Addis Ababa University (Since 1950)

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF NATURAL SCIENCES

DEPARTMENT OF MATHEMATICS

KHACHIYAN AND KARMARKAR POLYNOMIAL-TIME ALGORITHM

A SEMINAR PRESENTED TO THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN MATHEMATICS

BY: Minyahil Abera

Advisor: Prof. Dr.rer.nat.habil.R.Deumlich

June 2010

CONTENTS

ACKNOWLEDGEMENT

First of all, I thank to the almighty (Allah) for his endless grace, support and blessing on me during this seminar and all my life.

I am very grateful to Prof.Dr.rer.nat.habil.R.Deumlich, my Advisor and instructor, for his valuable comments, material assistance, teaching programming and suggestions that helped me much to improve and complete this seminar without limiting his valuable and golden time.

Next, I am very grateful to my wife Radiya Mohammed (Jaano) enormous amount of help for my successful studies besides perhaps more important still.

I want to express my deepest appreciation and thanks to Ato Solomon Soroto, Ato Yadeta Chimdesa, Ato Gashaw Shewangezaw, Ato Feleke Fekadu and Ato Sebhadin Sultan for their advice, suggestions and any other help they provide me.

Finally I am grateful to my father whose love, kindness, diligence made me learns and attend my education to this level.

1. PRELIMINARIES

Definition1.1: An $n \times n$ symmetric matrix B is said to be positive definite matrix if and only if

$$x^T Bx > 0 \ \ \forall x \in \mathbb{R}^n \setminus \{0\}.$$

Definition1.2: An ellipsoid with center $a \epsilon \mathbb{R}^n$ is a set

$$E := \{x \in \mathbb{R}^n : (x-a)^T B^{-1}(x-a) \leq 1\},$$

where B is $n \times n$ symmetric positive definite matrix.

Definition1.3: $\lambda \epsilon R$ is an eigenvalue of a matrix B if and only if there exist a vector $X \neq 0$ such that: $Bx = \lambda x$.

Definition1.4: A hyperplane in $\mathbb{R}^n$is a set having the form $\{x \in \mathbb{R}^n : ax = b\}$,

where $a \in \mathbb{R}^n \setminus \{0\}, b \in \mathbb{R}$.

Definition1.5: Let a line segment has end points P_1 *and* P_2 and elements of $\mathbb{R}^n$. The point (P) that divides the line segment in the ratio $r_1 : r_2$ has coordinate:

$$\left(\frac{r_2 x_{11} + r_1 x_{12}}{r_1 + r_2}, \frac{r_2 x_{21} + r_1 x_{22}}{r_1 + r_2}, \ldots \frac{r_2 x_{n1} + r_1 x_{n2}}{r_1 + r_2}\right)$$

Definition1.6: Let $P : x \to x$ be an endomorphism linear map. P is said to be a projection if and only if $P^2 = P$.

Definition1.7: A simplex in n-dimension is the convex hull of a set of $n+1$ non coplanar points in $\mathbb{R}^n$.

Theorem1.1: Let A be an $m \times n$ matrix, with $rank(A) = m$ and $H = \{x \in \mathbb{R}^n : Ax = 0\}$.The projection of p onto H is given by $q = (I - A^T(AA^T)^{-1}A)p$.

Linear Optimization problem

(P) $\quad c^T x \to max$

s. t. $Ax \leq b,$

$x \geq 0,$

where $A \in \mathbb{R}^{n\times m}, m, n \geq 2 \; and \; elements \; of \mathbb{N}, b \in \mathbb{R}^m$

Then we assign the of dual optimization problem by (D) and is given by:

(D) $b^T y \rightarrow min$

s.t. $A^T y \geq c,$

$y \geq 0,$

where $A^T \in \mathbb{R}^{m\times n}, m, n \geq 2, c \in \mathbb{R}^n.$

For the problem (P) and (D) we have:

Theorem1.2 :(Weak Duality Theorem)

If $S \; and \; S^*$ are feasible sets of the primal and dual problems respectively then

$c^T x \leq b^T y \quad \forall x \in S, \; \forall y \in S^*.$

Theorem1.3 :(Duality Theorem)

If $S \; and \; S^*$ are feasible sets of the primal and dual problems respectively the vectors $x_0 \in Sandy_0 \in S^*$ are solutions of (P) and (D), respectively if and only if

$c^T x_0 = b^T y_0.$

Theorem1.4: If $S \; and \; S^*$ are feasible sets of the primal and dual problems respectively the vectors $X_0 \in S \; and Y_0 \in S^*$ are solutions of (P) and (D), respectively, if and only if

$c^T x_0 \geq b^T y_0.$

Theorem1.5 :(Strong Duality Theorem)

If S *and* S^* are feasible sets to the primal and dual problems respectively and non empty then there are solution $x_0 \in S$ of(P)and$y_0 \in S^* of$(D) such that:

$$c^T x_0 = \max_{x \in S} c^T x = \min_{yS^*} b^T y = b^T y_0.$$

Theorem1.6: The vectors $x_0 \in \mathbb{R}^n$ *and*$y_0 \in \mathbb{R}^m$ are solutions of (P) and (D), respectively if and only if $x_0 \in \mathbb{R}^n$ *and*$y_0 \in \mathbb{R}^m$ are solutions of the system of inequalities:

$$Ax \leq b,$$

$$-A^T y \leq -c,$$

$$-c^T x + b^T y \leq 0,$$

$$-x \leq 0,$$

$$-y \leq 0.$$

2. INTRODUCTION

The development of simplex method for solving linear programming problems is a great achievement in the theory of optimization. An optimal solution of a linear programming problem always lies at a vertex of the feasible region which is a polyhedron .The simplex method moves along the edges of the polyhedron from one vertex to the next in an orderly fashion until it arrives at an optimal vertex .Since the number of vertices of the polyhedron associated with $m \times n$ coefficient matrix of the problem, is quite large for large values of m and n, there was apprehension that the simplex method not prove to be efficient. However, in practice, it was performed exceedingly well. It was observed that for most practical problems, the number of iterations the methods (simplex) needs only between m and $3m$. However, the fact remains that for certain problems the simplex method may require to examine all the vertices to arrive at an optimal and therefore the number of iterations can grow exponentially. An example given by Klee and ty[5] indeed shows that in the worst case, the simplex method needs 2^n iterations to find an optimal solution.

The search for methods with better complexity (algorithm efficiency) than the simplex method led to the development of polynomial-time algorithm, that is, where the computation time is bounded above by a polynomial in the size or the total data length of the problem. In the next two chapters we discuss in detail two polynomial-time algorithm namely Khachiyan and Karmarkar polynomial-time algorithm.

3. KHACHIYAN'S ELLIPSOID METHOD

3.1. Introduction

The first polynomial-time algorithm for linear programming was given by L.G. Khachiyan in1979.Khachiyan ellipsoid method approximates the optimum solution of linear programming problems by creating a sequence of ellipsoid that approaches the optimal solution.

3.2. Khachiyan's Ellipsoid Method

Khachiyan proposed a polynomial-time algorithm for determining a solution (if it exists) to the open set of linear inequalities $S = \{x \in \mathbb{R}^n : Ax \leq b\}$,where $A\ is\ m \times n, m, n \geq 2$ and where $A\ and\ b$ have all integer components. This algorithm either finds a solution in S, if one exists, or else concludes that S has no solution. There are several variants of Khachiyan's algorithm; we follow the interpretation given by Gacs and Lovask[2].

Consider the problem of determining a solution to the set of linear inequalities

$$S = \{x \in \mathbb{R}^n : Ax \leq b\}, \tag{3.1}$$

where A is an $m \times n$ matrix$b \in \mathbb{R}^m$; $m, n \geq 2$ and the data are all integers.

Assuming that S is non empty the algorithm starts by constructing an appropriate ball $E_o = S[0, r]$ centered at the origin, where r is so large that is $S[0, r]$ contains a certain part of S. If the center of the ball lies in S then the algorithm terminates. Otherwise, a sequence of ellipsoids $E_1, E_2, E_3, \ldots$. in decreasing volume are constructed each of which containing the region of S covered by E_0. Let at k iteration $(k = 0,1,2,3, \cdots)$ with center x_K if x_K lies in S the algorithm terminates. If not, some constraints are violated (i.e. there exist $a_i and\ b_i$such that $a_i x_k > b_i$ where $a_i$is the violated row).Let the most violated constraint (the largest distance between x_k and the hyperplane $a_v x = b_v$) be a_v such that:

$$a_v x_k > b_v . \tag{3.2}$$

A new ellipsoid E_{k+1} of smaller volume is then constructed which contains that region of S which was covered by E_k .

If the center of E_{k+1} lies in S the algorithm terminates and the center of ellipsoid E_{k+1} is the solution. Otherwise the above steps are repeated. Khachiyan has shown that if S is non empty then by $6(n+1)^2L$ iterations the algorithm will find a solution in S.

Hence, if no solution is found in $6(n+1)^2L$ iteration then S has no solution.

Here by L is the number of bits required to store the problem data in the computer and known as the input length of an instant of the problem and is given by:

$$L = \lceil 1 + \log_2 m + \log_2 n + \sum_i \sum_j (1 + \log_2(1 + |a_{ij}|)) + \sum_j (1 + \log_2(1 + |c_j|)) + \sum_i (1 + \log_2(1 + |b_i|)) \rceil,$$

where $\lceil . \rceil$ denotes the rounded up value.

3.3. Khachiyan's Algorithm

Starting with $\boldsymbol{k = 0}$, *suppose that at iteration k the ellipsoid* $\boldsymbol{E_k}$ *is given by:*

$$E_k = \{x \in \mathbb{R}^n : (x - x_k)^T B_K^{-1} (x - x_k) \leq 1\}, \tag{3.3}$$

where $x_k \in \mathbb{R}^n$ is the center and B_k is an $n \times n$ symmetric positive definite matrix.

To construct E_{k+1} see the fig 3.3.1.

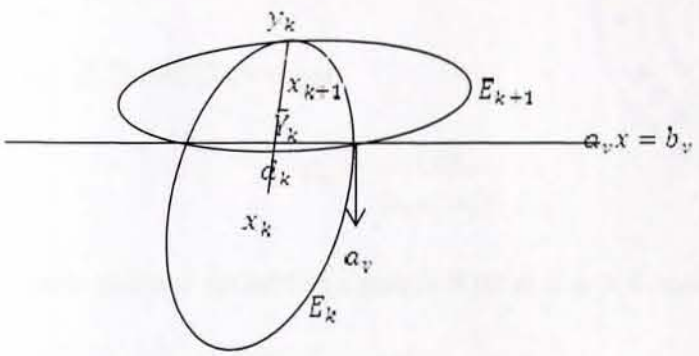

fig 3.3.1

We assume that the center of E_k does not lie on S .Then there exist a_v and b_v such that: $a_v x_k > b_v$ .Then geometrically translates the hyperplane $a_v x = b_v$ parallel to itself in the (feasible) direction of $-a_v^T$ until it becomes tangential to the ellipsoid E_k at the point y_k.

Lemma3.2.1: Let $y_k = x_k + d_k$ then $d_k = \frac{-B_k a_v^T}{(a_v B_k a_v^T)^{\frac{1}{2}}}$. (3.4)

Proof:

Since y_k is a boundary point of E_k from this we get:

$$(y_k - x_k)^T B_k^{-1}(y_k - x_k) = 1 \text{ or } d_k^\tau B_k^{-1} d_k = 1. \tag{3.5}$$

Hence d_k *and* $-a_v^\tau$ are parallel there exist $\rho > 0$ such that:

$$-\rho a_v^T = d_k. \tag{3.6}$$

From equation (3.6) and (3.5), we get:

$$(-\rho a_v^T)^T B_k^{-1}(-\rho a_v^T) = 1 \text{ or}$$

$$\rho^2 a_v B_k^{-1} a_v^T = 1 \text{ or}$$

$$\rho = \frac{1}{(a_v B_k^{-1} a_v^T)^{\frac{1}{2}}}. \tag{3.7}$$

From equation (3.7) and (3.6) we get:

$$d_k = \frac{-a_v^T}{(a_v B_k^{-1} a_v^T)^{\frac{1}{2}}}. \tag{3.8}$$

From symmetric positive definite of a matrix B there is $\mu > 0$ such that

$$B_k a_v^T = \mu a_v^T.$$

Therefore $a_v^T = \frac{B_k a_v^T}{\mu}$. (3.9)

Combining equation (3.9) and (3.8) we get that:

$$d_k = \frac{\frac{-B_k a_v^T}{\mu}}{\left(\left(\frac{B_k a_v^T}{\mu}\right)^T B_k^{-1} \frac{B_k a_v^T}{\mu}\right)^{\frac{1}{2}}} = \frac{-B_k a_v^T}{\frac{\mu\left(a_v B_k B_k^{-1} B_k a_v^T\right)^{\frac{1}{2}}}{\mu}} = \frac{-B_k a_v^T}{\left(a_v B_k a_v^T\right)^{\frac{1}{2}}}.$$

Now let $\bar{y}_k$ be the point of the intersection of the hyperplane $a_v x = b_v$ and $[x_k, y_k]$.

Lemma3.2.2: Let $\bar{y}_k = x_k + \lambda_k d_k$ then

i). $\lambda_k = \frac{a_v x_k - b_v}{(a_v B_k a_v^T)^{\frac{1}{2}}}$

ii). $\lambda_k \epsilon (0, 1]$.

Proof:

i).Since $\bar{y}_k$ lies in the hyper plane $a_v x = b_v$ and $\bar{y}_k = x_k + \lambda_k d_k$.

We have that:

$$a_v \bar{y}_k = b_v \text{ or } a_v(x_k + \lambda_k d_k) = b_v \quad \text{or} \quad a_v x_k + \lambda_k a_v d_k = b_v \text{ or}$$

$$\lambda_k a_v d_k = b_v - a_v x_k \text{ or } \frac{\lambda_k a_v(-B_k a_v^T)}{(a_v B_k a_v^T)^{\frac{1}{2}}} = b_v - a_v x_k \text{ (since } d_k = \frac{-B_k a_v^T}{(a_v B_k a_v^T)^{\frac{1}{2}}})$$

$$\text{or } -\lambda_k \frac{(a_v B_k a_v^T)}{(a_v B_k a_v^T)^{\frac{1}{2}}} = b_v - a_v x_k \text{ or } \lambda_k (a_v B_k a_v^T)^{\frac{1}{2}} = a_v x_k - b_v \text{or}$$

Thus $\lambda_k = \frac{a_v x_k - b_v}{(a_v B_k a_v^T)^{\frac{1}{2}}}$. (3.10)

ii).Since $\lambda_k = \frac{a_v x_k - b_v}{(a_v B_k a_v^T)^{\frac{1}{2}}}$ and $a_v x_k > b_v$

Therefore $\lambda_k = \frac{a_v x_k - b_v}{(a_v B_k a_v^T)^{\frac{1}{2}}} > 0$.

Next let us show $\lambda_k \leq 1$.

Suppose $\lambda_k > 1$.

From this we get:

$$\bar{y}_k = x_k + \lambda_k d_k > x_k + d_k = y_k.$$

Thus, $\bar{y}_k$ is not in the intersection of $a_v x - b_v$ *and* $[x_k, y_k]$ this is a contradiction.

Therefore $\lambda_k \leq 1$

Therefore $\lambda_k \epsilon (0, 1]$

Let x_{k+1} be the point on the line segment $[\bar{y}_k, y_k]$ that divides this in the ratio $1:n$.

Hence by definition (1.6) x_{k+1} given by:

$$x_{k+1} = \frac{n\bar{y}_k}{1+n} + \frac{y_k}{1+n} = \frac{n}{n+1}(x_k + \lambda_k d_k) + \frac{1}{n+1}(x_k + d_k)$$

$$= \frac{n}{n+1}x_k + \frac{n}{n+1}\lambda_k d_k + \frac{1}{n+1}x_k + \frac{1}{n+1}d_k = x_k + (1 + n\lambda_k)\frac{d_k}{n+1}$$

$$= x_k - \left(\frac{1+n\lambda_k}{n+1}\right)\frac{(B_k a_v^T)}{(a_v B_k a_v^T)^{\frac{1}{2}}}. \qquad (3.11)$$

Before we define the new ellipsoid E_{k+1} let us state one of the most important theorem without proof.

Theorem3.2.1: Let $B_{k+1} := \delta_k \left(B_k - \alpha_k \frac{(B_k a_v^T)(a_v B_k)}{(a_v B_k a_v^T)}\right)$, where $\delta_k = \frac{n^2}{n^2-1}(1 - \lambda_k^2)$

$\alpha_k = \frac{2(1+n\lambda_k)}{(n+1)(1+\lambda_k)}$ *and* $\lambda_k = \frac{a_v x_k - b_v}{(a_v B_k a_v^T)^{\frac{1}{2}}}$ then B_{k+1} is a symmetric positive definite matrix.

Now the ellipsoid E_{k+1} defined as $E_{k+1} := \{x \in \mathbb{R}^n : (x - x_{k+1})^T B_{k+1}^{-1}(x - x_{k+1}) \leq 1\}$, where B_{k+1}^{-1} $n \times n$ symmetric positive definite matrix.

3.4. Solving Linear Programming Problems by the Ellipsoid Method

Let us now see how Khachiyan algorithm can be used to solve a linear programming problem.

$$
\begin{aligned}
(P) \quad & c^T x \to min \\
\text{s.t.} \quad & Ax \geq b\,, \\
& x \geq 0.
\end{aligned}
$$

There are several ways to apply the ellipsoid method to solve the above problem. We here discuss the approach where we apply the ellipsoid method to a certain system of linear inequalities that yields an optimal solution of the linear programming problem.

Consider the above linear programming problem that is given in the canonical form.

$$
\begin{aligned}
(P) \quad & c^T x \to min \\
\text{s.t.} \quad & Ax \geq b\,, \\
& x \geq 0.
\end{aligned}
$$

Then the dual linear program is defined by:

$$
\begin{aligned}
(D) \quad & b^T w \to max \\
\text{s.t.} \quad & A^T w \geq c\,, \\
& w \geq 0.
\end{aligned} \tag{3.12}
$$

By duality theory in linear programming $x_0\, and\, y_0$ are optimal solutions to $(P)$and(D) respectively, if and only if $x_0\, and y_0$ solve the system of linear inequalities:

$$
\begin{aligned}
-Ax &\leq -b, \\
A^T w &\leq c, \\
-c^T x + b^T w &\leq 0, \\
-x &\leq 0, \\
-w &\leq 0.
\end{aligned} \tag{3.13}
$$

Now (3.12) is in the form of (3.1) and we can apply ellipsoid method to (3.12) and it produces optimal solutions to the primal and dual problems simultaneously.

Example: Find the solution of the following linear program by using ellipsoid method.

$$
\begin{aligned}
& -x_1 - 2x_2 \to min \\
\text{s.t.} \quad & -x_1 - x_2 \leq -1,
\end{aligned}
$$

$$-x_1 + x_2 \leq 2,$$

$$x_1 + x_2 \leq 4,$$

$$x_1, x_2 \geq 0.$$

To start Kachiyan algorithm draw the ellipse E_0 centered at the origin with x-intercept± 5 and y-intercept $\pm 5\sqrt{3}$. The center of E_0 *is* $x_0 = (0,0)$ not lies in S .Hence there exist violated row and the most violated constraint is $a_v = (-1,-1)$ and $b_v = -1$.

Now $(5,0)^T$, $\left(0,5\sqrt{3}\right)^T$ *and* $\left(\frac{5}{2}, \frac{15}{2}\right)^T$ are the boundaries of E_0 .

Let $B_0^{-1} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$.

Thus we have that:

$$(5,0)\begin{bmatrix} a & b \\ b & c \end{bmatrix}\begin{pmatrix} 5 \\ 0 \end{pmatrix} = 1,\ (0,5\sqrt{3})\begin{bmatrix} a & b \\ b & c \end{bmatrix}\begin{pmatrix} 0 \\ 5\sqrt{3} \end{pmatrix} = 1 \text{ and } \left(\frac{5}{2}, \frac{15}{2}\right)\begin{bmatrix} a & b \\ b & c \end{bmatrix}\begin{pmatrix} \frac{5}{2} \\ \frac{15}{2} \end{pmatrix} = 1.$$

From above equations we have that:

$$a = \frac{1}{25}, b = 0 and c = \frac{1}{75}.$$

Therefore $B_0^{-1} = \begin{bmatrix} \frac{1}{25} & 0 \\ 0 & \frac{1}{75} \end{bmatrix}\begin{pmatrix} \frac{1}{25} & 0 \\ 0 & \frac{1}{75} \end{pmatrix}$ and $B_0 = \begin{bmatrix} 25 & 0 \\ 0 & 75 \end{bmatrix}$.

Now let us find $Y_0, \bar{Y}_0 and\ X_1$.

$$Y_0 = X_0 + d_0$$

Where $X_0 = (0,0)^T$ *and* $d_0 = \frac{-B_0 a_v^T}{(a_v B_0 a_v^T)^{\frac{1}{2}}}$.

$$d_0 = \frac{-\begin{bmatrix} 25 & 0 \\ 0 & 75 \end{bmatrix}\begin{pmatrix} -1 \\ -1 \end{pmatrix}}{\sqrt{(-1,-1)\begin{bmatrix} 25 & 0 \\ 0 & 75 \end{bmatrix}\begin{pmatrix} -1 \\ -1 \end{pmatrix}}} = \frac{\begin{pmatrix} 25 \\ 75 \end{pmatrix}}{10} = \begin{pmatrix} \frac{5}{2} \\ \frac{15}{2} \end{pmatrix}.$$

Therefore $Y_0 = \begin{pmatrix} \frac{5}{2} \\ \frac{15}{2} \end{pmatrix}$.

$\bar{Y}_0 = X_0 + \lambda_0 d_0 = \lambda_0 \begin{pmatrix} \frac{5}{2} \\ \frac{15}{2} \end{pmatrix}$.

But $\lambda_0 = \frac{a_v x_0 - b_v}{(a_v B_v a_v^T)^{\frac{1}{2}}} = \frac{-b_v}{(a_v B_v a_v^T)^{\frac{1}{2}}} = \frac{1}{10}$.

Therefore $\bar{Y} = \begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix}$.

$$X_1 = \frac{2}{3}\begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix} + \frac{1}{3}\begin{pmatrix} \frac{5}{2} \\ \frac{15}{2} \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

Since x_1 lies in S therefore $\begin{pmatrix} 1 \\ 3 \end{pmatrix}$ is optimal solution of (P) and -7 is optimal value.

3.5. Description of Khachiyan's Algorithm

Step1:Start $k = 0$ and center at the origin. At iteration k x_k is the centerE_k if $x_k \in S$ then halt otherwise there exist a_v *and* b_v.

Such that:$a_v x > b_v$.

Step2: Compute $d_k = \frac{-B_k a_v^T}{(a_v B_k a_v^T)^{\frac{1}{2}}}$ and $\lambda_k = \frac{a_v x_k - b_v}{(a_v B_k a_v^T)^{\frac{1}{2}}}$.

Step3:Compute $y_k = x_k + d_k$and $\bar{y}_k = x_k + \lambda_k d_k$.

Step4:Compute $x_{k+1} = \frac{n}{n+1}\bar{y}_k + \frac{1}{n+1} y_k$ and $B_{k+1} = \delta_k \left(B_k - \alpha_k \frac{(B_k a_v^T)(a_v B_k)}{(a_v B_k a_v^T)}\right)$.

Step5:$E_{k+1} := \{x \in \mathbb{R}^n : (x - x_{k+1})^T B_{k+1}^{-1}(x - x_{k+1}) \leq 1\}$. If $x_{k+1} \in S$ then

stop otherwise go to 1.

4. KARMAKAR'S POLYNOMIAL-TIME ALGORITHM

4.1. Introduction

In1984 Karmarkar proposed a new polynomial-time algorithm for linear programming problems based on repeated projective transformation. This algorithm creates a series of interior points converging to optimal solution. A lot of variation of Karmarkar's algorithm has been made however in all variation Karmarkar algorithms the major work is repeated computation of the inverse of matrix and computation of the projection of gradient vector.

Karmarkar's consider the linear programming problem in the form:

$$\begin{aligned} & c^T x \to min \\ & \text{s.t.}\, Ax = 0, \\ & e^T x = 1, \\ & x \geq 0, \end{aligned} \tag{4.1}$$

where $x \in \mathbb{R}^n, e = (1,1,\cdots,1)^T \in \mathbb{R}^n$, A *is an* $m \times n$ matrix with rank $(A) = m, n \geq 2$ and the data are all integers and $c \in \mathbb{R}^n$ and its elements are integers with the following two assumptions hold.

1. The point $x_0 = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})^T \in \mathbb{R}^n$ is feasible in (4.1).

2. Optimal value of the objective functions in (4.1) is zero.

At first glance the form of linear programming (4.1) and the accompanying assumptions (1) and (2) may appear to be exclusively restrictive. However, as shown later, any general linear programming problem can be easily converted to this form.

4. 2. Karmarkar Algorithm

Let as proceed with the derivation of a polynomial-time algorithm to solve problem (4.1) with the assumption 1 and 2.

Starting $k = 0$ with feasible point $x_0 = \left(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\right)^T$ represents the center of an $(n-1)$ - dimensional simplex $\Lambda_x = \{x \in \mathbb{R}^n : e^T x = 1, x \geq 0\}$ the algorithms generates a sequence of feasible points which converges to the optimal solution in polynomial-time. Let at step k, x_k be any feasible solution and $c^T x_k \neq 0$.The current feasible point x_k will no longer be at the center of the simplex. For the procedures to be iterative, we use a projective transformation to bring this feasible point to the center of the simplex in transformed space.

Karmarkar uses the projective transformation:

$$y := \frac{D_k^{-1} x}{e^T D_k^{-1} x}, \tag{4.2a}$$

where D_k is the diagonal matrix i.e. $D_k = diag\{x_{k1}, \cdots, x_{kn}\}$ and X is an element of the feasible set.

The equivalent transformation of (4.2a) is:

$$y_i := \frac{\frac{x_i}{x_{ki}}}{\sum_{j=1}^{n} \frac{x_j}{x_{kj}}}. \tag{4.2b}$$

Under the transformation (4.2b) any point in the simplex $\Lambda_x = \{x \in \mathbb{R}^n : e^T X = 1, X \geq 0\}$

is transformed in to a point in the simplex $\Lambda_y = \{y \in \mathbb{R}^n : e^T y = 1, y \geq 0\}$. In particular, the current feasible point x_k is transformed into the center of the simplex Λ_y.

To algebraically describe this feasible region in the Y-space consider the inverse projective transformation obtained by solving for x in Λ_x from the transformation (4.2a) we have that

$$y = \frac{D_k^{-1}x}{e^T D_k^{-1}x}.$$

Thus we get $D_k y = \frac{x}{e^T D_k^{-1}x}$ or $x = (D_k y)(e^T D_k^{-1}x)$, moreover $e^T D_k y = \frac{e^T x}{e^T D_k^{-1}x} = \frac{1}{e^T D_k^{-1}x}$.

Therefore $x = \frac{D_k y}{e^T D_k y}$. (4.3)

Therefore, the transformation (4.2) and (4.3) maps points from the simplex Λ_x on to the simplex Λ_y and vice versa.

By transformation (4.3) the problem (4.1) is transformed into the following problem in the

Y-space:

$$\frac{c^T D_k y}{e^T D_k y} \to min$$

$$\text{s.t. } AD_k y = 0, \qquad (4.4)$$
$$e^T y = 1,$$

$$y \geq 0,\ y \in \mathbb{R}^m.$$

The constraint remain linear because $Ax = 0$ is homogenous, the objective function has become a quotient of linear functions. However, by assumption (2) the optimal objective value (4.4) is zero. Hence we can equivalently minimize the numerator in the problem, since the $e^T D_k y$ is positive and bounded away from zero for all $y \epsilon \Lambda_y$.

This leads us to the following problem:

$$\bar{c}^T y \to min$$

$$\text{s.t. } Py = P_0, \qquad (4.5)$$

$$y \geq 0,$$

where $y \in \mathbb{R}^n$, $\bar{c}^T = c^T D_k$, $P = \begin{bmatrix} AD_k \\ e^T \end{bmatrix}$, $P_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Rather than solve problem (4.5) which is equivalent to solving the original problem, we will optimize a particular simpler restriction at this problem. By iteratively repeating the step we will be able to polynomially obtain an optimal solution to the original problem. To define this restriction, consider the n −dimensional sphere or ball $B(y_0, r)$ with center at $y_0 = \left(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\right)^T$ and with appropriate radius r such that the intersection of this ball with the simplex $\Lambda_y = \{y \in \mathbb{R}^n : e^T y = 1, y \geq 0\}$ is an $(n-1)$ −dimensional ball with the same center and radius which is inscribed with in Λ_y.The radius r is the distance from the center $(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})^T$of the simplex to the center of its facets. But the facets of Λ_y are simply one lower dimensional simplices. Hence considering the facet of which the first component or y_1 is zero, for example, we obtain the distance from $(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})^T$to $(0, \frac{1}{n-1}, \frac{1}{n-1}, \cdots, \frac{1}{n-1})^T$.

This gives $r = \sqrt{\left(\frac{1}{n}\right)^2 + \left(\frac{1}{n-1} - \frac{1}{n}\right)^2 + \ldots + \left(\frac{1}{n-1} - \frac{1}{n}\right)^2} = \frac{1}{\sqrt{n(n-1)}}$. (4.6)

In the fig 4.2.1 we have that the simplex with dimension two with inscribed ball of dimension 3.

Consider a restriction of equation (4.5) in which one seeks to:

$$\bar{c}^T y \rightarrow min$$

$$\text{s.t.} \quad Py = P_0,$$

where $y \geq 0, y \in B(y_0, \alpha r)$, $0 < \alpha < 1$ is some constant.

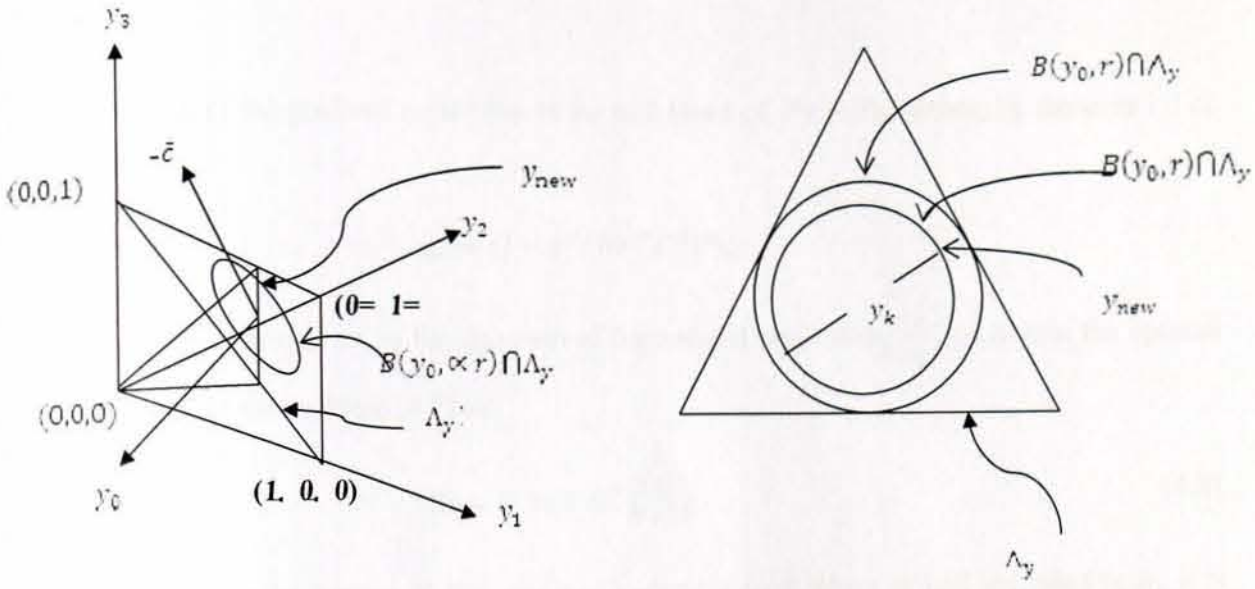

fig.4.2.1

Thus we have shrunk the inscribed ball by α and so its radius αr. Moreover, by remaining feasible to this restriction, the iterated remain strictly positive since $y \geq 0$ is implied by the intersection of the ball and the simplex constraint $\Lambda_y = \{y \in \mathbb{R}^n : e^T y = 1, y \geq 0\}$ this restriction is equivalent to the form:

$$\bar{c}^T y \to min$$

$$\text{S.t. } Py = P_0, \tag{4.7}$$

$$(y - y_0)^T (y - y_0) \leq \alpha^2 r^2.$$

where $\{y \in \mathbb{R}^n : (y - y_0)^T (y - y_0) \leq \alpha^2 r^2\}$, describes the ball $B(y_0, \alpha r)$.

Fig (4.2.1) shows the feasible region over the two dimensional simplex. Since the rank of the system $Py = P_0$ is $(m + 1)$ it defines an $(n - m - 1)$-dimensional affine subspace that passes through the center of the ball $B(y_o, \alpha r)$. Therefore, the feasible region in problem (4.7) is an $(n - m - 1)$-dimensional region.

So that, the optimal solution to problem (4.7) is obtained by simply projecting the negative gradient of the objective function$(-\bar{c})$ centered at y_0 on to the null space of the constraint surface of $Py = P_0$ and moving from y_0 along this projected direction to the boundary of the ball.Let c_p

the projection of the gradient vector $\bar{c}$on to the null space of $Py = P_0$. Hence, by theorem 1.1 c_P is given by:

$$c_P := (I - P^T(PP^T)^{-1}P)\bar{c}.$$

Taking a step of length αr in the direction of normalized projection$\frac{-c_P}{\|c_P\|}$ we obtain the optimal solution y_{new} of the problem (4.7) as:

$$y_{new} = y_0 - \alpha r\frac{c_P}{\|c_P\|}, \tag{4.8}$$

since y_{new} lies in the interior of the an $(n-1)$-dimensional sphere or ball inscribed in Λ_y it is greater than zero. Thus, the corresponding revised vector x_{k+1} in the X-space which is obtained by via the inverse transformation $\left(\frac{D_k y_{new}}{e^T D_k y_{new}}\right)$ is greater than zero. This completes one iteration. The step may be repeated after incrementing k by one. It can be shown in each step there is a reduction in the value of the objective function (4.7) and the process approaches the optimal value zero.

Example1: Perform two iterations of Karmarkar's algorithm on the following problem.

$$x_2 \to min$$

$$\text{S.t. } x_1 + x_2 - 2x_3 = 0,$$

$$x_1 + x_2 + x_3 = 1,$$

$$x_i \geq 0, i = 1,2,3.$$

Here$A = [1,1,-2]$,$c = (0,1,0)$,$e^T = (1,1,1)$, n=3, m=1 and the point $x_0 = \left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)^T = y_0$is feasible .We also have $r = \frac{1}{\sqrt{6}}$ and $\alpha = \frac{2}{9}$.

Iteration1

Starting with x_0 and k=0, we define $D_0 = diag\left\{\frac{1}{3},\frac{1}{3},\frac{1}{3}\right\}$.We now compute $\bar{c}^T = c^T D_k =$ $\left(0,\frac{1}{3},0\right)$, $P = \begin{bmatrix} AD_k \\ e^T \end{bmatrix} = \begin{bmatrix} \frac{1}{3},\frac{1}{3},\frac{1}{3} \\ 1,1,1 \end{bmatrix}$, $PP^T = \begin{bmatrix} \frac{2}{3} & 0 \\ 0 & 3 \end{bmatrix}$, $c_p = (I - P^T(PP^T)^{-1}P)\bar{c} = \begin{bmatrix} -\frac{1}{6} \\ \frac{1}{6} \\ 0 \end{bmatrix}$, and $\|c_p\| =$ $\frac{\sqrt{2}}{6}$.

Using (4.8), this gives us: $y_{new} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} - \frac{2}{9}\frac{1}{\sqrt{6}}\begin{bmatrix} -\frac{1}{6} \\ \frac{1}{6} \\ 0 \end{bmatrix}\frac{\sqrt{2}}{6} = \begin{bmatrix} 0.397484 \\ 0.269183 \\ 0.333333 \end{bmatrix}$. Transforming into the X-space, we obtain $x_1 = y_{new}$ for this iteration $c^T x_1 = 0.269183$.

Iteration2

Starting with x_1 and k=1,we define $D_1 = diag\{0.397484,0.269183,0.333333\}$.We now compute

$$\bar{c}^T = c^T D_k = \left(0,\frac{1}{3},0\right), P = \begin{bmatrix} AD_k \\ e^T \end{bmatrix} = \begin{bmatrix} 0.397484 & 0.269183 & -0.666666 \\ 1 & 1 & 1 \end{bmatrix},$$

$$c_p = (I - P^T(PP^T)^{-1}P)\bar{c} = \begin{bmatrix} -0.1324029 \\ 0.1505548 \\ -0.0181517 \end{bmatrix}, \text{and} \|c_p\| = 0.2013121 .$$

Using (4.8), this gives us $\boldsymbol{y_{new}} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} - \frac{2}{9}\frac{1}{\sqrt{6}} \frac{\begin{bmatrix} -0.1324029 \\ 0.1505548 \\ -0.0181517 \end{bmatrix}}{0.2013121} = \begin{bmatrix} 0.3930009 \\ 0.2654855 \\ 0.3415134 \end{bmatrix}$.

Transforming into the X-space, we obtain $\boldsymbol{x_2} = \begin{bmatrix} 0.457409 \\ 0.209258 \\ 0.333333 \end{bmatrix}$ for this iteration $\boldsymbol{c^T x_2} = 0.209258$.

4.3. Convergence and complexity of Karmakar's Polynomial-Time Algorithm

In this section we deal with a convergence and complexity of Karmakar's algorithm applied to problem (4.1) under assumption 1 and 2

In order to prove the convergence of the algorithm and established its polynomial complexity there are two constructs that we employ. The first is the consideration of a relaxation of problem (4.5) which complements its restriction (4.7).The second is the use of a novel function known as the potential function that measures the progress of the algorithm in a very clever fashion and helps establish its polynomial complexity. To construct the required relaxation of problem (4.5) consider the n-dimensional ball $B(Y_0, R)$ in the $Y - space$ that has center at $Y_0 = \left(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\right)^T \in \mathbb{R}^n$coinciding with that of the simplex Λ_y and that has radius R such that $B(Y_0, R)$ intersected with $\Lambda_y = \{Y \in \mathbb{R}^n : e^T Y = 1, Y \geq 0\}$ is an $(n-1)$-dimensional ball that circumscribes Λ_y. Hence R is the distance from the point $(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})^T$ to any vertex Λ_y say $(1, 0, \cdots, 0)^T$. Thus

$$R = \sqrt{\left(1 - \tfrac{1}{n}\right)^2 + \left(0 - \tfrac{1}{n}\right)^2 + \ldots\ldots \left(0 - \tfrac{1}{n}\right)^2} = \sqrt{\tfrac{(n-1)^2}{n^2} + \tfrac{n-1}{n^2}} = \sqrt{\tfrac{n-1}{n}} \qquad (4.9)$$

Now we consider the problem obtained by adding the redundant constraint $Y \epsilon B(Y_0, R)$ in problem (4.5) but relaxing (deleting) the non-negativity restrictions.

$$\bar{c}^T Y \to min$$

$$\text{S.t.} \quad PY = P_0 \ , \tag{4.10}$$

$$(Y - Y_0)^\tau (Y - Y_0) \leq R^2.$$

The feasible region of (4.10) is an $(n - m - 1)$ -dimensional ball centered atY_0, defined by the intersection of the $(n - m - 1)$-dimensional affine subspace$\{Y \in \mathbb{R}^n : PY = P_0\}$ with $B(Y_0, R)$.

Fig (4.3.1) shows this intersection for a case $n = 3$ and $m = 1$.

Hence, the optimal solution $\bar{Y}_{new}$ to problem (4.10) is obtained in a similar fashion to equation (4.8) and is given by:

$$\bar{Y}_{new} := Y_0 - R\frac{C_P}{\|C_P\|} \tag{4.11}$$

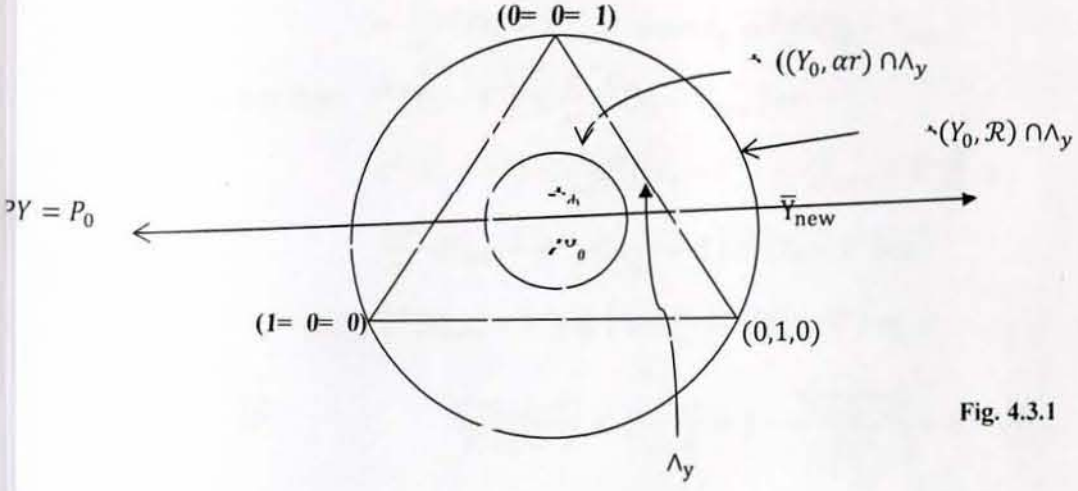

Fig. 4.3.1

Now we can estimate for the progress made with respect to the objective function in the Y-space as follows.

Firstly let's show that $\bar{c}^T Y_{new} \leq \bar{c}^T Y_0$.

Using equation (4.8) we get:

$$\bar{c}^T Y_{new} = \bar{c}^T Y_0 - \alpha r c^T \frac{C_P}{\|C_P\|} \tag{Δ}$$

But $c_p = \bar{c} - P^\tau (PP^\tau)^{-1} P\bar{c}$ or $c_p + P^T (PP^T)^{-1} P\bar{c} = \bar{c}$

$$\bar{c}^T = c_p^T + c^T P^T (PP^T)^{-1} P \,. \qquad (\Delta\Delta)$$

Thus inserting $(\Delta\Delta)$in (Δ) we have that:

$$\bar{c}^T Y_{new} = c^T Y_0 - \frac{\alpha r}{\|c_P\|}\left(\bar{c}_p^T c_p + c^T P^T (PP^T)^{-1} P c_p\right) \text{ since } Pc_P = 0.$$

$$c^T Y_0 - \bar{c}^T Y_{new} = \frac{\alpha r}{\|c_P\|}\|c_P\|^2 = \alpha r\|c_P\| \geq 0 \quad \text{since} \quad 0 < \alpha < 1, r \geq 0\,, \|c_P\| \geq 0.$$

Therefore $\bar{c}^T Y_{new} \geq c^T Y_0$.

Secondly let's denote Y^* as an optimal solution to problem (4.5) since problem (4.7) solved by Y_{new} is a restriction of the problem and problem (4.10) solved by $\bar{Y}_{new}$ is a relaxation of the problem. We have that:

$$\bar{c}^T \bar{Y}_{new} \leq \bar{c}^T Y^* \leq \bar{c}^T Y_{new} \leq \bar{c}^T Y_0$$

Moreover, using this along with equation (4.11) and we get that:

$$0 \leq \bar{c}^T (Y_0 - Y_{new}) \leq \bar{c}^T (Y_0 - Y^*) \leq \bar{c}^T (Y_0 - \bar{Y}_{new})$$

$$= R\frac{\bar{c}^T c_p}{\|c_P\|} = \frac{R}{\|c_P\|}\bar{c}^T (Y_0 - Y_{new})\frac{\|c_P\|}{\alpha r}$$

$$= \frac{R}{\alpha r}\bar{c}^T (Y_0 - Y_{new}) \quad \text{since } c_p = \frac{\|c_P\|}{\alpha r}(Y_0 - Y_{new}).$$

Hence this asserts that: $\bar{c}^T (Y_0 - Y^*) \leq \frac{R}{\alpha r}\bar{c}^T (Y_0 - Y_{new})$ or

$$\bar{c}^T (Y_0 - Y^*) \leq \frac{R}{\alpha r}\bar{c}^T ((Y_0 - Y^*) - (Y_{new} - Y^*)) \text{ or}$$

$$\frac{R\bar{c}^T}{\alpha r}(Y_{new} - Y^*) \leq \left(\frac{R}{\alpha r} - 1\right)\bar{c}^T (Y_0 - Y^*) \text{ or}$$

$$\bar{c}^T (Y_{new} - Y^*) \leq \left(1 - \frac{\alpha r}{R}\right)\bar{c}^T (Y_0 - Y^*) \text{ or}$$

$$\frac{\bar{c}^T (Y_{new} - Y^*)}{\bar{c}^T (Y_0 - Y^*)} \leq 1 - \frac{\alpha r}{R} = 1 - \frac{\alpha\left(\frac{1}{\sqrt{n(n-1)}}\right)}{\sqrt{\frac{n-1}{n}}} = 1 - \frac{\alpha}{n-1}. \qquad (4.12)$$

But under assumption A_2 we have that $\bar{c}^T Y^* = 0$, so that:

$$\frac{\bar{c}^T Y_{new}}{\bar{c}^T Y_0} \leq 1 - \frac{\alpha}{n-1}. \qquad (4.13)$$

Equation (4.12) tells us that at any iteration with respect to the objective function $\bar{c}^T Y$, the gap to optimality is reduced by $\frac{\alpha}{n-1}$. However, $\bar{c}^T = c^T D_k$ is dependent on the particular iteration k, and moreover equations (4.12) only guarantees a decrease in the numerator of problem (4.4). In fact, the fractional objective in problem (4.4) and hence the original objective function in problem (4.1), may not fall and may actually increase. Fortunately, there is another function that pre-

serves strict monotonicity and hence assures convergence to optimality. In fact, it assumes polynomial-time convergence. This function is known as potential function and Karmakar's potential function is given by;

$$f(x) \equiv \sum_{i=1}^{n} \ln\left[\frac{c^T X}{X_j}\right] = n \ln c^T X - \sum_{i=1}^{n} \ln X_j. \tag{4.14}$$

The other formulation of (4.14) is obtained by taking the antilog of the equation (4.14) and given by:

$$f(x) = \ln\left(\frac{(c^T X)^n}{\prod_{i=1}^{n} x_i}\right).$$

The projective transformation (4.2) does not preserve the linearity of a function as seen by the objectives in equation (4.1) and (4.4) it does preserve ratios of linear functions. Indeed under equation (4.3) we have that:

$$\frac{c^T X}{x_j} = \frac{\left(\frac{c^T D_k Y}{e^T D_k Y}\right)}{\left(\frac{x_{kj} Y_j}{e^T D_k Y}\right)} = \frac{c^T D_k Y}{x_{kj} Y_j} = \frac{c^T Y}{x_{kj} Y_j}$$

So that the potential function (4.14) transforms as:

$$f(x) = f\left(\frac{D_k Y}{e^T D_k Y}\right) = F(Y) = \sum_{j=1}^{n} \ln\left[\frac{\bar{c}^T Y}{X_{ki} Y_j}\right]$$

$$= n \ln \bar{c}^T Y - \sum_{j=1}^{n} \ln[Y_j] - \sum_{j=1}^{n} \ln[x_{kj}] \tag{4.15a}$$

Let's now measure the decrease in the value of the potential function (4.14) or equivalently in the potential function (4.15a) in the Y-space at iteration k using the potential function (4.15a) we can obtain that

$$F(Y_{new}) - F(Y_0) = n \ln \bar{c}^T y_{new} - \sum_{j=1}^{n} \ln y_{newj} - \sum_{j=1}^{n} \ln x_{kj} - n \ln \bar{c}^T y_0 + \sum_{j=1}^{n} \ln y_{0j} + \sum_{j=1}^{n} \ln x_{kj}$$

$$= n \ln \frac{\bar{c}^T \bar{Y}_{new}}{\bar{c}^T Y_0} - \sum_{j}^{n} \ln y_{newj} - \sum_{j}^{n} \ln n \quad \text{since } Y_0 = \left(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\right)^T$$

$$= n \ln \frac{\bar{c}^T \bar{Y}_{new}}{\bar{c}^T Y_0} - \sum_{j}^{n} \ln n y_{newj} \tag{4.15b}$$

From equation (4.13) we have that:

$$\frac{\bar{c}^T \bar{Y}_{new}}{\bar{c}^T Y_0} \leq 1 - \frac{\alpha}{n-1}$$

Hence, $\ln \frac{\bar{c}^T \bar{Y}_{new}}{\bar{c}^T Y_0} \leq \ln 1 - \frac{\alpha}{n-1}$.

But from Taylor series we have that: $\ln(1-x) \leq -x \ for\ 1-x>0$

Therefore $\ln \frac{\bar{c}^T \bar{Y}_{new}}{\bar{c}^T Y_0} \leq \frac{-\alpha}{n-1}$ (*)

Inserting equation (*) in to (3.15b) we can have

$$F(Y_{new}) - F(Y_0) \leq \frac{-n\alpha}{n-1} - \sum_{j=1}^{n} \ln(ny_{newj}) \quad (4.16)$$

Consequently, if $-\sum_{j=1}^{n} \ln(ny_{newj})$ does not increases too much in relation to $\frac{n\alpha}{n-1}$, we obtain a sufficient drop in the potential function. This will enable us to guarantee the desired convergence results. Before proceeding to the next discussion we need the following two lemmas. Let $\bar{\alpha} = \frac{n\alpha}{n-1}$ and $0 < \alpha < 1$. Thus $\sqrt{\frac{n-1}{n}}\bar{\alpha} = \sqrt{\frac{n}{n-1}}\alpha < 1$

Lemma4.3.1: If $|x| \leq \alpha < 1$ then $|\ln(1+X) - X| \leq \frac{x^2}{2(1-\alpha)}$ (4.17)

Proof

From Taylor series we have that:

$$\ln(1+X) = X - \frac{X^2}{2!} + \frac{2X^3}{3!} - \frac{6X^4}{4!} + \frac{24X^5}{5!} + \cdots \ldots$$

Thus $|\ln(1+X) - X| = \left|X - \frac{X^2}{2!} + \frac{2X^3}{3!} - \frac{6X^4}{4!} + \frac{24X^5}{5!} + \cdots - X\right|$

$$= \left|-\frac{X^2}{2} + \frac{X^3}{3} - \frac{X^4}{4} + \frac{X^5}{5} + \cdots\right|$$

$$= \frac{X^2}{2}\left|1 - \frac{2x}{3} + \frac{2x^2}{4} - \frac{2x^3}{5} + \cdots\right|$$

$$\leq \frac{X^2}{2}|1 + |x| + |X^2| + |X^3| + \ \ldots|$$

$$\leq \frac{X^2}{2}|1 + \alpha + \alpha^2 + \alpha^3 + \cdots|$$

$$= \frac{X^2}{2(1-\alpha)}.$$

Lemma4.3.2: If $\|nY - e\| \leq \beta \ for \beta = \sqrt{\frac{n}{n-1}}\alpha < 1, e^T Y = 1, Y > 0$ then

$$\left|\sum_{j=1}^{n} \ln(ny_j)\right| \leq \frac{\beta^2}{2(1-\beta)} \quad (4.18)$$

Proof

$$\|nY - e\| \leq \beta \text{ or } \left(\sum_{j=1}^{n}(ny_j - 1)^2\right)^{1/2} \leq \beta .$$

From this we have that:

$$(ny_j - 1)^2 \leq \beta^2 \quad \forall_j = 1, 2, \dots, n.$$

Using Lemma (4.3.1) we get:

$$|\ln(ny_j - 1 + 1) - (ny_j - 1)| \leq \frac{ny_j - 1}{2(1-\beta)} \qquad \forall_j = 1, 2, \dots \dots \dots n.$$

$$\sum_{j=1}^{n} |\ln ny_j - (ny_j - 1)| \leq \sum_{j=1}^{n} \frac{(ny_j - 1)^2}{2(1-\beta)} \leq \frac{\beta^2}{2(1-\beta)}$$

$$\left|\sum_{j=1}^{n} \ln ny_j - \sum_{j=1}^{n} (ny_j - 1)\right| \leq \sum_{j=1}^{n} |\ln ny_j - (ny_j - 1)| \leq \frac{\beta^2}{2(1-\beta)}$$

$$\left|\sum_{j=1}^{n} \ln ny_j - n\sum_{j=1}^{n} y_j + n\right| \leq \frac{\beta^2}{2(1-\beta)}$$

$$\left|\sum_{j=1}^{n} \ln ny_j - n + n\right| \leq \frac{\beta^2}{2(1-\beta)}, \text{ since } \sum_{j=1}^{n} y_j = 1.$$

Therefore $\left|\sum_{j=1}^{n} \ln ny_j\right| \leq \frac{\beta^2}{2(1-\beta)}$.

Moreover, $\sum_{j=1}^{n} \ln ny_j = \ln n^n \prod_{j=1}^{n} y_j = n \ln n + \ln \prod_{j=1}^{n} y_j = n \ln n + n \ln\left(\prod_{j=1}^{n} y_j\right)^{\frac{1}{n}}$

$= n \ln n \left(\prod_{j=1}^{n} y_j\right)^{\frac{1}{n}} \leq n \ln n \frac{\sum_{j=1}^{n} y_j}{n}$ since geometric mean$\leq$ arithmetic mean.

$= n \ln \sum_{j=1}^{n} y_j = n \ln 1 = 0$ since $\sum_{j=1}^{n} y_j = 1$

$$-\sum_{j=1}^{n} \ln ny_j \geq 0 \tag{4.19a}$$

Therefore $0 \leq -\sum_{j=1}^{n} \ln ny_j \leq \frac{\beta^2}{2(1-\beta)}$ (4.19b)

Now let's see that Y_{new} satisfies the condition of Lemma 4.3.2.

i) $e^T Y_{new} = 1$ since $Y_{new} \in \Lambda_y$.

ii) $Y_{new} > 0$, since $Y_{new} \in \Lambda_y$.

iii) $\|nY_{new} - e\| = \|nY_{new} - nY_0\|$ Since $nY_0 = e$

$= \|n(Y_{new} - Y_0)\| \quad = \left\|n\left(\frac{-\alpha r C_P}{\|C_P\|}\right)\right\| = n\alpha r = n\alpha \frac{1}{\sqrt{n(n-1)}} = \sqrt{\frac{n}{n-1}}\alpha = \sqrt{\frac{n-1}{n}}\bar{\alpha} < 1$

Hence, Y_{new} satisfies the condition of lemma (4.3.2) by equation (3.19b) we have that:

$$-\sum_{j=1}^{n} \ln(ny_{newj}) \leq \frac{\left(\sqrt{\frac{n-1}{n}}\bar{\alpha}\right)^2}{2\left(1-\sqrt{\frac{n-1}{n}}\bar{\alpha}\right)} \leq \frac{\bar{\alpha}^2}{2\left(1-\sqrt{\frac{n-1}{n}}\bar{\alpha}\right)}, \text{ but } 1 - \sqrt{\frac{n-1}{n}}\bar{\alpha} \geq 1 - \bar{\alpha}, 1 - \bar{\alpha} \leq 1$$

Therefore $-\sum_{j=1}^{n} \ln(ny_{newj}) \leq \frac{\bar{\alpha}^2}{2(1-\bar{\alpha})} \leq \frac{\bar{\alpha}^2}{2(1-\bar{\alpha})^2}$ (4.20)

Now using equation (3.20) in (3.15b) we obtain

$$F(Y_{new}) - F(Y_0) \leq -\bar{\alpha} + \frac{\bar{\alpha}^2}{2(1-\bar{\alpha})^2} \tag{4.21}$$

When $\bar{\alpha} = \frac{n\alpha}{n-1} = \frac{1}{3}$, equation (4.21) becomes:

$$F(Y_{new}) - F(Y_0) \leq \frac{-5}{24} < \frac{-1}{5}$$

Therefore, when $\bar{\alpha} = \frac{1}{3}$ the function $F(.)$ and consequently the potential $f(.)$ fall by $\frac{1}{5}$ in every iteration. This means that over k iterations:

$$f(X_k) - f(X_0) \leq \frac{-k}{5}$$

Here by $f(X_k) - f\left(X_0\right) = \text{nln}\left[\frac{C^\tau X_k}{C^\tau X_0}\right] - \sum_{j=1}^{n} \ln(nx_{kj})\Big)$

From the above two equation we can get that:

$$\text{nln}\left[\frac{C^\tau X_k}{C^\tau X_0}\right] \leq \sum_{j=1}^{n} \ln(nx_{kj}) - \frac{K}{5} \tag{∇}$$

But by equation (4.19a)

$$\sum_{j=1}^{n} \ln(nx_{kj}) \leq 0$$

Thus (∇) become:

$$\ln\left[\frac{C^\tau X_k}{C^\tau X_0}\right] \leq \frac{-k}{5n} \tag{4.22}$$

Hence, although the objective function value may increase iteration to the next iteration, a sufficient overall decrease from the original objective value at $k = 0$ is maintained as the iterations progress in fact from equation (4.22) we will have:

$$c^T X_k \leq c^T X_0 e^{\frac{-k}{5n}}.$$

For $k = 10nL$

$$c^T X_k \leq c^T X_0 e^{-2L}, \quad \text{since } c^T X_0 < 2^L \text{ see[6]}$$

$$\text{or } c^T X_k \leq e^{-2L} C^\tau X_0 \leq e^{-2L} 2^L < 2^{-2\iota} 2^\iota < 2^{-L} \tag{4.23}$$

where L the lower bound on the input length of the problem is given by:

$$L = \left[1 + \log|\mathrm{D}_{\mathrm{et.max}}| + \log\left(1 + |c_{jmax}|\right)\right] \tag{4.24}$$

where $|\mathrm{D}_{\mathrm{et.max}}|$ is the largest numerical value of the determinant of any basis of the problem and $|c_{jmax}|$ is the largest numerical value of any cost coefficientc_j. Hence strict monotonic function

bounded above is convergent. Further since at each iteration the number of arithmetic operation in the worst case is $0(n^3)$ the polynomial complexity of the algorithm is $0(n^4L)$.

4.4. A Modification to the Algorithm

In Karmarkar's algorithm the number of arithmetic operation in the worst case is $0(n^3)$ with polynomial complexity of $0(n^4L)$.But the computational effort of the algorithm discussed above is dominated by the computation of c_p and the projection of the gradient vector.

Thus in each iteration, we have to find the inverse of (PP^T) ,

Where:

$$PP^T = \begin{bmatrix} AD_k \\ e^T \end{bmatrix} [D_k A^T \quad e] = \begin{bmatrix} AD_k^2 A^T & AD_k e \\ e^T D_k A^T & n \end{bmatrix} = \begin{bmatrix} AD_k^2 A^T & 0 \\ 0 & n \end{bmatrix}$$

since $AD_k e = 0$, $e^T e = n$

and that the only change in this matrix from step to step is in the element of the diagonal trix D_k. Accordingly, Karmarkar shown how a slight modification of the algorithm, based on updating rather than re-computing an appropriate inverse in the gradient projection operation, can be made to run with $0(n^{2.5})$ effort per iterations. This results in an overall reduced polynomial complexity of $0(n^{3.5}L)$ for the modified algorithm.

4.5. Converting a General Linear Program into Karmarkar's Form

In this section we will see how to change a general linear programming into Karmarkar's form while satisfying assumptions 1 *and* 2.

Consider a general linear programming problem:

$$\begin{aligned} & c^T x \rightarrow max \\ & \text{s.t. } Ax \leq b, \\ & x \geq 0, \\ & x \in \mathbb{R}^n. \end{aligned} \tag{4.25}$$

where A is $m \times n$ matrix of rank(A)= m and the data are all integers.

The dual of the problem (4.25) is given by:

$$b^T x \to min$$

$$\text{s.t.} A^T y \geq c, \tag{4.26}$$

$$y \geq 0\,, y \in \mathbb{R}^m.$$

By duality theory, the system of inequalities in (4.27) has a solution if and only if the original problem has finite optimal solutions.

$$\begin{aligned} &Ax \leq b\,, \\ &A^T y \geq c\,, \\ &c^T x - b^T y = 0, \\ &x \geq 0,\ \ y \geq 0\,, x \in \mathbb{R}^n\,, y \in \mathbb{R}^m. \end{aligned} \tag{4.27}$$

Adding slack and surplus variables $x_{n+i},\ i = 1,2,\cdots,m$ and $y_{m+j},\ j = 1,2,\cdots,n.$ in the $Ax \leq b$ and $A^T y \geq c$ respectively. Then (4.27) become:

$$\begin{aligned} &\textstyle\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i \ \ i = 1,2,\cdots,m, \\ &\textstyle\sum_{i=1}^{n} a_{ij}y_i - y_{m+j} = c_j \ \ j = 1,2,\cdots,n, \\ &\textstyle\sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} b_i y_i = 0, \\ &x_j \geq 0, j = 1,2,\cdots,n,n+1,\cdots,n+m, \\ &y_i \geq 0, i = 1,2,\cdots,m,m+1,\cdots,m+n. \end{aligned} \tag{4.28}$$

Before proceeding to the next step, let us define bounding constraint Q by:

$$\textstyle\sum_{i=1}^{n+m} x_i + \sum_{j=1}^{m+n} y_j \leq Q, \tag{4.29}$$

here, Q is sufficiently large number and integer bound which derived from feasibility or optimality considerations. So that any solution of (4.28) satisfies (4.29)

Now by adding slack variables s_1 in (4.29) we have that:

$$\textstyle\sum_{i=1}^{n+m} x_i + \sum_{j=1}^{m+n} y_j + s_1 = Q \tag{4.30}$$

Having equation (4.30) in (4.28)

We have:

$$\begin{aligned} &\textstyle\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i \qquad i = 1,2,\cdots,m, \\ &\textstyle\sum_{i=1}^{m} a_{ij}y_i - y_{m+j} = c_j \qquad j = 1,2,\cdots,n, \\ &\textstyle\sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} b_i y_i = 0\,, \end{aligned} \tag{4.31}$$

$$\sum_{j=1}^{n+m} x_j + \sum_{i=1}^{m+n} y_i + s_1 \leq Q,$$

$$\text{all } x_j, y_i \geq 0, s_1 \geq 0.$$

At this stage, to obtain the problem in the homogenous form, we introduce additional slack variables s_2 with the restriction $s_2 = 1$ and express the problem (4.31) as:

$$\begin{aligned}
&\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} - b_i s_2 = 0 \quad i = 1,2,\cdots,m,\\
&\sum_{i=1}^{m} a_{ij}y_j - y_{m+j} - c_j s_2 = 0 \quad j = 1,2,\cdots,n,\\
&\sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} b_i y_i = 0,\\
&\sum_{j=1}^{n+m} x_j + \sum_{i=1}^{m+n} y_i + s_2 - Q s_2 = 0,\\
&\sum_{j=1}^{n+m} x_j + \sum_{i=1}^{m+n} y_i + s_1 + s_2 = Q + 1,\\
&\text{all } x_j, y_i \geq 0, s_1, s_2 \geq 0.
\end{aligned} \tag{4.32}$$

Where the last two constraints in (4.32) are equivalent to the constraint;

$$\sum_{j=1}^{n+m} x_j + \sum_{i=1}^{m+n} y_i + s_1 = Q \text{ in (4.31) and } s_2 = 1.$$

Next let us use the transformation

$$\begin{aligned}
x_i &= (Q+1)v_j \qquad j = 1,2,\cdots,n+m,\\
y_i &= (Q+1)v_{n+m+i} \quad i = 1,2,\cdots,m+n.,\\
s_1 &= (Q+1)v_{2m+2n+1},\\
s_2 &= (Q+1)v_{2m+2n+2},
\end{aligned}$$

in order to obtain units as the right-hand-side of the final equality constraint and hence equation (4.32) reduced to:

$$\begin{aligned}
&\sum_{j=1}^{m} a_{ij}v_j + v_{n+i} - b_i v_{2m+2n+2} = 0 \quad i = 1,2,\cdots,m,\\
&\sum_{j=1}^{n} a_{ij}v_{n+m+i} - v_{2m+n+i} - c_j v_{2m+2n+2} = 0 \quad j = 1,2,\cdots,n,\\
&\sum_{j=1}^{n} c_j v_j - \sum_{i=1}^{m} b_i\, v_{n+m+i} = 0,\\
&\sum_{j=1}^{n+m} v_j + \sum_{i=1}^{m+n} v + v_{2m+2n+1} - Q v_{2m+2n+2} = 0,\\
&\sum_{j=1}^{n+m} v_j + \sum_{i=1}^{m+n} v_{m+n+i} + v_{2m+2n+1} - v_{2m+2n+2} = 1,\\
&\text{all } v_i \geq 0, i = 1,2,\cdots,2m+2n+2.
\end{aligned} \tag{4.33}$$

The constraints in equation (4.33) are now of the form in problem (4.1) in order to satisfy assumption 1. Let us introduce an artificial variable λ in the constraints of (4.33) such that the sum

of the coefficients in each homogenous constraint and of λ is zero and the coefficient of λ in the last constraint must be unity.

We then consider the problem:

$$\lambda \to min$$

$$\begin{aligned}
\text{s.t.}\quad & \textstyle\sum_{j=1}^{m} a_{ij}v_j + v_{n+i} - b_i v_{2m+2n+2} + k_i\lambda = 0 \quad i = 1,2,\cdots,m, \\
& \textstyle\sum_{j=1}^{n} a_{ij}v_{n+m+i} - v_{2m+n+i} - c_j v_{2m+2n+2} + k_{m+j}\lambda = 0 \quad j = 1,2,\cdots,n, \\
& \textstyle\sum_{j=1}^{n} c_j v_j - \sum_{i=1}^{m} b_i\, v_{n+m+i} + k_{m+n+1}\lambda = 0, \qquad (4.34) \\
& \textstyle\sum_{j=1}^{2m+2n+1} v_j - Q v_{2m+2n+2} + \big(Q - (2m+2n+1)\big)\lambda = 0, \\
& \textstyle\sum_{j=1}^{2m+2n+2} v_j + \lambda = 1, \\
& \text{all } v_i \geq 0, i = 1,2,\cdots,2m+2n+2, \lambda \geq 0.
\end{aligned}$$

If the sum of the coefficients in a particular constraint of (4.33) is zero, the artificial variable λ need not be added to that constraint. That is, the corresponding k is of value zero. The minimum value of λ in (4.34) is zero if and only if the system of inequalities in (4.27) has a solution. The problem (4.18) is now in Karmarkar's form and assumptions 1 *and* 2 are also satisfied by the solving this problem by Karmarkar's algorithm. We obtain optimal solution to the primal and dual problems of the original linear program simultaneously. One disadvantage of this approach of combining the primal and the dual into a single problem is the increase in dimension of the system of equations which must be solved in each iteration.

Example2: To illustrate the use of duality in concept with the foregoing transformations consider the following linear programming problem.

$$2x_1 + x_2 \to max$$

$$\begin{aligned}
\text{s.t.}\quad & x_1 - x_2 \leq 2, \\
& x_1 + 2x_2 \leq 4, \\
& x_1, x_2 \geq 0.
\end{aligned}$$

The dual to this problem is given as follows:

$$2y_1 + 4y_2 \to min$$

$$\text{s.t.} \quad y_1 + y_2 \geq 2\,,$$

$$-y_1 + 2y_2 \geq 1,$$

$$y_1, y_2 \geq 0.$$

Adding slack variables (x_3, x_4) *and* (y_3, y_4) in the primal and dual problems respectively, we need to determine a solution to the following system in order to and a pair of primal and dual optimal solution simultaneously as follows:

$$x_1 - x_2 + x_3 = 2\,, \qquad y_1 + y_2 - y_3 = 2\,,$$

$$x_1 + 2x_2 + x_4 = 4\,, \qquad -y_1 + 2y_2 - y_4 = 1,$$

$$2x_1 + x_2 = 2y_1 + 4y_2 \;\; x_1, x_2 \geq 0 \; y_1, y_2 \geq 0.$$

Let us now introduce the bounding constraints by:

$$\sum_{i=1}^{4} x_i + \sum_{i=1}^{4} y_i \leq Q.$$

Adding slack variable s_1 this constraint becomes:

$$\sum_{i=1}^{4} x_i + \sum_{i=1}^{4} x_i + s_1 \leq Q.$$

Next we introduce slack variable s_2 with restriction $s_2 = 1$ using this variables, we have that

$$\sum_{i=1}^{4} x_i + \sum_{i=1}^{4} y_i + s_1 + s_2 \leq Q + 1.$$

This yields the system:

$$x_1 - x_2 + x_3 - 2s_2 = 0\,,$$

$$x_1 + 2x_2 + x_4 - 4s_2 = 0,$$

$$y_1 + y_2 - y_3 - 2s_2 = 0\,,$$

$$-y_1 + 2y_2 - y_4 - s_2 = 0\,,$$

$$2x_1 + x_2 - 2y_1 - 2y_2 = 0\,,$$

$$\sum_{i=1}^{4} x_i + \sum_{i=1}^{4} y_i + s_1 - Qs_2 = 0\,,$$

$$\sum_{i=1}^{4} x_i + \sum_{i=1}^{4} y_i + s_1 + s_2 = Q + 1\,,$$

$$x_1, x_2, y_1, y_2, s_1, s_2 \geq 0.$$

After this, we use a transformation to change the variables by:

$$x_j = (Q+1)v_j \qquad j = 1,2,\cdots 4,$$

$$y_j = (Q+1)v_{4+i} \qquad i = 1,2,\cdots,4,$$
$$s_1 = (Q+1)v_9\,,$$
$$s_2 = (Q+1)v_{10}\,.$$

This gives the system:

$$v_1 + v_2 + v_3 - 2v_{10} = 0,$$
$$v_1 + 2v_2 + v_4 - 4v_{10} = 0,$$
$$v_5 + 2v_6 - v_7 - 2v_{10} = 0,$$
$$-v_5 + 2v_6 - v_8 - v_{10} = 0,$$
$$2v_1 + v_2 - 2v_5 - 2v_6 = 0,$$
$$\sum_{i=1}^{9} v_i - Qv_{10} = 0,$$
$$\sum_{i=1}^{10} v_i = 1,$$
$$\text{all } v_i \geq 0, i = 1,2,\cdots,10.$$

Finally introducing the artificial variables λ with constant coefficients such that the sum of the coefficients in each homogenous constraint is zero and accommodating λ in the final equality constraint as well (so that assumption λ holds) we get the following problem:

$$\lambda \to min$$

$$\text{s.t.} \quad v_1 + v_2 + v_3 - 2v_{10} + \lambda = 0,$$
$$v_1 + 2v_2 + v_4 - 4v_{10} = 0,$$
$$v_5 + 2v_6 - v_7 - 2v_{10} + \lambda = 0,$$
$$-v_5 + 2v_6 - v_8 - v_{10} + \lambda = 0,$$
$$2v_1 + v_2 - 2v_5 - 2v_6 + 3\lambda = 0,$$
$$\sum_{i=1}^{9} v_i - Qv_{10} + (Q-9)\lambda = 0,$$
$$\sum_{i=1}^{10} v_i + \lambda = 1\,,$$
$$\text{all } v_i \geq 0, i = 1,2,\cdots,10.$$

The problem is now in the required form (4.1) and it satisfies assumptions 1 and 2 using Karmarkar's algorithm to solve this problem will yield a pair of primal and dual optimal solutions to the original linear program.

4.6. Description of Karmarkar's Algorithm

Step1.Initialization

compute $r = \frac{1}{\sqrt{n(n-1)}}$, $\alpha = \frac{n-1}{3n}$ and $x_0 = \left(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\right)^T$.

Step2. Main step

Start from $k = 0$ if at iteration k if $c^T x_k = 0$ stop otherwise, define:

$D_k = diag\{x_{k1}, x_{k2}, \cdots, x_{kn}\}, y_0 = \left(\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\right)^T P = \begin{bmatrix} AD_k \\ e^T \end{bmatrix}, \bar{c}^T = c^T D_k$.

Step3.compute$y_{new} = y_0 - \alpha r \frac{c_p}{\|c_p\|}, where\ c_p = [I - P^T(PP^T)^{-1}P]\bar{c}$.

Step 4: compute $x_{k+1} = \frac{D_k y_{new}}{e^T D_k y_{new}}$ and k=k+1 until $c^T x_k \leq 2^{-L}$.

Step5.Optimal Rounding Routine

Starting with X_k determine an extreme point solution $\bar{X}$ for problem (4.1)

$c^T \bar{X} \leq c^\tau X_k < 2^{-L}$ using earlier purification scheme terminate with $\bar{X}$ an optimal solution to problem (4.1).

Appendix

A.1.Programming for Solving a Linear Optimization Problem by Means of Karmarkar Projective Algorithm

```
Remove["Global`*"];
(*Input*)
c = {_, _, _, _, _};
aA = {{_, _, _, _, _}, {_, _, _, _, _} {_, _, _, _, _},};
e = {1, 1, 1, 1, 1};
it = m;
Print["c=", c];
Print["A= ", aA // MatrixForm];
Print["e= ", e];
n = Length[aA[[1]]];
(*Initialization*)
dD0 = DiagonalMatrix[Table[1/n, {i, 1, n}]];
dD0 // MatrixForm
x0 = Table[1/n, {i, 1, n}];
x[0] = x0;
Print["x[0]= ", x[0] // N];
y0 = x0;
y[0] = y0;
dD[0] = dD0;
uU[0] = aA.dD[0];

(*Iteration*)
r = 1/Sqrt[n (n - 1)];
α = (n - 1)/(3 n);
Print["r= ", r // N];
Print["α= ", α // N];
k = 0;
```

```
While[k ≤ it,
  w[k] = c.x[k]; Print["w(", k, ") = ", w[k]];
  If[w[k] ≤ 10^-6, Break[]];
  uU[k] = aA.dD[k] // N;
  Print["U(", k, ") = ", uU[k] // MatrixForm];
  pP[k] = Append[uU[k], e];
  Print["P(", k, ") = ", pP[k] // MatrixForm];
  gG[k] = (Transpose[pP[k]].Inverse[pP[k].Transpose[pP[k]]].pP[k]) // N;
  Print["G(", k, ") = ", gG[k] // MatrixForm];
  v[k] = (IdentityMatrix[n] - gG[k]).dD[k].c;
  Print["v(", k, ")=", v[k]];
  y[k + 1] = (y[0] - α * r v[k]/Norm[v[k]]) // N;
  x[k + 1] = (dD[k].y[k + 1]/e.dD[k].y[k + 1]) // N;
  Print["x(", k + 1, ")= ", x[k + 1]];
  dD[k + 1] = DiagonalMatrix[x[k + 1]];
  Print["D(", k + 1, ") = ", dD[k + 1] // MatrixForm];
  k = k + 1];
```

A.2. Programming for Solving Maximization of a Linear Optimization Problem by Means of Khachian's Ellipsoid Method

```
Remove["Global`*"];
<< LinearAlgebra`MatrixManipulation`;
(*Input*)
(*the positivity condition should not be considered in the input;
 it will be automatically add in course of the running program;
 Here we consider only problems with positivity because we use the dual problem*)
aA = {{_, _, _, _, _, _}, {_, _, _, _, _, _}, {_, _, _, _, _, _}, {_, _, _, _, _, _}};
b = {_, _, _, _};
c = {_, _, _, _, _, _};
r = v;
it = t; (*Number of iterations*)
(* This input is only for the graphic part in the case n0 = 2*)
n = n0 = Length[aA[[1]]];
paA = aA;
pb = b;
If[n0 == 2,
  {α1, α2} = {5, 3} ; (* choice of the breadth of picture; (x0-α1,x0+α1) *)
  {β1, β2} = {5, 3}; (* choice of the heigth of picture; (y0-β1,y0+β1) *)
  nn = 100; mm = 100;   (*number of points of the raster to fill the feasible set*)];
(*-----------------------------------------------------------------*)
Print["To solve the linear optimization problem:"];
tax = Table[ξi, {i, 1, n0}];
Print["      ", c.tax, " → min"];
Print["      s.t."];
Table[Print["      ", paA[[i]].tax, " ≤ ", pb[[i]]], {i, 1, n0}];
Print["      ξi ≥ 0 , i∈{1,...,", n0, "}"];
Print[""];
Print["A = ", aA // MatrixForm];
Print["b = ", b];
(*Maximization by the Mathematica command "Maximize"*)
constraint = Table[paA[[i]].tax ≤ pb[[i]], {i, 1, Length[paA]}]
tamax = Prepend[constraint, c.tax];
maxi = Maximize[tamax, tax];
xmax = tax /. maxi[[2]];
Print["The maximal value is f(xmax) = ", maxi[[1]]];
Print["The solution of the linear maximization problem is given by:"];
Print["    xmax = ", xmax];
```

```
{x1, x2} = {xmax[[1]] - α1, xmax[[1]] + α2};
{y1, y2} = {xmax[[2]] - β1, xmax[[2]] + β2};
Print[""];
Print["-------------------------------------------------"];
Print[""];
(*-------------------------------------------------------------*)
(*new settings for the extended inequality system*)
m = Length[aA];
o[μ_, λ_] := Table[0, {i, 1, μ}, {j, 1, λ}];
aA = BlockMatrix[{{aA, o[m, m]}, {-IdentityMatrix[n], o[n, m]}, {{-c}, {b}}, {o[n, n], -Transpose[aA]},
    {o[m, n], -IdentityMatrix[m]}  }];
Print["The new Matrix is A = ", aA // MatrixForm];
b = Join[b, Flatten[o[1, n]], {0}, -c, Flatten[o[1, m]]];
Print["The new right side is b = ", b];
tax = Table[ξi, {i, 1, n0}];
tay = Table[ηi, {i, 1, m}];
var = Join[tax, tay];
Table[aA[[i]].var ≤ b[[i]], {i, 1, Length[b]}] // TableForm
n = n + m;
pos = 1;
(*-----------------------------------*)
(*Initialization*)

bB0 = DiagonalMatrix[Table[r^2, {i, 1, n}]];
Print[bB0 // MatrixForm];
If[n0 == 2, px[0] = {0, 0}; pB[0] = {{r^2, 0}, {0, r^2}}];
(*-------------------------------------------------------------*)
(*Drawing the initial circle with the radius r and centre 0 *)
If[n0 == 2, p[0] = ContourPlot[{u, v}.Inverse[pB[0]].{u, v} - 1, {u, x1, x2}, {v, y1, y2}, Contours → {0},
    ContourShading → False, PlotPoints → 100, ContourStyle → {RGBColor[0, 0, 1], Thickness[0.009]},
    DisplayFunction → Identity]];
(*-------------------------------------------------------------*)
x0 = Table[0, {i, 1, n}];
x[0] = x0;
bB[0] = bB0;
k = 0;
w = And[Table[aA[[i]].x0 ≤ b[[i]], {i, 1, Length[aA]}]];
(* Those constraints which are satisfied by x0 = 0  *)
```

```
(* to collect and to calculate the distance of the centre 0 and the hyperplanes of all those
  constraints which are not in w; the centre 0 violates the constraints*)
vion = {}; (*table of list number for violating constraint*)
vio = {};   (*list of distances of violating constraints*)
viomax = 0; (*maximum of distances*)
v = 0;       (* v is the number of violating constraint with maximal distance*)
i = 1;
While[i ≤ Length[aA],
  If[w[[i]] == False,
   vion = Append[vion, i];
   vio = Append[vio, Abs[(aA[[i]].x[0] - b[[i]])/Norm[aA[[i]]]] // N];
   If[Abs[(aA[[i]].x[0] - b[[i]])/Norm[aA[[i]]]] ≥ Max[vio], viomax = Abs[(aA[[i]].x[0] - b[[i]])/Norm[aA[[i]]]]; v = i]];
  i = i + 1];
(*-----------------------------------------*)
av = aA[[v]]; (*normal vector of hyperplane of violating constraint with maximal distance*)
bv = b[[v]];
(* coordinate of vector b belonging to the violationg constraint with maximal distance*)
If[av.bB[0].av < 0, Print["In the ball B0 lies no feasible point"]; Break[]];
Print["quadratic form = ", av.bB[0].av];
(*-------------------------------------------------------------*)
(* Iteration *)

(* "it" is the number of iterations; input*)
k = 0;
While[k ≤ it,
 Print["Eigenvalues of B(", k, ") : ", Eigenvalues[bB[k]]];
 If[av.bB[k].av < 0, Print["In the ball B0 lies no feasible point"]; Break[]];
 Print["x(", k, ") = ", x[k]];
 w = And[Table[aA[[i]].x[k] ≤ b[[i]], {i, 1, Length[aA]}]];
 Print["w = ", w];
```

```
If[Intersection[{False}, w] == {}, Break[]];
(*i.e. the centre of the ellipse is feasible*)
(* now we make the same as before for k = 0 *)
vion = {};
vio = {0};
viomax = 0;
v = 0;
i = 1;
While[ i ≤ Length[aA],
  If[ w[[i]] == False,
   vion = Append[vion, i];
   vio = Append[ vio, Abs[ (aA[[i]].x[k] - b[[i]]) / Norm[aA[[i]]] ] // N ];
   If[ Abs[ (aA[[i]].x[k] - b[[i]]) / Norm[aA[[i]]] ] ≥ Max[vio], viomax = Abs[ (aA[[i]].x[k] - b[[i]]) / Norm[aA[[i]]] ]; v = i ] ];
  i = i + 1 ];
av = aA[[v]];
bv = b[[v]];
Print["v = ", v, "     av = ", aA[[v]], "  bv = ", b[[v]]];
d[k] = -bB[k].av / Sqrt[av.bB[k].av] // N;
Print["d(", k, ") = ", d[k]];
λ[k] = 0;

(*(av.x[k]-bv)/(av.av) Sqrt[av.bB[k].av] //N;*)
Print["λ(", k, ") = ", λ[k]];
α[k] = (1 + n*λ[k])/(n + 1) // N;
δ[k] = n^2/(n^2 - 1) (1 - (λ[k])^2) // N;
Print["δ(", k, ") = ", δ[k]];
σ[k] = 2 (1 + n*λ[k])/((n + 1)*(1 + λ[k])) // N;
Print["σ(", k, ") = ", σ[k]];
x[k + 1] = x[k] + α[k]*d[k] // N;
If[n0 == 2, px[k + 1] = {x[k + 1][[1]], x[k + 1][[2]]}];
Print["x(", k + 1, ") = ", x[k + 1]];
xx = x[1];
```

```
bB[k+1] = δ[k] (bB[k] - σ[k] Partition[bB[k].av, 1].{bB[k].av} / av.bB[k].av) // N;
  pB[k+1] = TakeMatrix[bB[k+1], {1, 1}, {2, 2}];
  Print["B(", k+1, ") = ", bB[k+1] // MatrixForm // N];
  If[n0 == 2, p[k+1] = ContourPlot[({u, v} - px[k+1]).Inverse[pB[k+1]].({u, v} - px[k+1]) - 1,
      {u, x1, x2}, {v, y1, y2}, Contours → {0}, ContourShading → False, PlotPoints → 100,
      ContourStyle → {RGBColor[1/(k+1), 0, 1/(k+1)], Thickness[0.009]}, DisplayFunction → Identity]];
  xxk = x[k+1];
  steps = k+1;
  Print["-----------------------------------------"];
  k = k+1];
If[n0 == 2,
  qua = Table[p[j], {j, 0, k}];
  poi = Append[Table[Graphics[{RGBColor[1/j, 0, 1/j], PointSize[0.02], Point[px[j]]}], {j, 1, k}],
    Graphics[{RGBColor[0, 0, 1], PointSize[0.02], Point[px[0]]}]];
  (*Definition of constraint functions*)
  Clear[x, y, u, v];
  const = Table[paA[[i]].{x, y} - pb[[i]], {i, 1, Length[pb]}];
  poscon = {-x, -y};  (*positivity condition for the variables in the form  ≤ 0 *)
  const1 = Join[const, poscon];
  nc = Length[const1];    (*number of constraints*)
  Do[g[i][x_, y_] = const[[i]], {i, 1, Length[const]}];
  g[Length[const] + 1][x_, y_] = -x;
  g[Length[const] + 2][x_, y_] = -y;
  conin = Table[g[i][u, v] ≤ 0, {i, 1, nc}];
  (*-----------------------------------------*)
  (*Calculation the raster of points in the square (x0-α1,x0+α2)×(y0-β1,y0+β2  *)
  p = 0; q = 0;
  lisx = {}; lisy = {};
  While[p ≤ nn,
   x = x1 + p*((x2 - x1)/nn);
   lisx = Append[lisx, x];
   p = p+1];
  While[q ≤ mm,
   y = y1 + q*((y2 - y1)/mm);
   lisy = Append[lisy, y];
   q = q+1];
  lis = Flatten[Table[{lisx[[i]], lisy[[j]]}, {i, 1, Length[lisx]}, {j, 1, Length[lisy]}], 1];
  (*-----------------------------------------*)
```

```
(*Graphic*)
a = Table[ContourPlot[g[i][x, y], {x, x1, x2}, {y, y1, y2}, Contours → {0}, ContourShading → False,
    PlotPoints → 100, ContourStyle → {RGBColor[1, 0, 0], Thickness[0.011]}, DisplayFunction → Identity], {i, 1, nc}];
ρ = 1; liscol = {};
While[ρ ≤ Length[lis],
  If[Union[And[Table[g[i][lis[[ρ, 1]], lis[[ρ, 2]]] ≤ 0, {i, 1, nc}]]] == {True},
   liscol = Append[liscol, {RGBColor[0.9, 1, 0], PointSize[0.015], Point[{lis[[ρ, 1]], lis[[ρ, 2]]}]}]];
  ρ = ρ + 1];
d = Graphics[{RGBColor[0, 0, 1], PointSize[0.03], Point[xmax]}];
Show[Graphics[liscol], a, poi, qua, d, Axes → True, AspectRatio → Automatic, DisplayFunction → $DisplayFunction] ];
Table[Print["The volume of qadric(", i, ") is ", 2 * √(π^n) / (√(Det[Inverse[bB[i]]]) * n * Gamma[n/2]) // N], {i, 0, k}];
```

A.3. Programming for Finding a Feasible Point by Means of Khachian's Ellipsoid Method

```
Remove["Global`*"];
(*Input*)
aA = {{_, _, _, _, _}, {_, _, _, _, _}, {_, _, _, _, _}, {_, _, _, _, _}};
b = {_, _, _, _};
c = {_, _, _, _, _}  (* vector to describe the objectiv function*)
it = t;  (*number of iterations*)
{α1, α2} = {4, 5} ; (* choice of the breadth of picture; (x0-α1,x0+α1) *)
{β1, β2} = {4, 5}; (* choice of the heigth of picture; (y0-β1,y0+β1) *)
pos = 2;  (* pos=1 means x≥0,y≥0; pos=2 means without nonnegative condition*)
nn = 100; mm = 100;    (*number of points of the raster to fill the feasible set*)
r = d;  (*radius of the initial ball*)
(*-----------------------------------*)
(*-----------------------------------*)
n = Length[aA[[1]]];
tax = Table[ξi, {i, 1, n}];
Print["To solve the linear optimization problem:"];
Print["      ", c.tax, " → min"];
Print["       s.t."];
Table[Print["          ", aA[[i]].tax, " ≤ ", b[[i]]], {i, 1, Length[aA]}];
If[pos == 1, Print["       ξi ≥ 0  , i∈{1,...,", n, "}"]];
Print[""];
Print["A = ", aA // MatrixForm];
Print["b = ", b];
Print["c = ", c];
n = Length[aA[[1]]];
(*Initialization*)
bB0 = DiagonalMatrix[Table[r^2, {i, 1, n}] // N];
bB0 // MatrixForm;
(*-----------------------------------------------------------*)
(*Drawing the initial circle with the radius r and centre 0 *)
```

```
If[n == 2, p[0] = ContourPlot[{u, v}.Inverse[bB0].{u, v} - 1, {u, -5, 5}, {v, -5, 5}, Contours → {0},
    ContourShading → False, PlotPoints → 100, ContourStyle → {RGBColor[0, 0, 1], Thickness[0.009]},
    DisplayFunction → Identity]];
(*-----------------------------------------------------------*)
x0 = Table[0, {i, 1, n}];
x[0] = x0; bB[0] = bB0;
k = 0;
w = And[Table[aA[[i]].x0 ≤ b[[i]], {i, 1, Length[aA]}]];
 (* Those constraints which are satisfied by x0 = 0  *)
norm[z_] := Sqrt[Sum[(z[[i]])^2, {i, 1, n}]];
(* to collect and to calculate the distance of the centre 0 and the hyperplanes of all those
  constraints which are not in w; the centre 0 violates the constraints*)
vion = {}; (*table of list number for violating constraint*)
vio = {};   (*list of distances of violating constraints*)
viomax = 0; (*maximum of distances*)
v = 0;       (* v is the number of violating constraint with maximal distance*)
i = 1;
While[i ≤ Length[aA],
  If[w[[i]] == False,
   vion = Append[vion, i];
   vio = Append[vio, Abs[(aA[[i]].x[0] - b[[i]])/Norm[aA[[i]]]] // N];
   If[Abs[(aA[[i]].x[0] - b[[i]])/Norm[aA[[i]]]] ≥ Max[vio], viomax = Abs[(aA[[i]].x[0] - b[[i]])/Norm[aA[[i]]]]; v = i]];
  i = i + 1];
(*-----------------------------------------------------------*)
av = aA[[v]]; (*normal vector of hyperplane of violating constraint with maximal distance*)
bv = b[[v]];
(* coordinate of vector b belonging to the violationg constraint with maximal distance*)
If[av.bB[0].av < 0, Print["In the ball B0 lies no feasible point"]; Break[]];
Print["quadratic form = ", av.bB[0].av];
(*-----------------------------------------------------------*)
```

```
(* Iteration *)
(* "it" is the number of iterations; input*)
k = 0;
While[k ≤ it,
 Print["Eigenvalues of B(", k, ") : ", Eigenvalues[bB[k]]];
 If[av.bB[k].av < 0, Print["In the ball B0 lies no feasible point"]; Break[]];
 Print["x(", k, ") = ", x[k]];

 w = And[Table[aA[[i]].x[k] ≤ b[[i]], {i, 1, Length[aA]}]];
 Print["w = ", w];
 If[Intersection[{False}, w] == {}, Break[]];
 (*i.e. the centre of the ellipse is feasible*)
 (* now we make the same as before for k = 0 *)
 vion = {};
 vio = {0};
 viomax = 0;
 v = 0;
 i = 1;
 While[i ≤ Length[aA],
  If[w[[i]] == False,
   vion = Append[vion, i];
   vio = Append[vio, Abs[(aA[[i]].x[k] - b[[i]])/Norm[aA[[i]]]] // N];
   If[Abs[(aA[[i]].x[k] - b[[i]])/Norm[aA[[i]]]] ≥ Max[vio], viomax = Abs[(aA[[i]].x[k] - b[[i]])/Norm[aA[[i]]]]; v = i]];
  i = i + 1];
 av = aA[[v]];
 bv = b[[v]];
 Print["v = ", v, "    av = ", aA[[v]], "  bv = ", b[[v]]];
 d[k] = -bB[k].av/Sqrt[av.bB[k].av] // N;
 Print["d(", k, ") = ", d[k]];
 λ[k] = 0;
 (*(av.x[k] - bv)/(av.av) Sqrt[av.bB[k].av] //N;*)
 Print["λ(", k, ") = ", λ[k]];
 α[k] = (1 + n*λ[k])/(n + 1) // N;
 δ[k] = n^2/(n^2 - 1) (1 - (λ[k])^2) // N;
```

```
Print["δ(", k, ") = ", δ[k]];
σ[k] = 2 (1 + n * λ[k]) / ((n + 1) * (1 + λ[k])) // N;
Print["σ(", k, ") = ", σ[k]];
x[k + 1] = x[k] + σ[k] * d[k] // N;
Print["x(", k + 1, ") = ", x[k + 1]];
xx = x[1];
{x1, x2} = {xx[[1]] - α1, xx[[1]] + α2};
{y1, y2} = {xx[[2]] - β1, xx[[2]] + β2};
bB[k + 1] = δ[k] {bB[k] - σ[k] Partition[bB[k].av, 1].{bB[k].av} / av.bB[k].av} // N;
Print["B(", k + 1, ") = ", bB[k + 1] // MatrixForm // N];
If[n == 2, p[k + 1] = ContourPlot[({u, v} - x[k + 1]).Inverse[bB[k + 1]].({u, v} - x[k + 1]) - 1, {u, x1, x2},
    {v, y1, y2}, Contours → {0}, ContourShading → False, PlotPoints → 100,
    ContourStyle → {RGBColor[1 / (k + 1), 0, 1 / (k + 1)], Thickness[0.009]}, DisplayFunction → Identity]];
xxk = x[k + 1];
steps = k + 1;
Print["----------------------------------------"];
k = k + 1];
If[n == 2,
qua = Table[p[j], {j, 0, k}];
poi = Append[Table[Graphics[{RGBColor[1/j, 0, 1/j], PointSize[0.02], Point[x[j]]}], {j, 1, k}],
  Graphics[{RGBColor[0, 0, 1], PointSize[0.02], Point[x[0]]}]];

(*Definition of constraint functions*)
Clear[x, y, u, v];
const = Table[aA[[i]].{x, y}, {i, 1, Length[b]}];
poscon = {-x, -y};  (*positivity condition for the variables in the form  ≤ 0 *)
const1 = Join[const, poscon];
nc = Length[const1];    (*number of constraints*)
Do[g[i][x_, y_] = const[[i]], {i, 1, Length[const]}];
Do[g[i][x_, y_] = poscon[[i - Length[const]]], {i, Length[const] + 1, nc}];
conin = If[pos == 1, Table[g[i][u, v] - b[[i]] ≤ 0, {i, 1, nc}], Table[g[i][u, v] - b[[i]] ≤ 0, {i, 1, Length[const]}]];
(*----------------------------------------------*)
(*--------------------------------------*)
(*Calculation the raster of points in the square (x0-α1,x0+α2)*(y0-β1,y0+β2  *)
```

```
p = 0; q = 0;
 lisx = {}; lisy = {};
 While[p ≤ nn,
   x = x1 + p* ((x2 - x1) / nn);
   lisx = Append[lisx, x];
   p = p + 1];
 While[q ≤ mm,
   y = y1 + q* ((y2 - y1) / mm);
   lisy = Append[lisy, y];
   q = q + 1];
 lis = Flatten[Table[{lisx[[i]], lisy[[j]]}, {i, 1, Length[lisx]}, {j, 1, Length[lisy]}], 1];
 (*-----------------------------------------------*)
 (*Graphic*)
 If[pos == 1,
   a = Table[ContourPlot[g[i][x, y] - b[[i]], {x, x1, x2}, {y, y1, y2}, Contours → {0}, ContourShading → False,
      PlotPoints → 100, ContourStyle → {RGBColor[1, 0, 0], Thickness[0.011]}, DisplayFunction → Identity],
     {i, 1, nc}]];
 If[pos == 2,
   a = Table[ContourPlot[g[i][x, y] - b[[i]], {x, x1, x2}, {y, y1, y2}, Contours → {0}, ContourShading → False,
      PlotPoints → 100, ContourStyle → {RGBColor[1, 0, 0], Thickness[0.011]}, DisplayFunction → Identity],
     {i, 1, Length[const]}]];
 r = 1; liscol = {};
 If[pos == 1,
   While[r ≤ Length[lis],
    If[Union[And[Table[g[i][lis[[r, 1]], lis[[r, 2]]] - b[[i]] ≤ 0, {i, 1, nc}]]] == {True},
     liscol = Append[liscol, {RGBColor[0.9, 1, 0], PointSize[0.015], Point[{lis[[r, 1]], lis[[r, 2]]}]}]];
    r = r + 1]];
 If[pos == 2,
   While[r ≤ Length[lis],
    If[Union[And[Table[g[i][lis[[r, 1]], lis[[r, 2]]] - b[[i]] ≤ 0, {i, 1, Length[const]}]]] == {True},
     liscol = Append[liscol, {RGBColor[0.9, 1, 0], PointSize[0.015], Point[{lis[[r, 1]], lis[[r, 2]]}]}]];
    r = r + 1]];
 d = Graphics[{RGBColor[0.3, 0.3, 0.1], PointSize[0.03], Point[xxk]}];
 Show[Graphics[liscol], a, poi, qua, Axes → True, AspectRatio → Automatic, DisplayFunction → $DisplayFunction]   ];
Table[Print["The volume of qadric(", i, ") is ", 2* √(π^n) / (√(Det[Inverse[bB[i]]]) *n* Gamma[n/2])], {i, 0, k}];
```

Bibliography

1.Bazaraa MS, JJ Jarvis and HD Sharali, Linear Programming and Network Flows, Second Edition, John Wiley and Sons, New York, 1990.

2. Gacs P and L Lovask "Khachiyan Algorithm for Linear Programming," Mathematical Programming Study 14, North-Holland, Amsterdam, pp 61-68, 1981.

3. George L.Nemhauser and Laurence A. Wolsey, Integer and Combinatorial Optimization ,John Wiley and Sons, United State of America,1999.

4. Hubertus TH. Jongen, Klaus Meer, Ebrehard Triesch, Optimization Theory, Kluwer Academic Publeshers.Boston, 2004.

5.Klee V and GJ Minty, "How Good is Simplex Algorithm?" in Inequalities III,O Shisha (ad), Academic Press, New York,pp159-175,1972

6.S.M.Sinha, Mathematical Programming Theory and Method, First Edition Reed Elsevier India Private Limited,2006.