数理解析研究所講究録
第250巻 1975年 76-120

76

大規模線型計画問題の緩和法による解法

DUAL RELAXATION METHOD FOR LINEAR PROGRAMMING WITH CONJUGATE GRADIENT ALGORITHM

KUNIO TANABE

THE INSTITUTE OF STATISTICAL MATHEMATICS
4-6-7 MINAMI-AZABU, MINATO-KU
TOKYO, JAPAN

ABSTRACT

An iterative algorithm is developed for solving large scale linear programming problems. The algorithm is essentially an iterative relaxation method for solving a system of linear equality and inequalities which is introduced by using the duality relation between the primal and dual linear programming problems. The relaxation process of the algorithm is performed by using the method of conjugate gradients for solving linear equations and a specially devised linear search algorithm, which can be viewed as a modified Gauss-Newton process for minimizing a certain function. It is shown that the algorithm always converges in finite number of steps to a vector, which is an optimal solution when it exists. The algorithm is conceptually very simple and shares with general iterative methods the desirable features; it always work with original data and is self-correcting and stable.

1. INTRODUCTION

Linear programming has become an important tool for solving a wide range of problems encountered in business, industry and government. There have been many ways to solve linear programming problems [5, 9, 19 - 22, 32, 36] but the Simplex method due to Dantzig and its extensions have almost been exclusively used to solve them. At the present time, linear programming practioners are required to solve larger and larger problems. However, the numerical solution of a large scale linear programming problem contains various difficulties. The main difficulties are the computational instabilities and the growth of quantity of data handled in the course of the Simplex iterations. There have been several alternative schemes [2, 3, 17, 50] for ensuring its numerical stability. The compact inverse techniques [7, 10, 12, 15, 33, 44, 51] have been designed to control the growth of the number of non-zero entries in the Simplex tableau. The decomposition principle [8, 11, 16, 42] have been developed to reduce the quantity of the data handled, by taking advantage of the special structure of problems. These techniques have been successfully applied to large scale problems but when the scale is vastly large the difficulties arise even with these methods; besides, they may pursue a very long laborious tour of vertices before the optimal solution is reached [26, 29, 30, 50], since each step of the Simplex method consists of searching for an adjoint vertex which gives an improved value of the objective function and of determining this vertex numerically by an elimination method for solving linear equations. Although the determination

of the numerical values of the vertices from step to step is indispensable to the Simplex method it is only a necessary mean to identify constraints active at the optimal solution. Hence, it is natural to investigate the way to speed up the convergence to the optimal solution of large scale problems. This consideration leads to the development of different techniques from the Simplex method. There have been several attempts on this line [19 - 22, 32, 36].

While the Simplex method and its variants adapt the direct methods for solving linear equations, the methods of Agmon [1], Cline and Pyle [5], Held and Karp [21] and Oettli [36] correspond to iterative relaxation method for solving linear equations. But the convergence of their methods is essentially linear and generally too slow for practical applications. The method presented here uses similar point of view, in Geoffrion's terminology [16], a 'dualization relaxation' approach, but different techniques are employed for relaxation. In the relaxation method of Agmon [1], Motzkin and Shoenberg [34] for solving a system of linear inequalities, the most violated inequality is relaxed at each step by 'Kaczmarz-like' projection method [27, 38, 46, 47] which results in the linear convergence of the method. In our method, all the violated constraints are relaxed at each step by using the conjugate gradient method which has been appraised as one of the best available algorithm for solving a sparse set of linear equations [23, 40, 41]. This ensures the finite termination property of our method even when the optimal solution is not unique or there exists no feasible solution at all for the given problem.

Our method is conceptually very simple and shares with general iterative methods the desirable feasures;

(1) it always works with original data hence can take advantage of the sparseness of given data, and

(2) it is self-correcting and stable, hence any approximation to the optimal solution can be incorporated profitably.

In what follows, we will use the following notations:

$F^{\dagger}$: The Moore-Penrose inverse of matrix F.

A^t, x^t : Transpose of matrix A or column vector x.

$Im\ F$: The range space of matrix F.

$Ker\ F$: The null space of matrix F.

$P_{S,T}$: The projection matrix onto S along T.

R^n : The vector space formed by n-dimensional column vectors

$\bar{C}$: The topological closure of set C.

2. FORMULATION OF THE PROBLEM

Consider the linear programming problem of the form;

maximize $$c_1^t x_1 + c_2^t x_2$$

subject to

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 &= b_1 , \\ A_{21}x_1 + A_{22}x_2 &\leqq b_2 \quad \text{and} \\ x_2 &\geqq 0 , \end{aligned} \tag{1}$$

where A_{11} and A_{22} are $m_1 \times n_1$ and $m_2 \times n_2$ matrices and b_1, b_2, c_1, c_2, x_1 and x_2 are column vectors of suitable dimensions. The dual of the problem is to

minimize $$b_1^t y_1 + b_2^t y_2$$

subject to

$$\begin{aligned} A_{11}^t y_1 + A_{21}^t y_2 &= c_1 , \\ A_{12}^t y_1 + A_{22}^t y_2 &\geqq c_2 \quad \text{and} \\ y_2 &\geqq 0 , \end{aligned} \tag{2}$$

where y_1 and y_2 are column vectors of suitable dimensions.

It is well-known that the optimal solutions $\bar{x}$ and $\bar{y}$ of these problems, if they exist, are the solution of the system of the linear equalities and inequalities (1) and (2) and the equality

$$c_1^t x_1 + c_2^t x_2 = b_1^t y_1 + b_2^t y_2 , \tag{3}$$

This system is equivalently expressed, for example, as the system of the equality

$$(3') \qquad \phi_0(z) \equiv c^t x - b^t y = 0 ,$$

and $2(m+n)$ inequalities

$$(4) \qquad \begin{aligned}
\phi_i(z) &\equiv \theta_i - \sum_{i=1}^{n} \alpha_{ij}\xi_j \geq 0, \quad (1 \leq i \leq m) \\
\phi_{i+m}(z) &\equiv \sum_{i=1}^{n} \alpha_{ij}\xi_j - \theta_i \geq 0, \quad (1 \leq i \leq m_1) \\
\phi_{j+m+m_1}(z) &\equiv \xi_{j+n_1} \geq 0, \quad (1 \leq j \leq n_2) \\
\phi_{j+m+m_1+n_2}(z) &\equiv \sum_{i=1}^{m} \alpha_{ij}\eta_i - \chi_j \geq 0, \quad (1 \leq j \leq n) \\
\phi_{j+m+n+m_1+n_2}(z) &\equiv \chi_j - \sum_{i=1}^{m} \alpha_{ij}\eta_i \geq 0, \quad (1 \leq j \leq n_1) \\
\phi_{i+m+2n+m_1}(z) &\equiv \eta_{i+m} \geq 0, \quad (1 \leq i \leq m_2) ,
\end{aligned}$$

where $m \equiv m_1 + m_2$, $n \equiv n_1 + n_2$,

$$A \equiv (\alpha_{ij}) \equiv \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad b \equiv (\theta_i) \equiv \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

$$c \equiv (\chi_i) \equiv \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad x \equiv (\xi_i) \equiv \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

$$y \equiv (\eta_i) \equiv \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \text{ and } z \equiv \begin{pmatrix} x \\ y \end{pmatrix}.$$

Conversely, if the system defined by (3') and (4) has a solution $\bar{z}^t = (\bar{x}^t : \bar{y}^t)$ then $\bar{x}$ and $\bar{y}$ are the solutions of the primal and dual problems respectively. Then the problem is equivalent to finding a solution of this system. We shall denote by S the (possibly empty) set of vectors z which satisfy the system.

There are many methods for solving a system of linear inequalities [1, 13, 18, 24, 25, 34, 35, 37]. In this paper, we develop a new method for finding a solution of the system (3) (4). Our approach is similar to that of Nagaraja and Krishna [35] for solving the special system of linear inequalities of the form

$$Ax \geqq 0 \ . \tag{5}$$

But different techniques are used by taking advantage of the inherent structure of the system introduced from the dual linear programming problems.

Given a point $z \in R^{m+n}$, if

$$\phi_k(z) \leqq 0 \quad \text{for some} \quad k \neq 0 \ , \tag{6}$$

then the constraint $\phi_k(z) \geqq 0$ and the corresponding hyperplane H_k defined by

$$H_k \equiv \{z: \phi_k(z) = 0\} \ , \tag{7}$$

are said to be active for z. The set of indexes of active constraints for z will be denoted by $A(z)$, i.e.,

$$A(z) \equiv \{k: \phi_k(z) \leqq 0 \quad \text{and} \quad 1 \leqq k \leqq 2(m+n)\} \ . \tag{8}$$

Note here that the equality sign is included in the expression (6), which will be seen to guarantee the finite convergence of our method. Hyperplanes H_k defined by the linear equalities

$$H_k: \phi_k(z) = 0, \quad k = 1, 2, \ldots, 2(m+n) \ , \tag{9}$$

divide the whole space R^{m+n} into mutually disjoint convex sets C_ℓ. More precisely, let C_ℓ, $\ell = 1, 2, \ldots, N$, be equivalent classes given by the equivalence relation $\sim$ between elements in R^{m+n}, which is defined by

$$z_1 \sim z_2 \iff A(z_1) = A(z_2) . \tag{10}$$

Each class C_ℓ thus defined is a convex set which possibly has no interior point. We shall call $\{C_\ell\}$ a partition of R^{m+n} by the system (3') - (4). We can well define active constraints for C_ℓ and the corresponding index set $A(C_\ell)$ by

$$A(C_\ell) \equiv A(z) \quad \text{for} \quad z \in C_\ell .$$

Let a function $V(z)$ be defined by

$$V(z) \equiv \sum_{i=1}^{2(m+n)} (\phi_i(z))_-^2 = \sum_{i \in A(z)} \phi_i^2(z) , \tag{11}$$

where

$$(d)_- = \begin{cases} 0 & \text{if} \quad d > 0 \\ d & \text{if} \quad d \leqq 0 . \end{cases}$$

Since $(d)_-^2$ is a continuously differentiable convex function in d and $\phi_i(z)$'s are linear functions in z, $V(z)$ is a continuously differentiable convex function in z. Also $V(z)$ is a quadratic spline function in z, that is, it is quadratic on every convex set C_ℓ. We note that

$$V(z) \geqq 0 ,$$

and the equality holds, iff, the system of inequalities (3') and (4) is consistent.

Let a linear manifold L be defined by

$$L \equiv \{z: \phi_0(z) = 0\} \tag{12}$$

and let $V_L(z)$ be the restriction of $V(z)$ on the linear manifold L, then $V_L(z)$, $z \in L$, is also a continuously differentiable convex quadratic spline function of z in L, more precisely, it is a quadratic function in any local coordinates of L on every convex set $C_\ell \cap L$. We note that

$$V_L(z) \geq 0 , \tag{13}$$

and the equality holds if and only if the system (3') and (4) is consistent and satisfied with $\bar{z} = (\bar{x}^t: \bar{y}^t)^t$ where $\bar{x}$ and $\bar{y}$ are the solutions of the primal and dual problems respectively.

Thus our problem is reduced either to minimize the function $V_L(z)$ or to solve the system of equations

$$\nabla_L V_L(z) = 0 , \tag{14}$$

where $\nabla_L \equiv P_L \nabla$ is a gradient projection operator, P_L is the orthogonal projector, defined by

$$P_L \equiv I - ee^t/\|e\|_2^2 \quad \text{and} \quad e^t \equiv (c^t:-b^t) , \tag{15}$$

and $\nabla \equiv (\partial/\partial x_1, \ldots, \partial/\partial x_n, \partial/\partial y_1, \ldots, \partial/\partial y_m)^t$. Since L is an $m+n-1$ dimensional linear manifolds unconstrained minimization methods coupled with the projection technique of Rosen [43] are applicable to this problem.

There are many available methods for numerical unconstrained minimization. However when we are to deal with a large scale

problem, the methods which require the storage for approximate Hessian matrices are discarded. In the next section, we shall discuss the use of modified Gauss-Newton method coupled with the conjugate gradient algorithm for minimizing the function $V_L(z)$. We shall also discuss briefly in section 4 the application of Fletcher-Reeves method [14, 31] to the function.

3. GAUSS-NEWTON METHOD WITH THE CONJUGATE GRADIENT ALGORITHM

The modified Gauss-Newton method applied to the peacewise quadratic function V_L requires a linearly constrained linear least squares solution at every step of the iteration. The solution of this restricted least squares can conveniently be obtained by applying the conjugate gradient method coupled with Rosen's projection method [43]. Before introducing the algorithm we give some notations used in it.

Given a vector $\tilde{z}$, let a matrix $F(\tilde{z})$ and a vector $d(\tilde{z})$ be defined respectively by

$$F(\tilde{z}) \equiv \begin{bmatrix} (\nabla\phi_{i_1})^t \\ (\nabla\phi_{i_2})^t \\ \vdots \\ (\nabla\phi_{i_k(\tilde{z})})^t \end{bmatrix} \quad \text{and} \quad d(\tilde{z}) \equiv \begin{bmatrix} \phi_{i_1}^{(0)} \\ \phi_{i_2}^{(0)} \\ \vdots \\ \phi_{i_k(\tilde{z})}^{(0)} \end{bmatrix},$$

where $A(\tilde{z}) = \{i_1, i_2, \ldots, i_{k(\tilde{z})}\}$. Then the sum of squares of active constraints for $\tilde{z}$ is expressed as

$$\sum_{i \in A(\tilde{z})} \phi_i^2(z) = \| d(\tilde{z}) - F(\tilde{z}) z \|_2^2 . \tag{16}$$

Hence we have

$$V_L(z) = \| d(z) - F(z) z \|_2^2 = \| d(z) - F(z) P_L z \|_2^2 , \tag{17}$$

(18) $$\nabla_L V_L(z) = 2\sum_{i\in A(z)} \nabla\phi_i\phi_i(z)$$
$$= -2(F(z)P_L)^t(d(z) - F(z)P_L z).$$

Note here that $\nabla_L V_L(z)$ is a linear spline function in z, i.e., it is linear on every convex set $C_\ell \cap L$ and continuous.

The following algorithm minimizes the function $V_L(z)$.

MGNCG ALGORITHM: Let $z_0 \in L$ be an initial approximation to a minimum point of V_L. Generate a sequence $\{z_j\}$ of approximations to the minimum point by the iterative formula

(19) $$z_{j+1} = \rho_j\tilde{z}_{j+1} + (1-\rho_j)z_j, \quad j = 0, 1, \ldots,$$

stopping if $\nabla_L V_L(z_j) = 0$, where the vector $\tilde{z}_{j+1}$ is a solution

(20) $$\tilde{z}_{j+1} \equiv (F(z_j)P_L)^\dagger d(z_j) + (I - (F(z_j)P_L)^\dagger F(z_j)P_L)z_j$$

of the problem of minimizing the sum of linear squares

(21) $$\sum_{i\in A(z_j)} \phi_i^2(z),$$

subject to $z \in L$, that is computed by the CG Algorithm (27) given below and $\rho_j = 1$ if $V_L(\tilde{z}_{j+1}) = 0$, otherwise ρ_j is so chosen as to minimize the continuously differentiable convex quadratic spline function

(22) $$V(\rho\tilde{z}_{j+1} + (1-\rho)z_j)$$

in ρ, which is computed by the linear search algorithm described below.

which corresponds to (25), we obtain the following algorithm which accomplishes the constrained minimization of the sum of linear squares (21) with the constraint $z \in L$ by putting $d = d(z_j)$, $F = F(z_j)$ and $w_0 = z_j$.

CG ALGORITHM: Given an $k \times (m+n)$ matrix F and a k-dimensional column vector d, generate a sequence $\{w_i\}$ of $(m+n)$-dimensional column vectors by the recurrence formula

$$
\begin{aligned}
r_0 &= d - FP_L w_0, \\
p_0 &= P_L F^t r_0, \\
\alpha_i &= \|P_L F^t r_i\|_2^2 / \|FP_L p_i\|_2^2, \\
w_{i+1} &= w_i + \alpha_i p_i, \\
r_{i+1} &= d - FP_L w_{i+1} = r_i - \alpha_i FP_L p_i, \\
\beta_i &= \|P_L F^t r_{i+1}\|_2^2 / \|P_L F^t r_i\|_2^2, \\
p_{i+1} &= P_L F^t r_{i+1} + \beta_i p_i,
\end{aligned}
\tag{27}
$$

starting with w_0 and stopping if p_i vanishes, where r_i and p_i are k- and $(m+n)$-dimensional column vectors respectively and the products of the form $P_L v$ should be computed by the formula

$$
P_L v = v - (e^t v / \|e\|_2^2) e . \tag{28}
$$

Note here that when the matrix $F(z_j)$ is sparse, this algorithm can fully exploit this property.

THEOREM 3.2: The sequence $\{w_i\}$ generated by the above algorithm converges in finite steps to the vector

which corresponds to (25), we obtain the following algorithm which accomplishes the constrained minimization of the sum of linear squares (21) with the constraint $z \in L$ by putting $d = d(z_j)$, $F = F(z_j)$ and $w_0 = z_j$.

CG ALGORITHM: Given an $k \times (m+n)$ matrix F and a k-dimensional column vector d, generate a sequence $\{w_i\}$ of $(m+n)$-dimensional column vectors by the recurrence formula

$$
\begin{aligned}
r_0 &= d - FP_L w_0, \\
p_0 &= P_L F^t r_0, \\
\alpha_i &= \|P_L F^t r_i\|_2^2 / \|FP_L p_i\|_2^2, \\
w_{i+1} &= w_i + \alpha_i p_i, \\
r_{i+1} &= d - FP_L w_{i+1} = r_i - \alpha_i FP_L p_i, \\
\beta_i &= \|P_L F^t r_{i+1}\|_2^2 / \|P_L F^t r_i\|_2^2, \\
p_{i+1} &= P_L F^t r_{i+1} + \beta_i p_i,
\end{aligned}
\tag{27}
$$

starting with w_0 and stopping if p_i vanishes, where r_i and p_i are k- and $(m+n)$-dimensional column vectors respectively and the products of the form $P_L v$ should be computed by the formula

$$
P_L v = v - (e^t v / \|e\|_2^2) e . \tag{28}
$$

Note here that when the matrix $F(z_j)$ is sparse, this algorithm can fully exploit this property.

THEOREM 3.2: The sequence $\{w_i\}$ generated by the above algorithm converges in finite steps to the vector

(29) $$\hat{w} = (FP_L)^{\dagger} d + (I - (FP_L)^{\dagger} FP_L) w_0 ,$$

which minimizes the sum of linear squares

(30) $$\| d - Fw \|_2^2$$

subject to the constraint $w \in L$ when w_0 belongs to L.

Before proving the theorem we prepare the following lemmas.

LEMMA 3.3: Let P be the orthogonal projector onto a subspace T. Then the equality

(31) $$P(FP)^{\dagger} = (FP)^{\dagger}$$

holds for an arbitrary matrix F which can form the product FP. The matrix $G \equiv (FP)^{\dagger}$ satisfies the equalities

(32) $FG = P_{Im\, FP,\, Ker\, PF^t}$ (an orthogonal projector) and

(33) $GF = P_{Im\, PF^t,\, Ker\, PF^t F}$

and F is a generalized inverse of G.

Proof) Since $Im(FP)^{\dagger} = Im(FP)^t = Im\, PF^t \subset Im\, P$, then we have (31). Since $FG = F(FP)^{\dagger} = FP(FP)^{\dagger}$, then we have (32). Since

$$(GF)^2 = (FP)^{\dagger} F (FP)^{\dagger} F = (FP)^{\dagger} FP (FP)^{\dagger} F = (FP)^{\dagger} F = GF,$$

GF is a projector. Since $Im\, GF \subset Im\, G = Im\, PF^t$ and

$$rank\, PF^t = rank\, FP = rank\, FP((FP)^{\dagger} F)P \leq rank\, (FP)^{\dagger} F = rank\, GF ,$$

then we have $Im\, GF = Im\, PF^t$. Since

$$Ker\, GF = Ker\, (FP)^{\dagger} F = Ker\, (PF)^t F = Ker\, PF^t F ,$$

thus we have (33). We have

$$GFG = (FP)^{\dagger}F(FP)^{\dagger} = (FP)^{\dagger}FP(FP)^{\dagger} = (FP)^{\dagger} = G ;$$

which completes the proof.

The following definition and lemma are due to Rao and Mitra [39].

Definition: A matrix G is called a T-restricted least squares (T-rls) inverse of F if for any d, the vector $w \equiv Gd$ is a T-restricted least squares solution which minimizes the function

(34) $$\|d - Fw\|_2^2$$

subject to $w \in T$. Further if the vector Gd gives the minimum $\|\cdot\|_2$-norm solution among the T-rls solutions, G is called a T-restricted minimum norm least squares inverse of F and is denoted by $F_T^{\dagger}$.

LEMMA 3.4: A matrix G is a T-restricted minimum norm least squares inverse of F, iff,

(35) $$Im\, G \subset T ,$$

(36) $$(FG)^t FP = FP \quad \text{and}$$

(37) $$G^t GFP = G^t P .$$

Then we have the following lemma.

LEMMA 3.5: The matrix $G \equiv (FP)^{\dagger}$ defined in Lemma 3.3 is a T-restricted minimum norm least squares inverse of F, i.e.,

(38) $$(FP)^{\dagger} = F_T^{\dagger}.$$

Proof) We have by (31) $Im\, G = Im\, P(FP)^{\dagger} \subset Im\, P = T$. We have by (32), $(FG)^t FP = (FG)FP = FP$. We have $G^t GFP = G^t((FP)^{\dagger}FP) = G^t((FP)^{\dagger}FP)^t = ((FP)^{\dagger}FPG)^t = ((FP)^{\dagger}FP(FP)^{\dagger})^t = ((FP)^{\dagger})^t = G^t$. Thus we have the desired result by Lemma 3.4.

Proof of Theorem 3.2) Noting that d, FP_L, w_i, r_i and p_i in (27) correspond to b, A, x_i, r_i and p_i in Theorem 3.1 respectively, we have by (24) the first half of the theorem. Since $P_L w_0 = w_0$ we have by (31) $F(I - (FP_L)^{\dagger}FP_L)w_0 = F(I - P_L(FP_L)^{\dagger}FP_L)P_L w_0 = (FP_L - FP_L(FP_L)^{\dagger}FP_L)w_0 = 0$ and $(I - (FP_L)^{\dagger}FP_L)w_0 = (I - P_L(FP_L)^{\dagger}FP_L)P_L w_0 = P_L(I - (FP_L)^{\dagger}FP_L)w_0$, hence the second term of the right-hand-side of (29) belongs to the subspace $Ker\, F \cap L$. Thus the last half of the theorem follows directly from Lemma 3.5.

The following lemma is easily seen by Lemma 3.3.

LEMMA 3.6: Let z_0 belong to L, the sequence $\{z_i\}$ generated by the algorithm (19) is contained in the subspace L.

Next we discuss a method for minimizing the function (22) in ρ. There are many available linear search methods to minimize the function, which are described, for example, in [31]. Since it is a continuously differentiable piecewise quadratic convex function in ρ we can give a special algorithm which takes advantage of this property. Before introducing the algorithm we give some results used in it.

Let Δ_j be a vector defined by

(39) $$\Delta_j \equiv \tilde{z}_{j+1} - z_j,$$

where $\tilde{z}_{j+1}$ and z_j are given in MGNCG Algorithm, then

$$\Delta_j = (F(z_j)P_L)^{\dagger}(d(z_j) - F(z_j)P_L z_j). \tag{40}$$

Let $z_j(\rho)$ be a vector defined by

$$z_j(\rho) \equiv \rho\tilde{z}_{j+1} + (1-\rho)z_j = \rho\Delta_j + z_j. \tag{41}$$

LEMMA 3.7: $V_L(z_j(\rho))$ is a continuously differentiable convex quadratic spline function in ρ, whose derivative is given by

$$\begin{aligned}\frac{d}{d\rho}V_L(z_j(\rho)) &= \langle \nabla_L V_L(z_j(\rho)), \Delta_j\rangle \\ &= 2(\rho\|F(z_j(\rho))P_L\Delta_j\|_2^2 \\ &\quad - \langle d(z_j(\rho)) - F(z_j(\rho))P_L z_j, F(z_j(\rho))P_L\Delta_j\rangle \\ &= 2(\rho\|F(z_j(\rho))\Delta_j\|_2^2 \\ &\quad - \langle d(z_j(\rho)) - F(z_j(\rho))z_j, F(z_j(\rho))\Delta_j\rangle,\end{aligned} \tag{42}$$

which is a monotone increasing, continuous and piecewise linear function in ρ.

Proof) Since $z_j(\rho)$ is a linear mapping from the non-negative axis to R^{m+n} and $V_L(z)$ is a continuously differentiable convex quadratic spline function in z, so is $V_L(z_j(\rho))$ in ρ. Hence its derivative in ρ is a continuous and piecewise linear function with finite number of knots in ρ and if ρ is not a knot point, then

$$\frac{d^2}{d\rho^2}V_L(z_j(\rho)) \geq 0. \tag{43}$$

which implies the derivative (42) is a monotone increasing function in ρ. We have

$$\frac{d}{d\rho}V_L(z_j(\rho)) = \langle \nabla_L V_L(z_j(\rho)), \frac{d}{d\rho}z_j(\rho)\rangle$$

$$= \langle \nabla_L V_L(z_j(\rho)), \Delta_j\rangle$$

(44)

$$= 2\langle -(F(z_j(\rho))P_L)^t(d(z_j(\rho)) - F(z_j(\rho))P_L z_j(\rho)), \Delta_j\rangle$$

$$= 2\langle -(d(z_j(\rho)) - F(z_j(\rho))P_L(\rho\Delta_j + z_j), F(z_j(\rho))P_L\Delta_j\rangle.$$

Since $P_L z_j = z_j$ and $P_L\Delta_j = \Delta_j$ (42) follows from (44). The monotone behaviour of the derivative (42) follows also from its form (42) since $\|F(z_j(\rho))\Delta_j\|_2^2 \geq 0$, where if the equality holds then $\langle d(z_j(\rho)) - F(z_j(\rho))z_j, F(z_j(\rho))\Delta_j\rangle = 0$ hence $dV_L(z_j(\rho))/d\rho = 0$. This completes the proof.

LEMMA 3.8: If $A(z_j(\rho)) = A(z_j)$ and $\rho < 1$ then

$$\frac{d}{d\rho}V_L(z_j(\rho)) = 2\|F(z_j)\Delta_j\|_2^2(\rho - 1) \leq 0, \tag{45}$$

where the equality holds if and only if $\nabla_L V_L(z_j) = 0$ in which case $\Delta_j = 0$.

Proof) In the proofs of this and subsequent lemmas $d(z_j)$, $F(z_j)$ and P_L will be abbreviated as d, F and P respectively. We have by (18), (41) and the assumption

$$-\frac{1}{2}\langle \nabla_L V_L(z_j(\rho)), \Delta_j\rangle = \langle d - FPz_j, FP\Delta_j\rangle - \rho\langle FP\Delta_j, FP\Delta_j\rangle$$

$$= ((d - FPz_j)^t FP(FP)^\dagger(d - FPz_j)$$

$$-\rho\langle FP\Delta_j, FP\Delta_j\rangle)$$

$$= \|(FP)(FP)^\dagger(d - FPz_j)\|_2^2(1 - \rho) \geq 0.$$

Hence the equality holds, iff,

$$(FP)^t(d - FPz_j) = 0 \quad \text{or} \quad (FP)^\dagger(d - FPz_j) = 0$$

which mean $\nabla_L V_L(z_j) = 0$ and $\Delta_j = 0$ respectively. This completes the proof.

Since the minimum of the convex function $V_L(z_j(\rho))$ is given by the solution of the following equation

$$W_j(\rho) = \frac{1}{2}\frac{d}{d\rho} V_L(z_j(\rho)) = 0 , \tag{46}$$

and the condition of Lemma 3.8 is satisfied for $\rho = 0$, it follows from Lemma 3.7 that there exists a positive number ρ which minimizes the function (22) unless $\nabla_L V_L(z_j) = 0$ and that the sequence $\{V_L(z_i)\}$ is a monotone decreasing function of i, i.e.,

$$V_L(z_1) > V_L(z_2) > V_L(z_3) \quad .....$$

- -

Fig. 1

- -

Now we give an algorithm for solving the equation (46), which takes advantage of the properties of the function $W_j(\rho)$ given in the preceding lemmas. We will use the following notations,

$$\sigma(\rho) \equiv \|F(z_j(\rho))\Delta_j\|_2^2$$

(47) $$\tau(\rho) \equiv \langle d(z_j(\rho)) - F(z_j(\rho))z_j, F(z_j(\rho))\Delta_j \rangle$$

and $$W(\rho) \equiv W_j(\rho) = \rho\sigma(\rho) - \tau(\rho) .$$

LINEAR SEARCH ALGORITHM: Given vectors z_j and $\tilde{z}_{j+1}$, compute a value $\rho = \rho_j$ which minimizes the function (22) by the following iterative linear search algorithm. It generates shrinking intervals (γ_i, Γ_i) which contain ρ_j and detects ρ_j within finite number of steps. See Fig. 2.

1. $\gamma_0 := 0$, (viz., ρ_j is contained in the interval $(0, \infty)$.)
   $\tilde{\gamma} := 1.$
   $i := 0.$

2. Evaluate the values $\sigma(\tilde{\gamma})$ and $\tau(\tilde{\gamma})$.
   $W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$.

3. If $W(\tilde{\gamma}) = 0$, (viz., $\rho = \tilde{\gamma}$ gives the minimum of the function (22),) then go to 11, otherwise if $W(\tilde{\gamma}) > 0$, (viz., ρ_j is contained in the interval $(\gamma_0, \tilde{\gamma})$,) then
   $\Gamma_0 := \tilde{\gamma}$ and go to 5,
   otherwise, (viz., ρ_j is contained in the interval $(\tilde{\gamma}, \infty)$,)
   $\gamma_0 := \tilde{\gamma}$.

4. $\tilde{\gamma} := \tau(\tilde{\gamma})/\sigma(\tilde{\gamma})$, (viz., compute the intersection of the ρ-axis and the extension of the line segments of the graph $\{(\rho, W(\rho)) : \rho \in R\}$, that contains the point $(\tilde{\gamma}, W(\tilde{\gamma}))$,)
   Go to 2.

5. $i := i+1$.

$\gamma_i := \gamma_{i-1}$.

$\Gamma_i := \Gamma_{i-1}$.

$\tilde{\gamma} := (\gamma_i W(\Gamma_i) - \Gamma_i W(\gamma_i))/(W(\Gamma_i) - W(\gamma_i))$, (viz., compute the point which divides the interval (γ_i, Γ_i) into two parts with the lengths of the ratio, $-W(\gamma_i) : W(\Gamma_i)$,)

Evaluate the values $\sigma(\tilde{\gamma})$ and $\tau(\tilde{\gamma})$.

$W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$.

6. If $W(\tilde{\gamma}) = 0$ then go to 11, otherwise if $W(\tilde{\gamma}) > 0$, (viz., ρ is contained in the interval $(\gamma_i, \tilde{\gamma})$,) then

$\Gamma_i := \tilde{\gamma}$ and go to 6,

otherwise (viz., ρ is contained in $(\tilde{\gamma}, \Gamma_i)$,)

$\gamma_i := \tilde{\gamma}$.

7. $\tilde{\gamma} := \tau(\tilde{\gamma})/\sigma(\tilde{\gamma})$.

If $\tilde{\gamma} \leq \gamma_i$ or $\tilde{\gamma} \geq \Gamma_i$ then go to 9, otherwise (viz., $\tilde{\gamma}$ is contained in (γ_i, Γ_i),) evaluates the values $\sigma(\tilde{\gamma})$ and $\tau(\tilde{\gamma})$.

$W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$.

8. If $W(\tilde{\gamma}) = 0$ then go to 11, otherwise if $W(\tilde{\gamma}) > 0$, then

$\Gamma_i := \tilde{\gamma}$ and go to 9,

otherwise

$\gamma_i := \tilde{\gamma}$.

9. $\tilde{\gamma} := (\gamma_i + \Gamma_i)/2$.

Evaluate the values $\sigma(\tilde{\gamma})$ and $\tau(\tilde{\gamma})$.

$W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$.

10. If $W(\tilde{\gamma}) = 0$ then go to 11, otherwise if $W(\tilde{\gamma}) > 0$ then
$\Gamma_i := \tilde{\gamma}$ and go to 5,
otherwise
$\gamma_i := \tilde{\gamma}$ and go to 5.

11. $\rho_j := \tilde{\gamma}$.
Stop.

Fig. 2

THEOREM 3.9: Linear Search Algorithm generates ρ_j which is a solution of the equation (46) in finite number of steps.

Proof) It follows from Lemmas 3.7 and 3.8 that the statements $1 \sim 4$ generate either ρ_j or an interval (γ_0, Γ_0) which contains ρ_j in finite number of steps. Suppose the algorithm does not yield a solution ρ_j in finite number of steps, it generates infinite number of intervals (γ_i, Γ_i) which contains ρ_j and satisfies

$$\Gamma_{i+1} - \gamma_{i+1} < (\Gamma_i - \gamma_i)/2, \quad \text{for} \quad i = 0, 1, \ldots$$

Then for a sufficiently large value of i the interval (γ_i, Γ_i) contains at most one knot point of the piecewise linear function $W_j(\rho)$. Hence at the next $(i+1)$-th step the statement 7 or 5 generates a value γ which satisfies $W_j(\tilde{\gamma}) = 0$, according as the interval (γ_i, Γ_i) contains a knot point or not, which contradicts the hypothesis.

Before proving the convergence of the MGNCG Algorithm we prepare the following lemmas.

LEMMA 3.10: If the algorithm (19) does not terminate within finite steps, then

$$\lim_{j\to\infty} W_j(0)/\|\Delta_j\|_2 = 0. \tag{48}$$

Proof) Suppose not, then there exist a subsequence $\{j_1, j_2, \ldots\}$ of indices and a positive number ε such that

$$W_{j_k}(0)/\|\Delta_{j_k}\|_2 < -2\varepsilon, \text{ for any } k \tag{49}$$

since by Lemma 3.8 we have $W_j(0) < 0$. Because $V_L(z)$ is a continuously differentiable non-negative convex quadratic spline function, $\nabla_L V_L(z)$ is uniformly continuous on a convex level set defined by

$$L(z_0) \equiv \{z \in L: V_L(z) \leq V_L(z_0)\}, \tag{50}$$

which contains the sequence $\{z_j\}$. Hence, there exists a positive number δ such that

$$\begin{aligned} &\|z_{j_k} - z_{j_k}(\rho)\|_2 < \delta \quad \text{implies} \\ &\|\nabla_L V_L(z_{j_k}) - \nabla_L V_L(z_{j_k}(\rho))\|_2 < \varepsilon \end{aligned} \tag{51}$$

for any k. Furthermore we have

$$\|z_{j_k} - z_{j_k}(\rho)\|_2 = \rho\|\Delta_{j_k}\|_2 \quad \text{and} \tag{52}$$

$$|W_{j_k}(0) - W_{j_k}(\rho)| = |\langle \nabla_L V_L(z_{j_k}) - \nabla_L V_L(z_{j_k}(\rho)), \Delta_{j_k}\rangle|$$

(53)
$$\leq \varepsilon \|\Delta_{j_k}\|_2 ,$$

so that if $0 < \rho < \delta/\|\Delta_{j_k}\|_2$ then

$$W_{j_k}(\rho) < W_{j_k}(0) + |W_{j_k}(0) - W_{j_k}(\rho)|$$

$$< -2\varepsilon\|\Delta_{j_k}\|_2 + \varepsilon\|\Delta_{j_k}\|_2 = -\varepsilon\|\Delta_{j_k}\|_2 ,$$

for any $k = 1, 2, 3, \ldots \infty$. Hence, for $\tilde{\rho} = \delta/\|\Delta_{j_k}\|_2$, we have

$$V_L(z_{j_k}(\tilde{\rho})) - V_L(z_{j_k}) = 2\int_0^{\tilde{\rho}} W_{j_k}(\rho)\, d\rho$$

(54)
$$< 2\int_0^{\tilde{\rho}} -\varepsilon\|\Delta_{j_k}\|_2 d\rho = -2\,\varepsilon\,\delta .$$

This contradicts the fact that $V_L(z) \geq 0$, since $V_L(z_{j_k+1}) \leq V_L(z_{j_k}(\tilde{\rho}))$.

LEMMA 3.11: The condition (48) implies

(55)
$$\lim_{j\to\infty} \nabla_L V_L(z_j) = 0 .$$

Proof) Since

$$\Delta_j = (F(z_j))^{\dagger} F(z_j)(F(z_j))^{\dagger}(d(z_j) - F(z_j)z_j)$$

$$= (F(z_j))^{\dagger} F(z_j)\Delta_j ,$$

we have by Lemma 3.8

$$
\begin{aligned}
-W_j(0)/\|\Delta_j\|_2 &= \|F(z_j)\Delta_j\|_2^2 / \|\Delta_j\|_2 \\
&= \|F(z_j)(F(z_j))^{\dagger}(d(z_j) - F(z_j)z_j)\|_2^2 / \\
&\quad \|\Delta_j\|_2 \\
&\geq \|F(z_j)(F(z_j))^{\dagger}(d(z_j) - F(z_j)z_j)\|_2 / \\
&\quad \|(F(z_j))^{\dagger}\|_2 \\
&\geq \|F(z_j)(F(z_j))^{\dagger}(d(z_j) - F(z_j)z_j)\|_2 / \\
&\quad (\max_z \|(F(z))^{\dagger}\|_2).
\end{aligned}
$$

The denominator of the last formula is finite. Hence, if the condition (48) is satisfied then the numerator converges to zero as j tends to infinity, which implies that

$$\nabla_L V_L(z_j) = (F(z_j))^t(d(z_j) - F(z_j)z_j)$$

converges to zero vector.

LEMMA 3.12: If $\nabla_L V_L(\tilde{z}_{j+1}) \neq 0$ then $A(\tilde{z}_{j+1}) \neq A(z_j)$, where $\tilde{z}_{j+1}$ and z_j are defined in MGNCG Algorithm.

Proof) Suppose $A(\tilde{z}_{j+1}) = A(z_j)$ then we have

$$
\begin{aligned}
\frac{1}{2}\nabla_L \dot{V}_L(\tilde{z}_{j+1}) &= (F(\tilde{z}_{j+1})P)^t(d(\tilde{z}_{j+1}) - F(\tilde{z}_{j+1})P\tilde{z}_{j+1}) \\
&= (FP)^t(d - FP\tilde{z}_{j+1}) = 0,
\end{aligned}
$$

which completes the proof.

LEMMA 3.13: If $\nabla_L V_L(z_j) \neq 0$ and $\nabla_L V_L(z_{j+1}) \neq 0$ then

$$A(z_j) \neq A(z_{j+1}),$$

where z_j and z_{j+1} are defined in MGNCG Algorithm.

Proof) Suppose $A(z_j) = A(z_{j+1})$ then we have

$$\begin{aligned}\frac{1}{2}\frac{d}{d\rho}V_L(z_{j+1}) &= \langle \nabla_L V_L(z_{j+1}), \Delta_j \rangle \\ &= -\langle (F(z_{j+1})P)^t (d(z_{j+1}) - F(z_{j+1})Pz_{j+1}), \Delta_j \rangle \\ &= -\langle (FP)^t (d - FP(\rho_j \Delta_j + z_j)), \Delta_j \rangle \\ &= \| (FP)(FP)^\dagger (d - FPz_j) \|_2^2 (\rho_j - 1).\end{aligned}$$

Since $\rho = \rho_j$ minimizes the differentiable function (22) in ρ,

$$\frac{d}{d\rho}V_L(z_{j+1}) = 0 ,$$

hence $\nabla_L V_L(z_j) = (FP)^t (FP)(FP)^\dagger (d - FPz_j) = 0$ or $\rho_j = 1$. Here $\rho_j = 1$ implies $z_{j+1} = \tilde{z}_{j+1}$, hence by Lemma 3.12 $\nabla_L V_L(z_{j+1}) = 0$. This completes the proof.

Now we prove the convergence of the MGNCG Algorithm.

THEOREM 3.14: If either of the problems (1) or (2) has an optimal solution, i.e., $S \neq \Phi$ then the sequence $\{z_j\}$ generated by the MGNCG Algorithm (19) converges in finite number of steps to a solution $\bar{z} \in S$.

Proof) If $\hat{z} \in \bar{C}_\ell \cap S$ then for any $k \in A(C_\ell)$,

$$\phi_k(\hat{z}) \geq 0 \quad \text{and} \quad \phi_k(z) \leq 0 \quad \text{for} \quad z \in C_\ell ,$$

hence

$$\phi_k(\hat{z}) = 0 \text{ for } k \in A(C_\ell) .$$

This implies that if $\bar{C}_\ell \cap S \neq \Phi$ then the linear system

$$d(z') - F(z')z = 0 , \tag{56}$$

is solvable in z for any $z' \in C_\ell$. Let U' be a union of the sets $C_\ell \cap L$ which touch the set S, i.e.,

$$U' \equiv \bigcup_{\ell=1}^{N} \{C_\ell \cap L: \bar{C}_\ell \cap S \neq \Phi\} \neq \Phi$$

Then there exists a neighbourhood U of S such that

$$U \equiv \{z \in L: V_L(z) < \delta\} \subset U' \subset L \tag{57}$$

hence the system (56) is solvable in z for any $z' \in U$. It follows from Lemma 3.11 that there exists an integer j_0 such that $z_j \in U$ for $j \geq j_0$. Then for $j \geq j_0$ the linear equation

$$d(z_j) - F(z_j)z = 0 \tag{58}$$

is solvable in z. Hence the vector $\tilde{z}_{j+1}$ is a solution of this equation, i.e., for $j \geq j_0$,

$$\phi_i(\tilde{z}_{j+1}) = 0 \quad \text{for} \quad i \in A(z_j)$$

which implies

$$A(z_j) \subset A(\tilde{z}_{j+1}) \tag{59}$$

thus

$$A(z_j) \subset A(z_j(\rho)) \subset A(\tilde{z}_{j+1}) \quad \text{for} \quad \rho \in [0, 1]. \tag{60}$$

Since $dW_j(\rho)/d\rho = \|F(z_j(\rho))\Delta_j\|_2^2 = \sum_{i \in A(z_j(\rho))} ((\nabla\phi_i)^t \Delta_j)^2$ $\geq \sum_{i \in A(z_j)} ((\nabla\phi_i)^t \Delta_j)^2 = \|F(z_j)\Delta_j\|_2^2 = dW_j(0)/d\rho$, for $\rho \in [0, 1]$ the zero ρ_j of $W_j(\rho)$ is contained in the closed interval $[0, 1]$. See Fig. 1. Hence

(61) $$A(z_j) \subset A(z_j(\rho_j)) = A(z_{j+1}) .$$

Suppose the algorithm (19) does not converge in finite number of steps, then by Lemma 3.13 and (61),

$$\#A(z_j) < \#A(z_{j+1}) \text{ for } j \geq j_0 ,$$

where $\#A(z)$ is the number of the elements of $A(z)$. Hence

$$\lim_{j\to\infty} \#A(z_j) = \infty ,$$

which contradicts the fact that the number of the constraints is finite.

LEMMA 3.15: The set C_V of all the critical points of the function $V_L(z)$, defined by

$$C_V \equiv \{z \in L: \nabla_L V_L(z) = 0\} .$$

is a (possibly unbounded) closed convex polyhedral convex set on which the function $V_L(z)$ takes the minimum value $V_L(C_V)$. When the system (3) - (4) is consistent then $S = C_V$.

Proof) The result is easily deduced from the form (18) of $\nabla_L V_L(z)$, since $V_L(z)$ is a continuous convex function and

$$C_V = \bigcup_{\ell=1}^{N} (C_\ell \cap \{z \in L: (F(z'))^t (d(z') - F(z')z) = 0 \text{ and } z' \in C_\ell\})$$

LEMMA 3.16: If $\phi_k(z') < 0$ for some $z' \in C_V$ then the function $\phi_k(z)$ is constant on C_V, i.e., $\phi_k(z) \equiv \phi_k(z') < 0$, for $z \in C_V$.

Proof) If C_V consists of a single point the result is trivial. Suppose there exists a critical points $z \in C_V$ which is different from z'. Let vectors Δ and $z(\rho)$ be defined respectively by

$$z(\rho) \equiv \rho\Delta + z' \quad \text{and} \quad \Delta \equiv z - z'.$$

Since C_V is a closed convex set, $z(\rho)$ is contained in C_V for $\rho \in [0, 1]$ hence $V_L(z(\rho))$ is constant for $\rho \in [0, 1]$. Thus

$$\begin{aligned} dV_L(z(\rho))/d\rho &= \rho\Big(\sum_{i\in A(z(\rho))} (\langle\nabla\phi_i, \Delta\rangle)^2\Big) \\ &\quad + \sum_{i\in A(z(\rho))} (\langle\nabla\phi_i, \Delta\rangle\phi_i(z')) \qquad (62) \\ &= 0 \quad \text{for } \rho \in (0, 1) . \end{aligned}$$

There exists a positive number δ such that

$$\phi_k(z(\rho)) < 0 \quad \text{for all} \quad \rho \in (0, \delta) ,$$

hence $k \in A(z(\rho))$ for $\rho \in (0, \delta)$. Therefore, noting the equality (62), we have

$$(\langle\nabla\phi_k, \Delta\rangle)^2 \leqq \sum_{i\in A(z(\rho))} (\langle\nabla\phi_i, \Delta\rangle)^2 = 0 ,$$

hence $\langle\nabla\phi_k, \Delta\rangle = 0$, which implies $\phi_k(z(\rho))$ is constant for all $\rho \in [0, 1]$. This completes the proof.

LEMMA 3.17: There exists an integer $\hat{\ell}$ such that $C_V \subset \bar{C}_{\hat{\ell}}$.

Proof) Suppose we assume contrary, there exist two different vectors z, $z' \in C_V$ and ϕ_k such that

$$\phi_k(z) > 0 > \phi_k(z')$$

since C_V is a closed set. This contradicts the previous lemma.

THEOREM 3.18: The sequence $\{z_j\}$ generated by the MGNCG Algorithm converges in finite number of steps to a critical point z which minimizes the function $V_L(z)$.

Proof) Since the result has been already proved in the case where $S \neq \Phi$, we need to prove only the case where $S = \Phi$. Let a finite set A^δ be defined by

$$A^\delta \equiv \{k: \phi_k(z) < 0 \quad \text{for} \quad z \in C_V\} ,$$

which is well defined by Lemma 3.16. Suppose $\hat{z} \in \overline{C_\ell} \cap C_V$ then for $k \in A(C_\ell)$ we have

$$\phi_k(z) \leqq 0 \quad \text{for} \quad z \in C_\ell ,$$

hence, from the continuity of ϕ_k, $\phi_k(\hat{z}) \leqq 0$, which implies

$$A(C_\ell) \subset A(\hat{z}) . \tag{63}$$

Further we have

$$\phi_k(\hat{z}) < 0 \quad \text{for} \quad k \in A^\delta$$

hence there exists $z \in C_\ell$ such that

$$\phi_k(z) < 0 \quad \text{for} \quad k \in A^\delta ,$$

which implies

$$A^\delta \subset A(C_\ell) . \tag{64}$$

It follows from the relations of (63) and (64) that the system of equations

$$\phi_k(z) = 0, \quad k \in A(C_\ell) - A^\delta$$

$$\sum_{k \in A^\delta} \phi_k(z) \nabla \phi_k = 0,$$

is solvable, since $\nabla_L V_L(\hat{z}) \equiv \sum_{k \in A(\hat{z})} \phi_k(\hat{z}) \nabla \phi_k = 0$.

Let U' be a union of the sets $C_\ell \cap L$ which touch the set C_V, i.e.,

$$U' = \bigcup_{\ell=1}^{N} \{C_\ell \cap L: \bar{C}_\ell \cap L \cap C_V \neq \Phi\}.$$

Then there exists a neighbourhood U of C_V such that

$$U = \{z \in L: V_L(z) < V_L(C_V) + \delta\} \subset U' \subset L,$$

hence the system of equations

$$\phi_k(z) = 0, \quad k \in A(z') - A^\delta,$$

$$\sum_{k \in A^\delta} \phi_k(z) \nabla \phi_k = 0,$$

is solvable in z for any $z' \in U$. It follows from Lemma 3.11 that there exists an integer j_0 such that $z_j \in U$ for $j \geq j_0$. Thus for any $j \geq j_0$, the vector $\tilde{z}_{j+1}$ is a solution of the system if z' is replaced by z_j. This implies that

$$A(z_j) \subset A(\tilde{z}_{j+1}).$$

Thus by the same argument as is given in the proof of the previous theorem, we have the desired result.

4. CONSIDERATIONS FOR IMPLEMENTATION

In this section several remarks are made on practical techniques for implementing our method.

A. STARTING APPROXIMATION

As may be seen from the preceding discussion, our method can use profitably any (possibly infeasible) approximation to the optimal solution. We suggest some tricks for incorporating these a priori information.

1) If we have approximations $\tilde{x}$ and $\tilde{y}$ to the optimal solutions of the primal and dual problems, then we should start our algorithm (19) with

$$z_0 = P_L\begin{pmatrix}\tilde{x}\\ \tilde{y}\end{pmatrix}. \tag{65}$$

2) If we have an approximation to one of the primal and dual problems, we should start it with

$$z_0 = \begin{pmatrix}\tilde{x}\\ (c^t\tilde{x}/b^tb)b\end{pmatrix} \quad \text{or} \quad \begin{pmatrix}(b^t\tilde{y}/c^tc)c\\ \tilde{y}\end{pmatrix}$$

accordingly we have $\tilde{x}$ or $\tilde{y}$.

3) If we have no a priori information about the optimal solutions, we should either start it with $z_0 = 0$ or try to find a better approximation by the following algorithm;

FR ALGORITHM: Starting with z_0 generate a sequence $\{z_i\}$ of vectors by the iterative formula,

$$r_0 = \nabla_L V_L(z_0),$$

$$p_0 = -r_0,$$

$$z_{i+1} = z_i + \rho_i p_i,$$

(where ρ_i is determined to minimize the function $V_L(z_i + \rho p_i)$ by the similar algorithm to Linear Search Algorithm)

$$r_{i+1} = \nabla_L V_L(z_{i+1}),$$

$$\beta_i = \|r_{i+1}\|_2^2 / \|r_i\|_2^2$$

$$p_{i+1} = -r_i + \beta_i p_i,$$

which is obtained by applying the Fletcher-Reeves method [14] to the function $V_L(z)$.

B. NUMERICAL ASSIGNMENT OF ACTIVE CONSTRAINTS

Since the finite termination property of our method is achieved by the special definition (8) of active constraints, care should be taken to protect against the incorrect assignment of active constraints at each step of the iteration (19). Hence in determining active constraints they should be tested against some 'tolerance', more explicitly, the following definition

$$A(z) \equiv \{k: \phi_k(z) < \varepsilon \max_i(|\pi_i \zeta_i|, |\pi_0|)\} \tag{67}$$

should be used instead of (8) where $z \equiv (\zeta_i)$,

$$\phi_k(z) \equiv \sum_{i=1}^{m+n} \pi_i \zeta_i + \pi_0$$

and ε is the maximum value such that machine computation of the logical expression $1 + \varepsilon = 1$ is true. It is safer to choose a larger number for ε than a smaller one.

C. SCALING OF MATRIX *A*

In order to use the algorithms effectively, we should equilibrate the matrix *A* before the algorithms are applied.

5. APPLICATION

In this section, it is pointed out briefly that a system of linear equalities/inequalities can be solved in the similar way to the method given in the preceding sections.

Given a system, transform it into a system of linear inequalities

$$\psi_i(z) \geqq 0, \quad 1 \leqq i \leqq k,$$

by converting every equality constraints to a pair of inequality constraints, where $\psi_i(z)$ is a linear function in z. Then form the function

$$V(z) \equiv \sum_{i=1}^{k} (\psi_i(z))^2$$

and apply the modified Gauss-Newton method coupled with the conjugate gradient method to the function in the same way as in section 3. Thus we obtain a finitely terminating iterative algorithm for solving a system of linear equalities/inequalities.

REFERENCES

[1] Agmon, S., The Relaxation Method for Linear Inequalities, Canadian Journal of Mathematics, Vol. 6, pp. 382-392, 1954.

[2] Bartels, R. H. and Golub, G. H., The Simplex Method of Linear Programming using LU Decomposition, Communications of the Association for Computing Machinery, Vol. 12, pp. 266-288, 1969.

[3] Bartels, R. H., A Stabilization of the Simplex Method, Numerische Mathematik, Vol. 16, pp. 414-434, 1971.

[4] Brayton, R. K., Gustavson F. G. and Willoughby, R. A., Some Results on Sparse Matrices, RC-2332, IBM Research Centre, Yorktown Heights, N.Y., 1969.

[5] Cline R. E. and Pyle, L. D., The Generalized Inverse in Linear Programming - An Intersection Projection Method and the Solution of a Class of Structured Linear Programming Problems, SIAM Journal on Applied Mathematics, Vol. 24, pp. 338-351, 1973.

[6] Cutler, L and Wolfe, P., Experiments in Linear Programming. Recent Advances in Mathematical Programming, Graves, R. L. and Wolfe, P., eds., McGraw-Hill, pp. 177-200, 1963.

[7] Dantzig, G. B. and Orchard-Hays, W., The Product Form of the Inverse in the Simplex Method. Mathematical Tables and Aids to Computation, Vol. 8, pp. 64-74, 1954.

[8] Dantzig, G. B. and Wolfe, P., Decomposition Principle for Linear Programs. Operations Research, Vol. 8, pp. 101-111, 1960.

[9] Dantzig, G. B., Linear Programming and Extensions, Princeton, New Jersey: Princeton University Press, 1963.

[10] Dantzig, G. B., Compact Basis Triangularization for the Simplex Method. Recent Advances in Mathematical Programming, Graves, R. L. and Wolfe, P., eds., pp. 125-132, 1963.

[11] Dantzig, G. B., Large Scale Linear Programming. Mathematics of the Decision Sciences, Dantzig, G. B. and Veinott, A. F., Jr., eds., American Mathematical Society Providence, Rhode Island, pp. 77-92, 1968.

[12] de Buchet, J., How to Take into Account the Low Density of Matrices to Design a Mathematical Programming Package. Relevant Effects on Optimisation and Inversion Algorithms, Large Sparse Set of Linear Equations, Reid, J. K., ed., Academic Press, London, pp. 211-217, 1971.

[13] Eaves, C., Piecewise Linear Retractions by Reflexion, Linear Algebra and Its Applications, Vol. 7, pp. 93-98, 1973.

[14] Fletcher, R. and Reeves, C. M., Function Minimization by Conjugate Gradients, Computer Journal, Vol. 7, pp. 149-1964.

[15] Forrest, J. J. H. and Tomlin, J. A., Updated Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method, Mathematical Programming, Vol. 2, pp. 263-278, 1972.

[16] Geoffrion, A. M., Elements of Large-Scale Mathematical Programming, Management Science, pp. 652-691, 1970.

[17] Gill, P. E. and Murray, W., A Numerically Stable Form of the Simplex Algorithm, Linear Algebra and Its Applications, Vol. 7, pp. 99-138, 1973.

[18] Goldstein, A. A. and Cheney, W., A Finite Algorithm for the Solution of Consistent Linear Equations and Inequalities and for the Tchebycheff Approximation of Inconsistent Linear Equations, Pacific Journal of Mathematics, pp. 415-427, 1958.

[19] Gould, F. J., Proximate Linear Programming: A Variable Extreme Point Method. Mathematical Programming, Vol. 3, pp. 326-338, 1972.

[20] Grinold, R. C., Steepest Ascent for Large Scale Linear Programs, SIAM Review, Vol. 14, pp. 447-464, 1972.

[21] Held, M. and Karp, R. M., The Traveling-Salesman Problem and Minimum Spanning Trees: Part II, Mathematical Programming, Vol. 1, pp. 6-25, 1971.

[22] Held, M., Wolfe, P. and Crowder, H. P., Validation of Subgradient Optimization. Mathematical Programming, Vol. 6, pp. 62-88, 1974.

[23] Hestenes, M. R. and Stiefel, E., Method of Conjugate Gradients for Solving Linear Systems. Journal of Research of National Bureau of Standard, Vol. 49, pp. 409-436, 1952.

[24] Ho, Yu-Chi and Kashyap, R. L., A Class of Iterative Procedures for Linear Inequalities, SIAM Journal on Control, Vol. 4, pp. 112-115, 1966.

[25] Ho, Yu-Chi and Kashyap, R. L., An Algorithm for Linear Inequalities and Its Applications, IEEE Transactions on Electronics and Computers, EC-14, pp. 683-688, 1965.

[26] Jeroslow, R. G., The Simplex Algorithm with the Pivot Rule of Maximizing Criterion Improvement, Discrete Mathematics, Vol. 4, pp. 367-377, 1973.

[27] Kaczmarz, S., Angenäherte Auflösung von Systemen Linearer Gleichungen, Bulletin International de l'Academie Polonaise des Sciences et des Lettres. Classe des Sciences Mathematiques et Naturelles. Serie A. Sciences Mathematiques, Vol. A16-17, pp. 355-357, 1937.

[28] Kammerer, W. J. and Nashed, M. Z., On the Convergence of the Conjugate Gradient Method for Singular Linear Operator Equations, SIAM Journal on Numerical Analysis, Vol. 9, pp. 165-181, 1972.

[29] Klee, V., A Class of Linear Programming Problems requiring a Large Number of Iterations, Numerische Mathematik, Vol. 7, pp. 313-321, 1965.

[30] Klee, V. and Minty, G. J., How Good is the Simplex Algorithm?, Inequalities III, Shisha, O., ed., Academic Press, New York, 1971.

[31] Kowalik, J. and Osborne, M. R., Methods for Unconstrained Optimization Problems, American Elsevier Publishing Company, New York, 1968.

[32] Kunzi, H. P. and Tzschach, H., The Douplex-Algorithm, Numerische Mathematik, Vol. 7, pp. 222-225, 1965.

[33] Markowitz, H. M., The Elimination Form of the Inverse and Its Application to Linear Programming, Management Science, Vol. 3, pp. 255-269, 1957.

[34] Motzkin, T. and Schoenberg, I. J., The Relaxation Method for Linear Inequalities, Canadian Journal of Mathematics, Vol. 6, pp. 393-404, 1954.

[35] Nagaraja, G. and Krishna, G., An Algorithm for the Solution of Linear Inequalities, IEEE Transactions on Computers, Vol. C-23, pp. 421-427, 1974.

[36] Oettli, W., An Iterative Method, having Linear Rate of Convergence, for Solving a Pair of Dual Linear Programs, Mathematical Programming, Vol. 3, pp. 302-311, 1972.

[37] Orden, A., On the Solution of Linear Equation Inequality Systems, Mathematical Programming, Vol. 1, pp. 137-152, 1971.

[38] Pyle, L. D., A Generalized Inverse ε-Algorithm for Constructing Intersection Projection Matrices, with Applications, Numerische Mathematik, Vol. 10, pp. 86-102, 1967.

[39] Rao, C. R. and Mitra, S. K., Theory and Application of Constrained Inverse of Matrices, SIAM Journal on Applied Mathematics, Vol. 24, No. 4, pp. 473-488, 1973.

[40] Reid, J. K., On the Method of Conjugate Gradients for the Solution of Large Sparse Systems of Linear Equations, Proceedings of the Conference on "Large Sparse Set of Linear Equations", Academic Press, New York, 1971.

[41] Reid, J. K., The Use of Conjugate Gradients for Systems of Linear Equations Possessing "Property A", SIAM Journal on Numerical Analysis, Vol. 9, pp. 325-332, 1972.

[42] Rosen, J. B., Primal Partition Programming for Block Diagonal Matrices, Numerische Mathematik, Vol. 6, pp. 250-260, 1964.

[43] Rosen, J. B., The Gradient Projection Method for Non-Linear Programming: Part I. Linear Constraints, SIAM Journal on Applied Mathematics, Vol. 8, pp. 181-217, 1960.

[44] Smith, D. M. and Orchard-Hays, W., Computational Efficiency in Product Form LP Codes. Recent Advances in Mathematical Programming, pp. 211-215, 1963.

[45] Takahashi, I., Variable Separation Principle for Mathematical Programming. Journal of the Operations Research Society of Japan, Vol. 6, pp. 82-105, 1964.

[46] Tanabe, K., Projection Method for Solving a Singular System of Linear Equations and Its Applications, Numerische Mathematik, Vol. 17, pp. 203-214, 1971.

[47] Whitney, T. M. and Meany, R. K., Two Algorithms related to the Method of Steepest Descent, SIAM Journal on Numerical Analysis, Vol. 4, pp. 109-118, 1967.

[48] Willoughby, R. A., ed., Proceedings of a Symposium on Sparse Matrices, IBM Watson Research Center, Yorktown Heights, New York, 1969.

[49] Wolfe, P., Error in the Solution of Linear Programming Problems, Error in Digital Computation, Vol. 2, pp. 271-284, 1965.

[50] Wolfe, P., and Cutler, L., Experiments in Linear Programming, Recent Advances in Mathematical Programming, New York, McGraw-Hill, pp. 177-200, 1963.

[51] Zoutendijk, G., A Product-Form Algorithm using Contracted Transformation vectors, Integer and Non-Linear Programming, pp. 511-523, 1970.

Figure 1

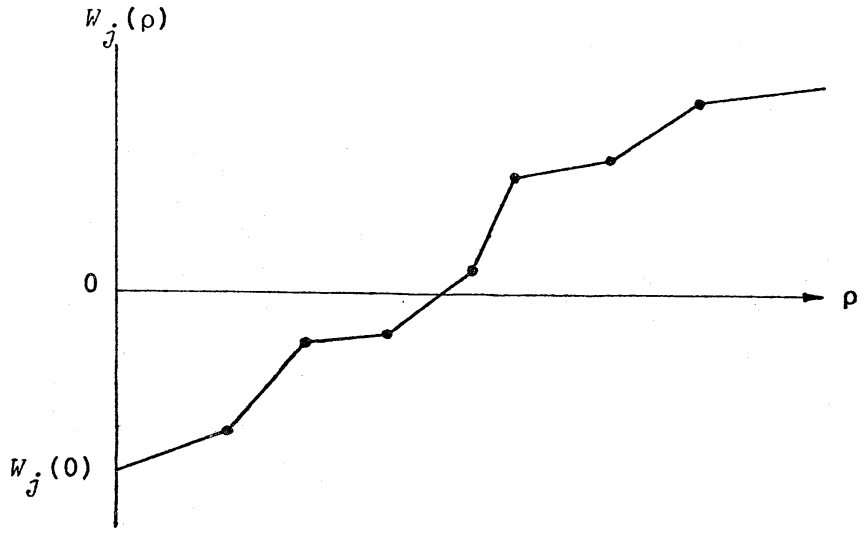
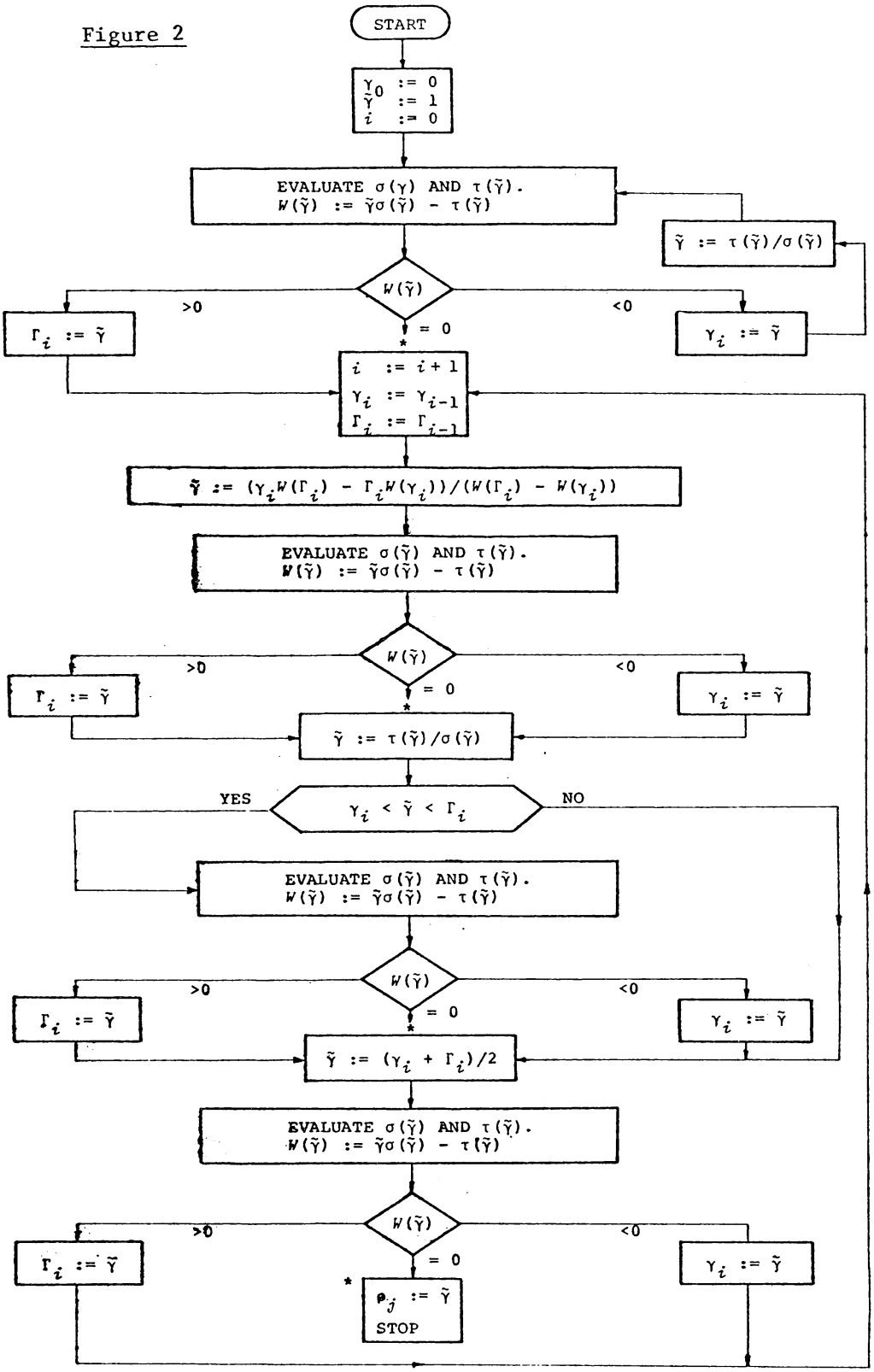


Figure 2


START
$\gamma_0 := 0$
$\tilde{\gamma} := 1$
$i := 0$
EVALUATE $\sigma(\gamma)$ AND $\tau(\tilde{\gamma})$.
$W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$
$\tilde{\gamma} := \tau(\tilde{\gamma})/\sigma(\tilde{\gamma})$
$W(\tilde{\gamma})$
>0
<0
= 0
$\Gamma_i := \tilde{\gamma}$
$\gamma_i := \tilde{\gamma}$
$i := i + 1$
$\gamma_i := \gamma_{i-1}$
$\Gamma_i := \Gamma_{i-1}$
$\tilde{\gamma} := (\gamma_i W(\Gamma_i) - \Gamma_i W(\gamma_i))/(W(\Gamma_i) - W(\gamma_i))$
EVALUATE $\sigma(\tilde{\gamma})$ AND $\tau(\tilde{\gamma})$.
$W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$
$W(\tilde{\gamma})$
>0
<0
= 0
$\Gamma_i := \tilde{\gamma}$
$\gamma_i := \tilde{\gamma}$
$\tilde{\gamma} := \tau(\tilde{\gamma})/\sigma(\tilde{\gamma})$
YES
$\gamma_i < \tilde{\gamma} < \Gamma_i$
NO
EVALUATE $\sigma(\tilde{\gamma})$ AND $\tau(\tilde{\gamma})$.
$W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$
$W(\tilde{\gamma})$
>0
<0
= 0
$\Gamma_i := \tilde{\gamma}$
$\gamma_i := \tilde{\gamma}$
$\tilde{\gamma} := (\gamma_i + \Gamma_i)/2$
EVALUATE $\sigma(\tilde{\gamma})$ AND $\tau(\tilde{\gamma})$.
$W(\tilde{\gamma}) := \tilde{\gamma}\sigma(\tilde{\gamma}) - \tau(\tilde{\gamma})$
$W(\tilde{\gamma})$
>0
<0
= 0
$\Gamma_i := \tilde{\gamma}$
$\gamma_i := \tilde{\gamma}$
$\rho_j := \tilde{\gamma}$
STOP