A program which implements a new algorithm for the solution of the linear minimax approximation problem

N. PAPAMARKOS

Department of Electrical Engineering, Democritus University of Thrace, 67100 Xanthi, Greece

This paper presents a Turbo-Basic program that implements a revised version of a new algorithm for the fast and optimum solution of the general linear minimax approximation problem. The algorithm proceeds iteratively and terminates when the global minimum value of the objective function is reached. The initial point may be selected arbitrarily or it may be optimally determined through a linear method to speed up algorithmic convergence. This algorithm is much faster and requires less storage than other approximation techniques. The program is implemented on an IBM-PC AT and tested by a number of approximation problems. Analytical examples are presented to illustrate how the program is used and the effectiveness of the algorithm.

Key Words: Minimax approximation, fast algorithm, linear programming,
Turbo-Basic program for IBM-PC AT, parameters estimation.

INTRODUCTION

The program presented in this paper is developed in order to solve fast and effectively the general linear minimax approximation problem (LMAP). The formulation of the LMAP consists of determining the variables x_j, $j = 1, 2, \ldots, N$ through the solution of the problem

minimize g

subject to

$$\left| \sum_{j=1}^{N} d_{ij} x_j - p_i \right| \leqslant g \qquad \text{for } i = 1, 2, \ldots, K \tag{1}$$

where the x_j variables are unrestricted in sign and d_{ij}, p_i are known coefficients. It is well known that the above approximation problem is frequently attended in many engineering applications, such as digital filters design[1,2,3] and digital image processing techniques.[4]

Accepted September 1989. Discussion closes April 1990.

Until recently, several algorithms have been proposed for the solution of the LMAP.[5,6] The majority of these solved the LMAP by exploiting basic Linear Programming (LP) methods, and more specifically with the Revised Simplex Algorithm (RSA).[7] This procedure is satisfactory for problems of small dimensionality, but for problems of large dimensionality is associated with known disadvantages such as high computational cost and large memory requirements.

The optimization algorithm employed in this article is based on a revised version of the method proposed by N. Papamarkos *et al.*[8] This method is based upon the full utilization of the formulation of the problem constraints and the feasible region. It specifies feasible direction of decreasing values of the objective function resulting always in the global minimum. A definite advantage of the algorithm relates to the arbitrary choice of an initial point, and as such, any feasible point may suffice. But speed of convergence is substantially improved by a judicial choice of an initial point through a fast linear procedure which gives an optimum initial point close to the problem's global minimum. In comparison with the primal form of the algorithm, this paper proposes several modifications which speed up algorithmic convergence. This final algorithm is much faster and requires less storage than other LMAP techniques and hence is ideally suitable for situations where the dimensionality of the approximation problem is large.

FORMULATION AND DEFINITIONS

Referring to the LMAP (1) and without loss of generality, it is considered that $p_i \geqslant 0$ for $i = 1, 2, \ldots, K$. Now, the approximation problem is converted to the following linear form

minimize g

subject to

$$-g + \sum_{j=1}^{N} a_{ij} x_j - c_i \leqslant 0 \qquad \text{for } i = 1, 2, \ldots, 2k \tag{2}$$

where the coefficients a_{ij} and c_i are defined by the relations:

$$a_{ij} = \begin{cases} d_{ij} & \text{for } i = 1, 2, \ldots, K \\ -d_{ij} & \text{for } i = K+1, K+2, \ldots, 2K \end{cases} \tag{3}$$

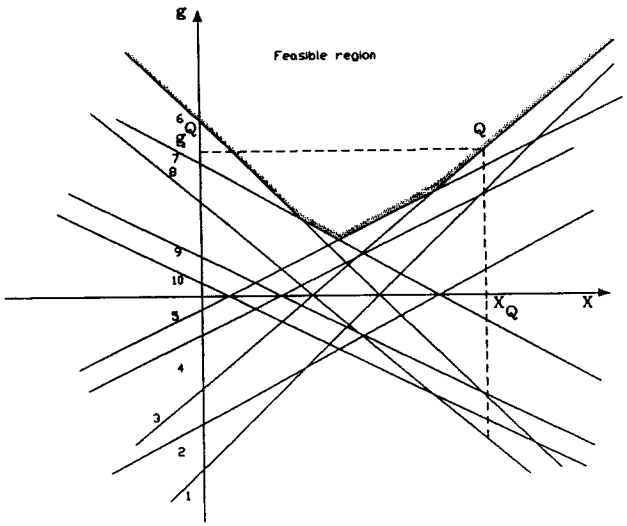

Figure 1. A geometrical interpretation of the minimax problem

$$c_i = \begin{cases} p_i & \text{for } i = 1, 2, \ldots, K \\ -p_i & \text{for } i = K+1, K+2, \ldots, 2K \end{cases} \quad (4)$$

If $X_Q = (x_1^Q, x_2^Q, \ldots, x_N^Q)^T$ is the vector of the x_j's variables at a point Q, then the intersections g_n^Q are defined as the quantities

$$g_n^Q = \sum_{j=1}^{N} a_{nj} x_j^Q - c_n \qquad \text{for } n = 1, 2, \ldots, 2K \quad (5)$$

From the above definitions, it follows that $c_i \geqslant 0$ for $i \leqslant K$ and $c_i \leqslant 0$ for $K < i \leqslant 2K$ whereas g_n's obey the relation

$$g_{n+K}^Q = -g_n^Q \qquad \text{for } n = 1, 2, \ldots, K \quad (6)$$

A geometrical interpretation of the minimax approximation problem, in $N+1$ space for $K = 5$, is depicted in Fig. 1. From this figure, we conclude that:

1. For every vector X_Q, there is only one solution point on the boundary surface of the feasible region, which corresponds to g and which satisfies the condition:

$$g = g^Q = \underset{i=1,2,\ldots,2K}{\text{maximum}} \left(\sum_{j=1}^{N} a_{ij} x_j^Q - c_i \right) \quad (7)$$

2. The minimax approximation problem has a convex Feasible Boundary Surface (FBS).
3. A line in $N+1$ space, defined by two points, Q_1 and Q_2, at least one of which lies on the FBS and on the q hyperplane, will either intersect the FBS at a second point on the hyperplane $p \neq q$ or will be identical with the initial feasible point.[8]

DESCRIPTION OF THE ALGORITHM

Let an initial feasible point Q exist on the hyperplane q, corresponding to X_Q and g, according to relations (5) and (7). At this feasible point, the intersections of all the hyperplanes are computed using relations (5) and (6). If the intersection g_k^Q of the k hyperplane is equal to g, then Q will be a common point of the hyperplanes q and k. All the hyperplanes with intersections equal to g are called basic hyperplanes.

The optimization procedure followed by the algorithm consists of specifying, for every feasible point Q that is found and belongs to FBS and on the hyperplane q, a feasible line of decreasing g. Thus, the line of decreasing g will intersect a hyperplane different from q and the point of intersection will be a feasible point with a corresponding minimax value smaller than g.

Various cases are distinguished depending upon the number of hyperplanes that share Q as a common point. A different procedure is employed for each case, as is described in the sequel.

Case 1: Q belongs to only one hyperplane

In this case the line of decreasing g is specified by Q and the point T, where T is the intersection of the hyperplane q with the g-axis and corresponds to $g^T = -c_q$. The line $\overline{TQ}$ is described by the relations

$$g = g^T + V(g^Q - g^T) = -c_q + V(g^Q + c_q) \quad (8)$$

and

$$x_j = V x_j^Q, \qquad \text{for } j = 1, 2, \ldots, N \quad (9)$$

where V is the scale factor of the line $\overline{TQ}$. The intersection of $\overline{TQ}$ with a constraint l will give a point L and a scale V for which the following holds true

$$\sum_{j=1}^{N} a_{lj} x_j^L - c_l = -c_q + V(g^Q + c_q) \quad (10)$$

or, via (9):

$$V \sum_{j=1}^{N} a_{lj} x_j^Q - c_l = -c_q + V(g^Q + c_q) \quad (11)$$

According though to (5), at the point Q it is true that

$$\sum_{j=1}^{N} a_{lj} x_j^Q = c_l + g_l^Q \quad (12)$$

and relation (11) gives

$$V(g_l^Q + c_l) = c_l - c_q + V(g^Q + c_q) \quad (13)$$

and, therefore

$$V = \frac{c_l - c_q}{g_l^Q + c_l - g^Q - c_q} \quad (14)$$

Now, knowing V, the point L may be determined through the relations (8) and (9) giving

$$g_n^L = V g_n^Q + c_n) - c_n \text{ for } n = 1, 2, \ldots, l-1, l+1, \ldots, 2K \quad (15)$$

The above relation is important in that it permits a fast computation of the intersections at L by avoiding the calculation of the sums $\Sigma_{j=1}^{N} a_{nj} x_j^L$. Obviously, and in agreement with relation (7), the point $L \neq Q$ is chosen for which $g_l \geqslant g_n^L \; \forall n = 1, 2, \ldots, 2K$.

Case 2: Q is a common point of two hyperplanes.

Let Q be a common point of two hyperplanes q_1 and q_2 and, moreover

$$|c_{q_1}| \leqslant |c_{q_2}| \quad (16)$$

We distinguish the following two subcases:

1. $c_{q_1} \cdot c_{q_2} \geqslant 0$ and $|c_{q_1}| > g$.
   In this case, Q is considered to lie only on hyperplane q_1 and therefore, the procedure of Case 1, as applied to the single point, is applicable here.
2. $c_{q_1} \cdot c_{q_2} < 0$ or the constraint $|c_{q_1}| \leqslant g$ is satisfied.
   In this situation, the feasible line of decreasing g is specified by Q and a second common point of hyperplanes q_1 and q_2. The second common point B, of q_1 and q_2 follows from the relations:

$$x_B = \frac{c_{q_1} - c_{q_2}}{a_{q_1 j} - a_{q_2 j}} \qquad \forall j = 1, 2, \ldots, N \tag{17}$$

and

$$g^B = \frac{a_{q_2 j} c_{q_1} - a_{q_1 j} c_{q_2}}{a_{q_1 j} - a_{q_2 j}} \qquad \forall j = 1, 2, \ldots, N \tag{18}$$

The directed line $\overline{BQ}$, defined by two points constitutes a feasible direction of decreasing g. It will, therefore, intersect a third hyperplane at a feasible point which will be common to three hyperplanes.

The line segment $\overline{BQ}$ is described by the equations

$$g = g^B + V(g^Q - g^B) \tag{19}$$

$$x_i = V x_i^Q \qquad \forall i \text{ with } i \neq j \tag{20}$$

$$x_i = x_i + V(x_i^Q - x_B) \qquad \text{for } i = j \tag{21}$$

The intersection of $\overline{BQ}$ with hyperplane l gives a point L for which

$$V = \frac{g^B + c_l - a_{lj} x_B}{g^B + c_l - a_{lj} x_B + g_l^Q - g^Q} \tag{22}$$

and

$$g_n^L = V(g_n^Q + c_n - a_{nj} x_B) + a_{nj} x_B - c_n \tag{23}$$

for $n = 1, 2, \ldots, 2K$ excluding the values $n = q_1$ and $n = q_2$.

Case 3: Q is a common point of T hyperplanes with $3 \leqslant T \leqslant N$.

Here, because Q is a common point of T hyperplanes in $(N+1)$-space, there will be an infinite number of common intersection points for the T hyperplanes. A feasible direction of decreasing g may be specified by a line segment through Q and a second point B which lies on the intersection of the T hyperplanes.

The method for determining the second point B is based on the solution of a system of linear equations. Specifically, since the corresponding value of g is of no consequence to B, the following linear system of T equations is arrived

$$\sum_{j=1}^{T} a_{id_j} x_{d_j}^B = c_i + r_1 g^Q - r_2 \sum_{j=T+1}^{N} a_{ig_j} x_{g_j}^Q \tag{24}$$

where the index i takes T values corresponding to T hyperplanes, r_1, r_2 are variational coefficients (usually $r_1 = 0$ and $r_2 > 1$), and the vectors d, g are chosen such as the linear system (24) to be non singular.

The solution of the linear system (24) gives the x_j^B values of B and, therefore, the line $\overline{QB}$ is described by the equations:

$$g = g^Q + V(g_B - g^Q) \tag{25}$$

$$x_j = x_l^Q + V(x_j^B - x_j^Q) \tag{26}$$

The intersection L, of $\overline{QB}$ with hyperplane l that does not belong to the T set, satisfies the following relation for the scale factor

$$V = \frac{g^Q - g_l^Q}{\sum_{j=1}^{N} a_{lj} x_j^B - g_l^Q - c_l + g^Q - g^B} \tag{27}$$

Case 4: Q is a common point of T hyperplanes with $T \geqslant N+1$.

Let $q_1, q_2, \ldots, q_T$ are the basic hyperplanes at the common Q point. In $(N+1)$-space, the point Q will obviously be the only common point of T hyperplanes. A feasible direction of decreasing g may be determined via the intersection of N hyperplanes passing through Q. One of these hyperplanes must be the hyperplane m which satisfies the following conditions

$$c_m \leqslant 0 \text{ and } |c_m| = \text{minimum } \{|c_{q_i}|\}$$
$$i = 1, 2, \ldots, T \tag{28}$$

Each line satisfying this condition must be examined through a procedure identical to that of Case 3.

If none of the candidate line segments form a feasible direction of decreasing g, then the global minimum value of g, corresponding to Q, has been found and the algorithm terminates successfully.

In the event that such a direction of decreasing g exists, then the line segment corresponding to it will reveal a new intersection point of T' hyperplanes, where $T' \geqslant N+1$, and the above procedure is repeated.

In all four cases outlined above, computation of the intersections is relatively straightforward and may be accomplished rapidly. The procedure for specifying a feasible point of intersection L is accelerated if the hyperplanes l, for which $g_l \geqslant 0$, are examined first since those hyperplanes most probably intersect q on the FBS.

OPTIMUM INITIAL POINT

Algorithmic convergence is further accelerated by the selection of an optimum initial point. An optimum initial point may result through the minimization of an L_2 criterion.

The objective function to be minimized is of the form:

$$J = \sum_{i=1}^{K} \left(\sum_{j=1}^{N} a_{ij} x_j - c_i \right)^2 \tag{29}$$

From (29), we may write

$$\frac{\partial J}{\partial x_k} = 2 \sum_{i=1}^{K} a_{ik} \left(\sum_{j=1}^{N} a_{ij} x_j - c_i \right) = 0 \tag{30}$$

And, therefore, the resulting linear system is of the form

$$\sum_{j=1}^{N} h_{ij} x_j = g_i \qquad \forall i = 1, 2, \ldots, N \tag{31}$$

where

$$h_{ij} = \sum_{k=1}^{K} a_{kj} a_{ki} \qquad \text{and} \qquad g_i = \sum_{k=1}^{K} c_k a_{ki} \tag{32}$$

The linear system (31) is symmetric and may be easily solved using the method of Cholesky[9] with low computational cost.

PROGRAM DESCRIPTION

The above minimax algorithm is coded in Turbo-Basic for IBM-PC and its compatible machines. Turbo-Basic has been used because it is a familiar compiled language, has effective block-structured programming statements, support 8087 and 80287 floating-point processors, and is clearly faster than IBM's Advanced Basic (commonly referred as BASICA) and Microsoft's GWBASIC. The program is menu driven and consists of a main program, one function and sixteen subroutines. All the input data are entered in the main program and in the self prompting mode. The input data required by the program are:

(i) The number of independent variables
(ii) The number of constraints
(iii) The known a_{ij} and c_i coefficients. These coefficients may be entered from a data file or from the keyboard. In the data file, the values of the coefficients, must be stored with the following sequence

$$a_{i1}, a_{i2}, \ldots, a_{iN}, c_i \qquad \text{for } i = 1, 2, \ldots, K$$

(iv) The option of the initial starting point (optimum or random initial point)
(v) The output option (final results only or detailed results for every step of the algorithm). If the detailed results option is selected then the program gives detailed results for the independent variables, intersections values, basic hyperplanes and the computation time in every step.
(vi) The Shift-value, which permits us to shift the origin of the independent variables.

Upon entering all the necessary data, the program can check and change the input data or solve the model. The best solution available at the beginning of each iteration cycle is displayed on the monitor if the option of the 'detailed results' is chosen. Otherwise, when the optimal solution has been found the program displays a new menu that allows the user to choose one of the following options:

(i) Display a list of the input data
(ii) Display the final results only
(iii) Print the input data and the final results
(iv) Terminate the program.

As final results the program gives the optimum values of the independent variables, the optimum value of g, the basic constraints on the optimum solution and the final computation time.

It is noted that additional information about the program's subroutines are given on the program list.

EXAMPLES

The program developed was tested by applying it to several LMAP. Owing to space limitation, here are only two examples for the test performed. The computation time is reported for an IBM-PC AT with 80287 coprocessor. The program was compiled by using the version 1.1 of the TURBO BASIC compiler.

Example 1

In the first example the program is applied to the linear minimax problem of approximating the function

$$G(\omega) = 2e^{-\omega}\sin(\omega) + e^{-2\omega} - x^2 + 4 \tag{33}$$

by the polynomial

$$F(\omega) = x_1 + x_2\omega + x_3\omega^2 + x_4\omega^3 + x_5\omega^4 \tag{34}$$

The sampling interval is $[-1, 1]$ and 41 sampling points were used in approximating $G(\omega)$.

The optimum initial point is found to be equal to

$x_1 = 5.003008045$ $x_4 = -1.088845527$
$x_2 = 0.097213830$ $x_5 = 0.835487554$
$x_3 = -1.059964161$ and $g = 0.044182389$

The print-out of the program at the optimal solution is as follows:

List of the Input Data

Number of variables = 5
Number of constraints = 41
Optimum initial point = Y
Shift-value = 100
Data file = A1.DAT

Final Results:

$g = 0.02464327360$ Computation time = 7.41

$x(1) = 5.00764412998$
$x(2) = 0.11672373652$
$x(3) = -1.09893356228$
$x(4) = -1.12202570794$
$x(5) = 0.87568971186$

Figure 2 shows the form of the approximated polynomial in comparison with the ideal response.

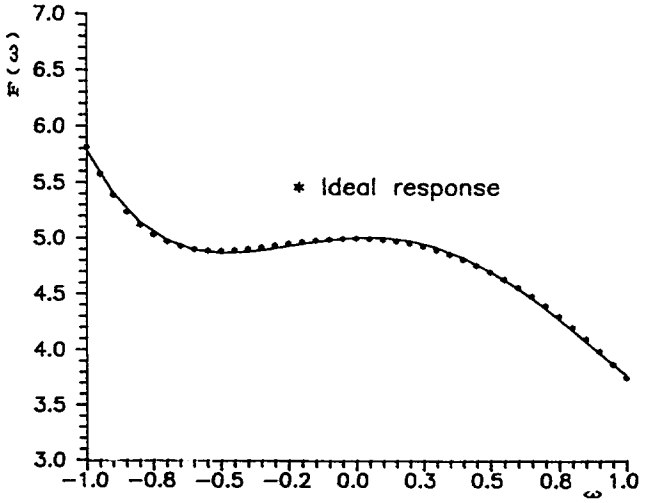

Figure 2. Approximation of $G(\omega) = 2e^{-\omega}\sin(\omega) + e^{-2\omega} - x^2 + 4$

Example 2

In this example the program is used to design a low-pass one-dimensional IIR digital filter in the time domain.[10] For this type of filter the transfer function has the following general form

$$H(z) = \frac{\sum_{i=0}^{N} a_i z^{-i}}{1 + \sum_{i=1}^{M} b_i z^{-i}} \tag{35}$$

where $z = \exp(j\omega)$ and $M \geqslant N$. The a_i, b_i are the unknown coefficients of the filter. For low-pass filters, the ideal magnitude frequency response is given by the relation:

$$|H_d(z)| = \begin{cases} 1 & \text{for} \quad 0 \leqslant \omega_k \leqslant \pi/4 \\ 0.5, & \text{for} \quad \omega_k = \pi/4 \\ 0, & \text{for} \quad \pi/4 \leqslant \omega_k \leqslant \pi \end{cases} \tag{36}$$

where $\omega_k = k\pi/\mathrm{K}$, with K the total number of sampling points. According to $|H_d(z)|$, the ideal impulse response is given by

$$h_k = \begin{cases} 0.25 \dfrac{\sin((k-2)\pi/4)}{(k-2)\pi/4}, & \text{if } k \neq 2 \\ 0.25, & \text{if } k = 2 \end{cases} \tag{37}$$

According to the above, the design problem is formulated as an LMAP of the following form:[10]

minimize g

subject to

$$\left| \sum_{n=1}^{M} b_n h_{i-n} - a_i' \right| \leqslant g \qquad \forall\, i = 1, 2, \ldots, K \tag{38}$$

where $a_0 = h(0)$ and'

$$a_i' = \begin{cases} a_i & \text{for } i = 1, \ldots, N \\ 0 & \text{for } i = N+1, \ldots, K \end{cases} \tag{39}$$

The design problem was solved for $N = 4$, $M = 5$ and $K = 50$ sampling points. The optimal initial point is determined equal to

$x_1 = -3.485409263$ $\quad$ $x_6 = -0.329641034$
$x_2 = 5.431850009$ $\quad$ $x_7 = 0.330013072$
$x_3 = -4.591292385$ $\quad$ $x_8 = -0.154404317$
$x_4 = 2.087293910$ $\quad$ $x_9 = 0.031224020$
$x_5 = -0.406800823$ and $g = 0.000172027$

The final print-out of the program is as follows:

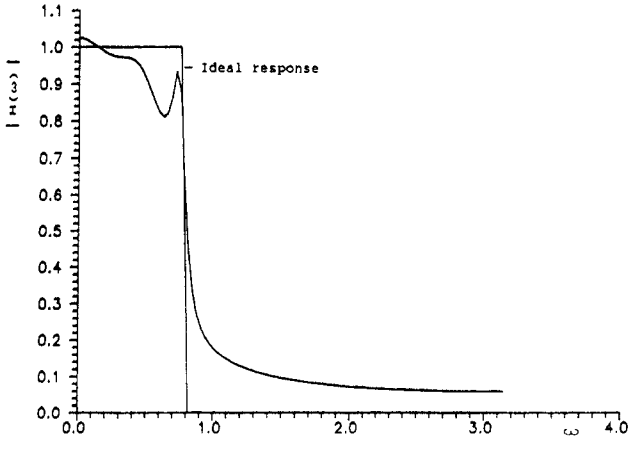

Figure 3. Magnitude response of the digital filter in Example 2

List of the Input Data

Number of variables = 9
Number of constraints = 50
Optimum initial point = Y
Shift-value = 10
Data file = A2.DAT

Final Results:

$g = 0.00011788945$ Computation time = 22.13

$x(1) = -3.48651468712$
$x(2) = 5.42199771541$
$x(3) = -4.56474400616$
$x(4) = 2.06202847716$
$x(5) = 0.39787733745$
$x(6) = -0.32993485697$
$x(7) = 0.32807833346$
$x(8) = -0.15279080233$
$x(9) = 0.03054861562$

Figure 3 shows the magnitude frequency response of the approximated filter and the ideal response according to equation (36)

CONCLUSIONS

This paper presents a Turbo-basic program which implements a modified form of a new algorithm for the optimum solution of the LMAP. This algorithm specifies feasible direction of decreasing values of the objective function, resulting always to the global minimum. In comparison with the solution of the LMAP with the revised Simplex algorithm, the new algorithm is much faster and has lower storage requirements.

The examples illustrated have shown that the proposed program is easy to use and is suitable to solve fast and optimum large dimensionality LMAP.

REFERENCES

1 Charalambous, C. The performance of an algorithm for minimax design of two-dimensional linear phase FIR digital filters, *IEEE Trans. Circuits Syst.*, 1985, CAS-32, 1016–1028
2 McClellan, J. H., Parks, T. W. and Rabiner, L. R. A computer program for designing optimum FIR linear phase digital filters, *IEEE Trans. Audio Electroacoustics*, 1973, AU-21, 506-625
3 Chottera, A, and Jullien, G. A. Design of two-dimensional recursive digital filters using linear programming, *IEEE Trans. Circuits Syst.*, 1982, CAS-29, 817–826
4 Lee, D. Some computational aspects of low-level computer vision, *Proceedings of the IEEE*, 1988, **76**, 890–898
5 Arthanari, T. S. and Dodge, Y. Mathematical programming in statistics, John Wiley and Sons, 1981
6 System/360 Scientific subroutine package, IBM Application Program, Version III, Programmer's Manual, prog. number 360A-CM-03X, 5th ed., 1970
7 Papademetriou, C. and Steiglitz, K. Combinatorial optimization: Algorithms and Complexity, Prentice-Hall, 1982
8 Papamarkos, N. and Vachtsevanos, G. A New algorithm for the solution of the linear minimax approximation problem, *Optimization*, 1988, **19**, 839–860
9 Forsythe, G. and Moler, C. B. Computer solution of linear algebraic systems, Prentice-Hall, 1967
10 Papamarkos, N. Time domain design of 1-D IIR digital filters with coefficients of finite word length, to appear in Proc. of International AMSE Conf. Signal and Systems, 1989

APPENDIX A

List of program minimax.bas

```
  DEFDBL A-H,O-Z
  DEFINT I-N
  CLS
PRINT"┌──────────────────────────────────────────────────────────────────────────┐"
PRINT"│                Linear minimax approximation                              │"
PRINT"│                   by   Nikos Papamarkos                                  │"
PRINT"│                                                                          │"
PRINT"│                                                                          │"
PRINT"│ Problem formulation:                                                     │"
PRINT"│                                                                          │"
PRINT"│   minimize g                                                             │"
PRINT"│                                                                          │"
PRINT"│ subject to                                                               │"
PRINT"│                                                                          │"
PRINT"│ |A(I,1)X(1)+A(I,2)X(2)+...+A(I,N)X(N)-C(I)| ≤ g    ,I=1,2,...,K        │"
PRINT"│                                                                          │"
PRINT"└──────────────────────────────────────────────────────────────────────────┘"
  LOCATE 22,5:PRINT" Press any key to continue "
  WHILE NOT INSTAT:WEND
  CLS
  DIM KASC(4)
  KASC(1)=121:KASC(2)=89:KASC(3)=78:KASC(4)=110
  INPUT ,AD$
WHILE AD$<>" "
LOCATE 3,3
PRINT " Input the following information for the linear minimax problem"
LOCATE 4,3
PRINT"──────────────────────────────────────────────────────────────────────────"
LOCATE 8,5:PRINT "                          Number of variables =       "
LOCATE 10,5:PRINT "                        Number of constraints =      "
LOCATE 12,5:PRINT"               Optimum initial point? [Y/N] =         "
LOCATE 14,5:PRINT "                                  Shift-value =      "
LOCATE 16,5:PRINT "                      Final results only? [Y/N] =    "
LOCATE 18,5:PRINT"Input coefficients from a data file? [Y/N] =          "
LOCATE 8,50:INPUT ,N:LOCATE 10,50:INPUT ,K
CALL EISAG(4,KASC(),ISTART$,12,50)
LOCATE 14,50:INPUT ,MET
CALL EISAG(4,KASC(),RES$,16,50)
CALL EISAG(4,KASC(),EIS$,18,50)
  IF EIS$="y" OR EIS$="Y" THEN
    LOCATE 20,25:PRINT " From which data file? =          "
    LOCATE 20,50:INPUT ,AM$
  END IF
  LOCATE 22,2
PRINT" Press [Esc] to changes the input data or any other key to continue"
  WHILE NOT INSTAT:WEND
  KK$=INKEY$
  IF KK$<>CHR$(27) THEN EXIT LOOP
  LOCATE 20,25:PRINT "                                                  "
WEND
  K2=2*K:TOL=0.000000001
  DIM A(K2,N),X(N+20),C(K2),XS(K2),XSL(K2),XQ(N),XL(N),XQN(N)
  DIM KQ12(K2),NK(K2),XS2(K2),BN(N,N+1)
  CLS
```

```
  IF EIS$="Y" OR EIS$="y" THEN
    OPEN AM$ FOR INPUT AS #1
    IJ=0
    FOR I=1 TO K:S=0
      FOR J=1 TO N
        INPUT #1,A(I,J)
        S=S+A(I,J):A(I+K,J)=-A(I,J)
      NEXT J
      INPUT #1,CI
      C(I)=CI+MET*S:C(I+K)=-C(I)
    NEXT I
  ELSE
    FOR I=1 TO K:S=0
      FOR J=1 TO N
        PRINT USING"A(## ##)=";I;J;:INPUT ,A(I,J)
        S=S+A(I,J):A(I+K,J)=-A(I,J)
      NEXT J
      PRINT USING"C(##)=";I;:INPUT ,CI
      C(I)=CI+MET*S:C(I+K)=-C(I)
    NEXT I
  END IF
  FOR I=1 TO K
    IF C(I)<0 THEN
      FOR J=1 TO N
        A(I,J)=-A(I,J):A(I+K,J)=-A(I+K,J)
      NEXT J
      C(I)=-C(I):C(I+K)=-C(I+K)
    END IF
  NEXT I
  XR2=0:XR1=TIMER
  IF ISTART$="Y" OR ISTART$="y" THEN
    T$=" OPTIMUM INITIAL STARTING POINT"
    CALL INIT(N,K,K2)
  ELSE
    T$=" RANDOM INITIAL STARTING POINT"
    CLS:LOCATE 5,15:PRINT "INPUT THE INITIAL STARTING POINT"
    FOR I=1 TO N
      PRINT USING" X(##)= ";I;:INPUT X(I)
      X(I)=X(I)+MET
    NEXT I
    XR1=TIMER
  END IF
  CLS
  IF RES$="Y" OR RES$="y" THEN
    LOCATE 14,20:PRINT" PLEASE WAIT"
  END IF
  CALL TOMES(N,K,X(),XS(),XSS,L)
  CALL MEGXS(N,K,XS(),XSS,KQ,KPOL,KQ12())
  IF RES$="N" OR RES$="n" THEN
    LOCATE 3,24:PRINT"------------------------------------------------------------"
    CALL SCR(T$,N,K,X(),XS(),XSS,KQ12(),KPOL)
  END IF
10  IF KPOL=1 THEN
      KQ=KQ12(1):T$="  T=1   CASE"
      CALL Q1(N,K,K2,KPOL,KQ,XSS)
      IF RES$="N" OR RES$="n" THEN CALL SCR(T$,N,K,X(),XS(),XSS,KQ12(),KPOL)
    END IF
20  IF KPOL=2 THEN
    CQ1=C(KQ12(1)):CQ2=C(KQ12(2))
      CALL ELAX(ABS(CQ1),ABS(CQ2),CMIN,KMIN)
```

```
    IF CQ1*CQ2>=0. AND CMIN>XSS THEN
      KQ=KQ12(KMIN):T$="  T=2.1   CASE"
      CALL Q1(N,K,K2,KPOL,KQ,XSS)
      IF RES$="N" OR RES$="n" THEN
        CALL SCR(T$,N,K,X(),XS(),XSS,KQ12(),KPOL)
      END IF
      GOTO 20
    ELSE
      CALL Q2(N,K,K2,KPOL,XSS)
      T$="  T=2.2   CASE"
      IF RES$="N" OR RES$="n" THEN
        CALL SCR(T$,N,K,X(),XS(),XSS,KQ12(),KPOL)
END IF:END IF:END IF
  IF KPOL>=3 AND KPOL<=N THEN
    IM3$=TIME$
    CALL Q3(N,KPOL,XSS,X(),XQN(),XSQN)
    CALL TOMES(N,K,XQN(),XS2(),XSQN,L)
    XSQN=XS2(KQ12(1))
    CALL Q1Q2L(N,X(),XQN(),XS(),XS2(),XSS,XSQN,KPOL,KQ12(),IR)
    CALL MEGXS(N,K,XS(),XSS,KQ,KPOL,KQ12())
    T$="  T=3   CASE"
    IF RES$="N" OR RES$="n" THEN
      CALL SCR(T$,N,K,X(),XS(),XSS,KQ12(),KPOL)
  END IF:END IF
  IF KPOL>N THEN
    IR=1
    KK=FNPAR(KPOL)/(FNPAR(N)*FNPAR(KPOL-N))
    NK(1)=0:KKM=0
    FOR I=2 TO KPOL:NK(I)=I-1:NEXT I
    FOR I=1 TO KPOL
      IF C(KQ12(I))<0 THEN
        IF KKM=0 THEN
          KMUST=I:KKM=1
        ELSE
          IF C(KQ12(I))>C(KMUST) THEN KMUST=I
      END IF:END IF
    NEXT I
    FOR I=1 TO KK
      CALL KQN(NK(),N,KPOL,KMUST,KIER)
      IF KIER=1 THEN
        CALL Q4(N,KPOL,XSS,X(),XQN(),XSQN,SIM)
        IF SIM<>0 THEN
          CALL TOMES(N,K,X(),XS(),XSS,L1)
          CALL TOMES(N,K,XQN(),XS2(),XSQN,L)
          XSQN=XS2(KQ12(NK(1)))
          CALL Q1Q2L(N,X(),XQN(),XS(),XS2(),XSS,XSQN,KPOL,KQ12(),IR)
          IF IR=0 THEN
            CALL MEGXS(N,K,XS(),XSS,KQ,KPOL,KQ12())
            T$="  T=4   CASE"
            IF RES$="N" OR RES$="n" THEN
              CALL SCR(T$,N,K,X(),XS(),XSS,KQ12(),KPOL)
            END IF
            EXIT FOR
          END IF
        END IF
        IF I=KK AND IR=1 THEN GOTO 200
      END IF
    NEXT I
  END IF
  GOTO 10
```

```
200 XR2=TIMER-XR1-DLAY
  CLS
  KASC(1)=49:KASC(2)=50:KASC(3)=51:KASC(4)=52
  KK=0
  DO UNTIL KK=4
    CLS:LOCATE 10,15:PRINT " OPTIMAL SOLUTION"
    LOCATE 12,15:PRINT "1 : Display the Input Data"
    LOCATE 14,15:PRINT "2 : Display the Final Optimal Results"
    LOCATE 16,15
    PRINT "3 : Print the Input Data and the Final Optimal Results"
    LOCATE 18,15:PRINT "4 : Terminate the Program"
    CALL EISAG(4,KASC(),KK$,20,20)
    KK=VAL(KK$):CLS
    IF KK=1 THEN
      CALL INDATA(N,K,MET,AM$,1)
      CALL PRES
    ELSEIF KK=2 THEN
      CALL SCR(T$,N,K,X(),XS(),XSS,KQ12(),KPOL)
      CALL PRES
    ELSEIF KK=3 THEN
      CALL INDATA(N,K,MET,AM$,2)
      CALL PRES
    END IF
  LOOP
  END
' ------------------------------------------------------------
  SUB PRES
  WHILE NOT INSTAT
    LOCATE 22,2
    PRINT"Press any key to return to the OPTIMAL SOLUTION menu"
  WEND
  END SUB
' ------------------------------------------------------------
  SUB INDATA(N,K,MET,AM$,IPR)
  LOCAL I
  SHARED X(),XSS,XR2,ISTART$
  IF IPR=1 THEN
LOCATE 2,5:PRINT "                  List of Input Data"
LOCATE 3,5:PRINT " ------------------------------------------------------"
LOCATE 6,5:PRINT "                     Number of variables =";N
LOCATE 8,5:PRINT "                   Number of constraints =";K
LOCATE 10,5:PRINT "                    Optimum initial point = ";ISTART$
LOCATE 12,5:PRINT "                              Shift-value =";MET
LOCATE 14,5:PRINT "                               Data  file = ";AM$
  ELSE
LPRINT TAB(5):LPRINT "                  List of Input Data"
LPRINT TAB(5):LPRINT " ------------------------------------------------------"
LPRINT TAB(5):LPRINT "                     Number of variables =";N
LPRINT TAB(5):LPRINT "                   Number of constraints =";K
LPRINT TAB(5):LPRINT "                   Optimum initial point = ";ISTART$
LPRINT TAB(5):LPRINT "                             Shift-value =";MET
LPRINT TAB(5):LPRINT "                              Data  file = ";AM$
LPRINT:LPRINT
LPRINT TAB(5):LPRINT  "                    Final Results:"
LPRINT TAB(5):LPRINT  " ------------------------------------------------------"
LPRINT TAB(5)
LPRINT USING " g = #######.###########  Computation time=####.##";XSS;XR2
LPRINT TAB(5):LPRINT  " ------------------------------------------------------"
  FOR I=1 TO N
    LPRINT TAB(5):LPRINT USING" x(##) = #######.###########";I;X(I)-MET
```

```
  NEXT I
LPRINT TAB(5):LPRINT  " ──────────────────────────────────────────────────────────────────────────────"
  END IF
  END SUB
' ──────────────────────────────────────────────────────────────
  SUB  EISAG(N,KASC(1),Y$,K1,K2)
  LOCAL I,II
  II=0
  DO UNTIL II=1
    LOCATE K1,K2:PRINT"    ":LOCATE K1,K2:INPUT ,Y$
    FOR I=1 TO N
        IF Y$=CHR$(KASC(I)) THEN II=1
    NEXT I
  LOOP
  END SUB
' ──────────────────────────────────────────────────────────────
  SUB KQN(NK(1),N,K,KMUST,KIER)
  LOCAL K1,K2,K3,I,J,KKK
    KIER=0:K1=NK(1):K2=NK(N)
  IF K1<K THEN
    FOR I=1 TO N
      NK(I)=NK(I)+1
            IF NK(I)>K THEN NK(I)=NK(I)-K
    NEXT I
  ELSEIF K1=K THEN
    K3=NK(2):NK(1)=1:NK(2)=2+K3
    FOR I=3 TO N
      NK(I)=NK(I-1)+1
            IF NK(I)>K THEN NK(I)=NK(I)-K
    NEXT I
  END IF
  KIER=0
  FOR I=1 TO K
    IF KMUST=NK(I) THEN
      KIER=1:EXIT SUB
    END IF
  NEXT I
  END SUB
' ──────────────────────────────────────────────────────────────
  DEF FNPAR(N)
'   Determines the N!
  LOCAL AK,I
  IF N=0 OR N=1 THEN
      FNPAR=1:EXIT DEF
  END IF
  AK=1
  FOR I=1 TO N:AK=AK*I:NEXT I
  FNPAR=AK
  END DEF
' ──────────────────────────────────────────────────────────────
  SUB INIT(N,K,K2)
' Determines the optimum initial point
  DIM H(N,N),G(N),U(N,N),UT(N,N)
  SHARED A(),C(),X()
  LOCAL I,J,KK
  FOR J=1 TO N:FOR I=1 TO N:FOR KK=1 TO K
     H(I,J)=H(I,J)+A(KK,J)*A(KK,I)
     IF J=1 THEN G(I)=G(I)+C(KK)*A(KK,I)
  NEXT KK:NEXT I:NEXT J
  FOR I=1 TO N
```

```
    S=0.
    FOR KK=1 TO I-1:S=S+U(I,KK)*U(I,KK):NEXT KK
    U(I,I)=SQR(H(I,I)-S):UT(I,I)=1./U(I,I)
    FOR J=1 TO N
      IF I<J THEN
        S=0.
        IF I>=2 THEN
                    FOR KK=1 TO I-1:S=S+U(I,KK)*U(J,KK):NEXT KK
        END IF
        U(J,I)=(H(I,J)-S)/U(I,I)
      END IF
  NEXT J:NEXT I
  FOR I=2 TO N:FOR J=1 TO N-1
    IF I>J THEN
      S=0.
      FOR KK=J TO I-1:S=S+U(I,KK)*UT(KK,J):NEXT KK
      UT(I,J)=-S/U(I,I)
    END IF
  NEXT J:NEXT I
  FOR I=1 TO N:FOR J=1 TO N
      IF I>=J THEN
      S=0.
      FOR KK=I TO N:S=S+UT(KK,I)*UT(KK,J):NEXT KK
      H(I,J)=S
      IF I<>J THEN H(J,I)=H(I,J)
    END IF
  NEXT J:NEXT I
  FOR I=1 TO N
    S=0.
    FOR J=1 TO N:S=S+H(I,J)*G(J):NEXT J
    X(I)=S
  NEXT I
  END SUB
' ------------------------------------------------------------
  SUB MEGXS(N,K,XS(1),XSS,KQ,KPOL,KQ12(1))
'       For known intersections determines the value of g and the
' vector of the basic hyperplanes
  LOCAL I
  TOL=1.E-9:XSS=ABS(XS(1)):KPOL=0
  FOR I=1 TO K
    IF ABS(ABS(XS(I))-XSS)<=TOL THEN
      KPOL=KPOL+1
      IF XS(I)>=0. THEN KQ12(KPOL)=I ELSE KQ12(KPOL)=I+K
    END IF
    IF ABS(XS(I))-XSS>TOL THEN
        KPOL=1
      IF XS(I)>=0 THEN KQ12(1)=I ELSE KQ12(1)=I+K
      XSS=ABS(XS(i))
      IF XS(I)>=0. THEN KQ=I ELSE KQ=I+K
    END IF
  NEXT I
  END SUB
' ------------------------------------------------------------
  SUB TOMES(N,K,X(1),XS(1),XSS,L)
' If the variable vector X() is known then this subroutine
' determines the intersections and the value of g
  SHARED A(),C()
  LOCAL I,J,S
  XSS=0
  FOR I=1 TO K
```

```
    S=0
    FOR J=1 TO N:S=S+A(I,J)*X(J):NEXT J
    XS(I)=S-C(I):XS(I+K)=-XS(I)
    IF XSS<ABS(XS(I)) THEN
      XSS=ABS(XS(I)):L=I
      IF XS(I)<0 THEN L=I+K
    END IF
  NEXT I
  END SUB
' ------------------------------------------------------------
  SUB ELAX(X1,X2,XMIN,L)
  IF X1<X2 THEN
    XMIN=X1:L=1
  ELSEIF X1=X2 THEN
     XMIN=X1:L=0
  ELSE
    XMIN=X2:L=2
  END IF
  END SUB
' ------------------------------------------------------------
  SUB Q1(N,K,K2,KPOL,KQ,XSS)
' This subroutine implements the Case 1 of the algorithm
  SHARED A(),C(),KQ12(),XS(),XSL(),X()
  LOCAL I,J,L,JJ,NQ1,NQ2,XSL
  NQ1=KQ12(1):NQ2=KQ12(2)
  IF NQ1<=K THEN
    NQ3=NQ1+K:NQ4=NQ2+K
  ELSE
    NQ3=NQ1-K:NQ4=NQ2-K
  END IF
  IF KPOL=1 THEN
    NQ2=0:NQ4=0
  END IF
  FOR L=1 TO K2
    IF L<>NQ1 AND L<>NQ2 AND L<>NQ3 AND L<>NQ4 THEN
      IF L<=K THEN LK=L+K ELSE LK=L-K
      V1=(C(L)-C(KQ)):V2=(XS(L)+C(L)-XSS-C(KQ))
      IF ABS(V2)>=1.E-10 THEN
                V=V1/V2:XSSL=V*(XS(L)+C(L))-C(L)
              IF XSSL>=0 AND ABS(XSSL)<XSS THEN
                    FOR I=1 TO K
                        IF I<>L AND L<>LK THEN
                            XSL(I)=V*(XS(I)+C(I))-C(I)
                            IF ABS(XSL(I))-ABS(XSSL)>1.E-9 THEN GOTO 2
              XSL(I+K)=-XSL(I)
                        END IF
                    NEXT I
                    XSL(L)=XSSL:XSL(LK)=-XSSL
              FOR J=1 TO N:X(J)=V*X(J):NEXT J
          EXIT FOR
    END IF:END IF:END IF
2   NEXT L
  CALL MEGXS(N,K,XSL(),XSSL,KL,KPOLL,KQ12())
  FOR I=1 TO K:XS(I)=XSL(I):XS(I+K)=-XSL(I):NEXT I
  KPOL=KPOLL:XSS=XSSL
  END SUB
' ------------------------------------------------------------
  SUB Q2(N,K,K2,KPOL,XSS)
' This subroutine implements the Case 2.2 of the algorithm
  SHARED A(),C(),KQ12(),XS(),XSL(),XL(),X()
```

```
  LOCAL I,J,L,JJ,V,XA,XSA,XAA
  FOR J=1 TO N
    NQ1=KQ12(1):NQ2=KQ12(2)
    IF NQ1<=K THEN
        NQ3=NQ1+K:NQ4=NQ2-K
    ELSE
        NQ3=NQ1-K:NQ4=NQ2+K
    END IF
    CQ1=C(NQ1):CQ2=C(NQ2):XAA=A(NQ1,J)-A(NQ2,J)
    IF ABS(XAA)>1.E-9 THEN
      XA=(CQ1-CQ2)/XAA:XSA=(CQ1*A(NQ2,J)-CQ2*A(NQ1,J))/XAA
      FOR L=1 TO K2
            IF L<>NQ1 AND L<>NQ2 AND L<>NQ3 AND L<>NQ4 THEN
          V=(XSA+C(L)-A(L,J)*XA)/(XSA+C(L)-A(L,J)*XA+XS(L)-XSS)
          XSSL=XSA+V*(XSS-XSA)
          IF XSSL>=0 AND XSSL<XSS THEN
            FOR JJ=1 TO K
              XSL(JJ)=V*(XS(JJ)+C(JJ)-A(JJ,J)*XA)+A(JJ,J)*XA-C(JJ)
              XSL(JJ+K)=-XSL(JJ)
              IF ABS(XSL(JJ))<=1.E-9 THEN
                XSL(JJ)=0.:XSL(JJ+K)=0.
                END IF
              IF ABS(XSL(JJ))-XSSL>1.E-9 THEN GOTO 12
                  NEXT JJ
            FOR I=1 TO N
              IF I<>J THEN XL(I)=V*X(I) ELSE XL(J)=XA+V*(X(J)-XA)
            NEXT I
            CALL MEGXS(N,K,XSL(),XSSL,KL,KPOLL,KQ12())
                IF XSSL<XSS THEN GOTO 5
            END IF:END IF
12        NEXT L
      END IF
  NEXT J
5 FOR I=1 TO N:X(I)=XL(I):NEXT I
  FOR I=1 TO K:XS(I)=XSL(I):XS(I+K)=-XSL(I):NEXT I
  KPOL=KPOLL:XSS=XSSL
  END SUB
' ------------------------------------------------------------
  SUB SCR(T$,N,K,X(1),XS(1),XSS,KQ12(1),KPOL)
  SHARED MET,DLAY,XR1,XR2
  LOCAL AK$,KK,I,II,XRTEL
  XR=TIMER:CLS
  IF XR2=0 THEN XRTL=XR-XR1-DLAY ELSE XRTL=XR2
  LOCATE  4,50:PRINT USING"Computing time=####.### sec";XRTL
  II=0
  LOCATE 2,25:PRINT T$:LOCATE 3,24:PRINT"--------------------"
  FOR I=1 TO N
    II=I MOD 12
    IF II=0 THEN
      LOCATE 18,3
      PRINT USING"  x(##) = ########.#########";I;X(I)-MET
      KSK$=INKEY$:LOCATE 21,5:PRINT:LOCATE 21,5
      PRINT" PRESS ENTER TO SEE THE NEXT VARIABLES":LOCATE 22,5
      PRINT" PRESS THE SPACE BAR TO SEE THE INTERSECTIONS"
      WHILE NOT INSTAT:WEND
55      AK$=INKEY$
      IF AK$=CHR$(32) THEN GOTO 65
      IF AK$<>CHR$(13) THEN GOTO 55
    ELSE
      LOCATE 6+II,3
```

```
      PRINT USING"  x(##) = ########.#########";I;X(I)-MET
    END IF
  NEXT I
  LOCATE 5,7:PRINT USING"  g = ########.######### ";XSS
  LOCATE 6,3:PRINT"  ------------------------------"
  LOCATE 17,5:PRINT"BASIC HYPERPLANES:"
  LOCATE 18,5:PRINT"--------------------":I2=1
  FOR I=1 TO KPOL
     I2=I2+4:LOCATE 19,I2+1:PRINT USING"##|";KQ12(I);
  NEXT I
  KSK$=INKEY$
  LOCATE 22,5:PRINT" PRESS THE SPACE BAR TO SEE THE INTERSECTIONS"
  LOCATE 23,5:PRINT" OR ANY OTHER KEY TO CONTINUE"
  WHILE NOT INSTAT:WEND:AK$=INKEY$
IF AK$=CHR$(32) THEN
65  LOCATE 6,50:PRINT "  INTERSECTIONS:"
  LOCATE 7,50:PRINT"----------------------------------"
  LOCATE 22,5:PRINT"                                                "
  LOCATE 23,5:PRINT"                                          "
    KK=K MOD 12:KK=K+12-KK
  FOR I=1 TO KK
    II=I MOD 12
    IF II=0 AND I<=K THEN
      LOCATE 19,50:PRINT USING"g(###)=#######.############";I;XS(I)
      KSK$=INKEY$:LOCATE 21,5:PRINT
      LOCATE 21,5:PRINT" PRESS ENTER TO SEE THE NEXT INTERSECTIONS"
      WHILE NOT INSTAT:WEND
75      AK$=INKEY$
      IF AK$<>CHR$(13) THEN GOTO 75
    ELSE
      IF I>K THEN
        IF II=0 THEN II=12
        LOCATE 7+II,50:PRINT "                                  "
      ELSE
        LOCATE 7+II,50
        PRINT USING"g(###)=#######.############";I;XS(I)
    END IF:END IF
  NEXT I
LOCATE 21,5:PRINT"                                                  "
LOCATE 23,5:PRINT" PRESS ANY KEY TO CONTINUE "
  WHILE NOT INSTAT:WEND
END IF
  DLAY=DLAY+TIMER-XR:CLS
  END SUB
' ------------------------------------------------------
  SUB Q3(N,KPOL,XSS,X(1),XQN(1),XSQN)
' Determines the second feasible point on the Case 3
' of the algorithm
  DIM CN(KPOL),KSTHLES(50),KMHSTL(50),XQN2(50)
  SHARED A(),C(),KQ12(),BN()
  LOCAL I,J,KT,S,EPS,KKPOL,JJ,JJ2
  EPS=1.E-10:R=1.2
  CALL MATMAT(N,KPOL,KSTHLES())
  JJ=0
  FOR J=1 TO N
    IF J<>KSTHLES(J-JJ) THEN
      JJ=JJ+1:KMHSTL(JJ)=J
    END IF
  NEXT J
  FOR I=1 TO KPOL
```

```
    KT=KQ12(I):S=0.
    FOR J=1 TO N-KPOL
      KAX=KMHSTL(J):XQN(KAX)=R*X(KAX)
      S=S+A(KT,KAX)*X(KAX)
    NEXT J
    CN(I)=C(KT)+R*XSS-R*S
  NEXT I
  FOR I=1 TO KPOL:BN(I,KPOL+1)=CN(I):NEXT I
  CALL SIMUL(KPOL,XQN2(),EPS,1,SIM)
  IF SIM=0 THEN
    PRINT "NO SOLUTION FOUND!!!":INPUT FSDF
  END IF
  FOR J=KPOL+1 TO N:XQN(KMHSTL(J))=1.2*X(KMHSTL(J)):NEXT J
  FOR J=1 TO KPOL:XQN(KSTHLES(J))=XQN2(J):NEXT J
  FOR I=1 TO KPOL
    L=KQ12(I):S=0
    FOR J=1 TO N:S=S+A(L,J)*XQN(J):NEXT J
    S=S-C(L)
  NEXT I
  XSQN=0:KQ=KQ12(1)
  FOR J=1 TO N:XSQN=XSQN+A(KQ,J)*XQN(J):NEXT J
  XSQN=XSQN-C(KQ)
  ERASE CN,AN,KMHSTL,KSTHLES
  END SUB
' ------------------------------------------------------------
  SUB Q1Q2L(N,X1(1),X2(1),XS1(1),XS2(1),XSS,XSS2,KPOL,KQ12(1),IR)
' This subroutine determines the intersection point of a line which
' is defined by the two known points {X1(),XSS} and {X2(),XSS2}
  SHARED A(),C(),XSL(),K,K2
  LOCAL I,J,L,V,XSSL
  FOR L=1 TO K2
    FOR J=1 TO KPOL
      IF L=KQ12(J) THEN GOTO 50
    NEXT J
    V=XSS-XSS2+XS2(L)-XS1(L)
    IF ABS(V)>1.E-9 THEN
      V=(XSS-XS1(L))/V
        XSSL=XSS+V*(XSS2-XSS)
      IF XSSL<XSS AND XSSL>=0 THEN
        IR=0
        FOR I=1 TO K
          XSL(I)=XS1(I)+V*(XS2(I)-XS1(I))
          IF ABS(XSL(I))-XSSL>1.E-10 THEN
            IR=1
            EXIT FOR
          END IF
          XSL(I+K)=-XSSL
        NEXT I
        IF IR=0 THEN EXIT FOR
    END IF:END IF
50  NEXT L
  IF IR=0 THEN
    FOR I=1 TO N:X1(I)=X1(I)+V*(X2(I)-X1(I)):NEXT I
    XSS=XSSL
    FOR I=1 TO K:XS1(I)=XSL(I):XS1(I+K)=-XSL(I):NEXT I
  END IF
  END SUB
' ------------------------------------------------------------
  SUB Q4(N,KPOL,XSS,X(1),XQN(1),XSQN,SIM)
' Basic subroutine for the Case 4 of the algorithm
```

```
  DIM CN(N)
  SHARED A(),C(),NK(),KQ12(),BN()
  LOCAL I,J,KT,S,EPS,AN
  EPS=1.E-10:R=1.2
  FOR I=1 TO N
    KT=KQ12(NK(I)):CN(I)=C(KT)+R*XSS
    FOR J=1 TO N:BN(I,J)=A(KT,J):NEXT J
  NEXT I
  FOR I=1 TO N:BN(I,N+1)=CN(I):NEXT I
  CALL SIMUL(N,XQN(),EPS,1,SIM)
  IF SIM<>0 THEN
    XSQN=0:KQ=KQ12(NK(1))
    FOR J=1 TO N:XSQN=XSQN+A(KQ,J)*XQN(J):NEXT J
    XSQN=XSQN-C(KQ)
  END IF
  ERASE CN,AN
  END SUB
' ------------------------------------------------------------
  SUB SIMUL(N,X(1),EPS,INDIC,SIM)
' This subroutine solves a system of linear equations
' by the Gauss-Jordan complete elimination method
  DIM IROW(50),JCOL(50),JORD(50),Y(50)
  SHARED BN()
  LOCAL IROW,JCOL,JORD,Y,I,J,K,MAX
        IF N>50 THEN
          SIM=0.:EXIT SUB
        END IF
  MAX=N
  IF INDIC>=0 THEN MAX=N+1
        DETER=1.
    FOR K=1 TO N
    KM1=K-1:PVOT=0.
    FOR I=1 TO N:FOR J=1 TO N
      IF K<>1 THEN
        FOR ISCAN=1 TO KM1:FOR JSCAN=1 TO KM1
          IF I=IROW(ISCAN) OR J=JCOL(JSCAN) THEN GOTO 240
        NEXT JSCAN:NEXT ISCAN
      END IF
      IF ABS(BN(I,J))<=ABS(PVOT) THEN GOTO 240
      PVOT =BN(I,J):IROW(K)=I:JCOL(K)=J
240   NEXT J:NEXT I
    IF ABS(PVOT)<=EPS THEN
      SIM=0.:EXIT SUB
    ELSE
          IROWK=IROW(K):JCOLK=JCOL(K)
    END IF
      DETER=DETER*PVOT
'       .....NORMALIZE PIVOT ROW ELEMENTS .....
    FOR J=1 TO MAX:BN(IROWK,J)=BN(IROWK,J)/PVOT:NEXT J
    BN(IROWK,JCOLK)=1./PVOT
      FOR I=1 TO N
        AIJCK=BN(I,JCOLK)
        IF I<>IROWK THEN
        BN(I,JCOLK)=-AIJCK/PVOT
        FOR J=1 TO MAX
          IF J<>JCOLK THEN BN(I,J)=BN(I,J)-AIJCK*BN(IROWK,J)
        NEXT J
      END IF
  NEXT I:NEXT K
  FOR I=1 TO N
```

```
  IROWI=IROW(I):JCOLI=JCOL(I):JORD(IROWI)=JCOLI
  IF INDIC>=0 THEN X(JCOLI)=BN(IROWI,MAX)
NEXT I
JNTCH=0.:NM1=N-1
FOR I=1 TO NM1:IP1=I+1
  FOR J=IP1 TO N
    IF JORD(J)<JORD(I) THEN
      JTEMP=JORD(J):JORD(J)=JORD(I):JORD(I)=JTEMP:JNTCH=JNTCH+1
    END IF
NEXT J:NEXT I
IF ((JNTCH/INT(2)*2.)<>JNTCH) THEN DETER=-DETER
IF INDIC>0 THEN
  SIM=DETER:EXIT SUB
      END IF
FOR J=1 TO N:FOR I=1 TO N
  IROWI=IROW(I):JCOLI=JCOL(I):Y(JCOLI)=BN(IROWI,I)
  NEXT I
  FOR I=1 TO N:BN(I,J)=Y(I):NEXT I
NEXT J
FOR I=1 TO N:FOR J=1 TO N
  IROWJ=IROW(J):JCOLJ=JCOL(J):Y(IROWJ)=BN(I,JCOLJ)
  NEXT J
  FOR J=1 TO N:BN(I,J)=Y(J):NEXT J
NEXT I
SIM=DETER
END SUB
```

```
SUB MATMAT(N,KPOL,KST(1))
DIM KGRAMH(KPOL)
SHARED A(),KQ12(),BN()
LOCAL I,J,JJ,KMAX,KLINE,KAPOR,KDIAF,KT,K1$
KMAX=0
FOR I=1 TO KPOL:FOR J=1 TO N
  IF ABS(A(I,J))>=1.E-10 THEN
    KGRAMH(I)=J
    IF KMAX<J THEN
      KMAX=J:KLINE=I
    END IF
    EXIT FOR
  END IF
NEXT J:NEXT I
JJ=0:KAPOR=0
FOR J=1 TO N
  K1$="OXI"
  FOR I=1 TO KPOL
    KT=KQ12(I)
    IF ABS(A(KT,J))>1.E-10 THEN
      K1$="NAI"
      EXIT FOR
    END IF
  NEXT I
  IF K1$="NAI" THEN
    JJ=JJ+1:KST(JJ)=J
    FOR I=1 TO KPOL:BN(I,JJ)=A(KQ12(I),J):NEXT I
  ELSE
    KAPOR=KAPOR+1
  END IF
  IF JJ=KPOL THEN EXIT FOR
NEXT J
KDIAF=KMAX-KPOL-KAPOR
```

```
IF KDIAF>=1 THEN
  JJ=0
  FOR J=KMAX TO N
    JJ=JJ+1
    IF JJ>KDIAF THEN EXIT FOR
    IF A(KQ12(KLINE),J)>1.E-10 THEN
      KST(KPOL+1-JJ)=J
      FOR I=1 TO KPOL:BN(I,KPOL+1-JJ)=A(KQ12(I),J):NEXT I
    END IF
  NEXT J
END IF
ERASE KGRAMH
END SUB
```