INTERVAL MATHEMATICS AND LINEAR PROGRAMMING APPLICATIONS

By

ZEYNEP AYSEGUL KARACAL

Bachelor of Science

Middle East Technical University

Ankara, Turkey

1982

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May 1988

INTERVAL MATHEMATICS AND
LINEAR PROGRAMMING APPLICATIONS

Thesis Approved:

J P Chandler
Thesis Adviser

D. E. Hedrick

M J Folk

Norman N. Durham
Dean of Graduate College

PREFACE

This thesis surveys the application of interval arithmetic to linear programming problems and presents an algorithm for solution of interval linear programming problems.

I would like to express my thanks to my advisor Dr. J.P. Chandler for his intelligent guidance and encouragement.

I am also thankful to my other comittee members Dr. Folk and Dr. Hedrick for their advice.

I am very grateful to my husband,Cem, and my parents, Mr. and Mrs. Izgu for their constant support, encouragement and understanding.

I wish to dedicate this thesis to my daughter, Sıla Gizem who made everything so worthwhile.

TABLE OF CONTENTS

Chapter		Page
I.	INTRODUCTION	1
II.	INTERVAL ARITHMETIC AND A SURVEY OF APPLICATIONS OF INTERVAL ARITHMETIC	4
III.	LINEAR PROGRAMMING	20
IV.	A METHOD FOR THE SOLUTION OF INTERVAL LINEAR PROGRAMMING PROBLEMS	34
V.	TESTING	43
VI.	SUMMARY AND CONCLUSIONS	48
VII.	SUGGESTIONS FOR FURTHER STUDY	51
BIBLIOGRAPHY		52
APPENDIX A - DEFINITIONS		56
APPENDIX B - PROGRAM LISTING		58

LIST OF TABLES

Table		Page
I.	Sign analysis of multiplication	8
II.	Determination of bounds for multiplication operation	13
III.	Computation of objective function at extreme points	22
IV.	Basic variables and ratios	30

LIST OF FIGURES

Figure		Page
1.	Intersection and Union of Two Intervals	5
2.	Representation of Rounding Endpoints	12
3.	A Nonconvex Solution Space	17
4.	Solution Space	21
5.	Graphical Representation of Solution Space	30
6.	Unbounded Solution	32
7.	Alternative Optimal Solution	33
8.	Adjacent Extreme Points	38
9.	Flowchart of the Algorithm	42
10.	Solution Space of Test Problem 1	44
11.	Optimum Solution of Test Problem 1	45

CHAPTER I

INTRODUCTION

Interval mathematics is a branch of applied mathematics. It has grown during the past two decades. Applications of interval mathematics which have been reported to date include diverse areas such as mathematical programming, operator equations, algebraic systems, even the re-entry of a spaceship into the earth's atmosphere.

In recent years there has been a growing interest in developing methods to solve interval Linear Programming (LP) problems. Various approaches have been suggested. Many of these approaches are based on methods already available for LP.

At present there is a considerable interest in the applications of interval mathematics to various areas including linear programming. A survey of recent developments in applications of interval mathematics is presented in chapter II. A summary of interval arithmetic operations and realization of interval operations on a computer are also given in chapter II. Chapter III gives an algebraic procedure for solving linear problems called simplex method. The geometric interpretation of the problem, and necessary iterations are presented with examples.

A linear programming problem is in the form of

$$\text{Maximize} \quad C^T x$$

$$\text{Subject to} \quad Ax \leqslant b \quad , \quad x \geqslant 0$$

The solution of this system can be obtained by using well-known simplex method. But the solution of LP problem may not be straight forward if the parameters are intervals. When the constraint set has only lower and upper bounds, LP problem is called Interval Programming problem. The problem becomes even more complex when all parameters are intervals. Interval Linear Programming problem can be defined as

$$\text{Maximize} \quad PZ$$

$$\text{Subject to} \quad AZ \leqslant B$$

where, A is mxm matrix with interval coefficents, B is m dimensional interval vector, and P is n dimensional interval vector.

In the light of acquired knowledge from literature survey, Krawczyk's method [40] which obtains an interval vector containing exact solution to ILP from an approximate solution of LP problem is studied. The cases which Krawczyk's method gives up or terminates are examined. Necessary modification is added to prevent the algorithm from giving up with no results.

To find the solution set for a given system, a software package based on FORTRAN is developed. Also a small interval package containing basic operations (*, /, +, -) is written to perform interval arithmetic operations on a computer and this package is used in the software whenever an interval operation becomes necessary.

A study of Krawczyk's method for the solution of interval linear programming problems and modification of the method is presented in chapter IV. The performance of the modified method is tested using test problems given in chapter V. Necessary mathematical definitions are given in Appendix A. Appendix B contains program listing.

CHAPTER II

INTERVAL ARITHMETIC AND A SURVEY OF APPLICATIONS OF INTERVAL ARITHMETIC

Finite arithmetic in computers and increasing demand of computers caused the development of a structure called interval analysis or ,later, interval mathematics. Interval analysis is a new branch of applied mathematics. It is an approach to computing which treats an interval of real numbers as a new kind of number represented by a pair of real numbers. An arithmetic introduced for such numbers is called interval arithmetic.

An interval is defined by its endpoints $\underline{X}$, $\overline{X}$, where $\underline{X} \leqslant \overline{X}$. Thus, $X = [\ \underline{X}, \overline{X}\ ]$. A Real number a is defined with the degenerate interval [a,a] having equal lower and upper endpoints; the term degenerate comes from the topological definition of a degenerate set as being a set consisting of a single point. The space of real numbers is regarded as a subspace of the space of intervals when the real number is defined as a degenerate interval. Thus, interval arithmetic includes real arithmetic as a special case. In other words, an interval number is a set of real numbers. Therefore, set theoretic operations can be applied to intervals. .

Two intervals are equal if their corresponding endpoints are equal. Thus $X = Y$ if $\underline{X} = \underline{Y}$ and $\overline{X} = \overline{Y}$.

The intersection of two intervals X and Y is empty, if either $\underline{X} > \overline{Y}$ or $\overline{X} < \underline{Y}$. The intersection of two interval is

$$X \cap Y = [\ \max(\underline{X}, \underline{Y})\ ,\ \min(\overline{X}, \overline{Y})\ ].$$

Figure 1 presents the geometric interpretation of intersection of two intervals. The intervals $X=[-1,4]$ and $Y=[2,6]$ will be used for numerical examples. So the intersection of $X \cap Y = [2,4]$ (See Figure 1.a).

If two intervals X and Y have nonempty intersection, their union is

$$X \cup Y = [\ \min(\underline{X}, \underline{Y})\ ,\ \max(\overline{X}, \overline{Y})\ ].$$

As it can be interpretted from Figure 1.c, $X \cup Y = [-1,6]$.

Set inclusion can be defined for intervals as

$$X \subseteq Y \text{ if and only if } \underline{Y} \leqslant \underline{X} \text{ and } \overline{X} \leqslant \overline{Y}.$$

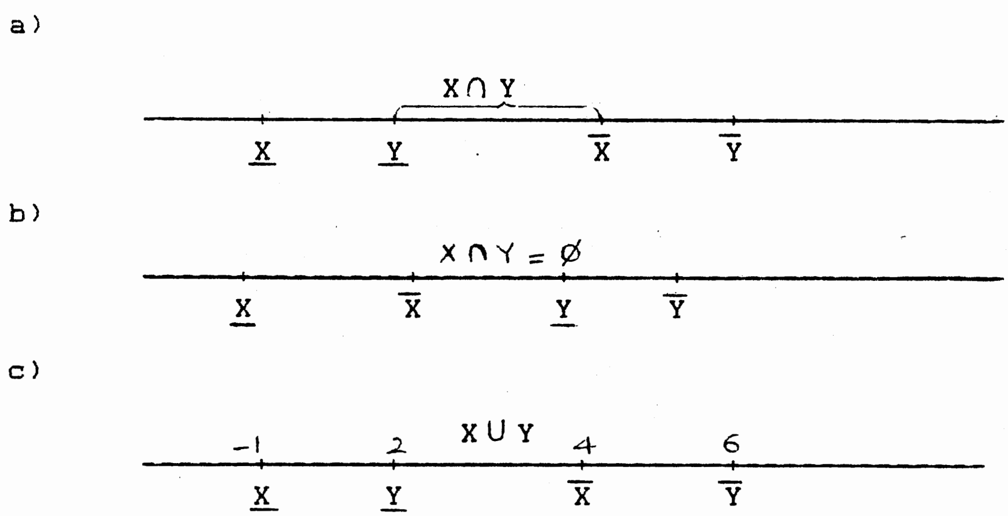

Figure 1. Intersection and Union of Two Intervals

The width of an interval is defined by

$$w(X) = \overline{X} - \underline{X} \quad \text{(e.g. } W(X)= 4-(-1)=5 \text{)},$$

the width of interval vector $X = (X_1, X_2, \ldots, X_n)$ is

$$w(X) = \max(w(X_1), w(X_2), \ldots, w(X_n)).$$

Absolute value of an interval is

$$|X| = \max(|\underline{X}|, |\overline{X}|) \quad \text{(e.g. } |X|= \max(|-1|,|4|)=4).$$

Midpoint of an interval is

$$m(X) = (\underline{X} + \overline{X})/2 \text{ , for given X } m(X) \text{ is } (-1+4)/2=1.5.$$

The vector norm for interval vectors $X = (X_1, X_2, \ldots, X_n)$ is

$$||X|| = \max(|X_1|, \ldots, |X_n|).$$

The rules of interval arithmetic are given in [32] and [33]. Let X and Y be intervals. The sum of two intervals is again an interval.

$$X + Y = [a,b] + [c,d] = [a+c \text{ , } b+d]$$

As an example X + Y= [-1,4]+[2,6] = [1,10].

The negative of an interval is defined as

$-X = -[a,b] = [-b,-a]$. Therefore the difference of two interval is

$$X - Y = X + [-Y] = [\, \underline{X}-\overline{Y} \text{ , } \overline{X}-\underline{Y} \,] \quad \text{for example}$$

$$X - Y = [-1,4]+[-6,-2] = [-7,2].$$

Before going further, it is beneficial to explain one shortcomings of interval arithmetic. Interval arithmetic has no additive inverse. If an interval X = [a,b] is subtracted from itself , the result is

$$X - X = [a-b \text{ , } b-a].$$

That is, $X - X \neq [0,0]$, different from the expected result (unless a=b) ,for example X= [-1,4] and X-X= [-5,5] . This kind of shortcoming causes interval arithmetic to produce nonsharp bounds. In general, if a given interval X occurs more than once in a computation, results are more likely to fail to be sharp. A generalized interval arithmeic which reduces this effect is introduced by Hansen [20].

The product of two intervals, X.Y, is again an interval whose endpoints can be computed from

$$X*Y = \min(\underline{X}\underline{Y}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{X}\overline{Y})$$

$$X*Y = \max(\underline{X}\underline{Y}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{X}\overline{Y})$$

The reciprocal of an interval is

$$1/X = [\, 1/\overline{X} \,,\, 1/\underline{X} \,] \text{ if } \underline{X} > 0 \text{ or } \overline{X} < 0.$$

If an interval X contains zero the set of 1/X is unbounded and can not be represented as an interval.

The quotient of two intervals can be defined as

$$X/Y = X*(1/Y) \text{ if } 0 \text{ is not contained in } Y.$$

The operation * and / can be simplified computationally by examining the signs of endpoints. The results for multiplication which are reduced to 9 cases are given in Table I.

TABLE I

SIGN ANALYSIS OF MULTIPLICATION

Case no	A $[\underline{A}, \overline{A}]$	B $[\underline{B}, \overline{B}]$	A * B =
1	$[\geq 0, \geq 0]$	$[\geq 0, \geq 0]$	$[\ \underline{A}*\underline{B}\ ,\ \overline{A}*\overline{B}\ ]$
2	$[< 0, \geq 0]$	$[\geq 0, \geq 0]$	$[\ \underline{A}*\overline{B}\ ,\ \overline{A}*\overline{B}\ ]$
3	$[\leq 0, \leq 0]$	$[\geq 0, \geq 0]$	$[\ \underline{A}*\overline{B}\ ,\ \overline{A}*\underline{B}\ ]$
4	$[\leq 0, \leq 0]$	$[< 0, \geq 0]$	$[\ \underline{A}*\overline{B}\ ,\ \underline{A}*\underline{B}\ ]$
5	$[\geq 0, \geq 0]$	$[< 0, \geq 0]$	$[\ \overline{A}*\underline{B}\ ,\ \overline{A}*\overline{B}\ ]$
6	$[\geq 0, \geq 0]$	$[\leq 0, \leq 0]$	$[\ \overline{A}*\underline{B}\ ,\ \underline{A}*\overline{B}\ ]$
7	$[< 0, \geq 0]$	$[\leq 0, \leq 0]$	$[\ \overline{A}*\underline{B}\ ,\ \underline{A}*\underline{B}\ ]$
8	$[\leq 0, \leq 0]$	$[\leq 0, \leq 0]$	$[\ \overline{A}*\overline{B}\ ,\ \underline{A}*\underline{B}\ ]$
9	$[< 0, \geq 0]$	$[< 0, \geq 0]$	$[\min(\underline{A}\overline{B}, \overline{A}\underline{B}), \max(\underline{A}\underline{B}, \overline{A}\overline{B})]$

In Table I, the notation ≤ 0 means the endpoint is negative. Similarly the notation ≥ 0 means that the endpoint is positive. For example the multiplication of X and Y can be computed from case 2 where

$$X * Y = [-1,4] * [2,6] = [-6,24].$$

All above operations contain real (infinite precision) arithmetic. Therefore they can not be implemented on a computer. In practice, real arithmetic on computers is impossible using floating point instructions because of limited precision and roundoff errors. When real arithmetic operations are carried on a computer, computations will be done in floating point arithmetic. Real numbers are approximated by floating point systems with a fixed number of digits in the mantissa. Any real number y can be written as

$$y = + .d_1 d_2 \ldots\ldots d_s d_{s+1} \ldots \beta^e$$

The IBM 3081 system which is used in this research represents y by chopping of all digits after the first s to get

$$fl(y) = +.d_1 d_2 \ldots\ldots d_s \beta^e$$

where s = 6 (for single precision) β = 16 and -64 < e < 63.

The unit roundoff error is defined

$$EPS = \beta^{1-s}$$

if y is a real number, then floating point y can be defined

$$fl(y) = y(1 + \delta) \quad \text{where} \quad |\delta| \leq EPS.$$

It is possible to find intervals containing the exact arithmetic results. Even if a mathematical equation can be solved exactly, it will still give an approximate description of the behavior of the real system which the mathematical equation is supposed to model. Basically, problems can be divided into two categories, problems with inexact and with exact initial data. In the first category, usually data are allowed to vary over an interval. In the category of problems where exact initial data are given, interval analysis is used to develop methods which generate convergent sequences of bounds converging to the solutions under comparatively weak conditions. When performing interval arithmetic on a digital computer, it is necessary to deal with roundoff error. When performing calculations, the basic properties of solutions such as monotonicty of sequential inclusion or convergency are assumed to be preserved. Therefore, the treatment of the machine interval arithmetic is limited to the realization of the interval operations on computer. Under these conditions , if interval arithmetic operations are performed just by plugging endpoints into equations , the procedure will be imprecise and will often produce much more pessimistic results than necessary. To overcome this deficiency , computer interval arithmetic is developed and its properties are discussed in [15],[27], and [47]. In order to perform computer interval arithmetic directed rounding, which has two parts (upward and downward directed rounding), is introduced in [47]. If x is a real number, upward rounding maps x to the smallest machine representable number greater than or equal to x. Downward rounding maps x to the greatest machine representable number less than or equal to x. To illustrate the distinction between interval arithmetic and computer interval arithmetic , the following example is taken from [31]. Let

$$X = [-.613*10^{-2}, -.610*10^{-2}]$$

$$Y = [+.100*10^{1}, +.300*10^{1}]$$

$$Z = X(1 + 1/Y)$$

Z can be computed using exact interval arithmetic.

$$Z = X(1 + 1/[1,3])$$

$$Z = X(1+[1/3,1])$$

$$= X[4/3,2]$$

$$= [-.1226*10^{-1}, -.8133...*10^{-2}]$$

When Z is computed using computer interval arithmetic based on 3 decimal digit mantissas and floating number representation , Z will be

$$1 = [+.100*10^{1}, +.100*10^{1}],$$

$$1/Y \quad [+.333*10^{0}, +.100*10^{1}],$$

$$1+ 1/Y \quad [+.133*10^{1}, +.200*10^{1}],$$

$$X(1+1/Y) \quad [-.123*10^{-1}, -.811*10^{-2}].$$

Therefore the final result contains the exact value of Z. Unfortunately these kinds of developments are mostly machine dependent. Another obstacle to experimentation with interval arithmetic is that supporting software may not be available. Realization of computer interval arithmetic in Algol 60 can be found in [1]. Since there are no software and hardware support for performing computer interval arithmetic , tje bounds of interval is expanded by EPS that is computed in a subroutine. An interval $X = [a,b]$,if $a \geq 0$, will be represented on computer

$$X = [a*(1-EPS) , b*(1+EPS)].$$

Thus, the lower bound will be shifted to left by epsilon*a and upper bound will be shifted to right by epsilon*b. Figure 2 represents different possibilities of rounding endpoints of an interval.

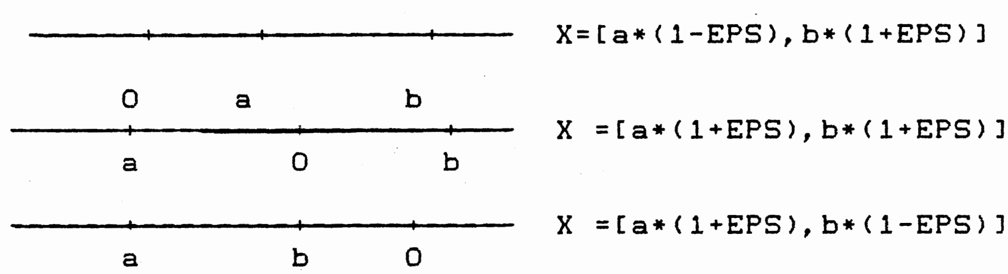

Figure 2. Representation of rounding endpoints

This representation is not valid for the overflow and underflow cases. But it is enough to perform operations used in this study. An operation for an interval wil

$$[a, b] \odot [c, d] = [\ a \odot c\ (1 \pm \text{EPS}),\ b \odot d\ (1 \pm \text{EPS})\ ]$$

For multiplication and division, the determination of bounds are not so trivial. The signs of the resulting interval must be considered by calculating end points. Table II shows how to expand the bounds of the resulting interval for multiplication operation. One drawback of this approach is unnecessary expansion of the width of the result interval when the result of the end point calculation was already a machine representable number.

TABLE II

DETERMINATION OF BOUNDS FOR MULTIPLICATION OPERATION

[A, B] * [C, D] = [E, F]		Implementation on the Computer	
Sign of E	Sign of F		
≥ 0	≥ 0	E [1 - EPS]	F [1 + EPS]
< 0	< 0	E [1 + EPS]	F [1 - EPS]
< 0	≥ 0	E [1 + EPS]	F [1 + EPS]

Since this approach will be used in the application of interval arithmetic to linear programming models, the worst case(representation of exact numbers) will not happen because of the characteristics of the model and coefficents.

A small interval arithmetic package containing only four arithmetic operations (*, /, +, -) is developed. This package employs the approach mentioned above. Each operation must be performed by a call on one of the subprograms. This means that the user must parse every expression himself and write his program to perform the calculation. Each subprogram requires lower and upper bounds of two intervals, and returns lower and upper bounds of the result interval and an error flag whenever it is necessary.

An interval matrix is a matrix whose coefficents are intervals. Let A and B be two mxn interval matrices with coefficients A_{ij} and B_{ij}, respectively. Then,

$$A \pm B = (A_{ij} + B_{ij})$$

defines an interval matrix addition and subtraction, respectively. Let A be an mxr interval matrix and let B be an rxn matrix. Then,

$$A * B = (\sum_{s=1}^{r} A_{is} \; B_{sj})$$

defines an interval matrix multiplication.

Some useful properties for operations on interval matrices are given below.

$$A + B = B + A$$

$$A + (B + C) = (A + B) + C$$

$A + O = O + A = A$ zero matrix

$A I = I A = A$ I = Unit matrix

$$(A + B)\,C \quad A\,C + B\,C$$

$$C\,(A + B) \quad C\,A + C\,B$$

$$(A + B)\,C_R = A\,C_R + B\,C_R$$

$$C_R\,(A + B) = C_R\,A + C_R\,B$$

where C_R is a real (point) matrix.

The starting point for the application of interval mathematics was to automate computational error analysis. But during the past two decades, interval mathematics has grown to include a much broader range of topics. The applications of interval arithmetic ranges from purely theoretical topics to computational methods, even computer architecture. Thus, it is impossible to give a complete representation and description of what has been developed recently under interval analysis or summarize all applications of interval mathematics. In this chapter, only the basic methods of interval arithmetics and remarkable applications will be mentioned.

The main objectives of the application of interval mathematics to computing are to find sets containing unknown solutions, to make these sets as small as possible, and to do all this as efficiently as possible. To achieve these objectives, point-to-point mappings are replaced by set-to-set mappings.

Some of the interval algorithms are extensions of corresponding real algorithms. On the other hand, some of them are very different. For example, many interval algorithms use the intersection of two intervals while there is no corresponding operation for real numbers.

Although interval arithmetic is known for real numbers, many of the properties and results for real interval arithmetic can be carried over to a complex interval arithmetic. The arithmetic in complex space that reduces to real arithmetic is introduced and the properties of this arithmetic are discussed in [43]. A method using circular arithmetic for finding the complex zeros of polynomials with error bounds is presented by Gargantini [9].

There has been a rapid development of new methods used for nonlinear problems. Various studies done for nonlinear equations and nonlinear optimization are discussed in [21],[23],[25],[31] and [45].

Hansen [18] has been developed a method to invert an interval matrix. He defined the set

$$(A^I)^{-1} = \{ A^{-1} : a_{ij} \in [\underline{A}_{ij}, \overline{A}_{ij}] \supset AA^{-1} = I \}$$

and computed $(A^I)^{-1}$ approximately and thus, bounded the errors due to roundoff. Hansen's method minimizes the loss of accuracy inherent in direct use of interval arithmetic. First, the problem is solved approximately in ordinary mathematic. Then ,it is reduced to a problem in which the solution is this approximate solution plus small quantities.

Interval mathematics is applied to linear algebraic systems in the form of

$$Ax = b.$$

Direct methods such as Gaussian elimination and indirect (iterative) methods are discussed in [31]. In an iterative method, the sequence of intervals is generated

$$X^{(k+1)} = \{ Yb + EX^{(k)} \} \cap X^{(k)} \quad , \; k=0,1,2,...$$

with

$$X^{(0)} = [-1,1] \|Yb\| / (1- E) \quad , \; i=1,2,...n$$

if $\|E\| < 1$, where E= I-YA and Y is an approximate inverse of m(A). The sequence will converge in a finite number of steps to an interval vectorcontaining the set of solutions to Ax=b. Gay[13] discussed different methods for solving linear equations Ax=b where A is an interval matrix and b is an interval vector. He proposed the ways of finding $x \subset \mathbb{R}^n$ such that

$$x^* = \{ G^{-1} h : \underline{A} \leqslant G \leqslant \overline{A} \;\; \underline{b} \leqslant h \leqslant \overline{b} \} \subset x$$

Oettli [34] shows that x* is the union of at most 2 convex polyhedra. When A and b have interval components, the solution set may be complicated and nonconvex. The following system is an example of nonconvex, star shaped solution space.

$$Ax = b$$

where

$$A = \begin{pmatrix} [\ 2,4] & [-2,1] \\ [-1,2] & [\ 2,4] \end{pmatrix} \qquad b = \begin{pmatrix} [-2,2] \\ [-2,2] \end{pmatrix}$$

The solution space is shown in Figure 3.

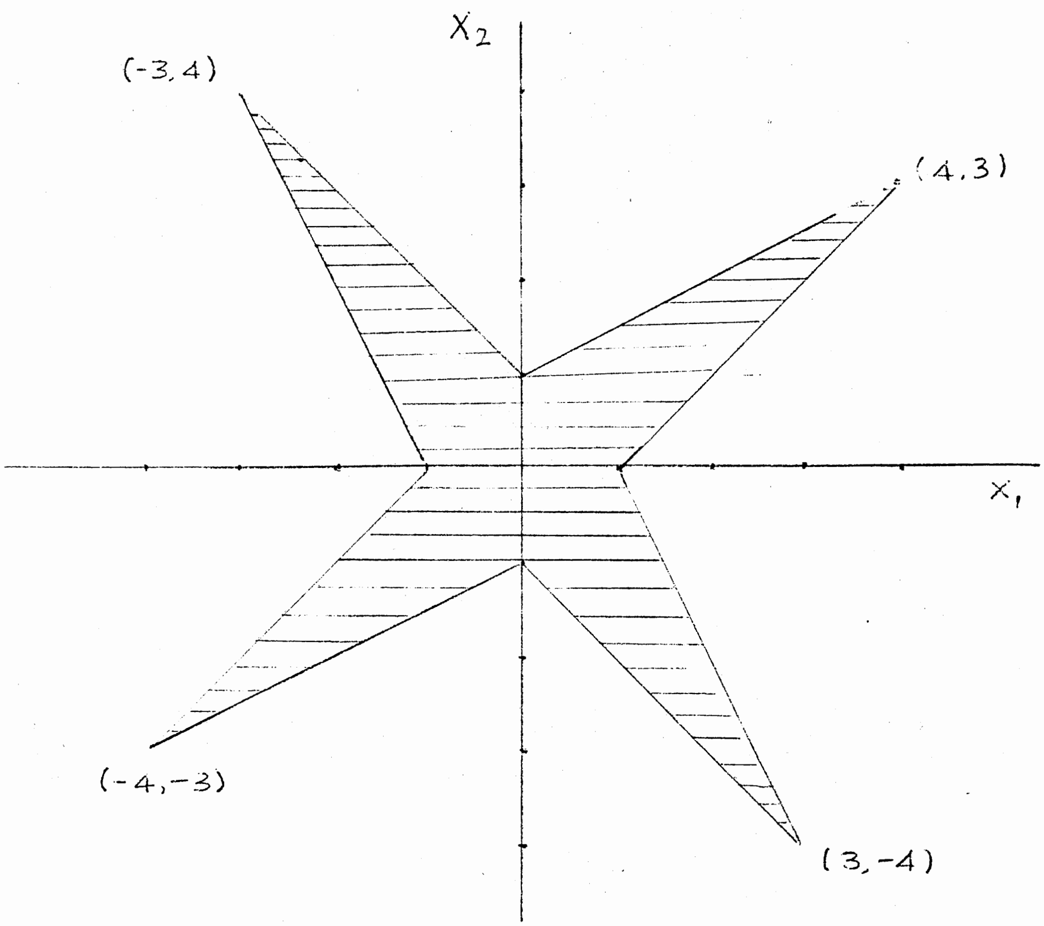

Figure 3. A nonconvex Solution Space

Finally, the applications of interval mathematics to linear programming problems will be considered. In literature an interval version of a linear programming problem is defined as

$$\begin{array}{ll} \text{Maximize} & c^T x \\ \text{s.to} & b^- \leq Ax \leq b^+ \end{array}$$

where the matrix A, vectors b^- , b^+ and c are given. There are several algorithms available that are primal or dual. A primal algorithm is developed in [19] to solve interval programming. The algorithm starts with a feasible solution and produces an extreme point to an interval problem resulting a better objective value. Then the algorithm proceeds by moving along adjacent extreme points until an optimal extreme point is generated. A detailed comparison of the simplex method for linear programming problem with the primal algorithm of [3] has been made and resulted that the algorithms are identical in the sense that the same sequence of extreme points can be generated by either algorithm [17]. The difference between methods is strategic in terms of choosing variables and resolving ties in the case of degeneracy. A dual method , SUBOPT , is developed by Ben_Israel and Robers [40]. Actually interval programming problems can be solved by ordinary linear programming . It has been claimed that interval programming problems occurs frequently enough in applications and to convert them to ordinary LP problems may increase the problem size. Also interval pogramming focuses attention on the role of bounds on the variables in a given model. Stewart [45] developed a revised simplex method to find the upper bounds for maximization problem.

So far linear programming problems with the constraints having upper and the lower bounds are considered. Although various aspects of rounding errors in linear programming received attention, LP problems whose parameters (both in constraints and in the objective function) are prescribed by intervals has not received much attention. Krawczyk [26] has applied interval mathematics methods to the simplex method for solving LP problem whose parameters are intervals. Necessary and sufficient conditions for strong solvability of interval linear programming problems are dicussed by Rohn[42]. A duality theorem and optimality criterion are also developed by Rohn[41].

CHAPTER III

LINEAR PROGRAMMING

Linear pprogramming problem is the minimization or the maximization of a linear function overa polyhedral set [see Appendix A]. In other words ,it is to find the way of efficient allocation of limited resources to known activities with the desired objective value. The functions representing constraints and the objective are linear.

A Linear Programming problem is in the form of

$$\text{Min (Max)} \quad c^T x$$

$$\text{subject to} \quad x \in S.$$

The set S is called the constraint set and c x is called the objective function. Although the well kown simplex method is used for solving LP problems, a graphical method is helpful for demonstrating basic concepts. Let's start with the following system.

$$\text{Max} \quad x_0 = 4*X1 + 3*X2$$

s.to

$$2*X1 + 3*X2 \leq 6 \qquad (1)$$

$$2*X1 + X2 \leq 4 \qquad (2)$$

$$X1, X2 \geq 0.$$

Figure 4 shows the solution space which is bounded by constraints 1 and 2.

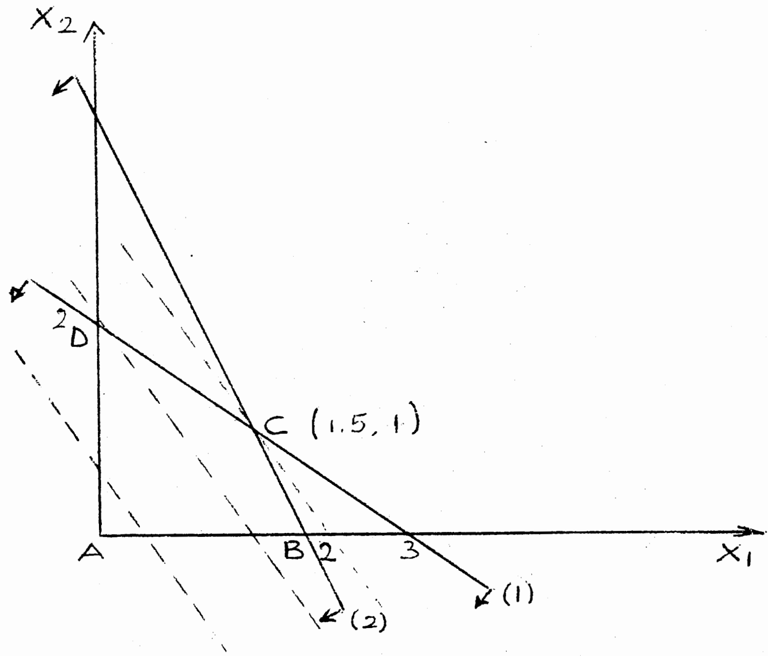

Figure 4. Solution Space

Every point within or on the boundaries of the area ABCD satisfies all the constraints. The optimum solution will be the point which maximizes objective function. In Figure 4, the contour lines of objective function are plotted. If x is increased beyond 9, the contour will not pass through feasible region. So x = 9 gives the optimum resultant point x = (1.5, 1). The value of objective function will be

x = 4(1.5)+3(1) = 9.

Corner points of feasible region are known as extreme points. The optimum will be always found at one of those extreme points. Table IV shows the values of objective function at extreme points.

TABLE III

COMPUTATION OF OBJECTIVE FUNCTION AT EXTREME POINTS

Corner point	Coordinates	Objective Function
A	X1=0 X2=0	0
B	X1=2 X2=0	8
C	X1=1.5 X2=1	9
D	X1=0 X2=2	6

For this problem, it is enough to evaluate objective function at the extreme point s and find the best result. But it is not a practical procedure for higher dimensions and a large number of variables.

To start with the simplex method , the solution space must be represented by the standard form [see Appendix A]. The standard form of the problem is

$$\text{Max} \quad x_0 = 4*X1 + 3*X2$$

$$\text{s.to} \quad 2*X1 + 3*X2 + X3 = 6$$

$$2X1 + X2 + X4 = 4$$

$$Xi \geq 0 \quad i=1,..4$$

X3 and X4 are known as slack variables. Now the system has two equations in four unknowns. The basic solution for a set of m linear equation in n unknowns is found by setting n-m variables to zero and solving the m equation m unknown system. n-m variables are called nonbasic , m variables are called basic variables.

In matrix definition the problem can be formulated as follow;

$$Ax = b$$

where A is an mxm matrix and b is m vector. For the given problem,

$$A = \begin{pmatrix} 2 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

After rearranging the columns of A, let A = [B,N] where B is mxm invertible matrix and N is mx(n-m) matrix and the solution point is

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} \qquad \text{where} \qquad \begin{aligned} x &= B^{-1} b, \\ x &= 0. \end{aligned}$$

x is called basic feasible soluiton if x $\geq$ 0 . B matrix is known as basic matrix (or basis) , N is called nonbasic matrix. The components of x_B vector are called nonbasic variables.

Let's consider the matrix A,

$$A = [a_1, a_2, a_3, a_4] = \begin{bmatrix} 2 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

Basic solution will correspond to finding $B_{2\times2}$ with $B^{-1}b>0$. All possible combinations of a_1, a_2, a_3, a_4 which gives 2x2 invertible matrix must be computed. Those computations are shown below.

1. $B = [a_1, a_2] = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix}$

$$x_B = B^{-1} b = \begin{bmatrix} -1/4 & 3/4 \\ 2/4 & -2/4 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 6/4 \\ 1 \end{bmatrix} \qquad x = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

2. $B = [a_1, a_3] = \begin{bmatrix} 2 & 1 \\ 2 & 0 \end{bmatrix}$

$$x_B = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = B^{-1} b = \begin{bmatrix} 0 & 1/2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \qquad x = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

3. $B = [a_1, a_4] = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix}$

$$x_B = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = B^{-1} b = \begin{bmatrix} 1/2 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \qquad x = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

4. $B = [a_2, a_3] = \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix}$

$$x_B = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = B^{-1} b = \begin{bmatrix} 0 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 \\ -6 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

5. $B = [a_2, a_4] = \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix}$

$$x_B = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = B^{-1} b = \begin{bmatrix} 1/3 & 0 \\ -1/3 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

6. $B = [a_3, a_4] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$x_B = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = B^{-1} b = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

The points corresponding to 1,2,5,6 are basic feasible solutions . The points obtained by 3 and 4 are not feasible, because they violate nonnegativity restrictions. Therefore four basic feasible solutions are

$$x_1 = \begin{bmatrix} 1.5 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad x_2 = \begin{bmatrix} 2 \\ 0 \\ 2 \\ 0 \end{bmatrix} \qquad x_3 = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix} \qquad x_4 = \begin{bmatrix} 0 \\ 0 \\ 6 \\ 4 \end{bmatrix}$$

When these solutions are projected in E^2 , the following points will be obtained.

$$x_1 = \begin{bmatrix} 1.5 \\ 1 \end{bmatrix} \qquad x_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \qquad x_3 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \qquad x_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

These points are actually extreme points that are found graphically. In general number of basic feasible solutions is less than or equal to

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

For example problemnumber of basic feasible solutions computed from above equation is 6.

Since the number of basic feasible soluiotns is bounded by C(n,m), all basic feasible solutions may be listed and the one with the best objective function value may be chosen. This procedure is not practical and satisfactory for many reasons. First of all the number of basic feasible solutions may be very large. Second, this procedure does not give an information about the problem's nature whether it is bounded or not. Also the feasible region may be empty. In that case this stituation is not realized until all computations of B b are done. Therefore the simplex method is the best way. It moves from one extreme point to another extreme point with a better objective function value and discovers whether the feasible region is empty or not. The foundation of simplex method is to find a new basic feasible solution with a better objective value.

Consider the following LP problem.

$$\text{Max} \quad c^T x$$

$$\text{s.to} \quad Ax = b, \quad x \geq 0$$

$(B^{-1}b>0)^T$ is an initial basic feasible solution with objective value z_0 is given by

$$z_0 = c\,(B^{-1}b \quad 0)^T = c_B B^{-1} b$$

Since $b=Ax = Bx_B + Nx_N$, then

$$x_B = B^{-1}b - B^{-1}Nx_N$$

$$x_B = B^{-1}b - \sum B^{-1} a_J x_J$$

Let's substitute z_0 and x_B into objective function. We find

$$\begin{aligned} z &= cx \\ &= c_B x_B + c_N x_N \\ &= c_B(B^{-1}b - \sum B^{-1} a_J x_J) + \sum c_J x_J \\ &= z_0 - \sum (z_J - c_J) x_J \end{aligned}$$

where $z_J = c_B B^{-1} a_J$ for each nonbasic variables. Since we are maximizing , it will be to our benefit to increase x_J from zero whenever $z_J - c_J < 0$. Therefore we have to find the most negative $z_J - c_J$ value (suppose it is $z_k - c_k$). If x_k is increased, the current basic variables must be modified. Let $x_B = B^{-1}b - B^{-1}a_k x_k$, $y_k = B^{-1}a$ and $b = B^{-1}b$, then we can write basis variables as follows.

$$x_B = \begin{pmatrix} x_{B1} \\ x_{B2} \\ \cdot \\ \cdot \\ x_{Br} \\ \cdot \\ x_{Bm} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_r \\ \cdot \\ b_m \end{pmatrix} - \begin{pmatrix} y_{1k} \\ y_{2k} \\ \cdot \\ \cdot \\ y_{rk} \\ \cdot \\ y_{mk} \end{pmatrix} x_k$$

As it could be observed from above equation , x_k can not be increased infinitely. Because nonnegativity restrictions must be considered. In order to be able to preserve nonnegativity , x_k can be increased until the first point in the basis drops to zero (assume it is x). This point can be calculated from the following equation.

$$b_r / y_{rk} = \underset{1 \leqslant i \leqslant m}{\text{minimum}} \{ b_i / y_{ik} \; : \; y_{ik} > 0 \} = x_k .$$

x_k is called the entering variable, X_{Br} which drops to zero first is called blocking or leaving variable.

$$\text{Let} \quad A = \begin{bmatrix} 2 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix} \text{ and } \quad B = [a_3 , a_4] \text{ and } z_c = 0.$$

$$x_B = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = B^{-1} b = \begin{bmatrix} 6 \\ 4 \end{bmatrix} \qquad\qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} .$$

In order to improve this basic feasible soluiton , $z_j - c_j$ values for nonbasic variables must be calculated.

$$\begin{aligned} z_1 - c_1 &= c_B B^{-1} a_1 - c_1 \\ &= (0 \; 0) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} - 4 \\ &= -4 \end{aligned}$$

$$\begin{aligned} z_2 - c_2 &= c_B B^{-1} a_2 - c_2 \\ &= (0 \; 0) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} - 3 \\ &= -3 \end{aligned}$$

Since $z_1 - c_1$ is the most negative , the solution will be

$$x_B = B^{-1} b - B^{-1} a_1 x_1$$

$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \end{bmatrix} x_1$$

The value of x_1 will be 2 and $x_4 = 0$. The new basis $x_B = (x_3 , x_1)^T$ with the objective value is 8. This procedure can be repeated until all $z_j - c_j \geqslant 0$, then the optimum z^* will be obtained.

All necessary background about simplex method was given in previous pages. The steps of simplex method are given below.

STEP 1: Define variables and write mathematical statement of the problem.

STEP 2: Put the problem in standart form.

STEP 3: Add or subtract slack variables.

STEP 4: Assemble the coefficients into an initial simplex tableau. If an initial basis is present go to step 6, otherwise go to step 5.

STEP 5: If a complete identity matrix is not visible in the initial tableau ,append the missing unit vectors to the tableau and associate with very high cost in the objective function (M-Technique).

STEP 6: Evaluate the solution represented therein, as to whether it is optimal. Calculate $z_j - c_j$ for each column vector.

STEP 7: If all $z_j - c_j$ are nonnegative (for maximization problem), or nonpositive (for minimization) stop. Otherwise identify a column vector with a nonterminal $z_j - c_j$ as a vector to come into the basis(say P_k)

STEP 8: Calculate the ratios b_i / y_{ik} for each y_{ik} and choose the smallest i=r ,replace variable k with variable r.

STEP 9: If a_{rk} is the pivotal element and a'_{ij} is the updated value of a_{ij} the iteration will be accomplished this way.

$$a'_{rj} = a_{rj} / a_{rk}$$

$$a'_{iJ} = a_{iJ} - a_{iK}\ a'_{rJ}$$

Actually the above calculation is the process of creating a unit vector in column k with $a'_{rk} = 1$. It can be done by elementary row operations.

The example problem is solved by using simplex method.

Step 1 & 2: Mathematical model

Max 4*X1 + 3*X2

s.to 2*X1 + 3*X2 $\leqslant$ 6

2*X1 + X2 $\leqslant$ 4 , X1,X2 $\geqslant$0

Step 3:

Max 4*X1 + 3*X2

s.to 2*X1 + 3*X2 + X3 $\leqslant$ 6

2*X1 + X2 + X4 $\leqslant$ 4

Xi $\geqslant$ 0 ,i=1,..4

Step4: Initial Simplex Tableau

Iteration 1:

Basic	X1	X2	X3	X4	Solution
x_0	-4	-3	0	0	0
X3	2	3	1	0	6
X4	2	1	0	1	4

The graphical representation of of solution space is given in Figure 5.

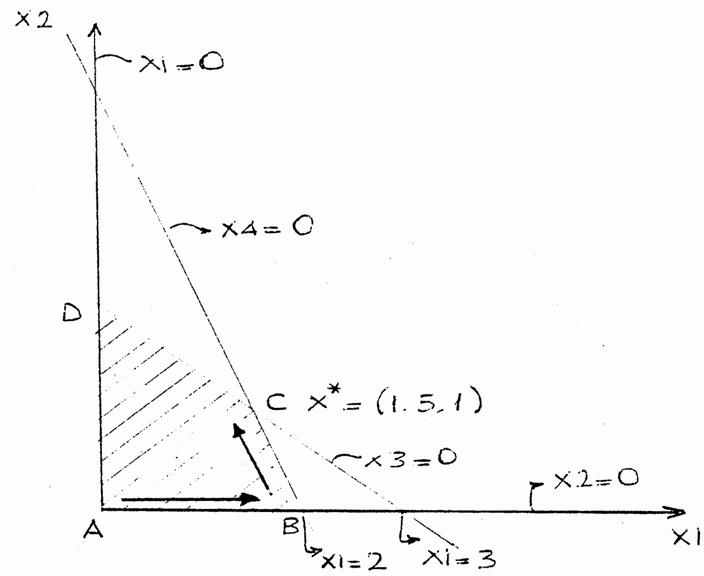

Figure 5. Graphical Representaion of Solution Space

Step 5: The initial solution is x_3 and x_4 .

Step 6 & 7:

An initial basic feasible solution is at point a with $x_1=x_2=0$ and $x_3=6$, $x_4=4$. Entering variable is the most negative coefficient in row x_0 from iteration 1. X1 is the entering variable with the value of -4. Table V represents basic variables with the ratios.

TABLE IV

BASIC VARIABLES AND RATIOS

Basic Var.	Solution	X1	Ratio
X3	6	2	6/2 = 3
X4	4	2	4/2 = 2 *

Since the ratio corresponding to the basic variable X4 is minimum, X4 is the leaving variable. The coefficient at the intersection of column X1 and row X4 is selected as pivot element.

Iteration 2:

Basic	X1	X2	X3	X4	Solution
X0	0	-1	0	2	8
X3	0	2	1	-1	2
X1	1	1/2	0	1/2	2

Iteration 3:

The coefficient of X2 is still negative , therefore X2 will be the entering variable and X3 will be leaving variable.

Basic	X1	X2	X3	X4	Solution
x_0	0	0	1/2	1/2	9
X2	0	1	1/2	-1/2	1
X1	1	0	-1/4	3/4	3/2

Since all coefficients are nonnegative, the optimum is reached with X1=1.5, X2=1 and z=9.

As it was mentioned before, the simplex method gives information about the nature of the problem. The solution is said to be degenerate when one or more basic variables becomes zero. An unbounded solution can occur when the solution space is unbounded. In that case the value of the objective function can be increased indefinitely. This case is illustrated in Figure 6.

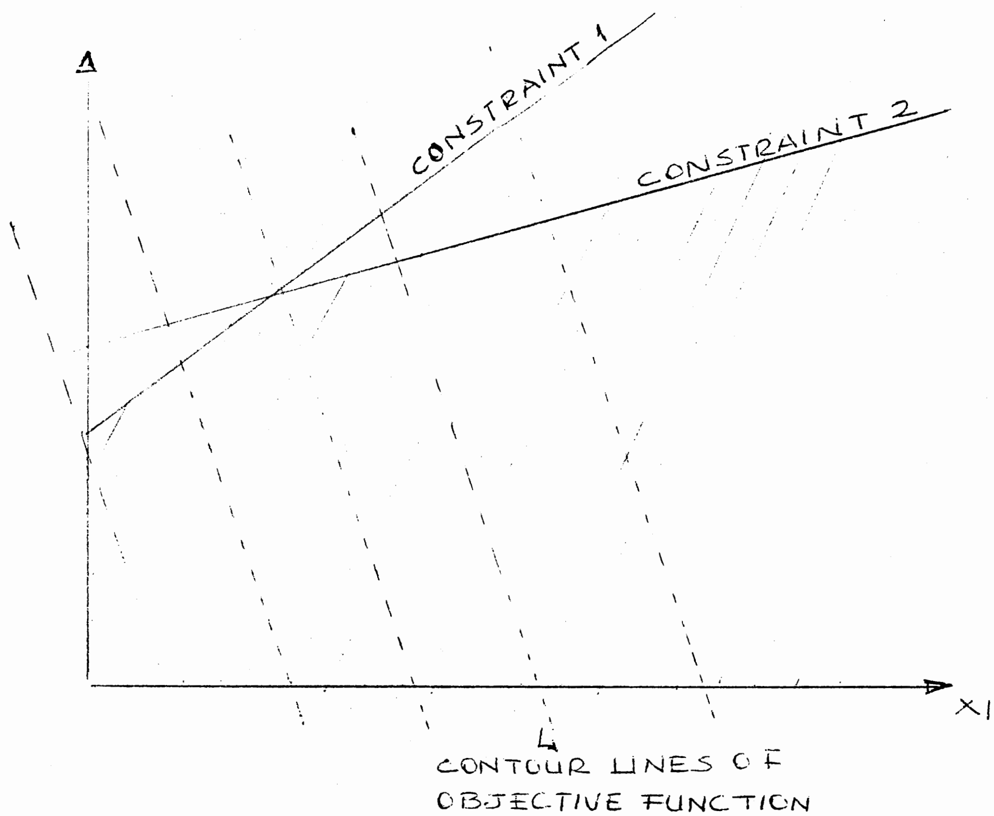

Figure 6. Unbounded Solution

Alternative optimal solution occurs when the objective function is parallel to a binding constraint. In such cases, problem has an infinite number of solutions with each solution yielding the same value of the objective function. Figure 7 an illustrates alternative optimal solution.

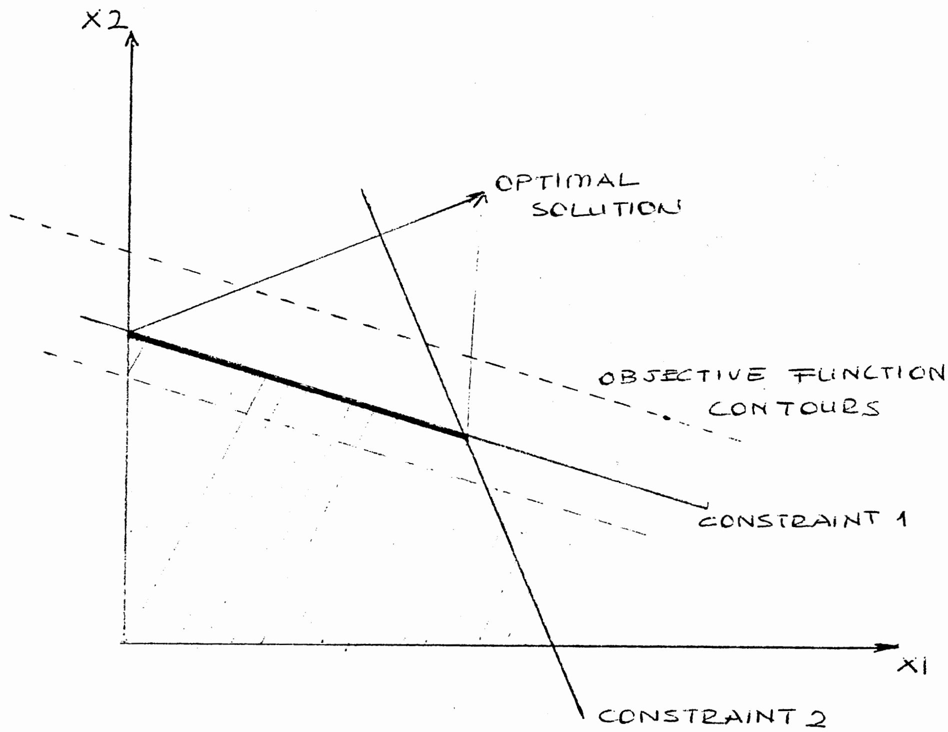

Figure 7. Alternative Optimal Solution

Finally ,nonexisting feasible solution occurs when the solution space is empty. In that case there is no point that satisfies all constraints.

CHAPTER IV

A METHOD FOR THE SOLUTION OF INTERVAL LINEAR PROGRAMMING PROBLEMS

A linear programming problem is the minimization or maximization of a linear function over a polyhedral set [see Appendix A]. The simplex method, which exploits extreme points and directions of the polyhedral set defining the problem, is widely used for the solution of LP problems. In this chapter, the solution of interval linear programming problem whose parameters are intervals will be discussed.

An interval linear programming problem is

$$\begin{aligned} &\text{Maximize} && PZ \\ &\text{subject to} && AZ \leq B \end{aligned}$$

where A is mxm interval matrix, P is n dimensional interval vector and B is m dimensional interval vector. The problem is to find an interval vector Z which contains the set of solutions to the above problem.

Before attempting to solve the problem, the solvability of the problem should be known. There are two ways to check this property. One is to apply the simplex methodto solve any LP problem which is a subsystem of a given ILP problem. The other one is to check all extremal subsytemsto determine whether they are feasible [see Appendix A]. The second approach requires checking of 2^p extremal subsystem (p is the number of rows of the matrix A).

The method developed by Krawczyk [26] consists of four parts:

(i) INITIAL APPROXIMATE SOLUTION

(ii) TEST FOR BASIS CHANGE

(iii) ALGORITHM FOR SOLUTION

(iv) TEST FOR NONNEGATIVITY

(i) INITIAL APPROXIMATE SOLUTION

To find an initial basis, an approximate solution is required. Particular matrices $p \in P$, $b \in B$ and $A_r \in A$ must be chosen. The problem for this particular selection will be

$$\begin{aligned} &\text{Maximize} \quad px \\ &\text{subject to} \quad A\,x = b \\ &\qquad\qquad\quad x \geq 0 \end{aligned}$$

where A_r is mxm real matrix , p and b are real vectors. $\bar{z}$ is the solution of above problem.

Let S be the index set of all basis variables of the solution z. After rearranging x vector, let $x^T = (x', x'')$. x' is an m dimensional vector consisting of basis components of n dimensional vector x. Similarly x'' is n-m dimensional vector consisting of all nonbasis components of x.

The following notations are used.

A_r' denotes an mxm matrix consisting of basis columns of A_r.

A_r'' mx(n-m) matrix consisting of the nonbasis columns of A_r

A' mxm interval matrix consisting of basis columns of A

A'' mx(n-m) interval matrix consisting of nonbasis columns of A

P' m dimensional interval vector consisting of objective coefficients of basis variables

P'' (n-m) dimensional interval vector consisting of objective coefficients of nonbasis variables

p' m dimensional real vector of basis variables of p

p'' (n-m) dimensional real vector

$A_r'^T$ transpose of A_r'

A''^T transpose of A''

(ii) TEST FOR BASIS CHANGE

To check the applicability of the method to a given problem, it must be determined that the set of all solutions has the same basis as $\bar{z}$.

To determine the possible basis changes, $z_J - c_J$ values for all nonbasic variables must be calculated. Consider the following linear programming problem,

$$\text{Maximize} \quad Q = c \;\; x$$

$$\text{subject to} \quad A_r x = b$$
$$x \geq 0$$

There is a basic feasible solution. The objective value z is given by

$$z_0 = c \begin{pmatrix} (A_r')^{-1} b \\ 0 \end{pmatrix} = c'(A_r')^{-1} b$$

The objective function can be written in terms of

$$Q = cx$$
$$= c'x' + c''x''$$

$$Q = z_0 - \sum (z_j - c_j)x_j$$

where $z_j = c' A_r'^{-1} a_j$ and a_j is the jth column of A_r' matrix. The basis changes can be tested by checking the values of $z_j - c_j$'s. For a maximization problem, if all $z_j - c_j$'s are greater than zero, then the optimum is reached. The test is basically the interval version of the $z_j - c_j$ test.

It is possible to find an interval vector V containing the set of solutions of $(A_r')^T v = p'$ for all $A_r' \in A'$ and $p' \in P'$. The method of Moore [31] for solution of linear systems is used to solve $(A_r')^T v = p'$. The interval solution vector V can be obtained from the following formula.

$$V^{(k+1)} = \{ Y P' + E V^{(k)} \} \cap V^{(k)} \qquad k=0,1,2,..$$

$$V_i^{(0)} = [-1, 1] \, \|Y P'\| / (1 - \|E\|) \quad i=1,2,..,n$$

where $Y = (m (A'))^{-1}$, $E = I - Y A'$.

It has been explained and proved that if $\|I - Y A'\| < 1$, the system has unique solution x for A and p , and the sequence V converges after a finite number of steps. For a given system , $z_j - c_j$ for nonbasic variables can be written as

$$z_j - c_j = p' (A_r')^{-1} a_j - P''$$

where a_j is the jth column of $(A'')^T$ and $p'(A_r')^{-1} = V$.

If $(A'')^T V - P'' > 0$, the optimum is reached and the set of solutions has the same basis as z. A problem arises when they do not have the same basis. In that case, the method terminates.

If at least one of the $z_J - c_J < 0$, there is a possibility of improvement in the objective function. By choosing the most negative $z_J - c_J$ value, the nonbasic variable (x_k) which will enter the basis can be determined. Now, the only problem left is to choose the leaving variable which will become nonbasic. The method suggested here to select the leaving variable is to find all of the adjacent extreme points, and eliminate the extreme points not containing x_k. If there is more than one left, the one resulting in the best objective value is chosen. Figure 8 shows the adjacent points and better objective value.

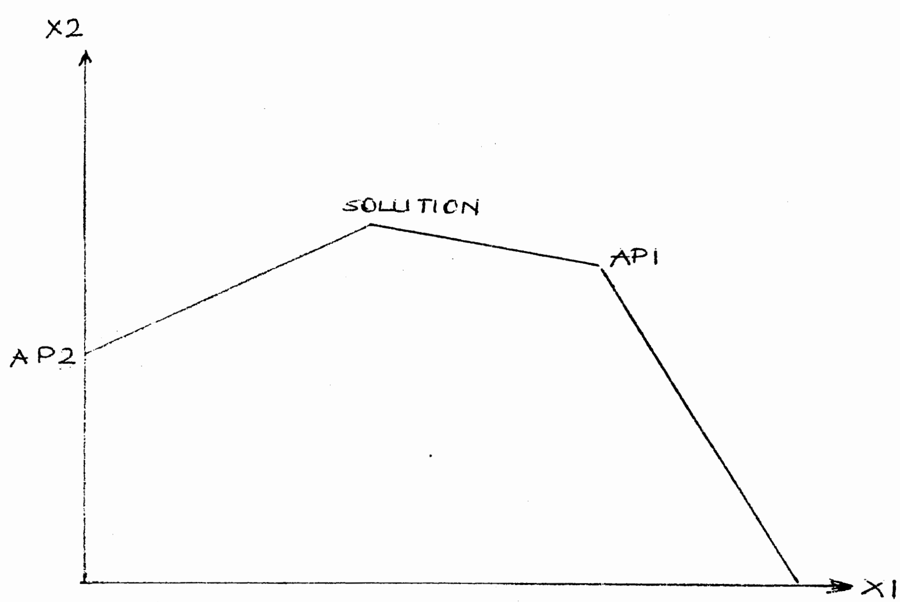

Figure 8. Adjacent Extreme Poinnts

AP1 and AP2 are the vertices adjacent to the solution, and AP1 has a better objective value.

After determining the basis change, the A_r' matrix can be rearranged to contain the new basis and used in the rest of the algorithm.

(iii) ALGORITHM FOR SOLUTION

Suppose that z' is the solution of $A_r' z' = b'$ for $A_r' \in A'$ and $b' \in B'$. Solve this specific system of $A_r' z' = b'$ and let $\bar{z}'$ be an approximate solution of $A_r' z' = b'$. The approximate inverse of the matrix A_r' which was used in the initial solution is the matrix Y.

The set of solutions to the interval linear programming problem is contained in the interval vector Z, computed as follows.

$$Z_i = \bar{z}_i + q\,[-1,\ 1] \qquad \text{(basis components of z)}$$

$$Z'' = 0 \qquad \text{(nonbasis components of z)}$$

where, $q = (||Y||\ ||A' \bar{z}' - B||) \,/\, (1 - R)$

$$R = ||I - YA'|| < 1$$

The formula for the computation of Z was developed and proved by Krawczyk [26]. The derivation of the formulas is given below.

The given system

$$A\ z = B$$

has an exact solution when

$$A_I\, z_I = b$$

or $\qquad A_I\ z_I - b = 0.$

It has an approximate solution when

$$A_1 \tilde{z}_1 - b = d \qquad (1)$$

where, $A_1 \in A$ and $b \in B$

$$A_1 \tilde{z}_1 - b - A_1 z_1 + b = d$$

$$A_1 (\tilde{z}_1 - z_1) = d$$

$$\tilde{z}_1 - z_1 = A_1^{-1} d \qquad (2)$$

By using the properties of interval, absolute value and matrix norm, it is possible to show that

$$| x | < || x || , \quad || x || < || X ||$$

and

$$|| Ax || < || A || \; || x ||$$

where x is a real vector and X is an interval vector. Therefore,

$$| \tilde{z}_1 - z_1 | < || A^{-1} || \; || d ||$$

since $R = I - Y A$. Then, $A^{-1} = (I - R)^{-1} Y$ and

$$|| A || < || Y || / (1 - || R ||) \qquad (3)$$

substituting equations (1) and (3) into equation (2)

$$\tilde{z}_1 - z_1 < \frac{|| Y || \; || A z_1 - B ||}{(1 - || R ||)} = q$$

$$\tilde{z}_1 - q < z_1 < \tilde{z}_1 + q$$

Therefore, initial solution $z^{(0)}$ can be written as

$$z^{(0)} = \tilde{z}_1 + q [-1, 1]$$

for basis components. It is possible to find a narrower interval vector containing the set of solutions with the following formula. Set $z^{(0)} = z'$,

$$z^{(k+1)} = z^{(k)} \cap \{ YB + (I - YA') z^{k} \}$$

The iterations will yield a nested sequence of interval vectors and converge in a finite number of steps when they are performed with limited precision on a computer. After the kth step the final solution will be

$$z = z^{(k)}$$

(iv) TEST FOR NONNEGATIVITY

For most practical problems, the variables represent physical quantities. Therefore, they must be nonnegative. Also, the simplex method is designed to solve linear programming where the variables are nonnegative. The nonnegativity of the solution vector Z must be checked.

Termination Criterion

For any iterative interval method which produces a nested sequence of intervals whose end points are represented by finite precision numbers on computer, a natural stopping criterion exists. Since the sequence will converge in a finite number of steps, the elements z^k of the sequence $\{ z^k \}$ can be computed until the condition $z^{k+1} = z^k$ is reached. The flowchart given in Figure 9 is the summary of the steps of the algorithm.

FLOWCHART OF THE ALGORITHM

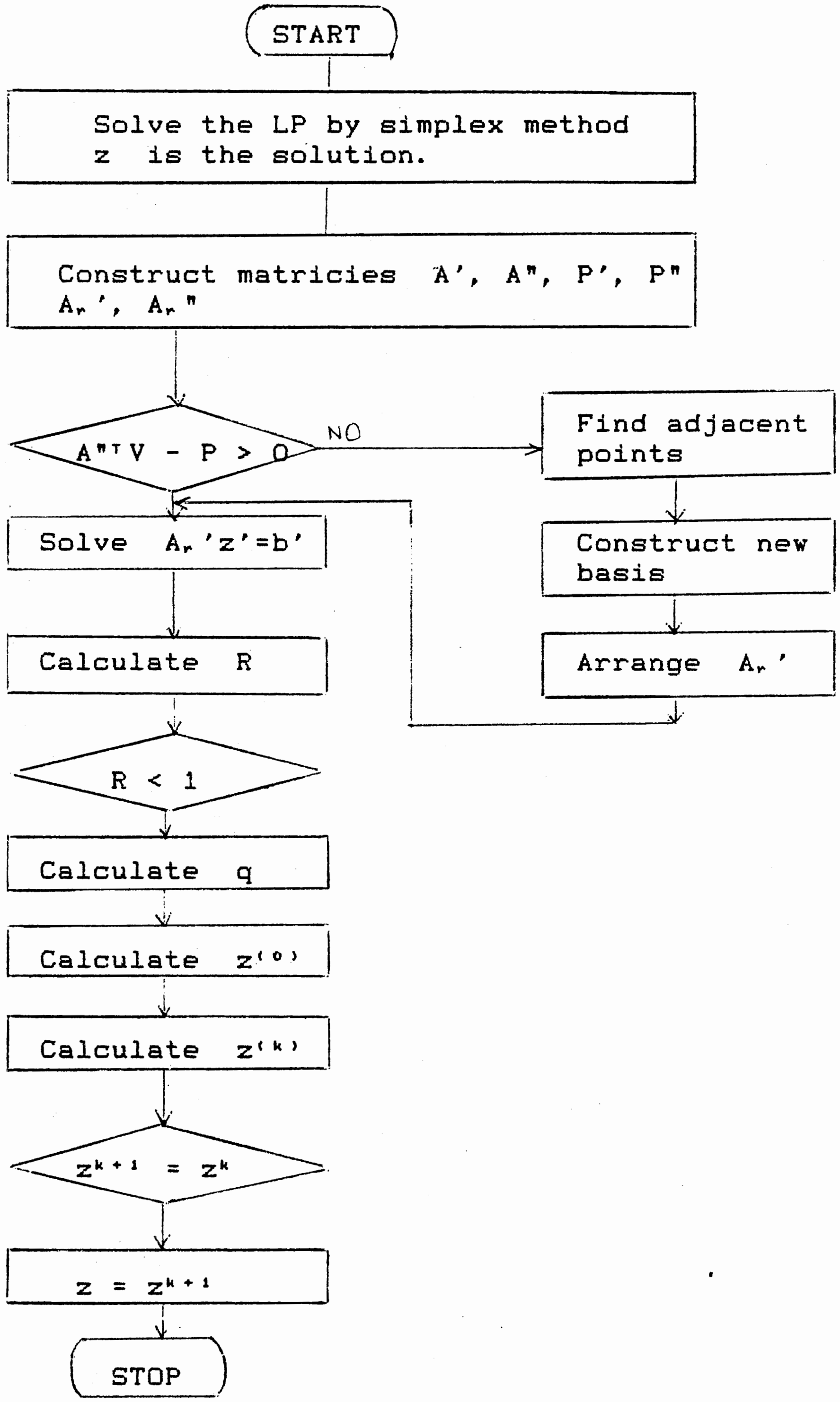

Figure 9. Flow Chart Of The Algorithm

CHAPTER V

TESTING

Test Problem 1.

Maximize [0.95, 1.05]*X1 + [2.85, 3.15]*X2

Subject to

[0.95, 1.05]*X1 + [0.95, 1.05]*X2 < [5.7, 6.3]

[-1.05,-0.95]*X1 + [1.9, 2.1]*X2 < [7.6, 8.4]

The LP problem solved by simplex method is

Maximize X1 + 3*X2

Subject to

$$X1 + X2 \leq 6$$

$$-X1 + 2*X2 \leq 8 , \quad \text{all} \quad Xi \geq 0$$

The solution space is shown in figure 10.

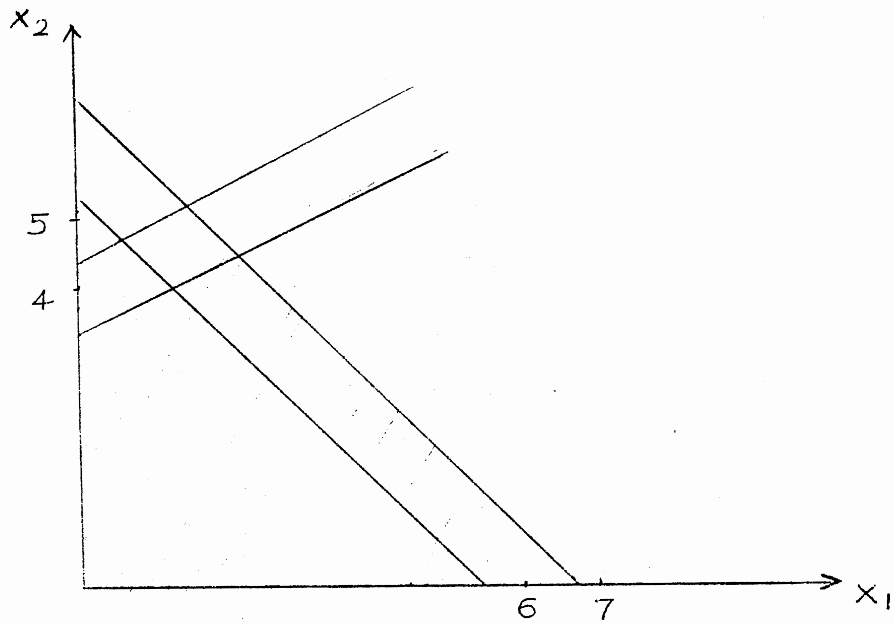

Figure 10 - Solution Space of Test Problem 1

The initial solution from simplex method is

$X1 = 4/3$ and $X2 = 14/3$

$R = \| I - Y A \| = 0.11667032$

$q = 1.15660477$

The initial interval vector is

$$z^{(C)} = \begin{pmatrix} [0.173395157, & 2.48660469] \\ [3.50339508\ , & 5.81660461] \end{pmatrix}$$

After six iterations, the sequence converges. The optimum solution is

$$z = \begin{pmatrix} [0.545046389\ , & 2.12162113] \\ [4.10096264\ , & 5.23237324] \end{pmatrix}$$

The boundaries of optimum solution is shown in figure 11.

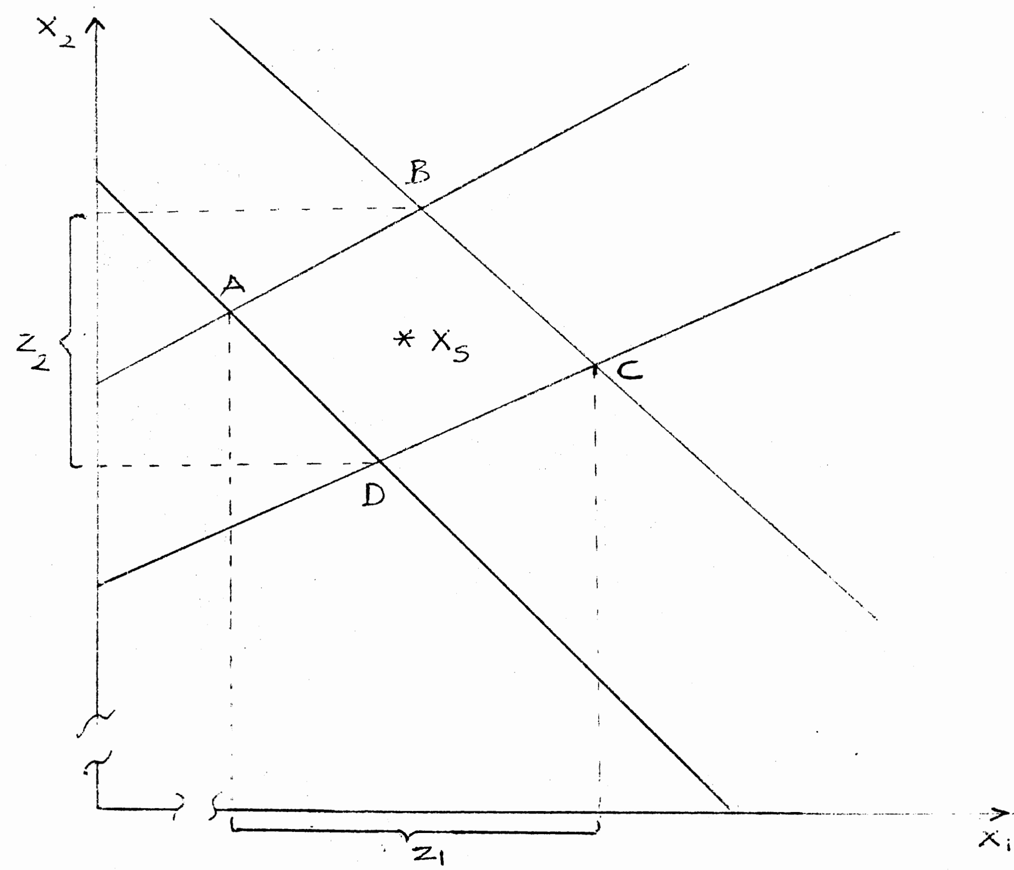

Figure 11 - Optimum Solution of Test Problem 1

If points A, B, C, D are computed graphically, their values are

A = (0.648910411, 4.779661017)

B = (1.473684211, 5.157894737)

C = (2.008354219, 4.623224728)

D = (1,206349206, 4.222222222)

When the two solutions are compared, it can be observed that the computer solution is very close to the graphical solution. The difference between the results is due to roundoff error.

Test Problem 2.

$$\text{Maximize} \quad [2.85, 3.15]X1 + [0.95, 1.05]X2 + [2.85, 3.15]X3$$

Subject to

$$[1.9, 2.1]*X1 + [0.95, 1.02]*X2 + [0.95, 1.05]*X3 \leq [1.9, 2.1]$$

$$[0.95, 1.05]*X1 + [1.9, 2.1]*X2 + [2.85, 3.15]*X3 \leq [4.75, 5.25]$$

$$[1.9, 2.1]*X1 + [1.9, 2.1]*X2 + [0.95, 1.05]*X3 \leq [5.7, 6.3]$$

$$\text{all} \quad Xi \geq 0$$

The LP problem used to obtain an initial solution is

$$\text{Maximize} \quad 3X1 + X2 + 3X3$$

Subject to

$$2*X1 + X2 + X3 \leq 2$$

$$X1 + 2*X2 + 3*X3 \leq 5$$

$$2*X1 + 2*X2 + X3 \leq 6$$

$$Xi \geq 0 \quad \text{for } i=1,2,3$$

The solution vector is

$$Xs = \begin{pmatrix} 0.2 \\ 0 \\ 1.6 \\ 0 \\ 0 \\ 4.0 \end{pmatrix}$$

$R = 0.17$ and

$q = 1.28675270$

The initial interval solution is

$$Z = \begin{pmatrix} [-1.08675194, 1.48675251] \\ 0 \\ [0.313245000, 2.88675000] \\ 0 \\ 0 \\ [2.713247000, 5.28675000] \end{pmatrix}$$

After five iteration, the optimum interval is reached as

$$Z = \begin{pmatrix} [0,\ 0.440287650] \\ 0 \\ [1.3154,\ 1.8446] \\ 0 \\ 0 \\ [3.6854,\ 4.3735] \end{pmatrix}$$

CHAPTER VI

SUMMARY AND CONCLUSIONS

In the previous chapters and in the program , the residual matrix R=(I-YA) and norm of R ($\|R\|$) are computed before we really start the algorithm. The necessary condition was $\|R\| < 1$. The width of the intervals affects the R . It was observed that whenever elements of A matrix has large intervals , it is quite possible to get R > 1 . It is true that even if one interval in the matrix has width> 1, the norm of R will be greater than one. But if $\|R\|$ < 1 , then for any real matrix $R_R \subset E$ it is possible to use power series representation

$$(I - R_R)^{-1} = I + R_R + R_R^2 + \ldots.$$

and

$$(I-R_R) < 1 + \|R\|^1 + \|R\|^2 + \ldots.. \leq 1/(1-\|R\|).$$

For n dimensional problems it is possible to get an $\|R\|$ value greater than one , while the simplex problem has an optimum solution. Therefore it may not be possible to find an interval solution for the given problem.

The second point is the test of the basis. In the original algorithm, whenever the test for basis check fails, the algorithm terminates without trying to find an alternative solution. This weakness is corrected by moving the basis to an adjacent point having better objective value. Therefore the modified method has high possibility of leading to the final solution.

The interval linear programming problem can be applied all linear programming problems, especially if their data are not certain. It is always possible to find physical models that may require the application of interval linear programming. In real life , most of the models developed for real physical system use approximations or estimations. Therefore their parameters and data can not be represented precisely. Interval linear programming is a good way to observe the system behaviour within given limits. The computer program written for the solution of interval linear programming problems can be used for n-dimensional cases with slight modifications. The small interval arithmetic package is very useful and portable eventhough it contains only basic operations. It can be used in any program whenevr an interval arithmetic operation becomes necessary.

Basic applications of interval arithmetic and developed methods are discussed in Chapter II. Most of the areas are open to discussion. Most of the studies done so far have concentrated on achieving the most efficient ways to find sharp upper and lower bounds on the solutions.

It may sound like that all computations should be carried out using interval techniques. In fact only interval methods really provide the tools which are helpful in analyzing computational errors, finding upper and lower bounds on set of solutions and providing termination criteria for iterative methods.

CHAPTER VII

SUGGESTIONS FOR FURTHER STUDY

Since interval mathematics is a new and growing area of applied mathematics, there have been many studies done in different areas of interval mathematics. But regardless of the class of application and study , the common point is the search for efficient ways of computation of intervals by machines. Studies in computer hardware and software can be done to find efficient representations of intervals. Also it has been observed that previously developed interval packages are quite slow. When the precision is vital , it is necessary and important to use intervals to eliminate roundoff errors. The interval arithmetic package can be modified to fit the available compiler or operations can be performad by using assembler language.

Also, the effects of changes of the model parameters can be studied. Particularly changing the endpoints of intervals , a kind of sensitivity analysis can be applied to study the behaviour of the model under different conditions.

BIBLIOGRAPGHY

(1) Alafeld, Gotz and Jurgen Herzberger. Introduction to Interval Computations. New York: Academic Press Inc., 1983.

(2) Bazaraa, Mokhtar S. and John J. Jarvis. Linear Programming and Network Flows. New York: John Willey and Sons, 1977.

(3) Charnes, A. , D. Granot, F. Granot."A Primal Algorithm for Interval Linear-Programming Problems." Linear Algebra and Appl. 17(1977), pp. 65-78

(4) Conte, S.D and Carl de Boor. Elementary Numerical Analysis. 3rd Ed. New York: McGraw-Hill Book Co., Inc., 1980.

(5) Dewer, J.K.S. "Procedure for Interval Arithmetic." Computer Journal, 14(1971), pp. 447-450

(6) Dwyer, Paul S. Linear Computations. New York: John Wiley and Sons Inc., 1951.

(7) Fishburn, Peter C. Interval Orders and Interval Graphs. New York: John Wiley and Sons Inc., 1985.

(8) Forsythe, George and C. B. Moler. Computer Solution of Linear Algebraic Systems. Englewood Cliffs, N.J: Prentice Hall Inc., 1967.

(9) Gargantini, I. "Further Applications of Circular Arithmetic: Schroder-like Algorithms with Error Bounds for Finding Zeros of Polynomials."SIAM J. Numer. Anal. 15(1978), pp. 497-510.

(10) Gass, Saul I. Linear Programming Methods and Applications. New York: McGraw-Hill Book Co., Inc., 1969.

(11) Gastinel, Noel. Linear Numerical Analysis. Paris: Herman , 1970.

(12) Gay, David M."Perturbation Bounds for Nonlinear Equations." SIAM J. Numer. Anal. 18(1981), pp. 654-663.

(13) Gay, David M. "Solving Linear Interval Equations." SIAM J. Numer. Anal. 19(1982), pp.858-870.

(14) Gibb, Allan. "Procedures for Range Arithmetic." Comm. of ACM. 4(1961), pp.319-320.

(15) Good, Donald I. and R. L. London. "Computer Interval Arithmetic: Definition and Proof of Correct Implementation." J. of ACM. 17(1970, pp.603-612.

(16) Gregory, Robert Todd. Error-Free Computation: Why It Is Needed and Methods for Doing It. Huntington, New York: Robert E. Krieger Publishing Co., Inc., 1980.

(17) Gunn, E. A. and G. J. Anders."A Comparison of Interval Linear Programming with the Simplex Method." Linear Algebra and Appl. 38(1981), pp. 149-159.

(18) Hansen, Eldon. "Interval Mathematic in Matrix Computations, Part I."SIAM J. Numer. Anal. 2(1965), pp. 308-320.

(19) Hansen, Eldon and R. Smith. "Interval Arithmetic in Matrix Computations, Part II." SIAM J. Numer. Anal. 4(1967), pp. 1-9.

(20) Hansen, Eldon. "A Generalized Interval Arithmetic." Interval Mathematics. Ed. Karl Nickel. Heidelberg: Springer-Verlag , 1975, pp. 7-18.

(21) Hansen, Eldon. "interval Forms of Newton's Method." Computing. 20(1978), pp. 153-163.

(22) Jansen, P. and P. Weidner. "High Accuracy Arithmetic Software- Some Tests of the ACRITH Problem-Solving Routines." ACM Trans. Math. Software. 12(1986), pp. 62-70.

(23) Jones, S. T. "Locating Safe Starting Regions for Iterative Method: A Heuristic Algorithm." Interval Mathematics 1980. Ed. Karl L.E. Nickel. New York: Academic Press Inc., 1980, pp. 377-386.

(24) King, J. T. Introductions to Numerical Computation. New York: McGraw-Hill Book Co., Inc., 1984.

(25) Kioutelidis, J.B. "Algorithmic Error Estimation for Approximate Solutions of Nonlinear Systems of Equations." Computing. 19(1978), pp. 313-320.

(26) Krawczyk, R. "Fehlerabschatzung bei Linearer Optimierung." Interval Mathematics. Ed. Karl Nickel. Heidelberg: Springer-Verlag, 1975.

(27) Kulish, U. "An Axiomatic Approach to Rounded Computations." Numer. Math. 18(1971), pp. 1-17.

(28) Kulish, U. and W. Miranker. Computer Arithmetic in Theory and Practise. New York: Academic Press, Inc., 1981.

(29) Luenberger, David G. Linear and Nonlinear Programming. Massachussets: Addison-Wesley Publishing Co., 1984.

(30) Moore, Ramon E. Interval Analysis. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1966.

(31) Moore, Ramon E. Methods and Application of Interval Analysis. Philadelphia: Society for Industrial and Applied Mathematics, 1979.

(32) Moore, Ramon E. and J.B. Kiostelidis. "A Simple Test for Accuracy of Approximate Solutions to Nonlinear (or Linear) Systems." SIAM J. Numer. Anal. 17(1980), pp. 521-529.

(33) Moore, Ramon E. "New Results on Nonlinear Systems." Interval Mathematics 1980. Ed. Karl L.E. Nickel. New York: Academic Press Inc., 1980.

(34) Oettli, W. "On the Solution Set of a Linear System with Inaccurate Coefficients." SIAM J. Numer. Anal 2(1965), pp. 115-118.

(35) Ortega, James M. Numerical Analysis- A Second Course. New York: Academic Press, Inc., 1972.

(36) Pachner, Jaroslav. Handbook of Numerical Analysis Applications. New York: McGraw-Hill Book Co., Inc., 1984.

(37) Rall, L.B. "Interval Methods for Fixed-Point Problems." Numer. Funct. Anal. and Optimiz. 9(1987), pp.35-59.

(38) Rice, John R. Numerical Methods, Software, and Analysis. New York: McGraw-Hill Book Co., Inc., 1983.

(39) Ris, F.N. "Tools for the Analysis of Interval Arithmetic." Interval Arithmetic. Ed. Karl Nickel. Heidelberg: Springer-Verlag, 1975, pp. 75-98.

(40) Robers, Philip d. and Adi Ben-Israel. "A Suboptimization Method for Interval Linear Programming: A New Method for Linear Programming." Linear Algebra and Appl. 3(1970), pp. 383-405.

(41) Rohn, J. "Duality in Interval Linear Programming." Interval Mathematics 1980. Ed. Karl Nickel. New York: Academic Press Inc., 1980.

(42) Rohn, J. "Strong Solvability of Interval Linear Programming Problems." Computing. 26(1981), pp. 79-82.

(43) Rokne, J. and P. Lancaster, "Complex Interval Arithmetic." Comm. ACM. 14(1971), pp. 111-112.

(44) Shampine, Lawrence F. and Richard C. Allen, Jr. Numerical Computing. Philadelphia: W.B. Saunders Co., Inc., 1973.

(45) Stewart, N.F "Interval Arithmetic for Guaranteed Bounds in Linear Programming." JOTA. 12(1973), pp. 1-5.

(46) Taha, Hamdy. Operations Research an Introduction. 2nd ed. New York: MacMillan Publishing Co., Inc., 1976.

(47) Yohe, J.M. "A Reasonably Portable Package." ACM Trans. Math. Software. 5(1979), pp. 50-63.

APPENDIX A

DEFINITIONS

Definition 1

Polyhedral Set

A nonempty set S in E_n is called a polyhedral set if it is the inter section of a finite number of closed half spaces, that is, $S = \{ x : p_i^t \; x < \alpha_i \;\; \text{for} \;\; i=1,2,\ldots,m \}$, where p_i is a nonzero vector and α_i is a scalar for $i=1,2,\ldots,m$.

Definition 2

Extremal Subsystem

For any $A_r \in A$ and $b \in B$, a system

$$A_r x = b$$

is called a subsystem of an interval linear system $A x = B$. A subsystem is called an external subsystem of $A x = B$ if for each $i=1,\ldots,m$, its ith equation has either the form $(\underline{A} \; x)_i = \overline{B}$ or the form $(\overline{A} \; x)_i = \underline{B}$.

Definition 3

Standart Form

All constraints are equations except for the nonnegativity constraints which remain inequalities ($\geqslant 0$). The right-hand side of each element is nonnegative. All variables are nonnegative . Objective function is of the maximization or the minimization type.

APPENDIX B

PROGRAM LISTING

```
C
C
C***********************************************************************
C
C      PREPARED BY:
C      ZEYNEP AYSEGUL KARACAL                         DECEMBER,1987
C
C      REFERENCES: RAMON E. MOORE , R. KRAWCZYK
C
C***********************************************************************
C
C      THIS PROGRAM IS FOR THE SOLUTION OF INTERVAL LINEAR PROGRAMMNIG
C      PROBLEMS. IT USES INTERVAL ARITHMETIC OPERATIONS(ADDITION,
C      SUBTRACTION,MULTIPLICATION AND DIVISION).
C
C***********************************************************************
C
C     A(*,*,2)        INTERVAL MATRIX CONSISTING BASIS COLUMNS
C     AI(*,*,2)       INVERSE OF A
C     IM(*,*,2)       IDENTITY MATRIX
C     ARP(*,*)        REAL MATRIX OF BASIS VARIABLES
C     BP(*)           RIGHT HAND SIDE USED IN SIMPLEX
C     B(*,1,2)        INTERVAL RIGHT HAND SIDE
C     Z(*,*,1,2)      SOLUTION VECTOR
C     PP(*,1,2)       INTERVAL VECTOR CONTAINING ONLY OBJECTIVE
C                     COEFFICENTS OF BASIS VARIABLES
C     PDP(*,1,2)      INTERVAL VECTOR CONTAINING OBJECTIVE COEFFICENTS OF
C                     NONBASIS VARIABLES
C     ADP(*,*,2)      INTERVAL MATRIX OF NONBASIS VARIABLES
C     ZS(*)           SOLUTION OF SIMPLEX METHOD
C     AA(*,*,2)       INTERVAL CONSTRAINT MATRIX
C     MAT(*,*)        REAL CONSTRAINT MATRIX USED IN SIMPLEX
C     IBASIS(*)       SET OF INDICES OF BASIS VARIABLES
C     P(*,1,2)        INTERVAL VECTOR OF OBJECTIVE FUNCTION
C     MD              NUMBER OF ROWS IN A
C     ND              NUMBER OF VARIABLES INCLUDING SLACKS
C     PS(*)           OBJECTIVE FUNCTION OF SIMPLEX PROBLEM
C
C***********************************************************************
C
C
       LOGICAL CHECK,GOON
       REAL A(2,2,2),AI(2,2,2),IM(2,2,2),YB(2,1,2),ZM(2,1,2),AZ(2,1,2),
      *ARP(2,2),BP(2),ZP(2),Y(2,2),B(2,1,2),Z(O:2O,2,1,2),BB(2),
      *WKAREA(18),NORMY,PP(2,1,2),PDP(2,1,2),ADP(2,2,2),ZS(4)
      *,TEMP(2,2),TI(2,2),ADPT(2,2,2),AA(2,4,2),P(4,1,2),ARPT(2,2)
      *,PS(2,1),AT(2,2,2),AV(2,1,2),MAT(2,4),TB(2),TBAS(2,2),
      *BS(2),EP(2),EXTP(6,2),ANB(2,2)
       INTEGER IBASIS(2),NB(2),M,N,IDGT
       COMMON EPS
```

```
      DATA (BP(I),I=1,2)/5.8,8.1/
      DATA (IBASIS(I),I=1,2)/1,2/
      DATA (ZS(I),I=1,4)/1.33,4.66,2*0.0/
      DATA ((B(I,1,K),K=1,2),I=1,2)/5.7,6.3,7.6,8.4/
      DATA ((P(I,1,K),K=1,2),I=1,4)/0.95,1.05,2.85,3.15,4*0.0/
      DATA (NB(I),I=1,2)/3,4/
      DATA (PS(I,1),I=1,2)/1.0,3.0/
      DATA (BS(I),I=1,2)/6.0,8.0/

      MD=2
      ND=4
C
C  READ INITIAL MATRIX AA
C
      DO 79 I=1,MD
         DO 79 J=1,ND
            DO 79 K=1,2
               READ(5,3)AA(I,J,K)
 3             FORMAT(F12.6)
 79   CONTINUE
C
C     READ THE MATRIX USED IN SIMPLEX FOR INITIAL SOLUTION
C
      DO 300 I=1,MD
         DO 300 J=1,ND
            READ(5,3)MAT(I,J)
 300  CONTINUE
      L=0
      K=0
      DO 310 J=1,ND
         DO 311 N=1,MD
            IF(J.EQ.IBASIS(N)) THEN
              L=L+1
              DO 312 I=1,MD
                 ARP(I,L)=MAT(I,J)
 312          CONTINUE
              GO TO 310
            ENDIF
 311     CONTINUE
         K=K+1
         DO 313 I=1,MD
            ANB(I,K)=MAT(I,J)
 313     CONTINUE
 310  CONTINUE
C
C  READ INTERVAL MATRIX A
C
      DO 70 I=1,MD
         DO 70 J=1,MD
            DO 70 K=1,2
               A(I,J,K)=AA(I,IBASIS(J),K)
               IM(I,J,K)=0.0
 70   CONTINUE
      DO 78 I=1,MD
         DO 78 J=1,(ND-MD)
            DO 78 K=1,2
```

```
                ADP(I,J,K)=AA(I,NB(J),K)
 78   CONTINUE
      DO 71 I=1,MD
         DO 71 K=1,2
            IM(I,I,K)=1.0
 71   CONTINUE
      DO 575 I=1,MD
         DO 575 K=1,2
            PP(I,1,K)=P(IBASIS(I),1,K)
 575  CONTINUE
      DO 576 I=1,(ND-MD)
         DO 576 K=1,2
            PDP(I,1,K)=P(NB(I),1,K)
 576  CONTINUE
C
C     CALL FINEPS TO FIND EPS
C
      CALL FINEPS(EPS)
C
      DO 400 I=1,MD
         DO 400 J=1,MD
          ARPT(J,I)=ARP(I,J)
          AT(J,I,1)=A(I,J,1)
          AT(J,I,2)=A(I,J,2)
 400  CONTINUE
      DO 401 I=1,MD
         DO 401 J=1,(ND-MD)
            ADPT(J,I,1)=ADP(I,J,1)
            ADPT(J,I,2)=ADP(I,J,2)
 401  CONTINUE
      DO 402 I=1,MD
         DO 402 J=1,MD
            TEMP(I,J)=(AT(I,J,1)+AT(I,J,2))/2.
 402  CONTINUE
C
C  CALL IMSL ROUTINE LINV2F TO GET INVERSE OF TEMP
C
      IA=2
      N=2
      IDGT=3
      CALL LINV2F(TEMP,N,IA,TI,IDGT,WKAREA,IER)
C
C  CALL TEST ROUTINE TO TEST THE APPLICABILITY OF ALGORITHM
C  TO GIVEN SPECIFIC SYSTEM
C
      CALL TEST(ARPT,ADPT,PS,PDP,TI,GOON,MD,ND,IND)
C
      IF(GOON.AND..TRUE.) GO TO 900
      PRINT *,'*** TEST FAILED ***'
      IF(IND.LT.0) GO TO 999
      PRINT *,' SET OF SOLUTIONS DO NOT HAVE THE SAME BASIS'
      PRINT *,'*** ALGORITHM CAN NOT BE APPLIED ***'
C
C FIND ADJACENT EXTREME POINTS
C
      DO 315 I=1,MD
```

```
 315        TB(I)=IBASIS(I)
      K=0
      M=1
      N=2
      IA=2
      IDGT=2
      DO 320 I=1,MD
         DO 321 J=1,(ND-MD)
            TB(I)=NB(J)
C
C  BUILD BASIS MATRIX
C
            DO 322 I1=1,MD
               DO 322 I2=1,MD
                  TBAS(I2,I1)=MAT(I2,TB(I1))
 322        CONTINUE
            DO 323 I1=1,MD
 323           EP(I1)=BS(I1)
C
C  CALL IMSL ROUTINE TO FIND THE BASIS VARIABLES
C
      CALL LEQT2F(TBAS,M,N,IA,EP,IDGT,WKAREA,IER)
            IFL=1
            DO 325 I1=1,MD
               IF(EP(I1).LT.0.0) THEN
                 IFL=0
                 GO TO 321
               ENDIF
 325        CONTINUE
            IF(IFL.GT.0)THEN
               K=K+1
               FIND=0
               DO 326 I1=1,MD
                  IF(TB(I1).EQ.IND) FIND=1
                  EXTP(K,I1)=TB(I1)
 326           CONTINUE
            ENDIF
            IF(FIND.GT.0)THEN
               OBJ=0.0
               DO 350 I1=1,MD
                  OBJ=PS(TB(I1),1)*EP(I1)+OBJ
 350           CONTINUE
               IF(OBJ.GT.MAX)THEN
                  MAX=OBJ
                  LEAVE=K
                  DO 355 I2=1,MD
 355                 ZS(I2)=EP(I2)
               ENDIF
               ENDIF
 321      CONTINUE
            TB(I)=IBASIS(I)
 320  CONTINUE
      DO 390 II=1,K
         PRINT *,(EXTP(II,JJ),JJ=1,MD)
 390  CONTINUE
      IF(FIND.GT.0) THEN
```

```
C
C  CHANGE THE BASIS
C
      K=LEAVE
      DO 360 I=1,MD
 360     IBASIS(I)=EXTP(K,I)
      DO 361 I=1,MD
         DO 361 J=1,MD
            ARP(J,I)=MAT(J,IBASIS(I))
 361  CONTINUE
      GO TO 370
      ELSE
          PRINT *,'LEAVING VARIABLE IS NOT ADJACENT TO THE SOLUTION'
          GO TO 999
      ENDIF
 900  PRINT *,'*** TEST SUCEEDED ***'
      PRINT *,'*** ALGORITHM CAN BE APPLIED ***'
C
C  SOLVE AR'Z=B'
C
C  COPY BP INTO ZP
C
 370  DO 72 I=1,MD
 72      ZP(I)=BP(I)
      M=1
      N=2
      IA=2
      IDGT=2
C
C  CALL IMSL ROUTINE LEQT2F TO SOLVE AR'Z=B'
C
      CALL LEQT2F(ARP,M,N,IA,ZP,IDGT,WKAREA,IER)
      PRINT *,'IER MAIN=',IER,'IDGT=',IDGT
      PRINT *,'  THE SOLUTION OF AR.Z=B '
       DO 111 I=1,MD
           WRITE(6,122) ZP(I)
 122       FORMAT(10X,F12.6)
 111   CONTINUE
C
C  CALL IMSL ROUTINE LINV2F TO GET INVERSE OF A'
C
      IA=2
      N=2
      IDGT=3
      CALL LINV2F(ARP,N,IA,Y,IDGT,WKAREA,IER)
      PRINT *,'    THE INVERSE OF A  '
      DO 133 I=1,MD
            PRINT *,(Y(I,J),J=1,MD)
 133  CONTINUE
C
C  CALCULATE YA'
C
      CALL RIMUL(Y,MD,MD,A,AI)
      PRINT *,'     MULTIPLICATION OF Y.A    '
      DO 144 I=1,MD
         DO 145 J=1,MD
```

```
          PRINT *,AI(I,J,1),'   ',AI(I,J,2)
 145     CONTINUE
         PRINT *,' '
 144  CONTINUE
      DO 73 I=1,MD
         DO 73 J=1,MD
            CALL SUBI(IM(I,J,1),IM(I,J,2),AI(I,J,1),AI(I,J,2),TEMP1,
     *TEMP2)
            AI(I,J,1)=TEMP1
            AI(I,J,2)=TEMP2
 73   CONTINUE
      PRINT *, '  AI=I-YA    '
      DO 166 I=1,MD
           PRINT *,((AI(I,J,K),K=1,2),J=1,MD)
 166  CONTINUE
C
C  AI=I-YA' , R=||I-YA'||
C
      CALL IMNORM(AI,MD,MD,R)
      PRINT *,'R=',R
      IF(R.GT.1.0) THEN
        PRINT *,'***ERROR***'
        GO TO 999
        ENDIF
C
C CALCULATE ||Y||
C
      CALL RMNORM(Y,MD,MD,NORMY)
      PRINT *,' THE NORM OF Y =',NORMY
C
C  CALCULATE A'Z'
C
       CALL IRMUL(ZP,MD,1,A,AZ)
C
C   CALCULATE (AZ'-B)
C
      DO 74 I=1,MD
         AZ(I,1,1)=AZ(I,1,1)-B(I,1,2)
         AZ(I,1,2)=AZ(I,1,2)-B(I,1,1)
 74   CONTINUE
C
C  CALCULATE ||AZ'-B||
C
      CALL IMNORM(AZ,MD,1,T)
      PRINT *,'T =',T
C
C  CALCULATE Q
C
      Q=(NORMY*T)/(1-R)
      PRINT *,'  Q = ',Q
C
C  CALCULATE Z
C
      DO 77 I=1,MD
         Z(0,I,1,1)=ZS(IBASIS(I))-Q
         Z(0,I,1,2)=ZS(IBASIS(I))+Q
```

```
         PRINT *,Z(O,I,1,1),Z(O,I,1,2)
 77   CONTINUE
C
C  CALCULATE Z(K+1)
C  CALCULATE YB
C
      CALL RIMUL(Y,MD,1,B,YB)
      DO 90 I=1,20
         CALL MULIM(AI,MD,I-1,MD,1,Z,ZM)
C
C  CALCULATE  YB+((I-YA')Z'=ZM)
C
         DO 91 J=1,MD
            DO 91 K=1,2
               ZM(J,1,K)=ZM(J,1,K)+YB(J,1,K)
 91      CONTINUE
         DO 93 J=1,MD
            CALL INTER(Z(I-1,J,1,1),Z(I-1,J,1,2),ZM(J,1,1),ZM(J,1,2),
     *Z(I,J,1,1),Z(I,J,1,2),IFLAG)
      PRINT *,'IFLAG=',IFLAG
 93      CONTINUE
         CALL CMP(Z,I,CHECK,MD)
         PRINT *,' CHECK =',CHECK
         IF(CHECK.AND..TRUE.) GO TO  100
         K=I
 90   CONTINUE
      PRINT *,'   NO SUCCESS   '
 100  DO 200 I=1,K+1
         DO 201 J=1,MD
            WRITE(6,101) Z(I,J,1,1), Z(I,J,1,2)
 101        FORMAT(10X,F12.6,10X,F12.6)
 201     CONTINUE
         PRINT *,' ------------------------'
 200  CONTINUE
      PRINT *,'************************************'
      PRINT *,'**    THE OPTIMAL SOLUTION        **'
      DO 203 I=1,MD
         PRINT *,(Z(K+1,I,1,J),J=1,2)
 203  CONTINUE
      PRINT *,'************************************'
 999  CONTINUE
      STOP
      END
```

```
C
C*********************************************************************
C*********************************************************************
C                        SUBROUTINES                                **
C*********************************************************************
C*********************************************************************
C
C
C
C*********************************************************************
C
C       THIS ROUTINE CHANGES THE ENDPOINTS OF AN INTERVAL BY GIVEN
C    PERCENTAGE.
C    A : LOWER POINT OF INTERVAL
C    B : HIGHER POINT OF INTERVAL
C    PERC : PERCENTAGE OF CAHNGE
C    CM = - OR +
C    INT(A,B) + CM.PERC(INT(A,B))
C
C********************************************************************
C
C
      SUBROUTINE CHANGE(A,B,CM,PERC)
      INTEGER CM
      WIDTH=B-A
      POFW=(WIDTH*PERC)/2.
      IF(CM.LT.0) THEN
        A=A+POFW
        B=B-POFW
      ELSE
        A=A-POFW
        B=B+POFW
      ENDIF
      RETURN
      END
```

```
C
C
C*********************************************************************
C
C     THIS ROUTINE FINDS THE UNIT ROUNDOFF ERROR,EPS.
C
C*********************************************************************
C
      SUBROUTINE FINEPS
      X=1.0
 11   X=X/2.0
      IF(1.0+X.GT.1.0) GO TO 11
      EPS=2.0*X
      RETURN
      END
```

```
C
C*****************************************************************
C
C  TEST ROUTINE - THIS ROUTINE TESTS THE BASIS CHANGE
C  IND    : INDICE OF THE MOST NEGATIVE ZJ-CJ
C  OK     : FLAG FOR TEST SUCCESS
C  ARPT(*,*) : TRANSPOSE OF ARP
C  ADPT(*,*,2) : TRANSPOSE OF ADP
C  Y(*,*): INVERSE OF ARPT
C  V(*,*,1,2) : SOLUTION OF (ARP.V=PDP)
C
C*****************************************************************
      SUBROUTINE TEST(ARPT,ADPT,PS,PDP,Y,OK,MD,ND,IND)
      REAL E(2,2),PS(MD,1),PDP((ND-MD),1,2),ARPT(MD,MD),
     *YP(2,1),EV(2,1,2),V(0:20,2,1,2),Y(MD,MD),ADPT((ND-MD),MD,2)
     * ,T(2),TEMP(2,1,2)
      LOGICAL OK
C
      PRINT *,'*******   TEST ROUTINE   ********'
      PRINT *,'                        '
      DO 405 I=1,MD
         PRINT *,(Y(I,J),J=1,MD)
 405  CONTINUE
C
C   A=YAT
C
      CALL RMUL(Y,MD,MD,ARPT,E)
      PRINT *,' **    Y*ARPT=E    '
      DO 418 I=1,MD
         PRINT *,(E(I,J),J=1,MD)
 418  CONTINUE
      PRINT *,'    I-YARPT'
      DO 406 I=1,MD
         DO 406 J=1,MD
            IF (I.EQ.J) THEN
                E(I,J)=1.-E(I,J)
            ELSE
                E(I,J)=-E(I,J)
            ENDIF
            PRINT *,E(I,J)
 406  CONTINUE
C
C      E=A
C      R IS THE NORM OF E
C
      CALL RMNORM(E,MD,MD,R)
      PRINT *,'R=',R
      IF (R.GT.1) GO TO 499
C
C      YP=Y*PS
C
      CALL RMUL(Y,MD,1,PS,YP)
```

```
C
C     P IS THE NORM OF YP
C
      CALL RMNORM(YP,MD,1,P)
      TMP=P/(1.-R)
      PRINT *,'TMP=',TMP
C
C     V(O)=(-1,1)P/(1-R)
C
      DO 407 I=1,MD
         V(O,I,1,1)=-TMP
         V(O,I,1,2)=TMP
 407  CONTINUE
C
C    CALCULATE    V(K+1)={Y.PS + E.V(K)}+ V(K)
C
      DO 410 I=1,20
         DO 419 J=1,MD
            EV(J,1,1)=0.0
            EV(J,1,2)=0.0
 419     CONTINUE

         DO 414 J=1,MD
            DO 414 K=1,MD
               DO 413 L=1,2
 413              T(L)=E(J,K)*V(I-1,K,1,L)
               IF(T(1).GT.T(2)) THEN
                  TP=T(1)
                  T(1)=T(2)
                  T(2)=TP
               ENDIF
               EV(J,1,1)=T(1)+EV(J,1,1)
               EV(J,1,2)=T(2)+EV(J,1,2)
 414     CONTINUE
         DO 411 J=1,MD
            DO 411 K=1,2
               EV(J,1,K)=EV(J,1,K)+YP(J,1)
 411     CONTINUE
C
C     TAKE INTERSECTION OF TWO  INTERVALS
C
         DO 412 J=1,MD
            CALL INTER(V(I-1,J,1,1),V(I-1,J,1,2),EV(J,1,1),EV(J,1,2),
     * V(I,J,1,1),V(I,J,1,2),IF)
 412     CONTINUE
C
C     COMPARE TWO INTERVALS
C
         CALL CMP(V,I,OK,MD)
         K=I
C
C     IF V(K)=V(K+1)  THEN TERMINATE.
C
         IF(OK   .AND..TRUE.) GO TO 460
 410  CONTINUE
      PRINT *,'    NO CONVERGENCE FOR   V '
```

```
      GO TO 499
 460  CONTINUE
      DO 470 I=1,K
         PRINT *,'*** ITERATION ',I
         DO 470 J=1,MD
            PRINT *,(V(I,J,1,K),K=1,2)
 470  CONTINUE
      CALL MULIM(ADPT,(ND-MD),K,MD,1,V,TEMP)
      DO 475 I=1,(ND-MD)
         CALL SUBI(TEMP(I,1,1),TEMP(I,1,2),PDP(I,1,1),PDP(I,1,2),
     * TEMP1,TEMP2)
         TEMP(I,1,1)=TEMP1
         TEMP(I,1,2)=TEMP2
 475  CONTINUE
C
C  TEST IF A"V-P" >= O
C
      OK=.TRUE.
      MIN=1.0
      DO 476 I=1,(ND-MD)
         IF(TEMP(I,1,2).LT.0.0) THEN
         IF(TEMP(I,1,2).LT.MIN) THEN
C
C  FIND THE MOST NEGATIVE A"V-P"
C
            MIN=TEMP(I,1,2)
            IND=I
         ENDIF
      OK=.FALSE.
         ENDIF
 476  CONTINUE
      GO TO 477
 499  IND=-1
 477  CONTINUE
      PRINT *,'************** END OF TEST ROUTINE **************'
      RETURN
      END
```

```
C**********************************************************************
C
C     THIS ROUTINE IS FOR THE MULTIPLICATION OF TWO REAL
C  MATRICES
C  A(M,M)*B(M,N)= C(M,N)
C
C**********************************************************************
      SUBROUTINE RMUL(A,M,N,B,C)
      REAL A(M,M),B(M,N),C(M,N)
      DO 699 I=1,M
         DO 699 J=1,N
 699        C(I,J)=0.0
      DO 700 I=1,M
         DO 700 J=1,M
            DO 700 K=1,N
               C(I,K)=C(I,K)+A(I,J)*B(J,K)
 700  CONTINUE
      RETURN
      END
```

```
C
C*************************************************************
C
C     THIS ROUTINE CALCULATES THE DIFFERENCE OF TWO INTERVALS,
C     (A,B)-(C,D) = (E,F)
C
C*************************************************************
      SUBROUTINE SUBI(A,B,C,D,E,F)
      REAL A,B,C,D,E,F
      COMMON EPS
            E=(A-D)
            F=(B-C)
            CALL ENLINT(E,F)
            RETURN
      END
```

```
C
C*********************************************************************
C
C   THIS ROUTINE IS FOR THE ADDITIONOF TWO INTERVALS
C   (A,B)+(C,D) = (E,F)
C
C*********************************************************************
C
      SUBROUTINE ADDI(A,B,C,D,E,F)
      COMMON EPS
         E=A+C
         F=B+D
         CALL ENLINT(E,F)
         RETURN
         END
```

```
C
C*********************************************************************
C
C    THIS ROUTINE PERFORMS COMPARISON OF TWO INTERVAL
C SOLUTION VECTOR
C    A(I,*,*,2) AND A(I-1,*,*,2)
C
C*********************************************************************
      SUBROUTINE CMP(A,II,CHK,MD)
      LOGICAL CHK,FLG(10)
      REAL A(0:20,MD,1,2)
           DEL=0.000001
           I=II
           DO 300 J=1,MD
              DELTA1=ABS(A(I,J,1,1)-A(I-1,J,1,1))
              DELTA2=ABS(A(I,J,1,2)-A(I-1,J,1,2))
              FLG(J)=DELTA1.LT.DEL.AND.DELTA2.LT.DEL
 300  CONTINUE
      CHK=.TRUE.
      DO 301 I=1,MD
         CHK=CHK.AND.FLG(I)
 301  CONTINUE
      RETURN
      END
```

```
C
C*********************************************************************
C
C     MULTIPLICATION OF INTERVAL MATRIX WITH REAL MATRIX
C     AP(*,*,2) * R(*,*) = C(*,*,2)
C
C*********************************************************************
      SUBROUTINE IRMUL(R,M,N,AP,C)
      DIMENSION R(M ),AP(M,M,2),C(M,N,2),TEMP(2)
      COMMON EPS
      DO 82 I=1,M
         DO 82 J=1,N
            DO 82 K=1,2
               C(I,J,K)=0.0
 82   CONTINUE
         DO 80 I=1,M
            DO 80 J=1,M
                 DO 81 L=1,2
 81                 TEMP(L)=AP(I,J,L)*R(J)
C
C     CHECK ENDPOINTS
C      IF LEFT  ENDPOINT > RIGHT ENDPOINT , REVERSE ENDPOINTS
C
                 IF(TEMP(1).GT.TEMP(2))THEN
                    TMP=TEMP(1)
                    TEMP(1)=TEMP(2)
                    TEMP(2)=TMP
                    ENDIF
                 C(I,1,1)=TEMP(1)+C(I,1,1)
                 C(I,1,2)=TEMP(2)+C(I,1,2)
                 CALL ENLINT(C(I,1,1),C(I,1,2))
 80      CONTINUE
         RETURN
         END
```

```
C
C*********************************************************************
C
C     MULTIPLICATION OF REAL MATRIX WITH INTERVAL MATRIX
C     R(*,*) * AP(*,*,2) = C(*,*,2)
C
C*********************************************************************
C
      SUBROUTINE RIMUL(R,M,N,AP,C)
      REAL R(M,M),AP(M,N,2),C(M,N,2),TMP,TEMP(2)
      COMMON EPS
      DO 7 I=1,M
         DO 7 J=1,N
            DO 7 K=1,2
               C(I,J,K)=0.0
 7    CONTINUE
           DO 5 I=1,M
              DO 5 J=1,M
                 DO 5 K=1,N
           DO 6 L=1,2
 6            TEMP(L)=R(I,J)*AP(J,K,L)
C
C     IF LEFT  ENDPOINT > RIGHT ENDPOINT , REVERSE ENDPOINTS
C
           IF(TEMP(1).GT.TEMP(2)) THEN
              TMP=TEMP(1)
              TEMP(1)=TEMP(2)
              TEMP(2)=TMP
              ENDIF
           C(I,K,1)=TEMP(1)+C(I,K,1)
            C(I,K,2)=TEMP(2)+C(I,K,2)
           CALL ENLINT(C(I,K,1),C(I,K,2))
 5     CONTINUE
       RETURN
       END
```

```
C
C***********************************************************************
C
C     IT CALCULATES THE MATRIX NORM OF AN INTERVAL MATRIX
C     A(*,*,2): INTERVAL MATRIX
C     M :ROW DIMENSION
C     N : COLUMN DIMENSION
C.    ROWSUM: SUMMATION OF MAXIMUM ENDPOINTS OF INTERVALS
C     NRM : NORM OF MATRIX
C
C************************************************************************
C
        SUBROUTINE IMNORM(A,M,N,NRM)
        REAL A(M,N,2),NRM,MX,ROWSUM
             NRM=0.0
             DO 10 I=1,M
                ROWSUM=0.0
                DO 11 J=1,N
                   IF(ABS(A(I,J,1)).GT.ABS(A(I,J,2)))THEN
                         MX=ABS(A(I,J,1))
                   ELSE
                         MX=ABS(A(I,J,2))
                   ENDIF
                   ROWSUM=ROWSUM+MX
 11             CONTINUE
C
C     FIND THE MAXIMUM ROWSUM AND ASSIGNED TO NRM
C     NRM IS THE MATRIX NORM
C
                IF (ROWSUM.GT.NRM) NRM=ROWSUM
 10          CONTINUE
          RETURN
          END
```

```
C
C***********************************************************************
C
C    THIS SUBROUTINE CALCULATES THE MATRIX NORM OF A GIVEN REAL MATRIX
C    NRM : MATRIX NORM
C    A(*,*) : REAL MATRIX
C
C***********************************************************************
      SUBROUTINE RMNORM(A,M,N,NRM)
      REAL A(M,N),NRM
           NRM=0.0
           DO 20 I=1,M
              ROWSUM=0.0
              DO 21 J=1,N
C
C     FIND ROW SUM OF EACH ROW
C
                 ROWSUM=ROWSUM+ABS(A(I,J))
 21           CONTINUE
C
C    ASSIGN MAXIMUM ROWSUM TO NRM
C
              IF (ROWSUM.GT.NRM) NRM=ROWSUM
 20        CONTINUE
      RETURN
      END
```

```
C
C***********************************************************************
C     IT FINDS THE DIFFERENCE OF TWO INTERVAL MATRIX
C     A(*,*,2) - B(*,*,2) = C(*,*,2)
C
C***********************************************************************
      SUBROUTINE SUBM(A,B,M,C)
      REAL A(M,M,2),B(M,M,2),C(M,M,2)
           DO 30 I=1,M
              DO 30 J=1,M
                 CALL SUBI(A(I,J,1),A(I,J,2),B(I,J,1),B(I,J,2),
     *           C(I,J,1),C(I,J,2))
 30        CONTINUE
      RETURN
      END
```

```
C
C***********************************************************************
C
C  MULTIPLICATION OF TWO INTERVAL
C  INT(R,S)*INT(T,U) = INT(E,F)
C
C***********************************************************************
      SUBROUTINE MULI(R,S,T,U,E,F)
      REAL A,B,C,D,E,F,X
      A=R
      B=S
      C=T
      D=U
C
C     TEST SIGN OF ENDPOINTS
C
      IF (A.LT.0.0) THEN
         IF(C.GE.0.0) THEN
C
C     A<0 AND C>0
C
            X=C
            C=A
            A=X
            X=D
            D=B
            B=X
            GO TO 40
            ENDIF
C
C     A <0 AND C<0
C
         GO TO 41
         ENDIF
C
C     A >0 AND C>0
C
      IF (C.GE.0.0) THEN
         E=A*C
         F=B*D
         GO TO 49
         ENDIF
C
C     A > 0 AND C < 0
C
 40   E=B*C
C
C     D > 0
C
      IF (D.GE.0.0) THEN
         F=B*D
         GO TO 49
```

```
            ENDIF
         F=A*D
         GO TO 49
 41      IF (B.GT.0.0) THEN
            IF (D.GT.0.0) THEN
C
C        A < 0 ,B >0 AND C<0 ,D>0
C
               X=A*D
               Y=B*C
               E=MIN(X,Y)
               X=A*C
               Y=B*D
               F=MAX(X,Y)
               GO TO 49
               ENDIF
C
C       A<0 B>0 AND C<0 D<0
C
             E=B*C
             F=A*C
             GO TO 49
          ENDIF
C
C       A<0 B<0
C
         F=A*C
         IF (D.LE.0.0) THEN
            E=B*D
            GO TO 49
            ENDIF
         E=A*D
 49      CONTINUE
         CALL ENLINT(E,F)
         RETURN
         END
```

```
C
C     ENLARGE INTERVAL WITH EPS
C
C***********************************************************************
C
C     THIS ROUTINES ENLARGES INTERVAL BOUNDARIES BY EPSILON AMOUNT
C
C***********************************************************************
C
      SUBROUTINE ENLINT(E,F)
      COMMON EPS
      IF(E.GT.O.O) THEN
        E=E*(1.-EPS)
      ELSE
        E=E*(1.+EPS)
      ENDIF
      IF(F.GT.O.O) THEN
        F=F*(1.+EPS)
      ELSE
        F=F*(1.-EPS)
      ENDIF
      RETURN
      END
```

```
C
C***********************************************************************
C
C    MULTIPLICATION OF TWO INTERVAL MATRIX
C
C***********************************************************************
C
       SUBROUTINE MULIM(A,M1,II,M,N,B,C)
      REAL A(M1,M,2),B(0:20,M,N,2),C(M1,N,2),TEMP1,TEMP2
           DO 50 I=1,M1
              DO 50 J=1,N
                C(I,J,1)=0.0
                C(I,J,2)=0.0
 50      CONTINUE
           DO 51 I=1,M1
              DO 51 J=1,M
                 DO 52 K=1,N
                    CALL MULI(A(I,J,1),A(I,J,2),B(II,J,K,1),B(II,J,K,2),
     *              TEMP1,TEMP2)
                    C(I,K,1)=C(I,K,1)+TEMP1
                    C(I,K,2)=C(I,K,2)+TEMP2
 52              CONTINUE
 51        CONTINUE
           RETURN
           END
```

```
C
C*********************************************************************
C
C    DIVISION OF TWO INTERVAL
C
C  INT(A,B)/INT(C,D)=INT(A,B)*1/INT(C,D) = INT(E,F)
C  INT(C,D) SHOULD NOT CONTAIN ZERO
C
C*********************************************************************
C
      SUBROUTINE DIVI(A,B,C,D,E,F,ERR)
      ERR=0
C
C     CHECK INVALID INTERVAL
C
      IF(A.GT.B.OR.C.GT.D) THEN
        ERR=1
        PRINT *,'INVALID PARAMETERS'
        GO TO 899
      ENDIF
C
C     DIVISION IS NOT DEFINED FOR INTERVALS CONTAINING ZERO
C
      IF(C.LE.0.0) THEN
         IF(D.GE.0.0) THEN
            ERR=2
            PRINT *,'INTERVAL CONTAINS ZERO'
            GO TO 899
         ENDIF
      ENDIF
C
C     TAKE RECIPROCAL OF THE INTERVAL
C
      TEMP=C
      C=1./D
      D=1./TEMP
      CALL MULI(A,B,C,D,E,F)
 899  CONTINUE
      RETURN
      END
```

```
C
C*********************************************************************
C
C     THIS ROUTINE TAKES THE INTERSECTION OF TWO INTERVALS
C     --------------------------------------
C        A      C     B            D
C               +++++++
C               INTERSECTION OF INT(A,B) AND INT(C,D)
C
C*********************************************************************
C
      SUBROUTINE INTER(A,B,C,D,E,F,IFL)
      IFL=0
C
C     INTERVALS ARE DISJOINT
C
      IF(A.GT.D )IFL=1
      IF(C.GT.B)IFL=2
      IF(IFL.GT.0) GO TO 60
      E=MAX(A,C)
      F=MIN(B,D)
 60   CONTINUE
      RETURN
      END
```

VITA 2

Zeynep Aysegul Karacal

Candidate for the Degree of

Master of Science

Thesis: INTERVAL MATHEMATICS AND LINEAR PROGRAMMING APPLICATIONS

Major Field: Computing and Information Science

Biographical:

Personal Data: Born in Istanbul, Turkey, March 1, 1960, the daughter of Mr. and Mrs. R.O. Izgu.

Education: Graduated from Ankara Fen Lisesi, Ankara, Turkey, in June 1977; received Bachelor of Science Degree in Industrial Engineering from Middle East Technical University, Ankara, Turkey in June 1982; completed requirements for the Master of Science Degree in Computer Science at Oklahoma State University, Stillwater, Oklahoma in May 1988.

Professional Experience: Research Assistant, Industrial Engineering Department, Middle East Technical University, June 1982, to April 1984.