Mathematical Theory and Modeling
ISSN 2224-5804 (Paper) ISSN 2225-0522 (Online)
Vol.3, No.2, 2013

Implementation of The Extended Dantzig – Wolfe Method

Awatif M. A. El Siddieg

Department of Mathematics, Faculty of Mathematical Sciences

Elneilain university-Sudan

*Corresponding author-mail:wigdan@hotmail.com

Abstract

In this paper implementation of the extended Dantzig - Wolfe method to solve a general quadratic programming problem is presented ,that is, obtaining a local minimum of a quadratic function subject to inequality constraints. The method terminates successfully at a KT point in a finite number of steps. No extra effort is needed when the function is non-convex.

The method solve convex quadratic programming problems. It is a simplex like procedure to the Dantzig - Wolfe method[1]. So, it is, the same as the Dantzig – Wolfe method when the Hessian matrix of the quadratic function is positive definite[7].

The obvious difference between our method and the Dnatzig – Wolfe method is in the possibility of decreasing the complement of the new variable that has just become non-basic.

In the practical implementation of the method we inherit the computational features of the active set methods using the matrices H, U and T, and in particular the stable features [5]. The features (i.e, the stable features) are achieved by using orthogonal factorizations of the matrix of active constraints when the tableau is complementary.

0. Introduction:

In our work we give full description to an algorithm applying the extended Dantzig-wolfe method as a model of the quadratic programming problems. [16].In section (**1**), general description of the algorithm of the Extended Dantzig-Wolfe method is given. In section (**2**) , we describe the steps to be followed when the current tableau is complementary. This is followed by describing the moves to be carried when the current tableau is non-complentary. Section (**3**) is summed up by outlining the main steps of the algorithms in a compact form that helps in writing a computer code to apply the method.

Section (**4**), describes the practical features of applying the algorithm. In that section we aim at producing an efficient code and is done by making use of the informations carried over from previous steps, the matrices H, T and U such updating factors of matrices (A = QR) rather than refactorizing them. Also we will try there not to carry out unnecessary calculations.

In section **(5)** we show how to update the QR-factors of $A_1^{(K)}$ when the tableau is complementary. We update the factors $L^{(K)}D^{(K)}L^{(K)^T}$ of $G_A^{(K)}$ at each iteration when the tableau is complementary.

As our method is a feasible-point method, in section (**6**), we show how to obtain an initial feasible point. The literature is full of methods for obtaining initial feasible points, but we prefer to follow the same steps as in [16].

1. General Description of the Algorithm of the Extended Dantzig-Wolfe Method:-

The description is given in two parts. The first part is to describe the steps to be carried out after a complementary tableau. The second part shows the moves to be followed when we are at a non-complementary tableau. In each part we show how the basic variables are updated and how decisions are made regarding the next move. The theoretical tools upon which the justification of the algorithm relies, were provided in chapter three.

Suppose at the kth iteration the tableau is complementary. Thus the basis matrix will have the form:

$$\begin{bmatrix} G & -A_1^{(K)} & 0 \\ -A_1^{(K)T} & 0 & 0 \\ -A_2^{(K)T} & 0 & I \end{bmatrix}$$

At this stage the basic variables $\underline{x}^{(K)}$, $\underline{\lambda}^{(K)}$ and $v_p^{(K)}$, $P \notin \eta$, are known. (Note: these values of the basic variables were carried over from the previous iteration, but not recalculated). The next step is to find the index which satisfies the minimality.

$$\min_{i \in \eta} \lambda_i^{(K)} \qquad (1.1)$$

If $\lambda_q^{(K)} \geq 0$, then the algorithm terminates at $\underline{x}^{(K)}$ which is a KT-point. If not, then the system:

$$\begin{bmatrix} G & -A_1^{(K)} & 0 \\ -A_1^{(K)T} & 0 & 0 \\ -A_2^{(K)T} & 0 & I \end{bmatrix} \begin{bmatrix} \underline{d}_x^{(K)} \\ \underline{d}_\lambda^{(K)} \\ \underline{d}_v^{(K)} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{e}_q \\ \underline{0} \end{bmatrix} \qquad (1.2.)$$

is solved to obtain $\underline{d}x^{(K)}$, $\underline{d}\lambda^{(K)}$ and $\underline{d}_v^{(K)}$, the next step is to select the index P_1 which solves:

$$\min_{p\notin\eta} \quad \frac{v_p^{(K)}}{dv_p^{(K)}} \qquad (1.3)$$

$$dv_p^{(k)} > 0$$

If $d\lambda_q^{(K)} < 0$ and $\frac{\lambda_q^{(K)}}{d\lambda_q^{(K)}} \le \frac{vp_1^{(K)}}{dv_{p1}^{(K)}}$ (1.4)

Then the next tableau is complementary with the following basic matrix:

$$\begin{bmatrix} G & -A_1^{(K+1)} & 0 \\ -A_1^{(K+1)^T} & 0 & 0 \\ -A_2^{(K+1)^T} & 0 & I \end{bmatrix}$$

Where $A_1^{(k+1)}$ is resulting by removing $\underline{a}_q$ from $A_1^{(k)}$ and $A_2^{(k+1)}$ results from adding $\underline{a}_q$ to $A_2^{(k)}$. Thus the index set η is updated by removing q.

The basic variables are now updated to

$$v_q^{(K+1)} = \frac{\lambda_q^{(K)}}{d\lambda_q^{(K)}} \qquad (1.5)$$

$$\underline{x}^{(K+1)} = x^{(K)} - \underline{d}_x^{(K)} v_q^{(K+1)} \qquad (1.6)$$

$$\lambda_S^{(K+1)} = \lambda_S^{(K)} - d_{\lambda_s}^{(K)} v_q^{(K+1)} \qquad (1.7)$$

$$v_P^{(K+1)} = v_P^{(K)} - dv_P^{(K)} \quad v_q^{(K+1)} \qquad \left(P \notin \eta U\{q\}\right) \qquad (1.8)$$

If (1.4) is not satisfied, then a non complementary tableau is obtained. The resulting basis matrix takes the form:

$$M_B^{(K+1)} = \begin{bmatrix} G & -A_1^{(K)} & \underline{0} & 0 \\ -A_1^{(K)^T} & 0 & \underline{e}_q & 0 \\ -\underline{a}_{P_1}^T & \underline{0}^T & 0 & \underline{0}^T \\ -A_2^{(K+1)^T} & 0 & \underline{0} & I \end{bmatrix}$$

Where $A_2^{(K+1)}$ is resulting by removing $\underline{a}_{P1}$ from $A_2^{(K)}$. The index set is thus updated by removing q and adding P_i. The new values of the basic variables are given by:

$$v_q^{(K+1)} = \frac{v_{P1}^{(K)}}{dv_{P1}^{(K)}} \qquad (1.9)$$

$$\underline{x}^{(K+1)} = \underline{x}^{(K)} - \underline{d}_x^{(K)} v_q^{(K+1)} \qquad (1.10)$$

$$\underline{\lambda}_1^{(K+1)} = \underline{\lambda}_1^{(K)} - \underline{d}_\lambda^{(K)} v_q^{(K+1)} \qquad (1.11)$$

$$v_P^{(K+1)} = v_P^{(K)} - d_{vP}^{(K)} v_q^{(K+1)} \qquad \left(P \notin \eta U\{q\}\right) \qquad (1.12)$$

2 . <u>The Move From Non-Complementary Tableau:-</u>

Suppose at the kth iteration the tableau was complementary, and the condition $\lambda_q^{(K)} < 0$ is fulfilled. Let the iterations after the kth is r add $P_1, P_2, \ldots\ldots, P_r$ respectively to η before restoring complementarity. Then , the current basic matrix $M_B^{(K+r)}$ is given by the formula:

$$M_B^{(K+r)} = \begin{bmatrix} G & -A_1^{(K)} & -W & \underline{0} & 0 \\ -A_1^{(K)T} & 0 & 0 & \underline{e}_q & 0 \\ -W^T & 0 & 0 & \underline{0} & 0 \\ -\underline{a}_{\Pr}^T & \underline{0}^T & \underline{0}^T & 0 & \underline{0}^T \\ -A_2^{(K+r)^T} & 0 & 0 & \underline{0} & I \end{bmatrix}$$

Where $W = \left[\underline{a}_{pr-1}, \ldots, \underline{a}_{p1}\right]$.

<u>Note 2.1.</u> There is a slight modification from the representation of $M_B^{(K+r)}$ In sections (**4**), (**5**) and (**6**) which is just a permutation of that one. At this stage $\underline{x}^{(K+r)}$, $\underline{\lambda}_1^{(K+r)}$, $\underline{\lambda}_w^{(K+r)}$, $v_q^{(K+r)}$ and $v_P^{(K+r)}$, $P \notin \eta U\{q\}$, are known. The first step is to solve the system

$$M_B^{(K+r)} = \begin{bmatrix} G & -A_1^{(K)} & -W & \underline{0} & 0 \\ -A_1^{(K)T} & 0 & 0 & \underline{e}_q & 0 \\ -W^T & 0 & 0 & \underline{0} & 0 \\ -\underline{a}_{\Pr}^T & \underline{0}^T & \underline{0}^T & 0 & \underline{0}^T \\ -A_2^{(K+r)^T} & 0 & 0 & \underline{0} & I \end{bmatrix} \begin{bmatrix} \underline{d}_x^{(K+r)} \\ \underline{d}_\lambda^{(K+r)} \\ \underline{d}_w^{(K+r)} \\ \underline{d}_{vq}^{(K+r)} \\ \underline{d}_{vp}^{(K+r)} \end{bmatrix} = \begin{bmatrix} -\underline{a}_{\Pr} \\ \underline{0} \\ \underline{0} \\ 0 \\ \underline{0} \end{bmatrix} \qquad (2.1)$$

for the basic variables $\underline{d}_x^{(K+r)}$, $\underline{d}_\lambda^{(K+r)}$, $\underline{d}_W^{(K+r)}$ and $d_{vP}^{(K+r)}$, $P \notin \eta U\{q\}$. Before moving to the next step we define σ as follows:

$$\sigma = \begin{cases} 1 \quad if \left(\left(d_{vq}^{(K+r)} > 0 \right) \quad or \; d_{vq}^{(K+r)} = 0 \;\; and \; d_{\lambda q}^{(K+r)} < 0 \right) \\ -1 \quad if \left(\left(d_{vq}^{(K+r)} < 0 \right) \quad or \; d_{vq}^{(K+r)} = 0 \;\; and \; d_{\lambda q}^{(K+r)} > 0 \right) \end{cases}$$

The next move will be to obtain P_{r+1} which solves

$$\min_{\substack{p \notin \eta \\ \sigma d v_p^{(k+r)} < 0}} \sigma \frac{v_P^{(K+r)}}{d_{vq_p}^{(K+r)}} \tag{2.2}$$

If $\sigma d_{\lambda_q}^{(K+r)} < 0 \quad and \quad \dfrac{\lambda_q^{(K+r)}}{\sigma d_{\lambda_q}^{(K+r)}} \le \dfrac{v_{p_{r+1}}^{(K+r)}}{\sigma d_{vp_{r+1}}^{(K+r)}}$ (2.3)

Then the next tableau is complementary and given by the form:

$$M_B^{(K+r+1)} = \begin{bmatrix} G & -A_1^{(K+r+1)} & 0 \\ -A_1^{(K+r+1)^T} & 0 & 0 \\ -A_2^{(K+r+1)^T} & 0 & I \end{bmatrix}$$

Where,

$$A_1^{(K+r+1)} = \left[A_1 : W : \underline{a}_{P_r} \right]$$

Here A_1, results from $A_1^{(K)}$ by removing $\underline{a}_q$, $A_2^{(K+r+1)}$ results from adding $\underline{a}_q$ to $A_2^{(K+r)}$. The new basic variables are updated and given by the following equations:

$$\lambda_{P_r}^{(K+r+1)} = \frac{\lambda_q^{(K+r)}}{d\lambda_q^{(K+r)}} \tag{2.4}$$

$$\underline{x}^{(K+r+1)} = \underline{x}^{(K+r)} - \underline{d}x^{(K+r)} \lambda_{P_r}^{(K+r+1)} \tag{2.5}$$

$$\lambda_S^{(K+r+1)} = \lambda_s^{(K+r)} - d_{\lambda_3}^{(K+r)} \lambda_{P_r}^{(K+r+1)}, \quad (S \in \eta) \tag{2.6}$$

$$v_P^{(K+r+1)} = v_P^{(K+r)} - d_{v_p}^{(K+r)} \lambda_{P_r}^{(K+r+1)}, \quad P \notin \eta \tag{2.7}$$

On the other hand, if (2.3) is not satisfied, then the resulting tableau is still non-complementary with $M_B^{(K+r+1)}$ having the form:

$$M_B^{(K+r+1)} = \begin{bmatrix} G & -A_1^{(K)} & -W^{\setminus} & \underline{0} & 0 \\ -A_1^{(K)T} & 0 & 0 & \underline{e}_q & 0 \\ -W^{\setminus T} & 0 & 0 & \underline{0} & 0 \\ -\underline{a}_{\mathrm{Pr}+1}^T & \underline{0}^T & \underline{0}^T & 0 & \underline{0}^T \\ -A_2^{(K+r+1)^T} & 0 & 0 & \underline{0} & I \end{bmatrix}$$

where $A_2^{(K+r+1)}$ is obtained from $A_2^{(K+r)}$ by removing $\underline{a}_{p_{r+1}}$ and $W^{\setminus} = \left[\underline{a}_{P_r} : W\right]$. The basic variables are updated and given by the system:

$$\lambda_{P_r}^{(K+r+1)} = \frac{v_{P_{r+1}}^{(K+r)}}{dv_{rpr+1}^{(K+r)}} \qquad (2.8)$$

$$\underline{x}^{(K+r+1)} = x^{(K+r)} - \underline{d}x^{(K+r)}\lambda_{P_r}^{(K+r+1)} \qquad (2.9)$$

$$\underline{\lambda}_1^{(K+r+1)} = \underline{\lambda}_1^{(K+r)} - \underline{d}\lambda^{(K+r)}\lambda_{P_r}^{(K+r+1)} \qquad (2.10)$$

$$\underline{\lambda}_W^{(K+r+1)} = \underline{\lambda}_W^{(K+r+1)} - \underline{d}w^{(K+r)}\lambda_{P_r}^{(K+r+1)} \qquad (2.11)$$

$$v_P^{(K+r+1)} = v_P^{(K+r)} - d_{vP}^{(k+r)}\lambda_{\mathrm{Pr}}^{(K+r+1)} \quad P \notin \eta \qquad (2.12)$$

Where ,the index set is then updated by adding P_{r+1}

In the following we give the detailed outlines of the algorithm. It combines the steps described in the previous sections in one whole unit. It references to the numbers of some of the conditions and equations appeared in (1. 1) and (1.2). the algorithm assumes the availability of an initial basic feasible point. The steps are:

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

2) Apply by (1.1) and (1.2) and solve the output for q .

3) If $\lambda_q^{(K)} \geq 0$ terminate with $\underline{x}^* = \underline{x}^{(K)}$ otherwise apply by (1.3) and (1.2) and solve the output for P_1.

4) If (1.4) is satisfied remove q from η; update the basic variables using (1.5) to (1.8); set $K = K + 1$ and go to 2) otherwise remove q from η; update the basic variables using (1.9) to (1. 12);

5) Let r=1 , kk =k , $KK + r$ and add P_r to η.

6) Apply by (2.2) , (2.1) and solve the output for P_{r+1} .

7) If (2.3) is satisfied, update the basic variables using (2.4) to (2.7) set K = K+1 and go to **2),** otherwise update the basic variables using (2.8) to (2.12).

8) Let r = r + 1 and go to **5)**. (See[8]).

3. Practical Application Of The Algorithm:-

The algorithm presented in the previous section represents a general outline of a method 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 midseventies until now (see [4,5,6]). The slight modifications are made to cope with the new forms of the matrices used in our method when G is indefinite. In the case when G is positive (semi definite) our method (which is in this case the Dantzig-Wolfe method) and the active set methods are considered to be equivalent, is pointed out. In[12] The auther gave a detailed description of that equivalence. He also remensioned this equivalence in [2].

The major computational work of the algorithm is in the solution of (1.2) and (2.1). We do not solve them directly; instead, we make use of the special structure of the matrices involved. We use the matrices H, T and U defined in (3.7). Thus, accordingly the solution of (1.2.) is given by:

$$\underline{d}_x^{(K)} = -T\underline{e}_q \qquad (3.1)$$

$$\underline{d}_\lambda^{(K)} = U\underline{e}_{\underline{e}} \qquad (3.2)$$

$$\underline{d}_v^{(K)} = A_2^{(K)^T}\underline{d}x^{(K)} \qquad (3.3)$$

$$\underline{d}_x^{(K+r)} = HW\underline{d}_W^{(K+r)} + T\underline{e}_q d_{v_q}^{(K+r)} - H\underline{a}_{P_r} \qquad (3.4)$$

And there for we get :

$$\underline{\lambda}_1^{(K+r)} = -T^T W\underline{d}_W^{(K+r)} - U\underline{e}_q\underline{d}_{v_q}^{(K+r)} + T^T\underline{a}_{P_r} \qquad (3.5)$$

$$\underline{d}_v^{(K+r)} = A_2^{(K+r)^T}\ \underline{d}_x^{(K+r)} \qquad (3.6)$$

Hence H, U,and 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.

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

$$T = S - Z\left(Z^T GZ\right)^{-1} Z^T GS \qquad (3.7)$$

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

Refering to (3.7), H, T and U are given by:

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

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

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

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

$$S^{(K)^T} A_1^{(K)} = I \qquad (3.8) \text{ and}$$

$$Z^{(K)^T} A_1^{(K)} = 0 \qquad (3.9)$$

The choice of $S^{(K)}$ and $Z^{(K)}$ to satisfy (3.8) and (3.9) respectively is generally open. Here we take the choice given in :

$$S = Q_1 R^{-T} \quad , \quad Z = Q_2 \qquad (3.10)$$

$$K\left(Z^T GZ\right) \le K(G) \qquad (3.11)$$

which is, according to (3.10) and (3.110 is advantageous as far as stability is concerned.

For the sake of making this section self contained we show how $S^{(K)}$ and $Z^{(K)}$ are obtained in away suitable to this section.

Let :

$$Q^{(K)} A_1^{(K)} = \begin{bmatrix} R^{(k)} \\ 0 \end{bmatrix} \qquad (3.12)$$

represent the QR factorization of $A_1^{(K)}$, where $A_1^{(K)}$ is $n \times L_K$. Thus $Q^{(K)}$ is $n \times n$ and $R^{(K)}$ is an $L_K \times L_K$ upper triangular matrix partition $Q^{(K)}$ into

$$Q^{(K)} = \begin{bmatrix} Q_1^{(K)} \\ Q_2^{(K)} \end{bmatrix}$$

where $Q_1^{(K)}$ is $L_K \times n$ and $Q_2^{(K)}$ is $(n - L_K) \times n$. Thus, from (3.11) we have

$$Q_1^{(K)} A_1^{(K)} = R^{(K)} \qquad (3.13) \text{ and}$$

$$Q_2^{(K)} A_1^{(K)} = 0 \qquad (3.14)$$

so from (3.13) and (3.14) we define $S^{(K)}$ and $Z^{(K)}$ by:

$$S^{(K)} = Q_1^{(K)^T} R^{(K)^T} \qquad (3.15)$$

and $$Z^{(K)} = \tilde{I} Q_2^{(k)^T} \qquad (3.16)$$

Where $\tilde{I}$ is the identity matrix whose columns are reversed. (This choice is meant for simplifying the work) . (1.3.3).

Thus we conclude by saying that the computation is focussed on using the QR - factorization of $A_1^{(K)}$, (when the kth iteration is complementary). So updating these factors is required at each iteration when the tableau is complementary. This to be shown in [9].

4. Updating The QR-Factors Of $A_1^{(K)}$:-

In this section we show how to update the QR-factors of $A_1^{(K)}$, when the tableau is complementary. 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 $A_1^{(K)}$ are updated to give those of $A_1^{(K+1)}$, and this is the case when a column, $\underline{a}_q$ say, is deleted from $A_1^{(K)}$. In the second case the factors of $A_1^{(K)}$ are used to give those of $A_1^{(K+r+1)}$, and this is the case when one column, $\underline{a}_q$ say, is deleted from $A_1^{(K)}$ and then r other columns are added to $A_1^{(K)}$. We follow the same steps carried in [5] with the appropriate modification in the second case.

In the first case, let $A_1^{(K+1)}$ be the $n\times(L_K-1)$ matrix obtained by deleting the qth column, $\underline{a}_q$, from $A_1^{(K)}$. Suppose the QR-factorization of $A_{12}^{(K)}$ 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 $n\times(q-1)$ and $A_{12}^{(K)}$ is $n\times(L_K-q)$. 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 $(q-1)\times(q-1)$ upper triangular, R_{12} is $(q-1)\times(L_K-q)$, R_{22} is $(L_K-q)\times(L_K-q)$ 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 $Q_1^{\backslash}$ be the product of the plane rotations which gives:

$$Q_1^{\backslash}\begin{bmatrix}\underline{\beta}^T\\ R_{22}\end{bmatrix}=\begin{bmatrix}R^{\backslash}\\ \underline{0}^T\end{bmatrix},$$

where R' is $(L_K - q)\times(L_K - q)$ upper triangular. In this case $Q_1^{\backslash}$ is an $(L_K - q + 1)\times(L_K - q + 1)$ orthogonal matrix

Thus if

$$Q^{\backslash} = \begin{bmatrix} I & 0 & 0 \\ 0 & Q_1^{\backslash} & 0 \\ 0 & 0 & I \end{bmatrix} \begin{matrix} q - L \\ L_K - q + 1 \\ n - L_K \end{matrix}$$

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 $Q^{(K+1)} = Q^{\backslash '} Q^{(K)}$ and

$$R^{(K+1)} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R^{'} \end{bmatrix} \qquad (4.1)$$

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

$$Q^{(K+1)} = \begin{bmatrix} Q_1^{(K+1)} \\ Q_2^{(K+1)} \end{bmatrix} \begin{matrix} L_{K-1} \\ n - L_{K+1} \end{matrix} \qquad (4.2)$$

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

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

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 an other alternative of choosing q in (1.1), such an alternative is

$$q = \max \{ i = \lambda_i < 0 \, , 1 \le i \le L_K \}$$

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 $W^{'} = [\underline{a}_{pr},..,-\underline{a}_{-p1}]$. Let $A_1^{(K+1)}$ be obtained from $A_1^{(K)}$ by removing $\underline{a}_q$. Thus

$$A^{(K+r+1)} = [A^{(K+1)} : W^{'}] \qquad (4.3)$$

Premultiplying both sides of (4.3) by $Q^{(K+1)}$ (defined in (4.1) we get

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

where $W_1^{'} = Q_1^{(K+1)} W^{'}$

and $W_2^{'} = Q_2^{(K+1)} W^{'}$

and $R^{(K+1)}$ is defined in (4.1)

let

$$Q_2^{''} W_2^{'} = \begin{bmatrix} R_2^{'} \\ 0 \end{bmatrix}$$

Define the QR-factorization of $W_2^{'}$. Here $Q_2^{''}$ is $(\hbar - L_K + 1) \times (n - L_K + 1)$ and orthogonal, and $R_2^{'}$ is $r \times r$ upper triangular.

If

$$Q^{\backslash\backslash} = \begin{bmatrix} 1 & 0 \\ 0 & Q_2^{''} \end{bmatrix} \begin{matrix} L_K - 1 \\ n - L_K + 1 \end{matrix}$$

Then

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

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

$$Q^{(K+r+1)} = Q^{"}Q^{(K+1)} = Q^{"}Q^{'}Q^{(K)}$$

and

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

5. Updating The LDL^T-Factors Of $G_A^{(K)}$:-

The factors $L^{(K)}D^{(K)}L^{(K)^T}$ of $G_A^{(K)}$ are updated at each iteration when the tableau is complementary. Near the end of this subsection we show that $G_A^{(1)}$ is always positive definite (on the assumption that $G_A^{(K)}$ is positive semi-definite). Updating these factors is very stable when $G_A^{(K)}$ 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) iteration, we are unable till now to explore a way of using the factors of $G_A^{(K)}$ in obtaining those of $G_A^{(K+r+1)}$. However $n - L_K - r$, the dimension of $G_A^{(K+r+1)}$, decreases with r, in which case the effort of refactorizing $G_A^{(K+r+1)}$ might not be so much, especially when $n - L_K$ is itself small. This calls for choosing the starting L_1 so that $n - L_1$ is small. In the case when the number of constraints is greater than *n*, L_1 is chosen to be equal to n; that is the initial guess $\underline{x}^{(1)}$ is a vertex. With this choice $G_A^{(1)} = 0$, and in the second iteration we might expect a constraint to be deleted 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 $G_A^{(3)} = 0$. In the former case the dimension of $G_A^{(2)}$ is 1. In general the dimension of $G_A^{(K)}$ 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 $G_A^{(K)}$ keeps on decreasing when constraints are added to the active set, and in this case we are faced with refactorizing the factors.

We return to the case when the (K+1)th iteration is complementary . In this case, as (4.3) shows we get :

$$G_2^{(K+1)} = \begin{bmatrix} q^T \\ Q_2^{(K)} \end{bmatrix}$$

and using (3.16) we have

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

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

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

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:

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

If we substitute (4.1) and (4.2) 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

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

and

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

The numerical stability of this scheme is based on the fact that, if $G_A^{(K+1)}$ is positive definite, the element d_{n-L_K+1} must be positive. In this event (4.5) ensures that arbitrary growth in magnitude can not occur in the elements of $\underline{L}$.

Before ending this subsection we show that when the kth iteration and the (K+1)th iteration are complementary then $G_A^{(K+1)}$ must be positive definite.

Let the tableau be complementary at the kth iteration. Let $A_1^{(K)}$ be the matrix whose columns correspond to the active constraints, and $\lambda_q^{(K)} < 0$. The increase of v_q changes f according to

$$f = f^{(K)} + \lambda_q v_q - 0.5 u_{qq} v_q^2 \qquad (5.1)$$

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

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

and $\underline{x}$ changes according to

$$\underline{x} = \underline{x}^{(K)} + T \underline{e}_q v_q \qquad (5.2)$$

For the next tableau (i.e. the (K+1)th) to be complementary u_{qq} must be negative, and the new value

$v_q^{(K+1)}\left(\equiv \dfrac{\lambda_q^{(K)}}{u_{qq}}\right)$ of v_q must not violate feasibility.

Thus, using (5.1), 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 $Z^{(K+1)^T} A^{(K+1)} = 0$

.

Premultiply both sides of (5.2) by $A^{(K+1)^T}$ to get

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

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

$$T\underline{e}_q = Z^{(K+1)}\underline{h} \qquad (5.3)$$

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

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

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.

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

subject to $A^{(K+1)^T}\underline{\delta} = \underline{0}$ (5.12)

using (5.3) $\delta^{(K+1)} = Z^{(K+1)}\underline{h}\, v_q^{(K+1)} = Z^{(K+1)}\delta_A^{(K+1)}$ thus $\underline{\delta}a^{(K+1)}$ solves the problem

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

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

6. 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 (See[11]). The basis of the technique is to define an artificial objective function, namely:

$$\tilde{F}(\underline{x}) = -\sum_{j\in v(\underline{x})}\left(\underline{a}_j^T\underline{x} - b_j\right),$$

where $v(\underline{x})$ is the set of indices of constraints which are violated at the point $\underline{x}$, and to minimize this function with respect to $\underline{x}$, subject to the constraints $\underline{a}_j^T\underline{x} - b_j \geq 0$, $j \notin v(\underline{x})$. The function $\tilde{F}(\underline{x})$ is linear and is

known as the sum of infeasibilities. If a feasible point exists the solution $\underline{x}^*$ of the artificial problem is such that $\widetilde{F}(\underline{x}^*) = 0$.

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 $\widetilde{F}(\underline{x})$. This process will ultimately lead to a feasible vertex. 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 in [4,5]. In [2] the number of methods including the above one have been described.

Gill and Murray in[3,4,5] are advantageous in that it makes available the QR-factorization of the initial matrix of active constraints which is then directly used in our algorithm.

References

[1] David G. Luenberger Linear and nonlinear programming 2nd Edition. Pearson Education, Inc. Publishing as Addison-Wesley, (2003).

[2] Fletcher R. Practical Methods of Optimization, 2nd Ed. John Wiley and Sons , (1987).

[3] Gill. P. E. and Murray. W. A numerically stable form of the simplex Algorithm. J. Linear Algebra Applors 7, 99-138 , (1973).

[4] Gill. P. E. and Murray. W. Numerical Methods for constrained optimization (Eds. P. E. Gill and W. Murray), Academic press, London , (1975) .

[5] Gill. P. E. and Murray. W. Numerically Stable methods for quadratic programming. Math Prog 14, 349 – 372 , (1978).

[6] Gill. P. E. and Murray. W. Practical Optimization. Academic Press, (1988).

[7] Kunisch K, Rendl F. An infeasible active set method for quadratic problems with simple bounds.siam journal on optimization volume 14,number1 pp35-52 , (2003).

[8] Mohsin H. A. Hashim An Extension to the Dantzig-Wolfe Method for general quadratic programming Ph.D. thesis, University of Khartoum , (1996)

[9] Awatif .M .A Elsiddieg. The Davidon-Fletcher and Powel Method Tested on Quadratic Functions, M.Sc. thesis, University of Khartoum ,(1996).

[10] Fletcher R. and Leyffer, S. Nonlinear programming without a penalty function, Mathematical programming series A, 91, pp. 239-269. (2002).

[11] Dantzig Linear Programming and Extensions,(1963).
[12] Fletcher.R.The Calculation Of Linear Betle –P Approximation,(1971).

This academic article was published by The International Institute for Science, Technology and Education (IISTE). The IISTE is a pioneer in the Open Access Publishing service based in the U.S. and Europe. The aim of the institute is Accelerating Global Knowledge Sharing.

More information about the publisher can be found in the IISTE's homepage: http://www.iiste.org

CALL FOR PAPERS

The IISTE is currently hosting more than 30 peer-reviewed academic journals and collaborating with academic institutions around the world. There's no deadline for submission. **Prospective authors of IISTE journals can find the submission instruction on the following page:** http://www.iiste.org/Journals/

The IISTE editorial team promises to the review and publish all the qualified submissions in a **fast** manner. All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Printed version of the journals is also available upon request of readers and authors.

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar