A COMPUTER STUDY

of

THE USE OF KHACHIYAN'S ALGORITHM

IN LINEAR PROGRAMMING

Submitted to the
University Honors Council

U H P 450

March 16, 1981

Charles David Gay

TABLE OF CONTENTS

CHAPTER 1

The author of the new mathematical theorem known as "Khachiyan's Algorithm" is twenty-seven year old Leonid Genrikhovich Khachiyan. He is a researcher at the Soviet Academy of Sciences in Moscow.

Mr. Khachiyan is of Armenian background and his father, Genrickh, is also a mathematician. Khachiyan still lives at home with his parents in Moscow. Khachiyan graduated from an ordinary Moscow high school in 1969 and from a school of engineering in 1974. He defended a dissertation last year in theoretical mathematics and has now reached a level of "kandidat", which is virtually the equivalent of a doctorate in the United States, at the Computer Center under the Academy of Sciences.

Khachiyan first submitted his paper concerning his new algorithm to the head of the Computer Center in October 1978. According to Khachiyan, the fact the head accepted it and had it printed meant he confirmed the importance of the paper. Khachiyan has announced his pleasure over the reaction to his paper, but is "surprised by the superlative terms in which it has been reported."

CHAPTER 2

Before beginning a description of Khachiyan's Algorithm, a short description of some ideas of linear programming is in order. A linear programming problem consists of a quantity to be maximized (or minimized) which is called the objective function. This objective function is subject to (that is, must stay within the bounds given by) a list of linear inequalities called constraints.

Two other points which form a basis behind linear programming and Khachiyan's Algorithm should be clarified. These points are consistent and inconsistent systems of linear equations. Observe the shaded region of the plane in R^2 shown in figure 1. This region is the intersection of the following four half-planes: x_1 _ 1, x_1 _ 2, x_2 _ 1, x_2 _ 2 which can be written in the following form:

$$\begin{array}{rcrcr} -x_1 & + & 0x_2 & \_ & -1 \\ x_1 & + & 0x_2 & \_ & 2 \\ 0x_1 & + & x_2 & \_ & -1 \\ 0x_1 & + & x_2 & \_ & 2 \end{array}$$

It is easy to see that there are points contained within this shaded region which will satisfy all of the inequalities simultaneously. This indicates that the system of inequalities is consistent, and thus has a solution.

In figure 2, there are two shaded regions (still in the R^2 plane). The region in the upper right is given by x_1 _ 2 and x_2 _ 2, while the one in the lower left is given by $x_1 \leq 1$ and $x_2 \leq 1$. These inequalities may also be combined into a single system:

$$\begin{aligned} x_1 + 0x_2 &\leq 1 \\ 0x_1 + x_2 &\leq 1 \\ -x_1 + 0x_2 &\leq -2 \\ 0x_1 - x_2 &\leq -2 \end{aligned}$$

It is easy to see that it is impossible to satisfy these four inequalities with a single point. There are no points which satisfy them all.

CHAPTER 3

Khachiyan's method is a procedure not only to check the consistency of a system of inequalities, but to also find the coordinates of a point which will satisfy all the inequalities or at least determine the points with a small margin of error. The revolutionary aspect of the algorithm is the number of steps required to find a solution (which affects the speed of computation). The maximum number of steps requird to find a solution increases as the number of variables within the inequalities increases.

Since the efficiency of an algorithm is measured by the time required to solve the worst possible case, the maximum number of steps, rather than the minimum, is the point of concern. With Khachiyan's method, the number of steps grows far more slowly than with any other known method.

According to Khachiyan, the execution of the algorithm will require at most $N = 16Ln^2$ iterations, where L and n are based on the inequalities given, with n being the number of columns of the matrix given by the starting system of inequalities. For example, for $0x_1 - x_2 \leq 1$ and $x_1 + 0x_2 \leq 2$ you get the following matrix:

$$\begin{array}{rcc} & \multicolumn{2}{c}{\text{columns}} \\ : & 0 & -1 \\ \text{rows}: & & \\ : & 1 & 0 \end{array}$$

which has n = 2 columns. L is computed by the following equation:

$$L = \left[\!\left[\sum_{i=1}^{m} \sum_{j=1}^{N} \log_2(|a_{ij}| + 1) + \sum_{i=1}^{m} \log_2 (|b_i| + 1) + \log_2 nm \right]\!\right] + 1$$

where a_{ij} is the element of the matrix which is found in the i^{th} row and the j^{th} column, b_i is the constraint for each inequality (i.e. the number to the right of the $\leq$ sign), m is the number of rows in the beginning matrix, and $[\![x]\!]$ denotes the greatest integer less than or equal to x. In an effort to control loss of accuracy and numerical instability due to extensive computation, the values computed at each step are required to be accurate to 2^{-27nL}.

The fact which gives Khachiyan's Algorithm its revolutionary concept is that as the inequalities increase in size and become more complicated, the number of steps in the algorithm increases as a polynomial of n. This means that determining the consistency of a set of inequalities belongs to the class of problems which are solvable in polynomial time. In other words, the number of steps increases as a fixed power of n (e.g. n^2 or n^3). This may not seem important, except for the fact that methods being used now increase their steps as exponential functions of n (e.g. 2^n or 3^n). An exponential function increases far more rapidly than one which is polynomial.

The principle behind Khachiyan's Algorithm is much like the traditional method of catching fish in a net: casting the net over such a large area that some of what is wanted must be inside, then gradually decreasing the volume of the net. When the volume is sufficiently reduced, it becomes obvious whether or not anything has been caught.

In place of the net, Khachiyan uses an ellipse. In the algorithm, Q_k denotes a matrix specifying an ellipse E_k centered at the point x_k subject to

$$E_k \doteq [y : y = x_k + Q_k z, \|z\| \leq 1]$$

The initial choice of x_0 and Q_0 gives a sphere of radius 2^L which is centered at the origin. It can be shown that this sphere (ellipse) will contain at least a minimum number of solution points. The ellipses then change position and shrink, but they still contain a minimum volume of solutions. According to Khachiyan, once the ellipse has shrunk to its minimum volume (which it will do within $16Ln^2$ steps) it can contain only solutions. Thus, either the center of the ellipse is a solution, or the system was inconsistent and there are no solutions.

The ellipses in Khachiyan's Algorithm are constructed, according to <u>BYTE</u>, in the following manner (consult figures 3 and 4), which will derive E_1 and x_1 from E_0 and x_0:

1) Draw a chord (line) through x_0, which is parallel to the boundary of the inequality most severely violated (this is determined within the algorithm). This chord cuts the ellipse E_0 at the points p_1 and p_2. The solution set (this is assuming x_0 was not a solution set) of the inequalities now lies on one side of this chord. Which side can be determined by examination.

2) The new ellipse E_1 passes through p_1 and p_2, has its center on the solution side of the chord, and is tangent to the old ellipse E_0 at point T

3) Choose the ellipse with the smallest volume out of the family of ellipses which will satisfy the above conditions. x_1 is the center of this new ellipse E_1

The above description of Khachiyan's Algorithm comes from the August issue of BYTE magazine. Though it gives a good description of the manner in which the algorithm works, the implementation is very difficult due to the manipulation of matrices needed to change the ellipses in this manner. It requires the computation of norms, magnitudes, and orthogonal matrices. For my program, I follow a slightly less exhaustive method of computation for Khachiyan's Algorithm found in an article by Bengt Aspvall and Richard E. Stone (in The Journal of Algorithms). A short description of this method follows, with a more extensive description found in Chapter 7.

The Aspvall and Stone version of Khachiyan's Algorithm begins much the same as the original with a matrix A formed from the system of inequalities. A column vector x with n entries is set to 0, and a n-by-n matrix B which is also set to 0 except for a non-zero constant which is inserted in the (1,1) to (n,n) diagonal of the matrix. The matrix B and vector x are then manipulated through matrix multiplication and scalar multiplication and division. The manipulations and computations continue until either a feasible solution is found, no solution exists, or the iterations exceed their bound.

CHAPTER 4

Unfortunately, Khachiyan's Algorithm is not without faults. Khachiyan is concerned with calculation procedures having only limited accuracy. As a result, the cumulative error of being unable to accurately represent numbers in the computer may be fatal, especially in determining the consistency or inconsistency of a system of inequalities. This defect could conceivably be overcome by using extreme precision with hundreds of thousands of digits to the right of the decimal point. This hardly seems, however, like a good solution.

Another problem is in the fact that Khachiyan's method only allows the use of integer coefficients whereas most coefficients of linear programming problems are not integers. This may be overcome by multiplying the coefficients by a sufficiently large multiple of ten as to make the smallest decimal fraction into an integer. Once again, however, this may not be, depending on conditions, an easy solution to a difficult problem.

The most significant problem is that of actual computing time. Though Khachiyan's method is polynomial bounded and the worst case behavior of the method is better, the popular theory among those who have done research into Khachiyan's Algorithm (see BYTE, Siam News, JMAP), is that the average running time is a good deal worse. It has been estimated through computation of bounds and running times for the steps of the algorithm, that a problem which can be solved in thirty minutes by the simplex method would take approximately fifty million years to solve using Khachiyan's method. This can be seen from a summary of a study done at Stanford on an IBM 370-168 computer. Using the following equations:

$$\bar{n} = n - m \qquad \bar{m} = n$$

$$m = \bar{m} - n \qquad n = \bar{m}$$

and letting m = 1000 and n = 3000 we get $\bar{n}$ = 2000 and $\bar{m}$ = 3000. On Stanford's computer, Khachiyan's algorithm requires $120\bar{m}\bar{n}^3$ steps which yields:

$$120 \times 3000 \times 2000^3 = 2.88 \times 10^{15} \text{ steps}$$

The simplex method requires 3.5m steps which yields:

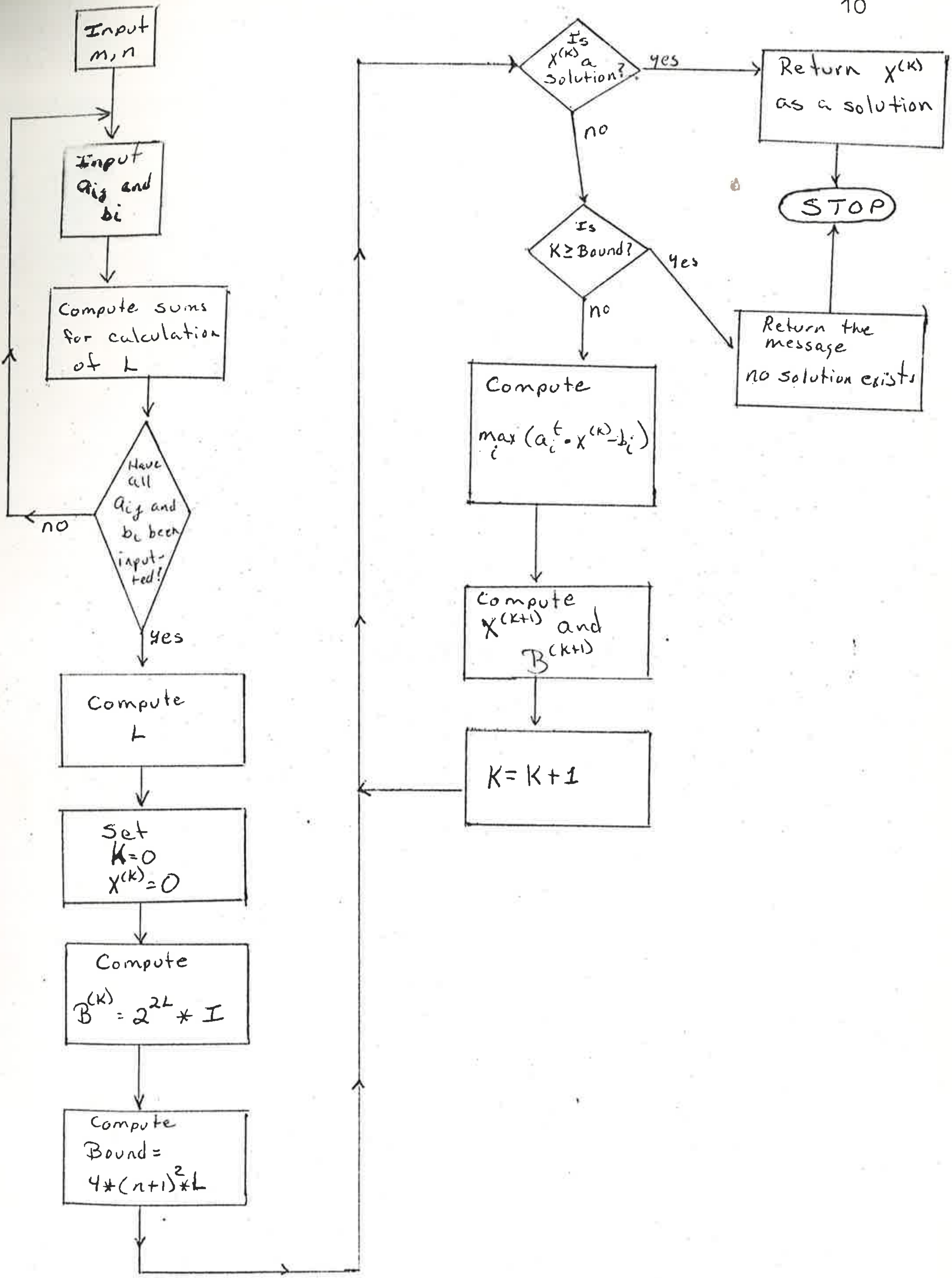
$$3.5 \times 1000 = 3.5 \times 10^3 \text{ steps}$$

Each step on the IBM 370-168 requires .5 seconds or at a rate of 6.3×10^7 steps per year. Thus the estimated (Khachiyan) and the actual (simplex) times to solve the problem are:

Khachiyan: 2.88×10^{15} steps = 50,000,000 years

simplex: 3.50×10^3 steps = 30 minutes

CHAPTER 5


Input m, n
Input a_{ij} and b_i
Compute sums for calculation of L
Have all a_{ij} and b_i been inputted?
no
yes
Compute L
Set K=0 $X^{(K)}=0$
Compute $B^{(K)} = 2^{2L} * I$
Compute Bound = $4*(n+1)^2*L$
Is $X^{(K)}$ a solution?
yes
no
Return $X^{(K)}$ as a solution
STOP
Is K ≥ Bound?
yes
no
Return the message no solution exists
Compute $\max_i (a_i^t \cdot X^{(K)} - b_i)$
Compute $X^{(K+1)}$ and $B^{(K+1)}$
K = K + 1

CHAPTER 6

```
 5   DOUBLE PRECISION  D,L,DENOMB,DENOM,C,SUM1,SUM2,SUM,
                       BOUND,LARG,HOLD,Q,R,W
 6   INTEGER  M,N,K,I,J,BIG,F,G
12   INTEGER A(M,N)
13   REAL  B(M),Z
14   DOUBLE PRECISION  NUMERX(N,1),DENOMX(1,N),B2(N,N),X(N),
                       THETA(M),NUMERB(N,N)
     READ (1,15) M,N
15   FORMAT (2I9)
     READ (1,16) F,G
16   FORMAT (2I9)
     C = ALOG(2.)
17   K = 0
     HOLD = 0
18   SUM1 = 0
20   SUM2 = 0
22   DO 23 I = 1,N
     X(I) = 0
23   CONTINUE
     DO 25 I = 1,N
       DO 24 J = 1,N
         B2(I,J) = 0
24     CONTINUE
25   CONTINUE
/* INPUTS MATRIX A AND VALUES FOR B */
26   DO 40 I = 1,M
27     DO 35 J = 1,N
         READ (1,33) A(I,J)
33       FORMAT (I9)
35     CONTINUE
```

```
       READ (1,37) B(I)
    37 FORMAT (F9.1)
    40 CONTINUE
/* CALCULATES THE VALUE OF L */
    42 DO 45 I = 1,F
         DO 44 J = 1,G
           SUM1 = SUM1 + (ALOG(IABS(A(I,J) + 1.0))/C
    44   CONTINUE
       SUM2 = SUM2 + (ALOG(ABS(B(I)) + 1.0))/C
    45 CONTINUE
       Z = M * N
       Q = SUM1 + SUM2 + ALOG(Z)/C
    46 L = Q + 1
/* CALCULATES THE MATRIX B2 */
    48 D = 2 ** (2*L)
       DO 50 I = 1,N
         B2(I,I) = D
    50 CONTINUE
    55 BOUND = 4 * ((N + L)**2) * L
/* COMPUTES SUMS TO DETERMINE WHETHER A SOLUTION HAS BEEN FOUND */
    60 DO 100 I = 1,M
         SUM = 0
           DO 80 J = 1,N
             SUM = SUM + A(I,J) * X(J)
    80     CONTINUE
         SUM = SUM - B(I)
```

```
      THETA(I) = SUM
100  CONTINUE
/* COMPUTES VALUE OF I WHICH CONTAINS THE LARGEST VALUE OF THETA */
/* AND DETERMINES IF A SOLUTION HAS BEEN FOUND */
     I = 1
     J = 2
120  R = THETA(I)
130  IF (J .GT. M) GO TO 150
       IF (THETA(I) .GE. THETA(J)) GO TO 135
         I = J
         J = J + 1
         GO TO 120
135    J = J + 1
140    GO TO 130
150  BIG = I
     IF (R .LT. 0) GO TO 500
/* DETERMINES IF ITERATION IS BEYOND BOUND */
     IF (K .LT. BOUND) GO TO 200
     WRITE (3,180)
180  FORMAT (' NO SOLUTIONS EXIST ')
     GO TO 999
/* CALCULATES NUMERATOR VALUE FOR CALCULATION OF NEXT ITERATION OF X */
200  DO 250 I = 1,N
       DO 220 J = 1,N
         HOLD = HOLD + B2(I,J) * A(BIG,J)
220    CONTINUE
     NUMERX(I,1) = HOLD
```

```
      HOLD = 0
250   CONTINUE
/* COMPUTES DENOMINATOR VALUE FOR CALCULATION OF NEXT ITERATION OF X */
255   DO 270 I = 1,N
        DO 260 J = 1,N
          HOLD = HOLD + A(BIG,J)*B2(I,J)
260     CONTINUE
      DENOMX(1,I) = HOLD
      HOLD = 0
270   CONTINUE
275   DO 280 I = 1,N
        HOLD = HOLD + DENOMX(1,I) * A(BIG,I)
280   CONTINUE
      DENOM = HOLD
      DENOMB = HOLD
290   DENOM = DSQRT(DENOM) * (N + 1)
      HOLD = 0
/* REVALUE X */
295   DO 300 I = 1,N
        X(I) = X(I) - NUMERX(I,1)/DENOM
300   CONTINUE
/* CALCULATES NUMERATOR VALUE FOR COMPUTING NEW B2 VALUES */
305   DO 315 I = 1,N
        DO 310 J = 1,N
          NUMERB(I,J) = NUMERX(J,1)
310     CONTINUE
315   CONTINUE
```

```
/* CALCULATES DENOMINATOR VALUE FOR COMPUTING NEW B2 VALUES */
317  DENOMB = DENOMB * (N + 1)
/* REVALUE B2 */
     W = N
     W = (W*W)/((W*W)-1)
319  DO 330 I = 1,N
       DO 320 J = 1,N
         B2(I,J) = (B2(I,J) - ((NUMERB(I,J)/DENOMB)*2))*W
320    CONTINUE
330  CONTINUE
     K = K + 1
340  GO TO 60
500  WRITE (3,510)
510  FORMAT (' THE FOLLOWING IS A FEASIBLE SOLUTION ')
520  WRITE (3,530) (X(I),I = 1,N)
530  FORMAT (D50.24)
     WRITE (3,535) K
535  FORMAT (' THE VALUE OF K IS ',I9)
999  STOP
     END
```

CHAPTER 7

In the following description of the program, all references to lines of calculations are in regard to the program found in the preceding chapter. Those lines whose meanings are obvious are omitted.

The first two lines of the program (5 and 6) are data descriptors. Double precision is used to extend the accuracy of the computations in the program. The variables D, BIG, SUM, SUM1, SUM2, HOLD, R, W, and Q are value holders used to perform computations. NUMERB, DENOMB, and DENOM are used in the calculation of the iterative values of B2 and X. L, BOUND, and C are numeric constants. The variables M and N are the inputted number of rows and columns respectively of the matrix A which is constructed an follows:

$$\begin{bmatrix} A & O_{(m,m)} \\ -I_n & O_{(n,m)} \\ O_{(n,n)} & -A^t \\ O_{(m,n)} & -I_m \\ c^t & -b^t \\ -c^t & b^t \end{bmatrix} z \leq \begin{bmatrix} b \\ O_{(n,1)} \\ -c \\ O_{(m,1)} \\ \epsilon \\ \epsilon \end{bmatrix}$$

with ϵ being an inputted bound which stops the program when the solution is within ϵ of being exact.

Where A, b, and c come from the original inequalities. F and G are the number of rows and columns of the system of constraints in the inequalities. K, I, and J are counter variables used in loops. Lines 12 and 14 are data descriptors for those values which need M and N to have numerical values before being described. A and B are the matrix and vector which come from the system of inequalities. NUMERX and DENOMX are used for the calculation of the next iterative of X. B2 and THETA are value holders used in computation and X is the vector in which the feasible solution will occur.

Lines 26 through 40 input the values which come from the system of inequalities and are put into matrix A and vector B. This is done with nested Do-loops, which is one loop inside of another. The loop ranging from lines 27 to 35 is contained in the loop in lines 26 to 40. The inner loop will run through all of its iteratives, in this case J will run from 1 to N, before going out of the loop and advancing the iterative I and beginning again.

Lines 42 through 46 calculate the value of L with INT(Q) giving the integer value of Q. Lines 48 through 50 insert the value 2^{2L} into the diagonal of the matrix B2 as explained in a previous chapter.

Lines 60 through 100 make the evaluation as to whether or not a solution has been found. The inner loop calculates a value called SUM, which is a summation of the equation $(\Sigma(a_{ij} * x_i)) - b_i$ for each row of beginning constraints. Once the inner loop is exited, the value of SUM is placed into THETA(I) and the process is then repeated until I is greater than M.

Lines 120 through 190 use a set of IF-Then statements to find the value of I where THETA has the greatest value. If this value of THETA, R, is less than zero, a solution has been found and is returned, otherwise the program continues. The value of I is held in the variable BIG and used in the calculation of the next iteratives of X and B2. Lines 200 through 250 calculate the numerator of the following equation which finds the next iterative of X:

$$X^{(k+1)} = X^{(k)} - \frac{1}{n+1} * \frac{B^{(k)} a_i}{\sqrt{a_i^T B^{(k)} a_i}}$$

Lines 255 through 290 evaluate the denominator for the same equation. Lines 295 to 300 insert the new values of X in place of the old ones.

Lines 305 through 315 and line 317 calculate the values for the numerator and denominator respectively of the following equation for the next iteration of B2:

$$B2^{(k+1)} = \frac{n^2}{n^2 - 1} \left[B2^{(k)} - \frac{2}{n+1} * \frac{(B2^{(k)} a_i)(B2^{(k)} a_i)^T}{a_i^T B2^{(k)} a_i} \right]$$

Upon inspection, you will find the two calculations are very similar. Instead of recomputing the like values, I used the values computed for $X^{(k+1)}$ to calculate $B2^{(k+1)}$. Lines 319 through 330 finish needed calculations and put the new values into the matrix. Line 340 sends the computer back to the beginning of the checking and calculation sequence, thus completing one step. Lines 500 through 530 print out X as a feasible solution if one is found.

CHAPTER 8

Once the program was in proper running order, two analysis runs were made to determine the efficiency of the program. In the first test, a trivial problem with n = 1 variables being used, my program required 54 steps to achieve a solution as opposed to 35 steps used by the BYTE algorithm and 1 step for the Simplex method. In the second run, the following problem was used:

$$\text{maximize } P = \begin{bmatrix} 2 & 3 \end{bmatrix} \circ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\text{subject to} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \circ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$\text{with} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 0$$

This problem is constructed in the September issue of BYTE, and was chosen for comparison of their program's efficiency to mine. For this test, my program required a total of 390 steps to reach a solution, the BYTE algorithm 525 steps, and the Simplex method required 3 steps.

NUMBER OF STEPS

n	GAY	BYTE	SIMPLEX
1	54 ($\epsilon = .01$)	35 ($\epsilon = .01$)	1
2	390 ($\epsilon = .01$)	525 ($\epsilon = .01$)	3

A few facts need to be interjected at this point. The authors of the BYTE algorithm used as a reference, the work by B. Aspvall and Richard Stone upon which my program is based. The BYTE program, however, is written in BASIC for the

TRS-80 mini-computer while my program as well as the Simplex program, is written in FORTRAN which is a much more powerful language. Also, the BYTE algorithm is more complicated than mine and produces its own dual matrix rather than requiring it to be input, as in my algorithm.

Even though my algorithm did require less steps than the BYTE method, both are far surpassed by the Simplex program. It must be remembered, however, these are very small problems. Khachiyan's method would certainly require less steps than the Simplex method in certain real world situations. Some of these real world problems, according to BYTE, may involve 10,000 inequalities and 50,000 variables.

So just as the Simplex method has been refined and developed into the powerful tool it is today, so must Khachiyan's Algorithm be manipulated and molded into what it has the potential to become. Much more work and experience with Khachiyan's method must be achieved before deciding if it will be of practical value, thus realizing its theoretical advantages.

CHAPTER 9

References for a further description of linear programming and the Simplex method may be found in most libraries by consulting the card catalogue. For information on Khachiyan's Algorithm, periodicals are the place to look. The Pembroke State University library has information on the algorithm in the following places:

1) Periodical Room - <u>BYTE</u> magazine
2) Micro film - <u>New York Times</u>
3) Bound Volumes - <u>Science</u> vol. 206 Nov., 1979.
<u>Scientific American</u> vol. 242

The Department of Mathematics may also be contacted for additional information on the algorithm. For all other sources, inter-library loan was consulted with literature coming from other schools in North Carolina. Also, the library at the Massachusetts Institute of Technology and the Russian embassy were contacted.

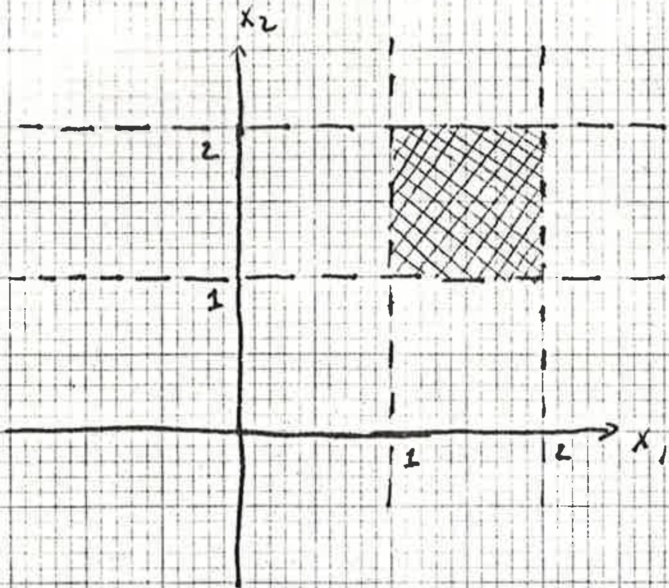


Figure 1

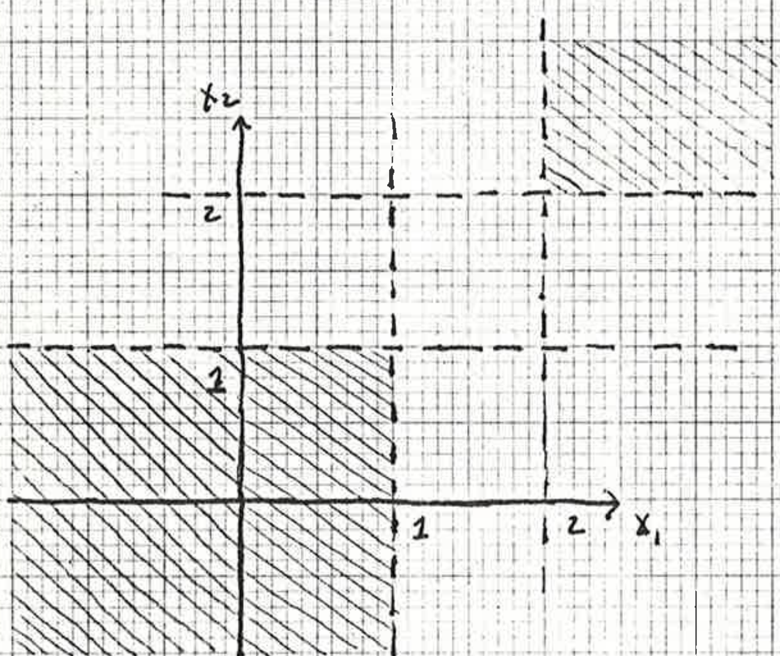
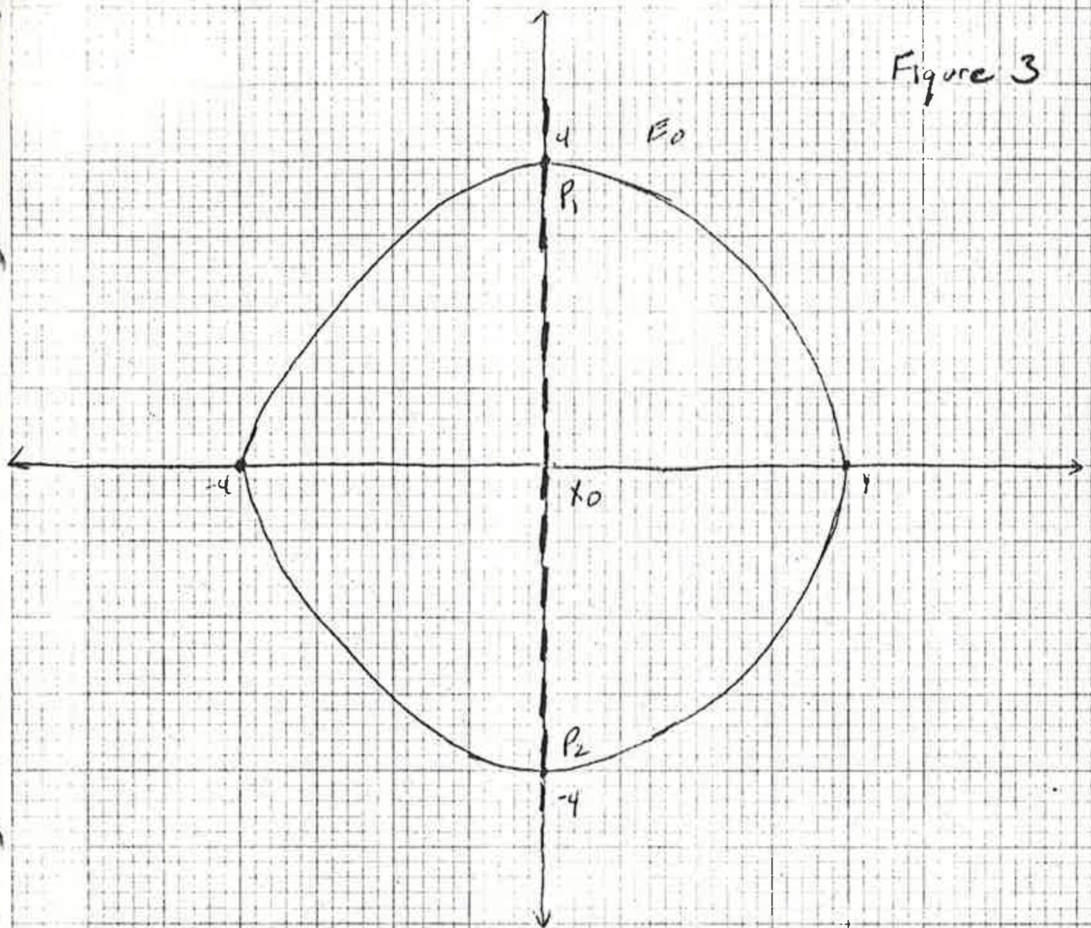


Figure 2

Figure 3



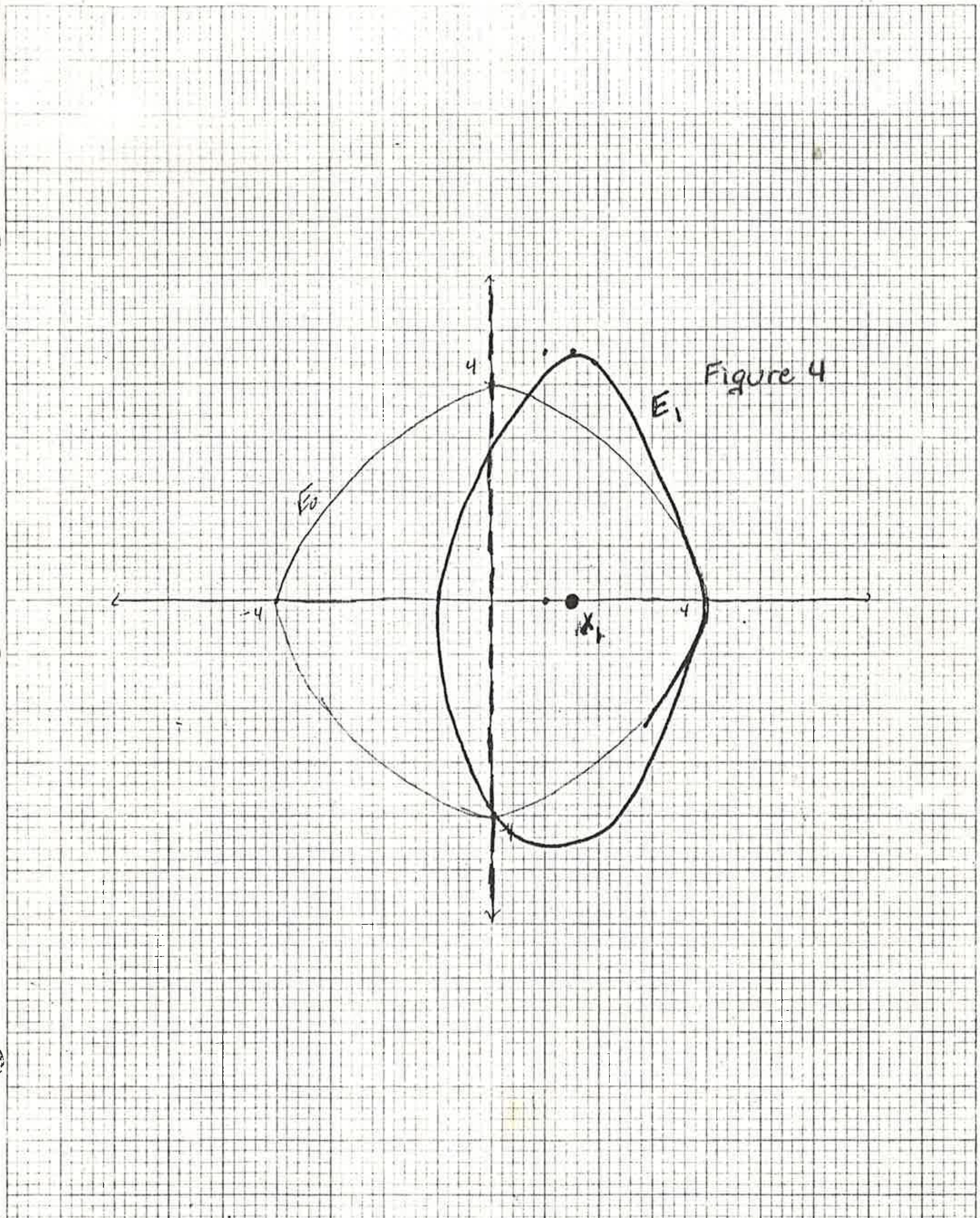


Figure 4
4
-4
4
E_0
E_1
X_1

A SELECTED BIBLIOGRAPHY

Aspvall, Bengi and Richard E. Stone. "Khachiyan's Linear Programming Algorithm." Journal of Algorithms, 1, (1980) , 1-13.

Berresford, G. C. , A. M. Rockett, and J. C. Stevenson. "Khachiyan's Algorithm, Part 1: a new Solution to Linear Programming Problems" (BYTE , 8, (Aug., 1980) , 198-108.

------------------ "Khachiyan's Algorithm Part 2: Problems with the Algorithm." BYTE, 9, (Sept., 1980), 242-255.

Dantzig, George B., KHACHIYAN'S ALGORITHM: A comment by George Dantzig" Siam News, vol. 13, no. 5, (1980) , 1.

Flanagan, Dennis., ed., "Fascinatin' Algorithm" Scientific American, vol. 242 no. 1, (1980) 80.

Giordano, Frank R. and Carroll O. Wilde. "ON JARGON-Khachiyan's Algorithm" The Journal of Undergraduate Mathematics and its Applications, unit 3, Autumn, (1980), 117-122.

Lawler, Eugene L., "The Great Mathematical Sputnik of 1979" The Mathematical Intelligencer, vol. 2, no. 4, (1980), 191-198.

Lovasz, L., "A New Linear Programming Algorithm-Better or Worse Than the Simplex Method?" The Mathematical Intelligencer, vol. 2, no. 3, (1980), 141-146.

Whitney, Craig B., "Soviet Mathematician is Obscure No More" The New York Times, (Nov. 27, 1979), c1,