SOLUTION OF CHANCE CONSTRAINED LINEAR PROGRAMMING
BY ELLIPSOID METHOD

5.1 INTRODUCTION
Charnes and Cooper (1969) initially developed the chance-constrained programming procedure for solving linear stochastic programming problem. The chance constraints are converted to deterministic nonlinear constraints distributed normally and independently of each other. In (1985 and 1983) Ecker and Kupferschmid provide a computational evidence that the ellipsoid algorithm is extremely robust and relative to efficiency. For a complete study of ellipsoid algorithm see the survey of ellipsoid algorithm given by Bland, Goldfarb and Todd (1983). A review of chance constrained modeling is given by Mukherjee (1980). A linear approximation for chance constrained programming was given by Olson and Scott (1987). Rakes, Franz and Wyne (1981) used a piecewise linear goal programming code for solving chance constrained models. Weintrub and Vera (1991) considered the constraints (5.2.2) of the problem defined in (5.2.1)-(5.2.2) of the next section for ≥ case taking only a_{ij} as a random variables distributed normally and solved it by cutting plane algorithm.

In this paper we consider the constraints (5.2.2) for ≤ case taking both a_{ij} and b_i as random variables and solving such problems using ellipsoid algorithm. Further, the
computational results are obtained when only the $a_{ij}$ are normally distributed random variables.

5.2 CHANCE CONSTRAINED PROGRAMMING PROBLEM

Let us consider a linear programming problem

$$\begin{align*}
\text{Minimize} & \quad C^T x \\
\text{s.t.} & \quad Ax \leq b, \quad x \geq 0
\end{align*} \quad \text{...(5.2.1)}$$

where $A \in R^{mxn}$, $b \in R^m$ and $x \in R^n$

The corresponding chance constrained programming problem is defined as

$$\begin{align*}
\text{Minimize} & \quad \sum_{j=1}^{n} C_j x_j \\
\text{s.t.} & \quad P\left(\sum_{j=1}^{n} a_{ij} x_j \leq b_i\right) \geq q_i, \quad i = 1, \ldots, m \\
x_j & \geq 0
\end{align*} \quad \text{...(5.2.2)}$$

where $i$ the constraint $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ in (5.2.2) has to be satisfied with a probability of at least $q_i$ where $0 \leq q_i \leq 1$, i.e., the constraint may be violated by a specified probability.

For simplicity, we consider that the decision variables $x_j$ are deterministic. Also assume that the random variables $a_{ij}, b_i$ are distributed normally and independently of each other.
5.3 DETERMINISTIC EQUIVALENT PROBLEM:

The deterministic equivalent of chance-constrained problem is the non-linear programming problem given below.

\[
\begin{align*}
\text{Minimize} & \quad \sum_{j=1}^{n} C_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} a_{ij} x_j + S_i \left( \sum_{j=1}^{n} a_{ij}^2 x_j^2 \right)^{1/2} \leq \omega_i \\
x_j \geq 0 \\
\end{align*}
\]

where \( E(a_{ij}) = \tilde{a}_{ij} \), \( V(a_{ij}) = a_{ij}^2 \), \( E(b_i) = \omega_i \), \( V(b_i) = u_i^2 \) and \( S_i = F^{-1}(1-q_i) \).

\( E(.) \) and \( V(.) \) are expected value and variance operators respectively and \( F(.) \) is the cumulative standardized normal probability distribution function of

\[
\frac{r_i - t_i}{\nu_i}
\]

where \( r_i = \sum_{j=1}^{n} a_{ij} x_j - b_i \)

\( t_i = E\left( \sum_{j=1}^{n} a_{ij} x_j - b_i \right) \)

\( \nu_i = V\left( \sum_{j=1}^{n} a_{ij} x_j - b_i \right) \)

Here the value of coefficient \( S_i \) is got from the normal distribution. The constraint set (5.3.1) is convex only if \( q_i \geq 0.5 \).

The model (5.3.1) is a type of stochastic problem. The objective
is to identify the feasible solution to the problem stated in (5.2.2). Therefore the concept of feasible direction assists in formulation of nonlinear constraints.

Separable technique was applied to chance constrained first by Seppala and Orpana (1984) and nonlinear constraints are linearized approximately. Olson and Lee (1985) used a gradient algorithm for chance constrained nonlinear goal programming. Later, Weintrub and Vera (1991) used a cutting plane algorithm for chance constrained linear program.

### 5.4 Ellipsoid Algorithm

Here, we present an ellipsoid algorithm that solves the nonlinear deterministic equivalent of the chance constrained problem. The method was first proposed by Shor (1977). We give a general description of this algorithm. The method is applicable to nonlinear programming of the form

\[
\begin{align*}
\text{Minimize} & \quad C^T x \\
\text{s.t.} & \quad h_i(x) \leq b_i, \quad i = 1, \ldots, m
\end{align*}
\]

where \( h_i \), \( i = 1, \ldots, m \) are convex function, i.e., the feasible set of this problem

\[
S = \left\{ x \in \mathbb{R}^n / f_i(x) = h_i(x) - b_i \leq 0 \right\}
\]

is convex such that \( f_i(x^*) \leq 0 \).

The method assumes that there exists an optimal point \( x^* \in S \) and \( S \subset E_0 \). We can take the centre of \( E_0 \) as the initial point \( x^0 \) with positive definite matrix \( Q_0 \) such that \( x^* \in E_0 \) and \( E_0 \cap S \) have positive volume relative to \( \mathbb{R}^n \).
We start with an $n$-dimensional ellipsoid of the form

$$E_0 = \left\{ x \in \mathbb{R}^n \mid (x-x^0)^T Q_0^{-1} (x-x^0) \leq 1 \right\} \quad \ldots (5.4.3)$$

The method is called the ellipsoid algorithm and basic steps are as follows:

Step 0: Select $x^0$ and ellipsoid matrix $Q_0$ such that $x^* \in E_0$ in (5.4.3). Set $k = 0$.

Step 1: Identify whether $x^k$ is feasible
- if $x^k \in S$, $i = 0$ stop,
- if $x^k \notin S$, $i$ = index of violated constraint.

Step 2: Find the unit normal vector to cutting hyperplane
$$g = \frac{\nabla f_i(x^k)}{\| \nabla f_i(x^k) \|}$$

Step 3: Calculate the direction vector
$$d = -Q_k g / \sqrt{g^T Q_k g} \quad \text{s.t.} \quad g^T Q_k g > 0.$$ 

Step 4: Calculate
$$x^{k+1} = x^k + \frac{d}{n+1}$$
$$Q_{k+1} = \frac{n}{n-1} \left[ Q_k - \frac{2}{n+1} dd^T \right]$$

Step 5: Check for convergence
- if $\| x^{k+1} - x^k \| < T$ stop, because $x^{k+1}$ is the minimum.

Step 6: $k \leftarrow k + 1$

Go to 1.

where $T$ denotes the convergence of tolerance.

The basic concept of the ellipsoid algorithm is to generate a sequence of successive smaller ellipsoids each containing the optimal point such that

$$S \subset E_k, \quad E_{k+1} \subset E_k \quad \text{for all } k$$
i.e., the current ellipsoid $E_k$ has its centre $x^k$ with matrix $Q_k$, then the next ellipsoid $E_{k+1}$ has centre $x^{k+1}$ with matrix $Q_{k+1}$.
as in step(4). This is done by cutting the half of previous ellipsoid \( E_0 \) to construct the new ellipsoid \( E_1 \).

Geometrically, the ellipsoid \( E_{k+1} \) is generated by using step(2), cutting the hyperplane \( H_k \) passing through the centre \( x^k \) of \( E_k \) and is given by

\[
H = \left\{ x \in \mathbb{R}^n \mid \nabla f_i(x^k)^T (x-x^k) \geq 0 \right\}
\]

and so \( x^k \in E_k \cap H_k \). This hyperplane support the contour \( f_i(x) = f_i(x^k) \) at the point \( x^k \), so it is termed as a supporting hyperplane. In real situation, the ellipsoid \( E_{k+1} \) defined in step(4) is the unique ellipsoid of minimum volume containing \( E_k \cap H_k \) and so

\[
x^k \in E_{k+1} \cap H_{k+1}
\]

Alternatively, the easiest way to start with ellipsoid \( E_0 \) we assume that upper bounds \( x^Z \) and lower bounds \( x^Y \) on the variables are known and

\[
x^0 = \frac{x^Y + x^Z}{2},
\]

\[
Q_0 = -\frac{n}{4} \begin{bmatrix} (x^Z - x^Y)^2 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & (x^Z - x^Y)^2
\end{bmatrix}
\]

where \( x^0 \) is the initial point with matrix \( Q_0 \). The ellipsoid algorithm requires that an initial ellipsoid \( E_0 \) can be selected as the smallest ellipsoid containing the optimal point defined by

\[
C = \left\{ \frac{x}{x^Y} \leq x \leq x^Z \right\}
\]

Many of the test problems have published upper and lower bounds and when available, we use these bounds to generate the
starting point. If published bounds are not available, then reasonably wide bounds are chosen so as to include the optimal vector.

5.5 CONVERGENCE OF ALGORITHM 1

Shor (1977) showed for the convex unconstrained case that ellipsoid algorithm converges to the optimal value as terms of a geometric series with a ratio $q_n$. Similar convergence result is given by Goffin (1983). By Ecker and Kupferschmid (1983) the average convergence of the ellipsoid algorithm is linear, so that the error in the solution decreases due to the volumes of the ellipsoid decreases in geometric progression with ratio $q(n)$ depending on the dimension and is given by

$$q(n) = \frac{\text{Volume of } E_{k+1}}{\text{Volume of } E_k}$$

$$= \frac{n^n \sqrt{(n-1)^{1-n}}}{\sqrt{(n+1)^{1+n}}} \leq 1 \quad \ldots \ldots (5.5.1)$$

The ratio $q(n)$ tends to 1 as $n$ increases, so the convergence of the algorithm is slower for larger values of $n$. One main shortcoming is that sometimes the algorithm converges to KKT point when the convexity is not satisfied.

5.6 EFFICIENCY OF ALGORITHM 1

The robustness of the ellipsoid algorithm is found to be most efficient at some levels of error. It is superior to both computer time and number of function required to reduce the relative error upto $10^{-2}$. It gives the quick solution even the iterates are larger.
5.7 ACCURACY OF ALGORITHM

It is capable of obtaining accurate results if run for many iterations. Ellipsoid algorithm displays relatively slow but sure convergence. Its initial trajectory is usually convex, so that the linear trend intersects the error below -1 or less. Ellipsoid algorithm remains stable until extremely small error level have been attained. It is always necessary for the initial ellipsoid to contain the optimal point to converge to the optimal point. To contain the optimal point for ellipsoid algorithm it is clear to generate larger ellipsoid to contain the entire feasible set.

5.8 EXAMPLE

An example is presented for chance constrained programming problem. Result of this problem is shown in table 1. The equivalent deterministic (nonlinear programming) problem is

Minimize \( 24x_1 + 40x_2 \)
\( x \in \mathbb{R}^2 \)

s.t. \( x_1 - x_2 + 1 \leq 0 \)
\( 6x_1 + 9x_2 + S(16x_1^2 + 36x_2^2)^{1/2} - 425 \leq 0 \)
\( -3x_1 - 5x_2 + 75 \leq 0 \)

where \( S = F^{-1}(.95) = 1.645 \)

Applying the ellipsoid algorithm to the example problem using \( T = 0.05 \), starting ellipsoid \( E_0 \) in figure (1) is defined by

\( x^0 = [16, 19] \) and

\( q_0 = \begin{bmatrix} 49 & 0 \\ 0 & 121 \end{bmatrix} \)

The starting point \( x^0 \) yields the sequence of iterates in phase 1.
and phase 2 in an irregular sequence in the Table 1, converging to the final values given on the last line.

**Table - 1**

<table>
<thead>
<tr>
<th>k</th>
<th>Phase</th>
<th>((x^k)^T)</th>
<th>(f_0(x^k))</th>
<th>(|x^{k+1} - x^k|)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>[16.000, 19.000]</td>
<td>1144.000</td>
<td>3.556</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>[15.262, 15.521]</td>
<td>987.128</td>
<td>3.043</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>[13.157, 17.718]</td>
<td>1024.488</td>
<td>2.481</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>[11.818, 15.629]</td>
<td>908.792</td>
<td>1.677</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>[10.751, 14.335]</td>
<td>831.424</td>
<td>1.119</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>[10.039, 13.472]</td>
<td>779.816</td>
<td>0.769</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>[09.524, 12.900]</td>
<td>744.576</td>
<td>0.463</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>[09.266, 12.516]</td>
<td>723.024</td>
<td>0.309</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>[09.095, 12.259]</td>
<td>708.640</td>
<td>0.207</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>[08.979, 12.088]</td>
<td>699.016</td>
<td>0.137</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[08.902, 11.974]</td>
<td>692.608</td>
<td>0.093</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>[08.850, 11.897]</td>
<td>688.280</td>
<td>0.061</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>[08.816, 11.846]</td>
<td>683.424</td>
<td>0.042</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>[08.792, 11.812]</td>
<td>683.488</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>∞</td>
<td>2</td>
<td>[08.421, 11.501]</td>
<td>662.144</td>
<td>0.000</td>
</tr>
</tbody>
</table>

All the constraints are satisfied at \(\tilde{x}\) which is shown in the feasible region in figure (2), so the objective function is used for defining the cutting hyperplane. But the calculation remains same for the second iteration as in first. Figure (1) and (2) show
an example of first and second iteration of the ellipsoid algorithm. In figure (1), $E_1$ is the smallest ellipsoid of $E_0$ which contains the feasible region $s$, $x^0$ is the optimal point defined by $E_0$ and $x^1$ is the interior point. $L_1$ is tangent hyperplane passing through $P_0$ and parallel to cutting hyperplane $H_0$. The constraint contour is shown by dotted line. Similarly in figure (2), $E_2$ is the smallest ellipsoid of $E_1$, $L_2$ the tangent hyperplane passing through $P_1$ is parallel to $H_1$ supporting the objective function contour leading to new ellipsoid $E_2$.

5.9 COMPUTATIONAL EFFORT:

Table 2 shows the 14 test problem presented by Dembo (1976) and Colville (1968) using by an ellipsoid algorithm which includes 4 convex and 10 nonconvex problem. All the problems were executed on IBM/370-3033 computer under MTS operating system and the CPU time was determined in seconds. The algorithm is acceptable at the error level of $10^{-3}$.

Table 2. Computational results for 14 test problem.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Convex</th>
<th>n</th>
<th>m</th>
<th>N</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>12</td>
<td>03</td>
<td>165</td>
<td>00.979</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>05</td>
<td>09</td>
<td>254</td>
<td>00.290</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>07</td>
<td>15*</td>
<td>442</td>
<td>00.838</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>08</td>
<td>04</td>
<td>404</td>
<td>00.670</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>08</td>
<td>06</td>
<td>467</td>
<td>00.790</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>13</td>
<td>18*</td>
<td>172</td>
<td>00.706</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>16</td>
<td>25*</td>
<td>2,219</td>
<td>13.560</td>
</tr>
</tbody>
</table>
Table - 2 Continued.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Convex</th>
<th>n</th>
<th>m</th>
<th>N</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Yes</td>
<td>07</td>
<td>04</td>
<td>215</td>
<td>00.343</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>05</td>
<td>15</td>
<td>203</td>
<td>00.105</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>15</td>
<td>20</td>
<td>482</td>
<td>00.759</td>
</tr>
<tr>
<td>11</td>
<td>No</td>
<td>05</td>
<td>16</td>
<td>230</td>
<td>00.120</td>
</tr>
<tr>
<td>12</td>
<td>No</td>
<td>04</td>
<td>08</td>
<td>138</td>
<td>00.058</td>
</tr>
<tr>
<td>13</td>
<td>No</td>
<td>06</td>
<td>04</td>
<td>053</td>
<td>00.426</td>
</tr>
<tr>
<td>14</td>
<td>Yes</td>
<td>03</td>
<td>20</td>
<td>060</td>
<td>00.312</td>
</tr>
</tbody>
</table>

* As constraints explicit bounds on variables.

- n = Number of variables
- m = Number of constraints
- N = Number of iterates
- T = CPU time in seconds.
FIGURE 1: A Phase 1 iteration of EA