NEW ITERATIVE METHODS FOR LINEAR INEQUALITIES

Kai Yang

Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, Michigan 48109

Katta G. Murty

Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, Michigan 48109

Technical Report 90-9

March, 1990

Revised September, 1990

NEW ITERATIVE METHODS FOR LINEAR INEQUALITIES

Kai Yang and Katta G. Murty

Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, Michigan 48109

March, 1990
Revised, September, 1990

Abstract

New iterative methods for solving systems of linear inequalities are presented. Each step in these methods consists of finding the orthogonal projection of the current point onto a hyperplane corresponding to a surrogate constraint which is constructed through a positive combination of a group of violated constraints. Both sequential and parallel implementations are discussed.

Key words: Linear inequalities, Surrogate constraint, Iterative methods, Sequential and Parallel Implementations.

1. INTRODUCTION

We consider the problem of finding a feasible solution $\bar{x} \in \mathbb{R}^n$ to a system of linear inequalities:

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

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Large scale versions of this problem have many applications. Some recent applications include *image reconstruction from projections* [5], which is becoming an important problem in many 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 arise in remote sensing [11], seismic tomography [2] 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 [15] [17] or the simplex method to solve the resulting linear program. These methods require matrix operations which are often impractical for the large scale systems that arise in applications such as image reconstruction. Also, in these applications the A matrix tends to have a special structure [5] which is hard to exploit in these methods.

The second approach involves using iterative methods. They always work with the original data and most of them do not need matrix manipulations. The basic computation step in them is extremely simple and easy to program. Because of these advantages, the linear inequality solvers employed in image reconstruction are most often iterative methods. One class of iterative methods are derived from the relaxation method for linear inequalities (Agmon, Motzkin and Schoenberg [1954]) [1] [16] and Kaczmarz's method [14] for linear equations.The word 'relaxation' in the name refers to the fact that they consider one constraint at a time, so in each step, all but one constraint are 'relaxed'. At each iteration a violated constraint is identified and an orthogonal projection is made onto it from the current point. So they are also called 'successive orthogonal projection' methods. Bregman [1965, 1967] [3] [4], Eremin [1965] [10] and Gubin et al. [1967] [13] extended this ' successive orthogonal projection' idea to find a point in a convex set defined by a system of inequalities involving convex functions. Making an orthogonal projection onto a single linear constraint is computationally inexpensive. However, when solving a huge system, considering only one constraint at a time would lead to slow convergence. Instead, it is better to process a group of constraints at a time. But making an orthogonal projection onto the affine space corresponding to a group of constraints is computationally expensive. The amount of work for this projection grows as a cube of the number of constraints in the group.

Another class of iterative methods are derived from Cimmino's algorithm [8] for linear equations. Y. Censor, T. Elfving [1982] [6] and A.R. De Pierro, A.N. Iusem [1985] [9] developed a Cimmino type algorithm for linear inequalities. This method makes orthogonal projections simultaneously onto each of the violated constraints from the current point and takes the new point to be a convex combination of those projection points. Cimmino's method can be implemented very easily on parallel computers. However, when dealing with large systems, with many violated constraints, making projections onto every violated constraint is computationally expensive.

In this paper we propose a class of new iterative algorithms called 'surrogate constraint methods'. They are able to process a group of violated constraints at a time but retain the same computational simplicity of the relaxation method and at the same time they are highly amenable to parallel implementations. In each iteration, a 'surrogate constraint' is derived from a group of violated constraints. The current point is orthogonally projected onto this surrogate constraint treated as an equation and the process is repeated until a feasible solution is found. We discuss three different surrogate constraint methods.

Section 2 defines notation. Sections 3, 4 and 5 describe various algorithms. Section 6 compares these methods with relaxation methods and provides some computational results. Section 7 presents the extension of these methods to solve linear equations.

2. NOTATION AND ASSUMPTIONS

$A = (a_{ij})$, $b = (b_i)$, and A_i denotes the ith row vector of A. We assume that all this data is integer and $A_i \neq 0$ for all i. We let **K** denote the set of feasible solutions of (1.1) and we assume that $\mathbf{K} \neq \emptyset$.

$\mathbf{I} \subset \{1,...,m\}$, denotes an index set, which identifies a subset of the constraints.
$\mathbf{K_i} = \{x \mid A_i x \leqq b_i\}$, is the half space corresponding to the ith constraint.
$\mathbf{H_i} = \{x \mid A_i x = b_i\}$, is the hyperplane corresponding to the boundary of the ith constraint.
$\mathbf{K_I} = \{\cap \mathbf{K_i}\}_{i \in I}$, $\mathbf{K_I} \neq \emptyset$, since $\mathbf{K_I} \supset \mathbf{K}$.
$|S|$ = Cardinality of the set **S**.
A_I = the $|\mathbf{I}|$xn matrix with rows A_i, $i \in \mathbf{I}$.
$b_I = (b_i,\ i \in \mathbf{I})$, a column vector.

$\|x\|$: the Euclidean norm of the vector x, $\|x\| = \sqrt{\sum x_j^2}$

$d(x, \mathbf{H_i})$ = 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 **K**.

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

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

The following lemma [12, 18] will be used in the convergence proofs.

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

Without any loss of generality we assume that first, each row of A is normalized so that $\|A_i\|=1$ for all i = 1 to m. This has no effect on **K**, or $\mathbf{K}_i$ or $\mathbf{H}_i$, but makes it easier to write down the projections of points on hyperplanes $\mathbf{H}_i$ and prove the convergence. Clearly, a point $x \in \mathbf{K}$ iff $\phi(x) = 0$. In practice, we are usually interested in getting an approximate solution for (1.1) within some tolerance. Given a tolerance $\varepsilon > 0$, a point x is said to be 'feasible to (1.1) within tolerance ε' if $\phi(x) \le \varepsilon$, or in other words, we want to find a point $x \in \mathbf{K}_\varepsilon = \{x: A_i x \le b_i + \varepsilon$ for all i = 1 to m$\}$. Clearly if $\varepsilon = 0$, $\mathbf{K}_\varepsilon = \mathbf{K}$.[1]

3.THE BASIC SURROGATE CONSTRAINT METHOD

In this method, at each iteration, the next point is some point in the line segment joining the current point and its reflexion in the hyperplane corresponding to a surrogate constraint which is a nonnegative combination of all the constraints violated by the current point. The actual point selected in this line segment depends on a parameter λ in the algorithm, which can be set by the user anywhere between 0 and 2 ($\lambda = 1$ corresponds to the orthogonal projection).

When x is the current point, let $I = I(x) = \{i: A_i x - b_i > 0\}$. For a given point x, the problem of finding the set I(x), the index set of violated constraints at x, is highly amenable to parallel implementation. If $I(x) \neq \emptyset$, The row vector $\pi = (\pi_i: i \in I(x))$ denotes a positive vector of weights. Given such x and π, the surrogate constraint defined by them is: $(\pi A_I)x \leqq (\pi b_I)$ and the corresponding surrogate hyperplane is $\mathbf{H}_s = \{x: (\pi A_I)x = (\pi b_I)\}$. Also, without any loss of generality, we assume that the π vector is normalized so that $\sum_{i \in I(x)} \pi_i = 1$. This has no effect on the following algorithms, so this assumption is purely for the sake of simplifying some statements.

Algorithm 1: Basic Surrogate Constraint Method

Initialization: Let $x^0 \in \mathbb{R}^n$ be some initial point (it could be 0, or some known near feasible point). Go to Step 1.

General Step k+1: Let x^k be the point obtained at the end of the previous step. Check whether $A_i x^k \leqq b_i$ holds for all i = 1 to m, and identify the index set, $I_k = I(x^k)$, of violated constraints. If $I_k = \emptyset$, x^k is feasible to (1.1), terminate. Otherwise, select a weight vector $\pi(k) = (\pi_i^k :, i \in I_k)$ and compute:

$$x^{k+1} = x^k - \frac{\lambda\,(\pi(k)A_{I_k}x^k - \pi(k)b_{I_k})(\pi(k)A_{I_k})^T}{\|\pi(k)A_{I_k}\|^2} \qquad (3.1)$$

[1] It is well known that if $\varepsilon = 2^{-L}$ and a solution x is found to be feasible with tolerance $\varepsilon = 2^{-L}$, then there exists a procedure with polynomial complexity which can construct a true feasible solution for (1.1) from this approximate solution x. [12] [18]

where $0< \lambda <2$, go to next step.

Remark 3.1. If $\lambda = 1$, x^{k+1} will be the orthogonal projection of the current point x^k on the surrogate hyperplane $\mathbf{H_s}$. (See FIG. 3.1)

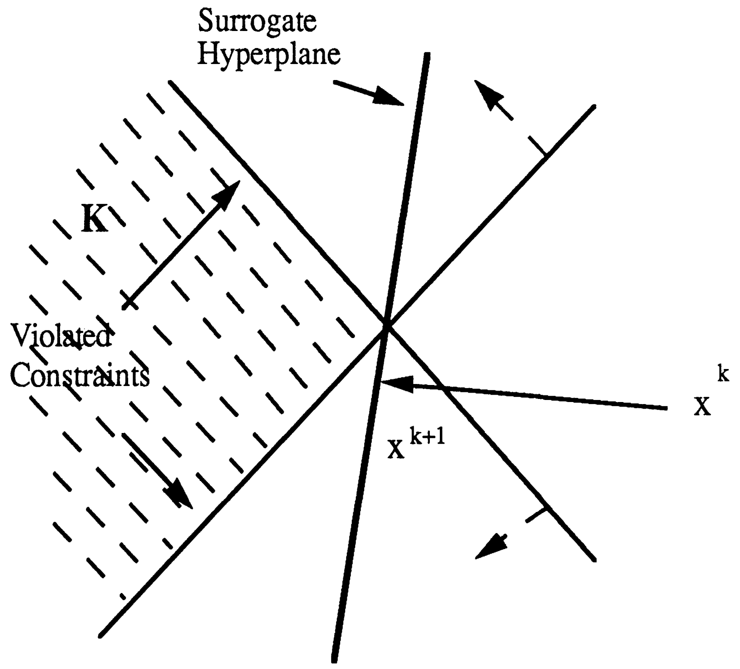


FIG 3.1 Illustration of a Step in the Surrogate Constraint Method with $\lambda=1$

Computationally, the most expensive piece of work in this method is that of finding the index set of violated constraints, $I(x^k)$, in each step. As mentioned above, this can be implemented for parallel computation very easily. This is one of the major advantages of these surrogate constraint methods.

Remark 3.2. Recommended choices of the weight vector π : The following are some of the rules that can be used to select the vector of weights in each step.

(1) Weight by error:

The quantity $r_i = (A_i x^k - b_i)$, is 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, for all $i \in I(x^k)$, that is:

$$\pi_i = \frac{r_i}{\sum_{i \in I(x^k)} r_i}$$

(2) Weight equally: in this rule, $\pi_i = \frac{1}{|I_k|}$, for $i \in I(x^k)$.

(3) Convex combination of the two weights given above:

$$\pi_i = \alpha \frac{r_i}{\sum_{i \in I(x^k)} r_i} + (1 - \alpha)\frac{1}{|I_k|} \text{ where } 0 < \alpha < 1, \text{ for } i \in I(x^k).$$

For the sake of simplicity, all our convergence proofs will be based on the assumption that the weight vector $\pi = (\pi_i : i \in I(x^k))$ is selected so as to satisfy $\pi_i > \gamma$ for all $i \in I(x^k)$, when x^k is the current point, where γ is some predetermined small positive quantity.

Convergence Results.

DEFINITION 3.1. *When $K \neq \varnothing$, a sequence $\{x^k\}_{k=1}^{\infty}$ in $\mathbb{R}^n$ is called strictly Fejer-monotone with respect to the set K if for every $x \in K$:*

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

Every Fejer-monotone sequence is bounded if $K \neq \varnothing$, since $\|x^k - x\|$ is always positive and monotonically decreasing with **K**.

THEOREM 3.1. *If $K \neq \varnothing$, 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: Select any point $x \in$ **K**. Define $e^k = x^k - x$, $k = 0, 1, \ldots$, for simplicity we denote $\pi(k)$ by π and I_k by I. Then, if $I \neq \varnothing$,

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

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

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

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

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

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

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

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

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

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

■

THEOREM 3.2. *If $K \neq \emptyset$, then any sequence $\{x^k\}_{k=1}^{\infty}$ generated by Algorithm 1 has the property:*

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

PROOF: Fejer-monotonicity implies that the sequence $\{\|e^k\|\}_{k=1}^{\infty}$ is monotonically decreasing. Since the sequence $\|e^k\|$ is positive and bounded below thus it converges. Which implies that $\lim_{k\to\infty} \|e^{k+1}\| = \lim_{k\to\infty} \|e^k\|$. It follows from (3.3) that $\lim_{k\to\infty} (\pi(k)A_{I_k}x^k - \pi(k)b_{I_k}) = 0$. Since $\pi_i(k) > \gamma$ for all $i \in I(x^k)$, and $\sum_{i\in I_k} \pi_i^k = 1$ and also since $A_{I_k}x^k - b_{I_k} > 0$ for all $i \in I_k$, this implies that either $\lim_{k\to\infty} (A_i x^k - b_i) = 0$ for all $i \in I_k$, or at some $\bar{k} < \infty$, $I(x^{\bar{k}}) = \emptyset$. Therefore

$$\lim_{k\to\infty} (\phi(x^k)) = \sup_{i\in\{1,..,m\}} d(x^k, \mathbf{K}_i) = 0$$

■

LEMMA 3.1: *If $K \neq \emptyset$, 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) $\lim_{k\to\infty} \phi(x^k) = 0$

*then x^k converges to an $x \in$ **K***

PROOF: Follows from Lemma 5 and Lemma 6 of Gubin et al.[13]

In practice, we are more interested in finding a point $x \in \mathbf{K}_\varepsilon$ quickly. We will show if we set $I(x) = \{i: A_i x - b_i > \varepsilon\}$ in Algorithm 1, then Algorithm 1 converges to a point $x \in \mathbf{K}_\varepsilon$ in a finite number of iterations if $\mathbf{K} \neq \emptyset$.

THEOREM 3.3. *If $K \neq \emptyset$, and we set $I(x^k) = \{i: A_i x^k - b_i > \varepsilon\}$ in Algorithm 1, then it converges to a point $x \in K_\varepsilon$ in a finite number of iterations.*

PROOF:
I) First we show that for any $x \in \mathbf{K}$ if $x^k \notin \mathbf{K}_\varepsilon$, then $\|x^{k+1} - x\| < \|x^k - x\|$ for all $k \geq 0$.This follows directly form the fact that if $x^k \notin \mathbf{K}_\varepsilon$, then $I(x^k) \neq \emptyset$ so Algorithm 1 will not terminate, from (3.3) we have

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

Hence the result follows.

II) Then we show that in this case any sequence$\{x^k\}_{k=1}^{\infty}$ generated by Algorithm 1 has the property: $\lim_{k\to\infty} \phi(x^k) \leq \varepsilon$. This follows from that fact that the sequence$\{\|e^k\|\}_{k=1}^{\infty}$ is positive and monotonically decreasing thus it converges. So: $\lim_{k\to\infty} \|e^{k+1}\| = \lim_{k\to\infty} \|e^k\|$. So it follows from (3.3) that $\lim_{k\to\infty} (\pi(k)A_{I_k}x^k - \pi(k)b_{I_k}) = 0$. But since $\pi(k) > 0$ and $\sum_{i\in I_k} \pi_i^k = 1$ and also since $A_i x^k - b_i > \varepsilon$ for all $i \in I_k$, this implies that $\pi(k)A_{I_k}x^k - \pi(k)b_{I_k} > \varepsilon$ as long as $I_k \neq \emptyset$, which contradicts $\lim_{k\to\infty} (\pi(k)A_{I_k}x^k - \pi(k)b_{I_k}) = 0$. So there must exist $\bar{k} < \infty$, such that $I(x^{\bar{k}}) = \emptyset$. Which implies at iteration $\bar{k}$, $\phi(x^{\bar{k}}) = \sup_{i\in\{1,..,m\}} d((x^{\bar{k}}, \mathbf{K}_i) \leq \varepsilon$. In other words, $x^{\bar{k}} \in \mathbf{K}_\varepsilon$.

III) Bounds on $\bar{k}$: It follows from II) that the sequence of points $\{x^k\}$ produced by Algorithm 1 converges to a point $x^{\bar{k}} \in \mathbf{K}_\varepsilon$. Let $e^k = x^k - x$, where $x \in \mathbf{K}$, from Lemma 2.1, we have $\|e^0\| \leq 2^{L-1}/\sqrt{n}$, If at iteration k, $I(x^k) \neq \emptyset$, $A_i x^k - b_i \geq \varepsilon$ for all $i \in I(x^k)$, it follows that $\pi A_{I_k} x^k - \pi b_{I_k} \geq (\sum_{i\in I_k} \pi_i)\{\min_{i\in I_k}(A_i x^k - b_i)\} \geq (\sum_{i\in I_k}\pi_i)\varepsilon \geq \varepsilon$

and $\|\pi A_{I_k}\| \leq \|\pi\|\|A_{I_k}\| \leq \|\pi\|\|A_{I_k}\|_F = (\sum_{i\in I_k}\pi_i^2)^{1/2}(\sum_{i\in I_k}\sum_{j=1}^{n}|a_{ij}|^2)^{1/2} < m$

where $\|A_{I_k}\|_2 = \sup_x \frac{\|A_{I_k}x\|}{\|x\|}$, $\|A_{I_k}\|_F$ is the Frobenius norm of $A_{I_k} = (\sum_{i \in I_k} \sum_{j=1}^{n} |a_{ij}|^2)^{1/2}$.

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

Therefore, Algorithm 1 converges within $\bar{k}$ steps where $\bar{k} \leqq \frac{m^2 \cdot 2^{2L-2}}{n\lambda(2-\lambda)\cdot\varepsilon^2}$

■

4.THE SEQUENTIAL SURROGATE CONSTRAINT METHOD

In many applications requiring the solution of linear inequalities, the coefficient matrix of (1.1), that is, the A matrix, is often very large (m and n are of the magnitude 10^5 or more) and sparse (less than 0.1% 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 compatibly into p subvectors, as follows

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

where A^t is a m_txn matrix, b^t has m_t rows , for t = 1 to p, and

$$\sum_{t=1}^{p} m_t = m .$$

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

For any $x \in \mathbb{R}^n$, define $I^t(x)$ = {i: ith constraint in t-th subsystem is violated by x }. We denote by $\pi^t = (\pi^t_1, ..., \pi^t_{m_t})$, the weight vector for the t-th subsystem, t = 1 to p. When the current point is x^k, and we have to operate on the t-th subsystem next, we will set $\pi^t_i > 0$ if $i \in I^t(x^k)$, 0 otherwise. Then the corresponding surrogate constraint for the t-th subsystem is: $\pi^t A^t x \leqq \pi^t b^t$, and the surrogate hyperplane of the t-th subsystem is: $H^t_s = \{x: \pi^t A^t x = \pi^t b^t\}$.

The algorithm goes through major cycles. In every major cycle, each of the p subsystems is operated on once, in serial order t =1 to p.

Algorithm 2: Sequential Surrogate Constraint Method

Initialization is the same as in Algorithm 1. Considers a major cycle. In this major cycle, operate on subsystems in the order $t = 1$ to p.

Let x^k be the current point and let the t-th subsystem be the one to be operated next. Find $I^t(x^k)$.

If $I^t(x^k) = \varnothing$, define $x^{k+1} = x^k$ and go to the next subsystem with x^{k+1}, if $t < p$. If $t = p$, this completes the major cycle. If there is no change in the current point throughout this major cycle, then the current point is feasible to (1.1), terminate. Otherwise, go to the next major cycle with the current point.

If $I^t(x^k) \neq \varnothing$, select a weight vector π^t and define $x^{k+1} = x^k - \lambda d^k$

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

and $0 < \lambda < 2$. With x^{k+1}, go to the next subsystem if $t < p$, or to the next major cycle if $t = p$.

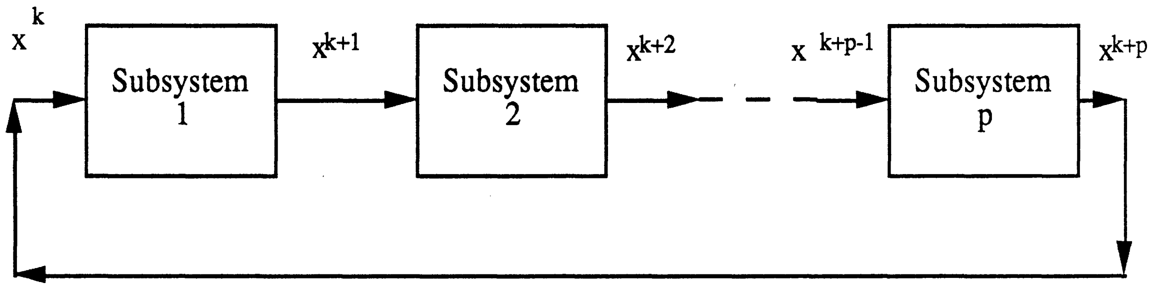


FIG 4.1 Diagram of the Sequential Surrogate Constraint Method

Convergence Results

LEMMA 4.1. *Let $c^0 \in \mathbb{R}^n$ and C^T a row vector in $\mathbb{R}^n$. Let Γ be the half space $\{x: C^T x \leq C^T c^0\}$. If $z \notin \Gamma$ is such that its orthogonal projection on Γ is c^0. Then : $\|y - z_\lambda\| < \|y - z\|$ for all $y \in \Gamma$ and $z_\lambda = z - \lambda(z - c^0)$ for $0<\lambda<2$.*

PROOF: See [1] [10].

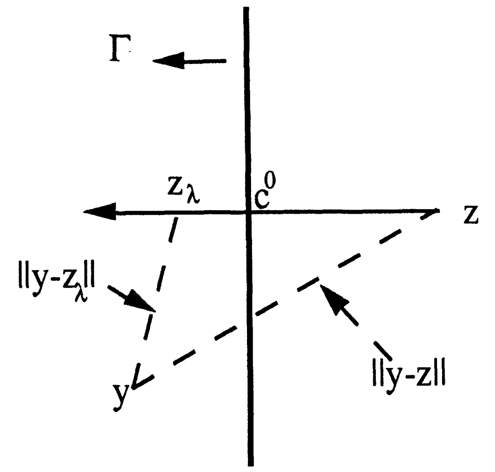


FIG. 4.2 Illustration for Lemma 4.1

Suppose the point x^{k+1} is obtained by operating on the t-th subsystem with x^k as the current point. Then from Algorithm 2, $x^{k+1} = x^k - \lambda d^k$ for some $0 < \lambda < 2$, with d^k as defined in (4.2). The surrogate hyperplane at this iteration is $\mathbf{H}_s^t = \{x: \pi^t A^t x = \pi^t b^t\}$, and $x^k - d^k$ is the orthogonal projection of x^k onto $\mathbf{H}_s^t$. So: $\pi^t A^t(x^k - d^k) = \pi^t b^t$. Also recall that $\mathbf{K} = \{x|Ax \leqq b\}$, clearly for all $x \in \mathbf{K}$, $\pi^t A^t x \leqq \pi^t b^t$ has to be satisfied for any t = 1 to p. Using the above arguments and Lemma 4.1 the following Lemma 4.2 follows directly.

LEMMA 4.2. *Suppose the point x^{k+1} is obtained by operating on the t-th subsystem with x^k as the current point. Let Γ^t be the half-space corresponding to the surrogate constraint in this step, $\pi^t A^t x \leqq \pi^t b^t = \pi^t A^t(x^k - d^k)$. Then $\|y - (x^k - \lambda d^k)\| < \|y - x^k\|$ for all $0 < \lambda < 2$ and for all $y \in \Gamma^t$. Also $K \subset \Gamma^t$.*

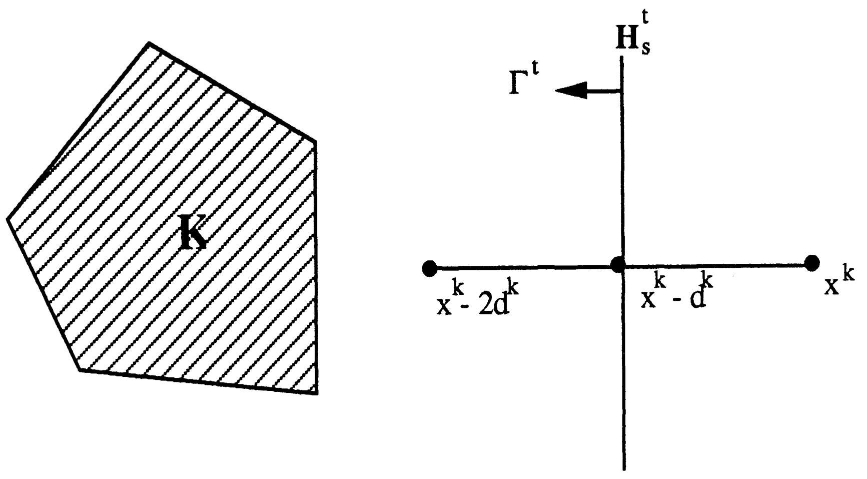


FIG 4.3 Illustration for Lemma 4.2.

THEOREM 4.1. *In Algorithm 2, if $x^{k+1} \neq x^k$, then for all $x \in K$, $\| x - x^{k+1} \| < \| x - x^k \|$.*
PROOF: Follows directly from Lemma 4.2.

THEOREM 4.2. *If $K \neq \emptyset$, any sequence $\{x^k\}_{k=1}^{\infty}$ generated by Algorithm 2 has the property:*

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

PROOF: For any $x \in \mathbf{K}$, Theorem 4.1 implies that the sequence $\{\|x^k - x\|\}_{k=1}^{\infty}$ is monotonically decreasing, thus it converges. So if t(k) denotes the subsystem operated upon when x^k is the current point, and $\pi^{t(k)}$ the weight vector used in that step, then $\lim_{k\to\infty} (\pi^{t(k)}A^{t(k)}x^k - \pi^{t(k)}b^{t(k)}) = 0$. Since $\pi_i^{t(k)} > 0$ for all $i \in I^{t(k)}(x^k)$ and sums to one over these i, and $A_i^{t(k)} x^k - b_i^{t(k)} > 0$ for all $i \in I^{t(k)}(x^k)$, this implies that either

$\lim_{k\to\infty} (A_i^{t(k)} x^k - b_i^{t(k)}) = 0$ for all $i \in I^{t(k)}(x^k)$ or there exist a $\bar{r} < \infty$ such that at major cycle $\bar{r}$, $I^t(x^k) = \emptyset$ for all t = 1 to p. Therefore: $\lim_{k\to\infty} \phi(x^k) = 0$ ■

THEOREM 4.3. *If $K \neq \emptyset$ and, we define $I^t(x^k) = \{i: A_i^t x^k - b_i^t > \varepsilon\}$, then Algorithm 2 converges to a point $x \in K_\varepsilon$ in a finite number of major cycles.*

PROOF: As in the proof of Theorem 3.3, and from Theorem 4.1 we know that if $x^k \notin \mathbf{K}_\varepsilon$, then for all $x \in \mathbf{K}$, $\| x - x^{k+1} \| < \| x - x^k \|$. Define $e^k = x^k - x$, since $\|e^k\|$ is monotonically decreasing and bounded below so it converges. Which implies that $\lim_{k\to\infty} \|e^{k+1}\| = \lim_{k\to\infty} \|e^k\|$. But this happens only if there exists an $\bar{r} < \infty$ such that at major cycle $\bar{r}$, $I^t(x^k) = \emptyset$ for all t = 1 to p. Otherwise for all k there exists a subsystem t such that $I^t(x^k) \neq \emptyset$, then $\pi^t A^t x^k - \pi^t b^t > \varepsilon$ hence $\lim_{k\to\infty} \|e^k\| \neq \lim_{k\to\infty} \|e^{k+1}\|$, a contradiction. So Algorithm 2 must converge to a point in $\mathbf{K}_\varepsilon$ in a finite number of major cycles.

Now we derive a bound on $\bar{r}$. From lemma 2.1 it follows that: $\|e^0\| \leq 2^{L-1}/\sqrt{n}$. If at the begining of major cycle r, the current point $x^k \notin \mathbf{K}_\varepsilon$, there exists at least one subsystem t, and at least one constraint in it, such that $A_i^t x^k - b_i^t > \varepsilon$. So a change of point must occur in some step in this major cycle. Let t be the subsystem operated on in that step, and x^g, x^{g+1} the point at the begining and end of this step, and π^t the weight vector used. So,

$$e^{g+1} = e^g - \lambda \frac{(\pi^t A^t x^g - \pi^t b^t)(\pi^t A^t)^T}{\|\pi^t A^t\|^2} \quad \text{Thus:}$$

$\|e^{g+1}\|^2 \leq \|e^g\|^2 - \lambda(2-\lambda)\dfrac{(\pi^t A^t x^g - \pi^t b^t)^2}{\|\pi^t A^t\|^2}$. As in the proof of Theorem 3.3, we get:

$\|e^{g+1}\|^2 < \|e^g\|^2 - \dfrac{\lambda(2-\lambda)\varepsilon^2}{m^2}$. From this it follows that Algorithm 2 converges within $\bar{r}$ major cycles where $\bar{r} \leqq \dfrac{m^2 \cdot 2^{2L-2}}{n\lambda(2-\lambda)\cdot\varepsilon^2}$

■

5. PARALLEL SURROGATE CONSTRAINT METHOD

The surrogate constraint method can also be implemented to work on *ALL* of the subsystems, $A^t x \leqq b^t$, for t =1 to p of (1.1) *SIMULTANEOUSLY* . This is particularly suited for parallel computation. This algorithm generates one new point in each step and an operation is carried out with the current point on each subsystem in a parallel manner.

Algorithm 3: Parallel Surrogate Constraint Method

Initialization is the same as in Algorithm 1.

General Step k+1: Let x^k be the point obtained at the end of the previous step.

Do the following for each subsystem, $A^t x \leqq b^t$, for t =1 to p parallelly.

Find $I^t(x^k)$ as in Algorithm 2. If $I^t(x^k) = \emptyset$, define $P_t(x^k) = x^k$.

If $I^t(x^k) \neq \emptyset$, select the weight vector π^t as in Algorithm 2, and define: $P_t(x^k) = x^k - d^k$

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

If $I^t(x^k) = \emptyset$, for all t = 1 to p, then x^k is feasible to (1.1), terminate. Otherwise,

$$\text{Define } P(x^k) = \sum_{t=1}^{p} \tau_t P_t(x^k) ,$$

where τ_t are nonnegative numbers summing to 1 with $\tau_t > 0$ for all t such that $I^t(x^k) \neq \emptyset$

Define $x^{k+1} = x^k + \lambda(P(x^k) - x^k)$ where $0 < \lambda < 2$ (5.2)

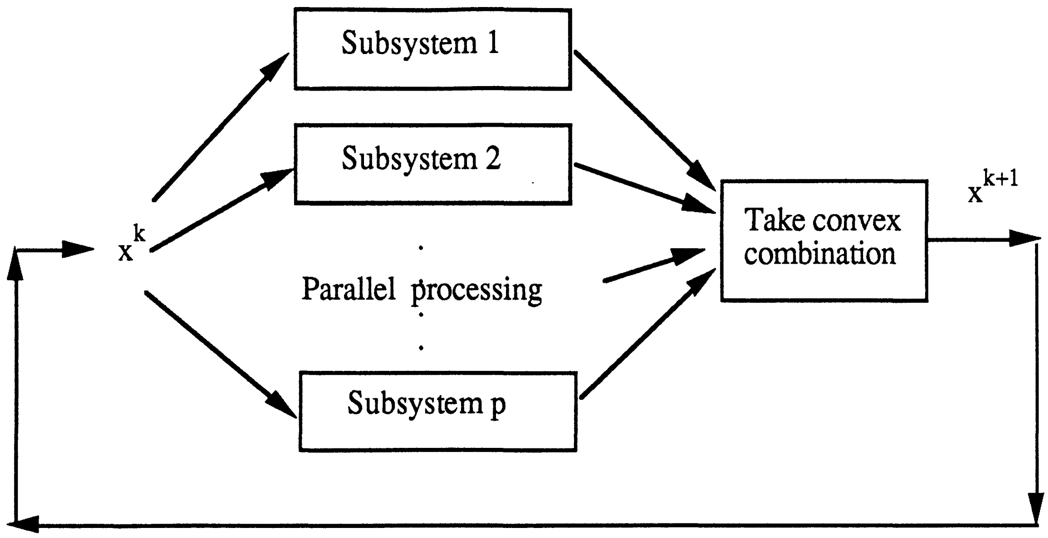


FIG 5.1 Diagram of the Parallel Surrogate Constraint Method

Convergence Results

LEMMA 5.1. *Let $V_t \in \mathbb{R}^n$ for $t = 1$ to p, and $V = \sum_{t=1}^{p} \tau_t V_t$, where $\sum_{t=1}^{p} \tau_t = 1$ and*

$0 \leqq \tau_t \leqq 1$ for all $t = 1$ to p, p is a positive integer, then $\|V\|^2 \leqq \sum_{t=1}^{p} \tau_t \|V_t\|^2$.

PROOF: By induction on p. For $p = 2$, $V = \tau_1 V_1 + \tau_2 V_2$, where $\tau_1 + \tau_2 = 1$, $\tau_1 \geqq 0$ and $\tau_2 \geqq 0$. So $\|V\|^2 = \|\tau_1 V_1 + \tau_2 V_2\|^2 = \tau_1^2 \|V_1\|^2 + \tau_2^2 \|V_2\|^2 + 2\tau_1\tau_2 (V_1)^T V_2$. So,

$$\begin{aligned}
\tau_1\|V_1\|^2 + \tau_2\|V_2\|^2 - \|V\|^2 &= \tau_1\|V_1\|^2 + \tau_2\|V_2\|^2 - \tau_1^2\|V_1\|^2 - \tau_2^2\|V_2\|^2 - 2\tau_1\tau_2(V_1)^T V_2 \\
&= \tau_1(1-\tau_1)\|V_1\|^2 + \tau_2(1-\tau_2)\|V_2\|^2 - 2\tau_1\tau_2(V_1)^T V_2 \\
&= \tau_1\tau_2\|V_1\|^2 + \tau_2\tau_1\|V_2\|^2 - 2\tau_1\tau_2(V_1)^T V_2 \\
&= \tau_1\tau_2\|V_1 - V_2\|^2 \geqq 0 .
\end{aligned}$$

It follows that $\|V\|^2 \leqq \tau_1\|V_1\|^2 + \tau_2\|V_2\|^2$. Hence the assertion is valid for $p = 2$.
Induction Hypothesis: Assume that the assertion is valid for $p - 1$. We will prove that the assertion is also valid for p. Let $V = \sum_{t=1}^{p} \tau_t V_t = \sum_{t=1}^{p-1} \tau_t V_t + \tau_p V_p = (1-\tau_p)\bar{V} + \tau_p V_p$ where

$\bar{V} = \sum_{t=1}^{p-1} \frac{\tau_t}{1-\tau_p} V_t$. From the above argument,

$$\|V\|^2 \leqq \tau_p\|V_p\|^2 + (1-\tau_p)\|\bar{V}\|^2 \leqq \tau_p\|V_p\|^2 + (1-\tau_p)\sum_{t=1}^{p-1} \frac{\tau_t}{1-\tau_p}\|V_t\|^2 = \sum_{t=1}^{p}\tau_t\|V_t\|^2$$

by the induction hypothesis. So, the assertion is valid for all finite positive integers p. ■

THEOREM 5.1. *In Algorithm 3, if $x^{k+1} \neq x^k$, then for all $x \in$* ***K***, *$\| x - x^{k+1} \| < \| x - x^k \|$.*

PROOF:

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

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

$$e^{k+1} = e^k - \lambda\sum_{t=1}^{p} \tau_t \frac{(\pi^t A^t x^k - \pi^t b^t)(\pi^t A^t)^T}{\|\pi^t A^t\|^2} \tag{5.3}$$

Let $V_t = \dfrac{(\pi^t A^t x^k - \pi^t b^t)(\pi^t A^t)^T}{\|\pi^t A^t\|^2}$ and $\displaystyle\sum_{t=1}^{p} \tau_t \frac{(\pi^t A^t x^k - \pi^t b^t)(\pi^t A^t)^T}{\|\pi^t A^t\|^2} = V$,

it is clear that $V = \displaystyle\sum_{t=1}^{p} \tau_t V_t$.

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

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

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

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

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

$$\leq \|e^k\|^2 + \lambda^2 \|V\|^2 - 2\lambda \sum_{t=1}^{p} \frac{\tau_t(\pi^t A^t x^k - \pi^t b^t)^2}{\|\pi^t A^t\|^2}$$

$$= \|e^k\|^2 + \lambda^2 \|V\|^2 - 2\lambda \sum_{t=1}^{p} \tau_t \|V_t\|^2$$

From Lemma 5.1 we have: $\lambda^2 \|V\|^2 \leq \lambda^2 \sum_{t=1}^{p} \tau_t \|V_t\|^2$, So

$$\|e^{k+1}\|^2 \leq \|e^k\|^2 - \lambda(2-\lambda) \sum_{t=1}^{p} \tau_t \|V_t\|^2 < \|e^k\|^2 \tag{5.4}$$

■

THEOREM 5.2. *If $K \neq \varnothing$, any sequence $\{x^k\}_{k=1}^{\infty}$ generated by Algorithm 3 has the property:*

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

PROOF: For any $x \in \mathbf{K}$, Theorem 5.1 implies that the sequence $\{\|x^k - x\|\}_{k=1}^{\infty}$ is monotonically decreasing, thus it converges. It follows from (5.4) that

$$\lim_{k\to\infty} \sum_{t=1}^{p} \frac{\tau_t(\pi^t A^t x^k - \pi^t b^t)^2}{\|\pi^t A^t\|^2} = 0$$

Since $\tau_t > 0$ for all t such that $I^t(x^k) \neq \varnothing$, and the τ_t sum to 1, the above implies that $\lim_{k\to\infty} (\pi^t A^t x^k - \pi^t b^t) = 0$ for all such t. Since for each subsystem π_j^t are strictly positive for all $j \in I^t(x^k)$ and sum to 1, this implies that either $\lim_{k\to\infty} (A_j^t x^k - b_j^t) = 0$ for all $j \in I^t(x^k)$ and for all subsystems, t = 1 to p or there exists a $\bar{k} < \infty$ such that $I^t(x^{\bar{k}}) = \varnothing$ for all t = 1 to p. Therefore $\lim_{k\to\infty} \phi(x^k) = 0$.

■

For the sake of simplicity, our following finite convergence proofs will be based on the assumption that τ_t are all equal for all t = 1 to p, that is, $\tau_t = \frac{1}{p}$ for all t. In practice, a different choice may yield better computational performance, this has to be determined in a computational experiment.

THEOREM 5.3. *If $K \neq \emptyset$, and we define $I^t(x^k) = \{i: A_i^t x^k - b_i^t > \varepsilon\}$ in Algorithm 3 , then it converges to a point $x \in K_\varepsilon$ in a finite number of steps.*

PROOF: As in the proof of Theorem 3.3, and from Theorem 5.1 we know that if $x^k \notin \mathbf{K}_\varepsilon$, then for all $x \in \mathbf{K}$, $\| x - x^{k+1} \| < \| x - x^k \|$. Define $e^k = x^k - x$, since $\|e^k\|$ is monotonically decreasing and bounded below so it converges. Which implies that

$\lim_{k\to\infty} \|e^{k+1}\| = \lim_{k\to\infty} \|e^k\|$. But this happens only if there exist a $\bar{k} < \infty$ such that at $\bar{k}$th iteration, $I^t(x^{\bar{k}}) = \emptyset$ for all $t = 1$ to p. Otherwise for all k there exists at least one subsystem t such that $I^t(x^k) \neq \emptyset$ then $\pi^t A^t x^k - \pi^t b^t > \varepsilon$ .Since $\|e^{k+1}\| \leq \|e^k\|^2 +$

$(\lambda^2 - 2\lambda)\sum_{t=1}^{p} \frac{\tau_t(\pi^t A^t x^k - \pi^t b^t)^2}{\|\pi^t A^t\|^2}$, and this implies that $\lim_{k\to\infty} \|e^k\| \neq \lim_{k\to\infty} \|e^{k+1}\|$, a contradiction. So Algorithm 3 must converge to a point in $\mathbf{K}_\varepsilon$.

From lemma 2.1 it follows that: $\|e^0\| \leq 2^{L-1}/\sqrt{n}$. If $I^t(x^k) \neq \emptyset$ for at least one subsystem, say, the t-th subsystem , then from (5.3) and (5.4) we get

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

Since: $\tau_t = \frac{1}{p} < \frac{1}{m}$,

$$\pi^t A^t x^k - \pi^t b^t \geq (\sum_i \pi_i)\cdot\varepsilon = \varepsilon, \text{ and } \|\pi^t A^t\| \leq \|\pi^t\|_2 \|A^t\|_2 \leq \|\pi^t\|_2 \|A^t\|_F < m$$

It follows that $\|e^{k+1}\|^2 < \|e^k\|^2 - \frac{\lambda(2-\lambda)\varepsilon^2}{m^3}$

Therefore, Algorithm 3 converges to a point in $\mathbf{K}_\varepsilon$ within $\bar{k}$ steps where

$$\bar{k} \leq \frac{m^3 \cdot 2^{2L-2}}{n\lambda(2-\lambda)\cdot\varepsilon^2}$$

■

A different parallel method

In the parallel method discussed above, we obtained for the t-th subsystem a surrogate constraint

$$\pi^t A^t x \leqq \pi^t b^t \tag{5.5}$$

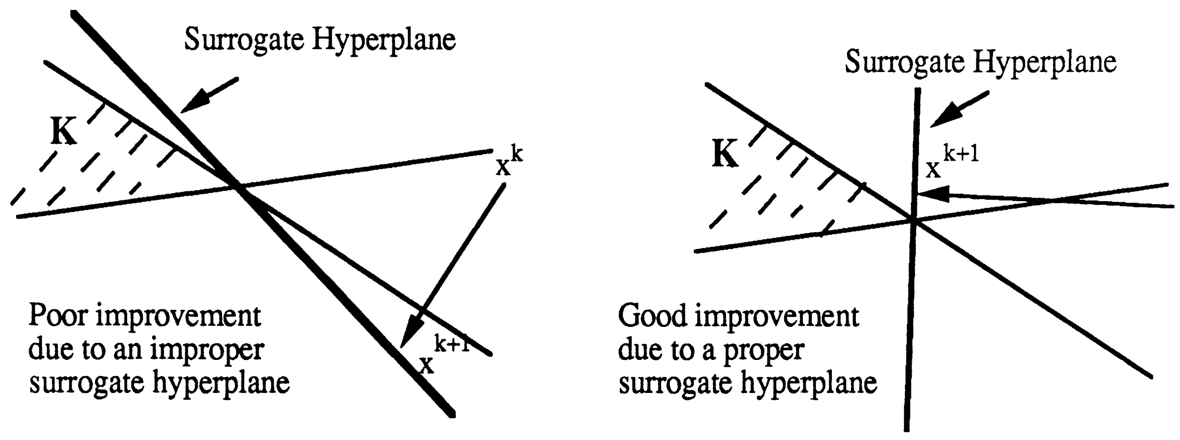
and a point $P_t(x^k)$ by projecting x^k onto this surrogate hyperplane, for each $t = 1$ to p such that $I^t(x^k) \neq \emptyset$. And the new point x^{k+1} is derived from a weighted average of these $P_t(x^k)$. So this method can be viewed as a Cimmino type method using groups of constraints instead of individual constraints, and surrogation within each group. Another parallel method would just obtain the surrogate constraint (5.5) for each subsystem t such that $I^t(x^k) \neq \emptyset$. Then it would take a positive combination of all such surrogate constraints generated, leading to a surrogate constraint for the entire original system (1.1). If the weight assigned to t is $\delta_t > 0$, this constraint will be:

$$\sum_{t \in \{I^t(x^k) \neq \emptyset\}} \delta_t (\pi^t A^t x) \leqq \sum_{t \in \{I^t(x^k) \neq \emptyset\}} \delta_t (\pi^t b^t) \tag{5.6}$$

The point $P(x^k)$ is then defined to be the orthogonal projection of x^k onto (5.6) treated as an equation, and the next point x^k is obtained as in (5.2) using this $P(x^k)$. This method is essentially Algorithm 1 using a parallel implementation for identifying all the violated constraints, with a different processor examing the constraints in each subsystem.

6. COMPARISONS WITH EARLIER METHODS

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 , see Figure 6.1.


Surrogate Hyperplane
K
x^k
x^{k+1}
Poor improvement due to an improper surrogate hyperplane
Surrogate Hyperplane
K
x^{k+1}
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 6.1 The Effect of the π Vector on the Performance of the Surrogate Constraint Methods

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. In the relaxation

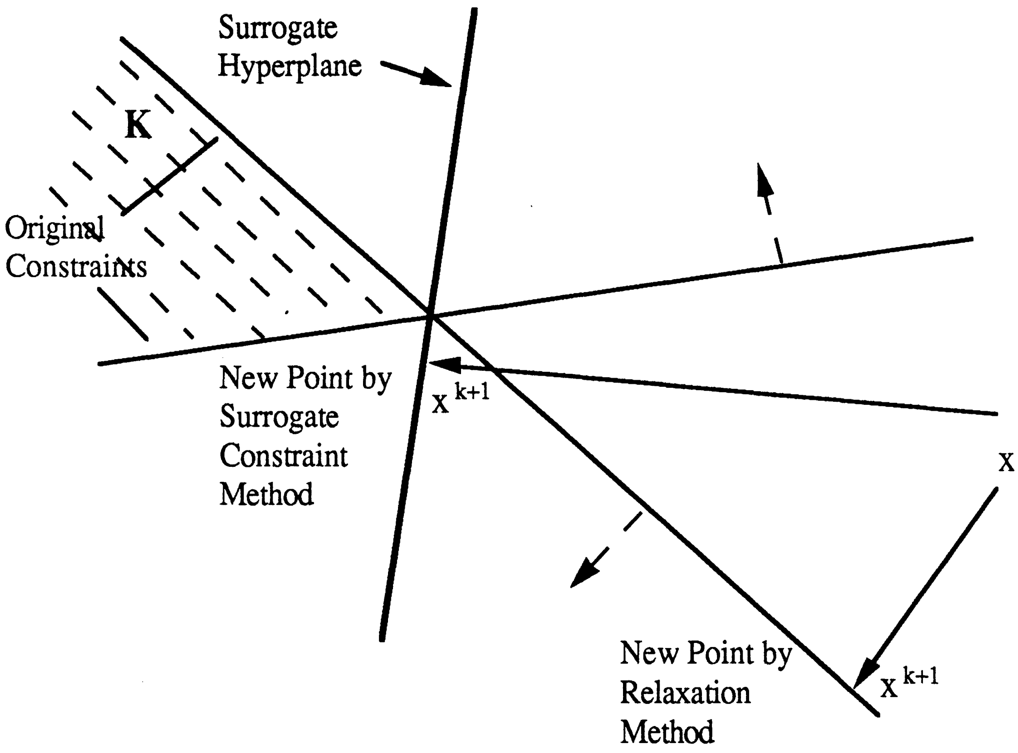

Surrogate Hyperplane
K
Original Constraints
New Point by Surrogate Constraint Method
x^{k+1}
x^k
New Point by Relaxation Method
x^{k+1}

FIG. 6.2 Comparison of the Surrogate Constraint Method with the Relaxation Method

method at each iteration an orthogonal projection is made from current point x^k onto an individual $\mathbf{K}_i$ for some 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 is expected to generate a better new point than the relaxation method. (See FIG. 6.2)

Cimmino's method for linear inequalities identifies all violated constraints in each iteration. Othogonal projections are made simultaneously onto all violated constraints from the current point and the new point is a convex combination of those projection points. (See FIG. 6.3)

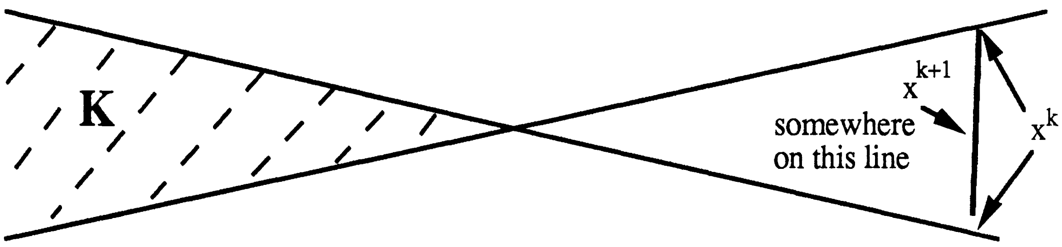


FIG. 6.3. Geometric Interpretation of Cimmino's Method.

Computational experiments have been carried out to compare the sequential surrogate constraint method (Algorithm 2) with the version of the relaxation method that processes the inequalities in cyclical order. We give below our preliminary computational results on randomly generated large sparse problems carried out on the IBM 3090-400/VM main frame computer at the University of Michigan. The problems are generated in such a way that the system would have an interior feasible solution. The sequential surrogate constraint method is implemented using in each step the weights suggested in (3) of Remark (3.2) with $\alpha = 0.2$. The value of λ for both methods was taken to be 1.7.

The results are listed in Table 6.1. Five test problems were generated in each dimension. The speedup of surrogate constraint method over the relaxation method ranged from 30 to 60. The speedup increases as the problem size increases.

TABLE 6.1
Comparison of the Relaxation Method and Algorithm 2
(Sequential Surrogate Constraint Method)

Problem Size		**Sparsity of the Problem**	**Relaxation Method**			**Sequential Surrogate Constraint Method**			
Rows	Columns	%	*	**	+	#	~	+	##
5000	2500	2.0	10500	4.9	17.04	2	3.4	0.511	2500
5000	5000	1.0	10675	5.2	23.49	2	3.2	0.544	2500
10000	2500	1.0	22375	6.3	45.11	5	2.7	0.806	2000
10000	5000	0.4	20985	5.9	54.06	5	3.7	1.146	2000
10000	10000	0.4	23125	7.1	84.22	5	2.8	1.371	2000
18000	5000	0.5	41125	8.4	213.00	9	3.4	3.332	2000
18000	9000	0.2	44750	9.3	255.13	9	3.8	4.002	2000

Five test problems were generated for each dimension and the accuracy = 10^{-9}

*	Average number of projections	+	Average CPU time (seconds)
#	Number of subsystems	##	Number of rows in each subsystem
**	Number of sweeps	~	Number of major cycles

In the relaxation method, in each sweep, all the constraints are examined once from top to bottom. The average number of sweeps before termination varied from 5 to 10 among the problem sizes. Since the current point changes after each projection, it is not possible to implement a sweep in this method in a parallel fashion.

In the sequential surrogate constraint method, in each major cycle, the number of projections made is at most equal to the number of subsystems. In each major cycle, each constraint is examined once, but as explained earlier, this work can easily be parallelized. Also, the number of major cycles needed in the surrogate constraint method is much less than the number of sweeps needed in the relaxation method to achieve the same accuracy.

These computational results are very encouraging. More extensive experimentation is necessary to determine the strategies to implement the surrogate constraint methods for obtaining the best performance, things such as the best choice for the weight vector in each step, etc.

7. EXTENSIONS TO LINEAR EQUATIONS

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

$$Ax = b \tag{7.1}$$

by applying them on the following equivalent systems of linear inequalities

$$\begin{aligned} Ax &\leqq b \\ -Ax &\leqq -b \end{aligned} \tag{7.2}$$

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 $A^TAx = A^Tb$. 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.

For each i, system (7.2) has both the constraints $A_ix \leqq b_i$ and $A_ix \geqq b_i$ .When x^k is the current point, if $A_ix^k = b_i$, both these constraints are satisfied. Otherwise, $A_ix^k \neq b_i$, and exactly one of the constraints in the above is violated, while the other one is satisfied. Thus, when x^k is the current point, the set of violated constraints in (7.2) includes at most one of the constraints from the pair $A_ix \leqq b_i$ and $A_ix \geqq b_i$ .Using this, simplifications can be made in executing Algorithm 1, 2 or 3 on the system (7.2).

Acknowledgements: We are grateful to a refree for pointing out the result in Lemma 5.1 and its importance in the proof of Theorem 5.1.

REFERENCES

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

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

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

[4] 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*

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

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

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

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

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

[10] 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*

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

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

[13] 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*

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

[15] KARMARKAR, N.; A New Polynomial Algorithm for Linear Programming, *Comninatorica*, 4 (1984) 373-395

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

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

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

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