sci. Lett. A. 35 (1937) 355-357*

[18] MANGASARIAN, O.L., Normal Solutions of Linear Programs, *Mathematical Programming Study 22 (1984) 206-216*

[19] MIFFLIN, R., A Superlinearly Convergent Algorithm for One-Dimensional Constraint Minimization Problems with Convex Functions, *Mathematics of Operations research, 8 (1983) 185-195*

[20] MOTZKIN, T.S.; SCHOENBERG,I.T., The Relaxation Method For Linear Inequalities, *Canad. J. Math. 6 (1954) 393-404*

[21] MURTY, K.G., *Linear Complementarity, Linear and Nonlinear Programming, 1988, Heldermann Verlag Berlin.*

[22] MURTY, K.G., *Linear Programming, 1983, John Wiley & Sons.*

[23] PAIGE,C.C.; SAUNDERS,M.,A., LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares *ACM Transactions on Mathematical Software, Vol. 8, No. 1, March 1982, Pages 43-71*

[24] RHEINBOLDT,W.C., A Unified Convergence Theory for a Class of Iterative Processes, *SIAM J. Numer. Anal. 5 (1968), 42-63*

[25] TELGEN, J., On Relaxation Methods for System of Linear Inequalities, *European Journal of Operational Research 9 (1982), 184-189*

APPENDIX

This Appendix provides some proofs for Section 10.

LEMMA A.1. *(Gordan's theorem of the alternatives)[21]*

$$B^T x = 0$$

$$x \geq 0 \quad (I) \qquad By > 0 \quad (II)$$

Either (I) or (II) has solutions but not both.

LEMMA A.2. *The system:*

$$\begin{aligned} B^T B u &= B^T v \\ Bu &\geq v \\ v &> 0 \end{aligned} \tag{A.1}$$

has no feasible solution.

PROOF: Suppose (A.1) has a feasible solution, let

$$\mu = Bu - v \geq 0$$

then, (A.1) is equivalent to:

$$\begin{aligned} B^T \mu &= 0 \\ \mu &\geq 0 \end{aligned} \tag{A.2}$$

If (A.2) has a solution, by Lemma A.1, $Bu > 0$ has no solution.

Which contradicts $Bu \geq v > 0$. Hence, (A.2) has no feasible solution.

Q.E.D.

THEOREM A.3. *If the system $A_I y = b_I$ is inconsistent, and let $\hat{x} = x^k + d^k$, where $\hat{x}$ is the solution of (9.1). Then $\hat{x} \notin K_I$.*

PROOF: Since the system $A_I\, y = b_I$ is inconsistent, it follows that

$\mathrm{Min}_d(\| A_I (x^k - d) - b_I \|^2) = \| A_I(x^k - d^k) - b_I\|^2 \neq 0$

Also, the solution of this least squares problem , d^k , must satisfy the following normal equation:

$$A_I^T A_I d^k = A_I^T (A_I x^k - b_I)$$

Now assume that: $\hat{x} \in \mathbf{K}_p$, and since $A_I y = b_I$ is inconsistent, then:

$$A_I \hat{x} = A_I x^k - A_I d^k \leq b_I$$

In other words, the following system:

$$A_I^T A_I d^k = A_I^T (A_I x^k - b_I)$$

$$A_I d^k \geq (A_I x^k - b_I)$$

$$(A_I x^k - b_I) > 0$$

is feasible. But Lemma A.2 has shown that the above system can not have a feasible solution, which leads to a contradiction. Hence $\hat{x} \notin \mathbf{K}_I$.

Q.E.D

Theorem A.3. indicates that when $A_I y = b_I$ is inconsistent, the geometric picture of the Algorithm 4 is quite different from that when $A_I y = b_I$ is consistent. Now we will show that $\hat{x} = x^k - d^k$ must hit or cross at least one of the hyperplanes $\{\mathbf{H}_i\}_{i \in I}$.

LEMMA A.4. *Let y and z are both m-vector, in addition, $y \geq 0$, and $z \leq 0$, If $y^T z \doteq 0$, then $\exists \varepsilon > 0$, such that:*

$$\|y + \varepsilon z\|^2 < \|y\|^2$$

PROOF: Define $f(\varepsilon) = (y+\varepsilon z)^T (y+\varepsilon z) = \|y\|^2 + 2\varepsilon z^T y + 2\varepsilon^2 \|z\|^2$

Solve ε for $\dfrac{df(\varepsilon)}{d\varepsilon} = 0$, yields: $\varepsilon = \dfrac{-z^T y}{\|z\|^2} > 0$

Substitute ε into $f(\varepsilon)$, yields: $f(\varepsilon) = \|y\|^2 - \dfrac{\|z^T y\|^2}{\|z\|^2} < \|y\|^9$

Q.E.D.

THEOREM 10.9. *If the system $A_I\, y = b_I$ is inconsistent, and let $\hat{x} = x^k + d^k$, where $\hat{x}$ is the solution of (9.1). Then $A_I \hat{x} \not> b_I$. In other words, there exists at least one $i \in I$, such that $A_i \hat{x} \leq b_i$.*

PROOF: From Lemma 10.6. $\hat{x}$ is a least squares solution to system $A_I\, y = b_I$. Assume that: $A_I \hat{x} > b_I$. Pick up any $y \in \mathbf{K}_I$, and consider $\delta d = y - \hat{x} \neq 0$, then:
$A_I \delta d = A_I\, y - A_I \hat{x}$. Since: $A_I\, y \leqq b_I$ and $A_I \hat{x} > b_I$. It follows: $A_I \delta d < 0$. Consider the system: $\|A_I(\hat{x} + \varepsilon\delta d) - b_I\|^2 = \|A_I \hat{x} - b_I + \varepsilon A_I \delta d)\|^2$, Since: $A_I \hat{x} - b_I > 0$ and $A_I \delta d < 0$. From the result of Lemma A.4, $\exists \varepsilon > 0$, and $\tilde{x} = \hat{x} + \varepsilon\delta d$, such that:
$\|A_I \tilde{x} - b_I\|^2 < \|A_I \hat{x} - b_I\|^2$, which contradicts $\hat{x}$ is a least squares solution of $A_I\, y = b_I$. So: $A_I \hat{x} \not> b_I$. And it immediately follows that there exists at least one $i \in I$, such that $A_i \hat{x} \leq b_i$

Q.E.D.

COROLLARY 10.10. *If the system $A_I\, y = b_I$ is inconsistent, $\exists \lambda$ such that $0 < \lambda \leqq 1$, $\lambda = Min\left(\dfrac{A_i x^k - b_i}{-A_i d^k}\right)_{i \in I}$. Let: $x^{k+1} = x^k + \lambda d^k$, and for i such that: $Min\left(\dfrac{A_i x^k - b_i}{-A_i d^k}\right)_{i \in I}$ is achieved, there must be: $A_i x^{k+1} = b_i$.*