A Method for Solving Indefinite Quadratic Programming Problems

CH.SUBRAHMANYAM[1], Sudhir Patel[2]
[12] *Dept. of Mathematics, New Horizon College Of Engineering, Bangalore, Karnataka*

Abstract- In this paper, we give in section (1) compact description of the algorithm for solving general quadratic programming problems (that is, obtaining a local minimum of a quadratic function subject to inequality constraints) is presented. In section (2), we give practical application of the algorithm, we also discuss the computation work and performing by the algorithm and try to achieve efficiency and stability as possible as we can. In section (3), we show how to update the QR-factors of A1 (K), when the tableau is complementary ,we give updating to the LDLT-Factors of (K) A G . In section (4) we are not going to describe a fully detailed method of obtaining an initial feasible point, since linear programming literature is full of such techniques.

I. INTRODUCTION

In this section we give the detailed outlines of the algorithm of indefinite quadratic programming problems. It references to the numbers of some equations and conditions appeared in the following equations [1-8]:

$$\min_{i\in\eta} \lambda_i^{(K)} \tag{1}$$

and

$$\begin{bmatrix} G & -A_1^{(K)} & 0 \\ -A_1^{(K)T} & 0 & 0 \\ -A_2^{(K)T} & 0 & I \end{bmatrix} \begin{bmatrix} \underline{d}_x^{(K)} \\ \underline{d}_\lambda^{(K)} \\ \underline{d}_v^{(K)} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{e}_q \\ \underline{0} \end{bmatrix} \tag{2}$$

the algorithm assumes the availability of an initial basic feasible point [9-15]. The steps are:

1. Given $\underline{x}^{(1)}$, $\lambda_1^{(1)}$, $v_2^{(1)}$ and η, set $K=1$.

2. Solve (1) for q.

3. If $\lambda_q^{(K)} \geq 0$ terminate with $\underline{x}^* = \underline{x}^{(K)}$ otherwise solve $\min_{p\in\eta} \frac{v_p^{(K)}}{dv_p^{(K)}}$ $dv_p^{(k)} > 0$ for P_1.

4. If If $d\lambda_q^{(K)} < 0$ and $\frac{\lambda_q^{(K)}}{d\lambda_q^{(K)}} \leq \frac{vp_1^{(K)}}{dv_{p1}^{(K)}}$ is satisfied remove q from η; update the basic variables using [16-20] $v_q^{(K+1)} = \frac{\lambda_q^{(K)}}{d\lambda_q^{(K)}}$ to : $v_p^{(K+1)} = v_p^{(K)} - dv_p^{(K)}\ v_q^{(K+1)}$ $\left(P \notin \eta U\{q\}\right)$; set $K=K+1$ and go to (b)

otherwise remove q from η [21-28]; update the basic variables using $v_q^{(K+1)} = \frac{v_{p1}^{(K)}}{dv_{p1}^{(K)}}$ $v_p^{(K+1)} = v_p^{(K)} - d_{vp}^{(K)} v_q^{(K+1)}$ $\left(P \notin \eta U\{q\}\right)$

5. Set r=1 and k k=k, set $KK+r$ and add P_r to η .

6. Solve $\min_{\substack{p\in\eta \\ \sigma dv_p^{(k+r)}<0}} \sigma \frac{v_p^{(K+r)}}{d_{vq_p}^{(K+r)}}$ for P_{r+1}

7. If $\sigma d_{\lambda_q}^{(K+r)} < 0$ and $\frac{\lambda_q^{(K+r)}}{\sigma d_{\lambda_q}^{(K+r)}} \leq \frac{v_{p_{r+1}}^{(K+r)}}{\sigma d_{vp_{r+1}}^{(K+r)}}$ is satisfied, update the basic variables using $\lambda_{P_r}^{(K+r+1)} = \frac{\lambda_q^{(K+r)}}{d\lambda_q^{(K+r)}}$ to $v_p^{(K+r+1)} = v_p^{(K+r)} - d_{v_p}^{(K+r)} \lambda_{P_r}^{(K+r+1)}$, $P \notin \eta$.

Set K=K+1 and go to (b).

Otherwise update the basic variables using $\lambda_{P_r}^{(K+r+1)} = \frac{v_{p_{r+1}}^{(K+r)}}{dv_{vp_{r+1}}^{(K+r)}}$ to $v_p^{(K+r+1)} = v_p^{(K+r)} - d_{vp}^{(k+r)} \lambda_{P_r}^{(K+r+1)}$ $P \notin \eta$; set r=r+1 and go to (e)

Practical Application of the Algorithm

The algorithm presented above represents a general outline of a method for solving indefinite quadratic programming problems rather than an exact definition of a computer implementation. In this section we discuss the computational work performed by the algorithm, and try to achieve efficiency and stability as possible as we can. In doing so we follow, with slight modifications, the work of Gill and Murray which has been applied to active set methods since mid-seventies until now [7-10]. The slight modifications are made to cope with the new forms of the matrices used in the method when G is indefinite. In the case when G is positive (semi definite) the active set methods are considered to be equivalent, [20], pointed out. There he gave a detailed description of that equivalence. He also re-mentioned this equivalence [6]. The major computational work of the algorithm is in the solution of

$$\begin{bmatrix} G & -A_1^{(K)} & 0 \\ -A_1^{(K)T} & 0 & 0 \\ -A_2^{(K)T} & 0 & I \end{bmatrix} \begin{bmatrix} \underline{d}_x^{(K)} \\ \underline{d}_\lambda^{(K)} \\ \underline{d}_v^{(K)} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{e}_q \\ \underline{0} \end{bmatrix} \tag{3}$$

and

$$M_B^{(K+r)} = \begin{bmatrix} G & -A_1^{(K)} & -W & \underline{0} & 0 \\ -A_1^{(K)T} & 0 & 0 & \underline{e}_q & 0 \\ -W^T & 0 & 0 & \underline{0} & 0 \\ -\underline{a}_{P_r}^T & \underline{0}^T & \underline{0}^T & 0 & \underline{0}^T \\ -A_2^{(K+r)^T} & 0 & 0 & \underline{0} & I \end{bmatrix} \begin{bmatrix} \underline{d}_x^{(K+r)} \\ \underline{d}_\lambda^{(K+r)} \\ \underline{d}_w^{(K+r)} \\ \underline{d}_{vq}^{(K+r)} \\ \underline{d}_{vp}^{(K+r)} \end{bmatrix} = \begin{bmatrix} -\underline{a}_{P_r} \\ \underline{0} \\ \underline{0} \\ 0 \\ \underline{0} \end{bmatrix} \tag{4}$$

$$H = Z^{(K)}\left(Z^{(K)^T} G Z^{(K)}\right)^{-1} Z^{(K)^T}$$

$$T = S^{(K)} - Z^{(K)}\left(Z^{(K)^T} G Z^{(K)}\right)^{-1} Z^{(K)^T} G S^{(K)} \tag{5}$$

We do not solve them directly; instead, we make use of the special structure of the matrices involved. We use the matrices H and T are defined in eqn. (5). Thus, accordingly the solution in eqn. (3) is given by:

$$\underline{d}_x^{(K)} = -T\underline{e}_q \tag{6}$$

$$\underline{d}_1^{(K)} = U\underline{e}_g \tag{7}$$

$$\underline{d}_v^{(K)} = A_2^{(K)^T}\underline{dx}^{(K)} \tag{8}$$

$$\underline{d}_x^{(K+r)} = HW\underline{d}_W^{(K+r)} + T\underline{e}_q d_{v_q}^{(K+r)} - H\underline{a}_{P_r} \tag{9}$$

$$\text{and } \underline{\lambda}_1^{(K+r)} = -T^T W\underline{d}_W^{(K+r)} - U\underline{e}_q\underline{d}_{v_q}^{(K+r)} + T^T\underline{a}_{P_r} \tag{10}$$

$$\underline{d}_v^{(K+r)} = A_2^{(K+r)^T}\underline{d}_x^{(K+r)} \tag{11}$$

H, U, T define the inverse of the upper left partition of the basis matrix when the tableau is complementary. This calls for making them available at every complementary tableau. In other words they are to be updated from a complementary tableau to another [12].

Referring to $(Z^TGZ)\underline{y} = -Z^T(\underline{g} + GS\underline{b})$, H, T and U are given by:

$$H = Z^{(K)}\left(Z^{(K)^T}GZ^{(K)}\right)^{-1}Z^{(K)^T}$$

$$T = S^{(K)} - Z^{(K)}\left(Z^{(K)^T}GZ^{(K)}\right)^{-1}Z^{(K)^T}GS^{(K)} \tag{12}$$

$$U = S^{(K)^T}GZ^{(K)}\left(Z^{(K)^T}GZ^{(K)}\right)^{-1}Z^{(K)^T}GS^{(K)} - S^{(K)^T}GS^{(K)}$$

Where $S^{(K)}$ and $Z^{(K)}$ satisfy

$$S^{(K)^T}A_1^{(K)} = I \tag{13}$$

$$\text{and } Z^{(K)^T}A_1^{(K)} = 0 \tag{14}$$

The choice of S(K) and Z(K) to satisfy in eqns. (13) and (14), respectively is generally open. Here we take the choice given in S=Q1R-T, Z=Q2 which is, according to K(ZTGZ)≤K(G), is advantageous as far as stability is concerned. For the sake of making this section selfcontained we show how S(K) and Z(K) are obtained in away suitable to this section. Let:

$$Q^{(K)}A_1^{(K)} = \begin{bmatrix} R^{(K)} \\ 0 \end{bmatrix} \tag{15}$$

represent the QR factorization of $A_1^{(K)}$, where $A_1^{(K)}$ is $n\times L_x$. Thus $Q^{(K)}$ is $n\times n$ and $R^{(K)}$ is an $L_{K_x}L_K$ upper triangular matrix partition $Q^{(K)}$ into $Q^{(K)} = \begin{bmatrix} Q_1^{(K)} \\ Q_2^{(K)} \end{bmatrix}$, where $Q_1^{(K)}$ is $L_x\times n$. and $Q_2^{(K)}$ is $(n-L_K)\times n$. Thus, in eqn. (15) we have:

$$Q_1^{(K)}A_1^{(K)} = R^{(K)} \tag{16}$$

$$\text{and } Q_2^{(K)}A_1^{(K)} = 0 \tag{17}$$

so in eqns. (16) and (17) we define $S^{(K)}$ and $Z^{(K)}$ by:

$$S^{(K)} = Q_1^{(K)^T}R^{(K)^T} \tag{18}$$

$$\text{and } Z^{(K)} = \tilde{I}Q_2^{(k)^T} \tag{19}$$

Where I the identity matrix is whose columns are reversed. Thus we conclude by saying that the computation is focused on using the QR factorization of A K , (when the kth iteration is complementary)

So updating these factors is required at each iteration when the tableau is complementary [25].

Updating the QR-Factors of 1

A(K)

In this section we show how to update the QR-factors of ( )1*A K* , when the tableau is complementary, also we give updating to the *LDLT*Factors of (*K*) *A G* . Following the stream of our discussions, two cases are to be considered separately. The case when the (k+1)th iteration results in a complementary tableau, and the case when complementarity is restored at the (k+r+1)th iteration after r successive non-complementary tableaux. In the first case the factors of A1 (k) are updated to give those of (1) 1*A K* + , and this is the case when a column, *q a* say, is deleted from A1 (k). In the second case the factors of A1 (k) are used to give those of A1 (k+r+1), and this is the case when one column, *q a* say, is deleted from A1 (k) and then r other columns are added to A1 (k). We follow the same steps carried [9], with the appropriate modification in the second case. In the first case, let A1 (k) be the n×(Lk-1) matrix obtained by deleting the *q*th

column, *q a* , from A1 (k). Suppose the QR-factorization of A12 (k) is given.

by: $Q^{(K)}A_1^{(K)} = \begin{bmatrix} R^{(K)} \\ 0 \end{bmatrix}$, partition $A_1^{(k)}$ into: $A_1^{(K)} = \begin{bmatrix} A_{11}^{(K)} & \underline{a}_q & A_{12}^{(K)} \end{bmatrix}$

Where $A_{11}^{(K)}$ is n×(q-1) and $A_{12}^{(K)}$ is n×(L_K-q). Let $R^{(K)}$ have the form

$$R^{(K)} = \begin{bmatrix} R_{11} & \underline{\alpha} & R_{12} \\ \underline{0}^T & \gamma & \underline{\beta}^T \\ 0 & \underline{0} & R_{22} \end{bmatrix}$$

where R_{11} is (q-1)×(q-1) upper triangular, R_{12} is (q-1)×(L_k-q), R_{22} is (L_k-q)×(L_k-q) upper triangular, $\underline{\alpha}$ is a (q-L) vector, β is an (L_K-q) vector and γ is a scalar. Since $A_1^{(K+1)} = \begin{bmatrix} A_{11}^{(K)} & A_{12}^{(K)} \end{bmatrix}$, $Q^{(K)}A^{(K+1)}$ will have the form:

$$\begin{bmatrix} R_{11} & R_{12} \\ \underline{0}^T & \underline{\beta}^T \\ 0 & R_{22} \end{bmatrix}$$

Now, let Q_1' be the product of the plane rotations which gives:

$$Q_1' \begin{bmatrix} \underline{\beta}^T \\ R_{22} \end{bmatrix} = \begin{bmatrix} R' \\ \underline{0}^T \end{bmatrix}$$

where R' is (L_k-q)×(L_k-q) upper triangular. In this case Q_1' is an (L_k-q+1)×(L_k-q+1) orthogonal matrix. Thus if $Q' = \begin{bmatrix} I & 0 & 0 \\ 0 & Q_1' & 0 \\ 0 & 0 & I \end{bmatrix} \begin{matrix} q-L \\ L_K-q+1 \\ n-L_K \end{matrix}$

Which is orthogonal, then $Q'Q^{(K)}A_1^{(K+1)} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R' \\ 0 & 0 \end{bmatrix} \begin{matrix} q-1 \\ L_K-q \\ n-L_K+1 \end{matrix}$

So we obtain $Q^{(K+1)} = Q'Q^{(K)}$ and $R^{(K+1)} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R' \end{bmatrix}$ (20)

Thus, only the rows from the q^{th} to the L_k^{th} of $Q^{(K)}$ are altered in obtaining $Q^{(K+1)}$, so if $Q^{(K+1)}$ is partitioned into

$$Q^{(K+1)} = \begin{bmatrix} Q_1^{(K+1)} \\ Q_2^{(K+1)} \end{bmatrix} \begin{matrix} L_{K-1} \\ n-L_{K+1} \end{matrix} \quad (21)$$

then $Q_2^{(K+1)}$, in particular, takes the form:

$$Q_2^{(K+1)} = \begin{bmatrix} \underline{q}^T \\ Q_1^{(K)} \end{bmatrix} \quad (22)$$

Note also that the first q-1 rows of $Q_1^{(k)}$ are not changed. This fact might be helpful as far as efficiency is concerned if we want to think of another alternative of choosing *q* in eqn. (1), such an alternative is:

$$Q=\max\{i=\lambda_i<0, 1\le i\le L_k\} \quad (23)$$

So that increase the number of rows of $Q_1^{(k)}$ and $R^{(K)}$ which are unaltered in iteration *(K+1)*, which in turns reduces the effort, especially when L_k is relatively large. We now consider the second case when complementarity is restored at the $(k+r+1)^{th}$ iteration. Let $W' = \begin{bmatrix} \underline{a}_{p+r}, \ldots, \underline{a}_{p1} \end{bmatrix}$. Let $A_1^{(K+1)}$ be obtained from $A_1^{(k)}$ by removing $\underline{a}_q$. Thus

$$A^{(K+r+1)} = \begin{bmatrix} A^{(K+1)} : W' \end{bmatrix} \quad (24)$$

Pre-multiplying both sides in eqn. (24) by $Q^{(k+1)}$ (defined in eqn. (21)) we get

$$Q^{(K+1)}A^{(K+r+1)} = \begin{bmatrix} R^{(K+1)} & W_1' \\ 0 & W_2' \end{bmatrix} \begin{matrix} L_{K-1} \\ n-L_{K+1} \end{matrix} \quad (25)$$

where $W_1' = Q_1^{(K+1)}W'$, $W_2' = Q_2^{(K+1)}W'$ and $R^{(k+1)}$ is defined in eqn. (20)

$$\text{let } Q_2'W_2' = \begin{bmatrix} R_2' \\ 0 \end{bmatrix} \quad (26)$$

Define the QR-factorization of W_2'. Here Q_2' is (h-L_k+1)×(n-L_k+1) and orthogonal, and R_2' is r×r upper triangular. If $Q'' = \begin{bmatrix} 1 & 0 \\ 0 & Q_2' \end{bmatrix} \begin{matrix} L_K-1 \\ n-L_K+1 \end{matrix}$

Then $Q'Q^{(K+1)}A^{(K+r+1)} = \begin{bmatrix} R^{(K+1)} & W_1' \\ 0 & R_2' \\ 0 & 0 \end{bmatrix} \begin{matrix} L_K-1 \\ r \\ n-L_K-r-1 \end{matrix}$ (27)

Thus we obtain the QR-factorization of $A^{(k+r+1)}$ with

$$Q^{(K+r+1)} = Q'Q^{(K+1)} = Q'Q'Q^{(K)} \quad (28)$$

And $R^{(K+r+1)} = \begin{bmatrix} R^{(K+1)} & W_1' \\ 0 & R_2' \end{bmatrix}$ (29)

Updating The *LDLT*-Factors of (*K*)
A G

The factors *L*(*K*)*D*(*K*)*L*(*K*)*T* of (*K*) *A G* are updated at each iteration when the tableau is complementary. Near the end of this section we show that (1) *A G* is always positive definite (on the assumption that (*K*)
A G is positive semi-definite). Updating these factors is very stable when (*K*)*A G* is positive definite as we shall see. This fact is counted as one of the good numerical features of the method. We consider the case when the (k+1)th iteration results in a complementary tableau. Unfortunately,in the other case when complementarity is restored at the (k+r+1)thiteration, we are unable till now to explore a way of

using the factorsof (*K*)*A G* in obtaining those of (*K r* 1)*A G* + + . However n-Lk-r, the dimensionof (*K r* 1)*A G* + + , decreases with r, in which case the effort of re-factorizing(*K r* 1)*A G* + + might not be so much, especially when n-Lk is itself small.This calls for choosing the starting *L1* so that n-L1 is small. In the casewhen the number of constraints is greater than *n, L1* is chosen to beequal to n; that is the initial guess *x*(1) is a vertex. With this choice(1) 0 *A G* = , and in the second iteration we might expect a constraint to bedeleted from the active set (which is the case when the second iteration is complementary). Otherwise the third iteration will definitely restore complementarity at another vertex leaving (3) 0 *A G* = . In the former case the dimension of (2) *A G* is 1. In general the dimension of (*K*)*A G* keeps on increasing when constraints are deleted, and updating the factors is straight forward as will be shown. On the other hand the dimension of (*K*)*A G* keeps on decreasing when constraints are added to the active set, and in this case we are faced with re-factorizing the factors. We return to the case when the (k+1)th iteration is complementary. Here we

are almost copying the work of in eqn. (9). In this case, as in eqn. (25)

shows $G_2^{(K+1)} = \begin{bmatrix} q^T \\ Q_2^{(K)} \end{bmatrix}$, and using in eqn. (19) we have :

$$Z^{(K+1)} = \tilde{I}\, G_2^{(K+1)^T} = \left[Z^{(K)} q \right] \tag{30}$$

the matrix $G_A^{(K+1)}$ is given by:

$$G_A^{(K+1)} = Z^{(K+1)^T} G Z^{(K+1)} = \begin{bmatrix} G_A^{(K)} & Z^{(K)T} G\underline{q} \\ q^T G Z^{(K)} & \underline{q}^T G q \end{bmatrix} \tag{31}$$

It can be shown that when a symmetric matrix is augmented by a single row and a column, the lower-triangular factor is augmented by a single row. Define:

$$L^{(K+1)} = \begin{bmatrix} L^{(K)} & \underline{0} \\ 1^T & 1 \end{bmatrix}, \quad D^{(K+1)} = \begin{bmatrix} D^{(K)} & \underline{0} \\ \underline{0}^T & d_{n-L_K+1} \end{bmatrix} \tag{32}$$

If we substitute in eqn. (31) and in eqn.(32) into the identity:

$G_A^{(K+1)} = L^{(K+1)} D^{(K+1)} L^{(K+1)^T}$, we obtain $\underline{L}$ and d_{n-L_K+1} as the solution of the equations

$$L^{(K)} D^{(K)} \underline{1} = Z^{(K)T} G\underline{q} \tag{33}$$

and $$d_{n-L_K+1} = \underline{q}^T G \underline{q} - \underline{L}^T D^{(K)} \underline{L} \tag{34}$$

The numerical stability of this scheme is based on the fact that, if (*K* 1) *A G* + is positive definite, the element *n LK* 1 *d* − + must be positive. In this event in eqn. (34) ensures that arbitrary growth in magnitude cannot occur in the elements of *L* . Before ending this section we show that when the kth iteration and the (k+1)th iteration are complementary then (*K* 1) *A G* + must be positive definite. Let the tableau be complementary at the kth iteration. Let ( ) 1

A K be the matrix whose columns correspond to the active constraints, and (*K*) 0 *q* λ<. The increase of vq changes *f* according to

$$f = f^{(K)} + \lambda_q v_q - 0.5 u_{qq} v_q^2 \tag{35}$$

$$\left(u_{qq} = \underline{e}_q^T U \underline{e}_q \right)$$

$$\lambda_q = \lambda_q^{(K)} - u_{qq} v_q \tag{36}$$

and $\underline{x}$ changes according to

$$\underline{x} = \underline{x}^{(K)} + T\underline{e}_q v_q \tag{37}$$

For the next tableau (i.e., the $(k+1)^{th}$)) to be complementary u_{qq} must be negative, and the new value $v_q^{(K+1)} \left(\equiv \frac{\lambda_q^{(K)}}{u_{qq}} \right)$ of v_q must not violate feasibility.

Thus, using in eqn. (35), we have $\frac{d^2 f}{dv_q^2} = -u_{qq} > 0$ which reflects the fact that *f* possesses a positive curvature along the direction $T\underline{e}_q$.

Now let $A_1^{(K+1)}$ be obtained from $A_1^{(K)}$ by removing $\underline{a}_q$ and let $Z^{(k+1)}$ be defined so that $Z^{(K+1)^T} A^{(K+1)} = 0$.

Pre-multiply both sides of in eqn. (37) by $A^{(K+1)^T}$ to get:

$$A^{(K+1)^T} T\underline{e}_q = \underline{0} \tag{38}$$

showing that $T\underline{e}_q$ lies in the space spanned by the columns of $Z^{(k+1)}$, so

$$T\underline{e}_q = Z^{(K+1)} \underline{h} \tag{39}$$

for some (n-L_k+1) vector $\underline{h}$. Since along $T\underline{e}_q$ at $v_q^{(K+1)}$, $\frac{df}{dv_q} = 0$ and $\frac{d^2 f}{dv_q^2} > 0$, then *f* is minimum along $T\underline{e}_q$ at $\underline{x}^{(K+1)}$, where

$$\underline{x}^{(K+1)} = \underline{x}^{(K)} + T\underline{e}_q v_q^{(K+1)} \tag{40}$$

We therefore conclude, in the active set methods sense, that the direction $\delta^{(K+1)} = T\underline{e}_q v_q^{(K+1)}$ solves the equality problem:

$$\text{minimize } 0.5\underline{\delta}^T G \underline{\delta} + \underline{\delta}^T \left(G\underline{x}^{(K)} + g \right)$$

$$\text{Subject to } A^{(K+1)^T} \underline{\delta} = \underline{0} \tag{41}$$

Using (39), $\delta^{(K+1)} = Z^{(K+1)} \underline{h}\, v_q^{(K+1)} = Z^{(K+1)} \delta_A^{(K+1)}$, thus $\underline{\delta a}^{(K+1)}$ solves the problem

$$\text{minimize } 0.5\underline{\delta}_A^T \left(Z^{(K+1)^T} G Z^{(K+1)} \right) \underline{\delta}_A + \underline{\delta}_A^T \left(G\underline{x}^{(K)} + g \right),$$

from which we conclude that $G_A^{(K+1)} = Z^{(K+1)^T} G Z^{(K+1)}$ is positive definite.

Finding an Initial Feasible Point

In this section we are not going to describe a fully detailed method of obtaining an initial feasible point, since linear programming literature is full of such techniques. The method of finding a feasible point has been resolved in linear programming by a technique known as phase 1 simplex [27]. The basis of the technique is to define an artificial objective function, namely: set of indices of constraints which are violated at the point x , and to minimize this function with respect to x , subject to the constraints $T\ 0\ j\ j\ a\ x - b \geq ,j \notin v(x)$. The function $F\square\ (x)$ is linear and is known as the sum of infeasibilities. If a feasible point exists the solution x^* of the artificial problem is such that $F\square\ (x^*) = 0$. In the case when m exceeds n, a non-feasible vertex is available as an initial feasible point to phase 1 and the simplex method is applied to minimize $F\square\ (x)$. This process will ultimately lead to a feasible vertex [28]. Direct application of this method to finding a feasible point in the case when m is less than n is not feasible since, although a feasible point may exist a feasible vertex will not. Under these circumstances artificial vertices can be defined by adding simple bounds to the variables, but this could lead to either a poor initial point, since some of these artificial constraints must be active, or exclusion of the feasible region. A way out of this dilemma is described [6-9] a number of methods including the above one have been described. Gill and Murray is advantageous in that it makes available the QR-factorization of the initial matrix of active constraints which is then directly used in our algorithm.

II. REFERENCES

[1]. Bazaraa SM, Sherali DH, Shetty CM (1994) Nonlinear Programming: Theory and Algorithm.

[2]. Coleman LLS (1990) Numerical Optimization. SIAM Books.

[3]. Cottle RW (1990) The Principle Pivoting method positive visited math program. 48: 369-385.

[4]. David GL (2003) Linear and nonlinear programming (2nd edn.), Pearson Education.

[5]. Dennis, Schnabel (1996) Numerical Methods for unconstrained Optimization and nonlinear equation classics in applied Mathematics. Society for Industrial and Applied Mathematics.

[6]. Fletcher R (2013) Practical Methods of Optimization. (2nd edn.), John Wiley and Sons.

[7]. Gill PE, Murray W (1973) A numerically stable form of the simplex Algorithm. Journal Linear Algebra Applors 7: 99-138.

[8]. Gill PE, Murray W (1975) Numerical Methods for constrained optimization. Academic press.

[9]. Gill PE, Murray W (1978) Numerically Stable methods for quadratic programming. Math Programme 14: 349-372.

[10]. Gill PE, Murray W, Margaret W (1981) Practical Optimization. Academic Press.

[11]. Kunisch K, Rendl F (2003) An infeasible active set method for quadratic problems with simple bounds. SIAM Journal on Optimization 14: 35-52.

[12]. Mohsin HAH (1996) An Extension to the Dantzig-Wolfe Method for general quadratic programming. University of Khartoum.

[13]. Byrd RH, Gillbert JC, Nocedal J (2000) A trust region method based on interior point techniques for nonlinear programming. 9: 149-185.

[14]. Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University.

[15]. Jorge N (2006) Stephen Wright books. Springer Series in Operations Researchnd Financial Engineering.

[16]. Stephen G N, Ariela S (1996) Linear and non -liner programming. McGraw Hill,New York.

[17]. Anitescu M (2005) On solving mathematical Programs with complementarityconstraints as nonlinear programs. SIAM Journal on Optimization 15: 1203-1236.

[18]. Byrd RH, Hribar ME, Nocedal J (1999) An interior point algorithm for largescalenonlinear programming. Society for Industrial and Applied Mathematics9: 877-900.

[19]. Gould NIM, Toint PL (2005) An interactive working set method for large scalenonlinear optimization, ActaNumerica 14: 299-361.

[20]. Gould NIM, Tiont PL (2002) An iterative working set method for large scale nonconvexquadratic programming. Applied Numerical Mathematics 43: 109-128.

[21]. Fletcher R, Leyffer S (2002) Nonlinear programming without a penalty function.Mathematical Programming 91: 239-269.

[22]. Gill PEM, Murray W, Wright M (1991) Numerical Linear Algebra andOptimization.

[23]. Kočvara M, Stingl MP (2003) PENNON: A code for non-converx non-linear andsemi-definite programming Optimization Methods and software. 18: 317-333.

[24]. Vanderbi RJ, Shanno DF (1999) An interior point algorithm for non-convex nonlinearprogramming. Computation and Applications 13: 231-252.

[25]. Vavasis SA (1990) Quadratic programming is NP. Information Processing

[26]. Letters 36: 73-77.

[27]. Gill PEM, Murray W, Wright MH (1981) Practical Optimization. Academic Press.

[28]. Vavasis SA (1991) Nonlinear Optimization. Oxford University Press, New Yorkand Oxford.

[29]. Higham NJ (1996) Accuracy and Stability Of Numerical Algorithms. SIAM

[30]. Publications Philadelphia.