INTERNAL REPORTS IN

SIMULATION, OPTIMIZATION AND CONTROL

No. SOC-281

MMLA1Q - A FORTRAN PACKAGE FOR LINEARLY CONSTRAINED MINIMAX OPTIMIZATION

J. Hald

(Adapted and Edited by J.W. Bandler and W.M. Zuberek)

December 1981

FACULTY OF ENGINEERING
McMASTER UNIVERSITY
HAMILTON, ONTARIO, CANADA

No. SOC-281

MMLA1Q - A FORTRAN PACKAGE FOR LINEARLY CONSTRAINED MINIMAX OPTIMIZATION

J. Hald

(Adapted and Edited by J.W. Bandler and W.M. Zuberek)

December 1981

MMLA1Q - A FORTRAN PACKAGE FOR LINEARLY CONSTRAINED

MINIMAX OPTIMIZATION

J. Hald

Adapted and Edited by

J.W. Bandler and W.M. Zuberek

Abstract

This report provides a user-oriented description of a program package written in Fortran IV for linearly constrained minimax optimization. The new subroutine MMLA1Q is very similar to MINLA1, which was presented by Madsen and Schjaer-Jacobsen, the main difference being that MMLA1Q accumulates and uses approximate second-order information as described by Hald and Madsen. Both routines require first-order partial derivatives of the nonlinear functions defining the minimax problem. The list of parameters is described herein, and a listing of the complete program package including the linear programming part is given. Instead of the revised simplex algorithm used in MINLA1, a reduced gradient algorithm has been developed. Finally, a couple of simple examples illustrate the use of the program. The program and documentation have been adapted for the CDC 170/730 system.

The work of verifying, editing and adapting the original material due to J. Hald was supported by the Natural Sciences and Engineering Research Council of Canada under Grant G0647.

J. Hald is with the Institute of Numerical Analysis, Technical University of Denmark, Lyngby, Denmark.

J.W. Bandler and W.M. Zuberek are with the Group on Simulation, Optimization and Control, and the Department of Electrical and Computer Engineering, McMaster University, Hamilton, Canada L8S 4L7.

W.M. Zuberek is on leave from the Institute of Computer Science, Technical University of Warsaw, Warsaw, Poland.

I. INTRODUCTION

Prepared by J.W. Bandler and W.M. Zuberek

This report gives a user-oriented description of a program package for linearly constrained minimax optimization of a set of differentiable nonlinear functions. The package has been developed in Fortran IV by Jorgen Hald at the Institute for Numerical Analysis of the Technical University of Denmark in Lyngby* and has been adapted for the CDC 170/730 (System B) installation at McMaster University. Sections II-VII contain the body of Hald's report edited and arranged for use at McMaster University. Also given is the listing (Appendix) and tests of the package.

The package is available as a permanent group file in the form of a library of binary relocatable subroutines. The name of the library is LIBRMML. The package is linked with the user's program by the appropriate call of the main subroutine of the package, namely, subroutine MMLA1Q. The general sequence of NOS commands to use the package can be as follows:

```
/GET(LIBRMML/GR)    - fetch the library LIBRMML,
/LIBRARY(LIBRMML)   - indicate the library to the loader,
/FTN(...,GO)        - compile, load and execute the program.
```

---

* J. Hald, "MMLA1Q, a Fortran subroutine for linearly constrained minimax optimization", Inst. for Numerical Analysis, Technical Univ. of Denmark, Lyngby, Denmark, Report NI-81-01, April 1981.

The user must prepare programs which should be composed (at least) of:

- the main segment, which prepares parameters and calls the main subroutine of the package (subroutine MMLA1Q),
- the segment which calculates the values of the nonlinear functions and their first derivatives w.r.t. all variables at points determined by the package; the name of this subroutine can be arbitrary because it is transferred to the package as one of the parameters.

II. GENERAL DESCRIPTION

Given a set of nonlinear functions $f_i(\underset{\sim}{x})$, $i = 1,2,\dots,m$, of n variables, it is the purpose of the package to find a local minimum of the minimax objective function

$$F(\underset{\sim}{x}) = \max_{1 \leq i \leq m} f_i(\underset{\sim}{x}) \tag{1}$$

subject to the constraints

$$\underset{\sim}{c}_i^{'T} \underset{\sim}{x} + c_i = 0 , \quad i = 1,2,\dots,\ell_{eq}, \tag{2}$$

$$\underset{\sim}{c}_i^{'T} \underset{\sim}{x} + c_i \geq 0 , \quad i = \ell_{eq}+1,\dots,\ell,$$

where $\underset{\sim}{c}_i^{'}$ and c_i, $i = 1,2,\dots,\ell$, are constants.

The algorithm has two stages, as described by Hald and Madsen [1]. The stages are summarized as follows.

Stage 1 (MLA1QS)

At each point the nonlinear functions are approximated by linear functions using the first-derivative information. The linearized problem is solved as follows.

An updated steepest descent direction is followed until a minimum or a step size limit has been reached. These steepest descent directions are reduced gradients [2]. After each linear problem the step size limit is updated according to the goodness of the linear approximations, as described in [3]. Only first-order information is used during this stage. However, approximate second-order information is accumulated (the BFGS updating formula is employed for this purpose) to be used in case a switch to Stage 2 is made.

Stage 2 (S2LA1Q)

This algorithm is a modified quasi-Newton algorithm, attempting to solve the set of nonlinear equations which correspond to the Kuhn-Tucker equations in nonlinear programming [1],[3].

Comments

The iterative procedure is always started in Stage 1. The Stage 2 iteration is introduced in order to speed up the final rate of convergence for problems which are singular at the solution [1],[3]. For such problems the Stage 1 algorithm may give very slow convergence. A shift to Stage 2 is made, when certain criteria seem to indicate that a reasonably good approximation to the solution, $\underset{\sim}{x}^*$, has been obtained. However, too early a switch is not disastrous, since then a switch back will be performed. Several switches between the two stages are allowed.

The main criterion for switching to Stage 2 is based on the current "active set" of constraints and nonlinear functions. The active set in a Stage 1 step is defined as the set of constraints and functions, which are active at the end of the step. When the distance to the solution $\underset{\sim}{x}^*$ is sufficiently small, this active set will equal the set which determines the solution. This means that, if the iteration converges, the active set will remain constant after some point. Actually, if the active set has not changed for a user-specified number of successive steps, and if a few more conditions are satisfied, then a switch to Stage 2 is made.

Convergence theorems of the Stage 1 algorithm have been proved by Hald [4], but are identical to those of the algorithm of Madsen and Schjaer-Jacobsen [3], [5]. The convergence properties of a two stage algorithm for unconstrained optimization have been investigated in [1]. As the algorithm described here is equivalent, the convergence properties are the same. Normally the final rate of convergence is either quadratic for regular solutions or superlinear for singular solutions [4].

The package does not require a feasible starting point, i.e., a point satisfying the constraints. Initially, a feasible point is found (in the subroutine FEASI) and after this point feasibility is retained. This means that the nonlinear functions are only evaluated at feasible points.

The package is applicable also to unconstrained minimax problems.

III. STRUCTURE OF THE PACKAGE

A block diagram of the package is shown in Fig. 1. The subroutine MMLA1Q is the main subroutine, the aim of which is only to simplify the use of the package. Check of input parameters and subdivision of the work space (provided by the user) is done in MMLA1Q. The Stage 1 algorithm is implemented in MLA1QS and the Stage 2 algorithm in S2LA1Q. FEASI finds a feasible starting point, and the linear subproblems of Stage 1 are solved by MMLPA. Both MMLPA and FEASI use the subroutine package REGRAD for projected gradient calculations. The subroutine BFGS is an implementation of the BFGS formula for updating an approximate Hessian matrix, containing second-order information. LINSYS uses Gaussian elimination for solving a system of linear equations.

The main program MAIN and the subroutine FDF for evaluation of functions and derivatives must be supplied by the user.

IV. LIST OF ARGUMENTS

The main subroutine call is

```
CALL MMLA1Q (FDF,N,M,L,LEQ,C,DC,IC,X,DX,EPS,MAXF,KEQS,W,IW,IFALL)
```

The arguments of this call statement are defined as follows.

FDF is the name of a subroutine written by the user. It must have the form

```
SUBROUTINE FDF(N,M,X,DF,F)
REAL X(N), DF(M,N), F(M)
```

and must calculate the values of the nonlinear functions $f_i(\underset{\sim}{x})$

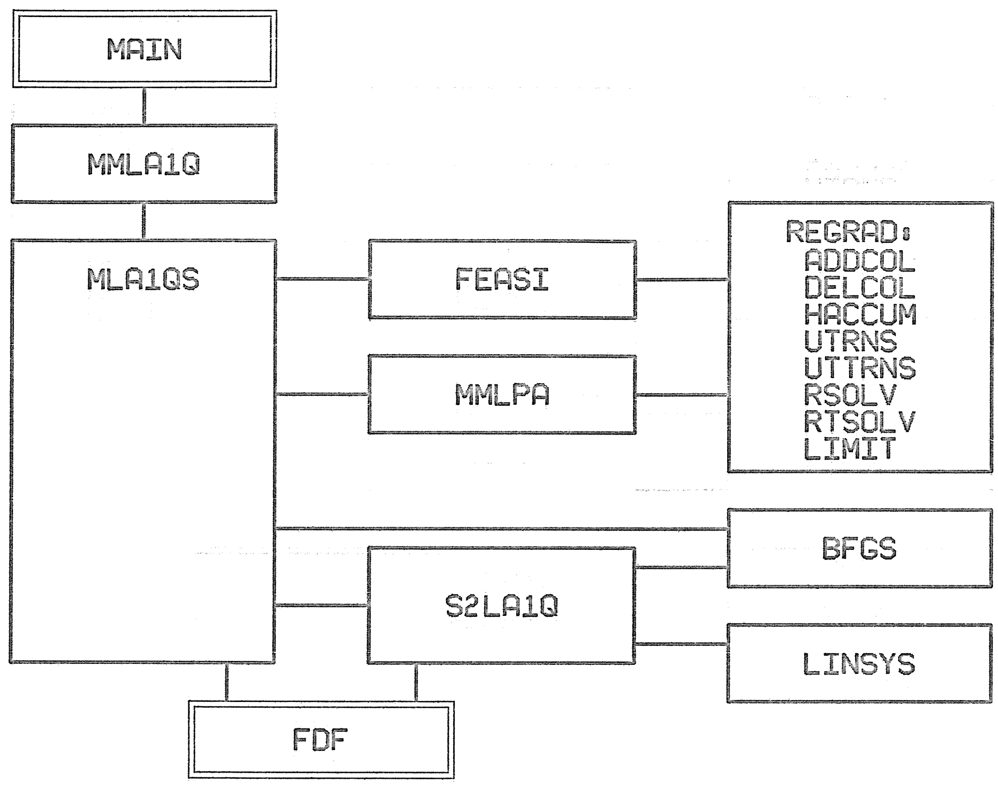

Fig. 1 Structure of the MMLA1Q package for nonlinear minimax optimization subject to linear equality and inequality constraints.

and their derivatives at the point $\underset{\sim}{x}$ corresponding to X(1),X(2),...,X(N), and store these in the following way:

$$F(I) = f_I(\underset{\sim}{x}), \qquad I = 1,\ldots,M,$$

$$DF(I,J) = \partial f_I/\partial x_J(\underset{\sim}{x}), \quad I = 1,\ldots,M,\ J = 1,\ldots,N.$$

Note: The name of this user-supplied subroutine, which can be any name of the user's choice, must appear in an EXTERNAL statement in the calling program.

N is an INTEGER variable and must be set to n, the number of optimization parameters. Its value must be positive, and it is not changed by the package.

M is the INTEGER variable and must be set to m, the number of nonlinear functions defining the minimax objective function. Its value must be positive, and it is not changed by the package.

L is an INTEGER variable and must be set to ℓ, the total number of linear equality and inequality constraints. Its value must be positive or zero, and it is not changed by the package.

LEQ is an INTEGER variable and must be set to ℓ_{eq}, the number of equality constraints. Its value must be positive or zero and less than N and less than or equal to L. Its value is not changed by the package.

C is a REAL array of length IC $\geq$ L. Its elements must be set to the constant terms in the linear constraints (2), i.e., C(I)=c_I, I=1,...,L. The content of C is not changed by the package.

DC is a REAL two-dimensional matrix of dimensions (IC,N). Its first L rows must be set to the coefficients of $\underset{\sim}{x}$ in the linear constraints (2), i.e.,

$$(DC(I,1),DC(I,2),\ldots,DC(I,N)) = \underset{\sim}{c}'^{T}_{I} , \ I = 1,\ldots,L.$$

IC is an INTEGER variable which must be set to the number of rows of the array DC. Its value must be equal to or greater than L, and it is not changed by the package.

X is a REAL array of length at least N and must be initialized to an approximation of the solution, $X(I) = x_I^0$, I=1,...,N. <u>On exit</u> X will contain the best feasible solution found by the package.

DX is a REAL variable which controls the step length of the iterative methods used. If the functions f_i are nearly linear, DX should be set to an approximate value of the distance between the initial approximation $\underset{\sim}{x}^0$ and the solution. But if more curvature is present, this value may be too large. Normally DX $= 0.1*\|\underset{\sim}{x}^0\|$ is a reasonable choice, but this is not critical, since the value of DX is adjusted by the package during the iteration. The value of DX must be positive.

EPS is a REAL non-negative variable which specifies the required accuracy of the solution. The iteration is stopped when $\|\underset{\sim}{h}_k\| \leq$ EPS$*\|\underset{\sim}{x}_k\|$, where $\underset{\sim}{h}_k$ is the change given by the algorithm to the kth approximation $\underset{\sim}{x}_k$. If EPS is chosen too small, the package will return with IFALL = 2, when no better estimate to the solution can be obtained because of rounding errors. Therefore EPS = 0 is an acceptable input value. <u>On exit</u> EPS contains the length of the last step taken in the iteration.

MAXF is an INTEGER variable which must be set to an upper bound on the number of calls of FDF. <u>On exit</u> MAXF contains the number of times FDF has been called.

KEQS is an INTEGER variable which must be set to the number of successive approximations to the solution that must have identical active sets in order that a switch to Stage 2 will be allowed. Normally KEQS = 3 will give reasonably early shifts. <u>On exit</u> KEQS contains the number of times a switch to Stage 2 has taken place.

W is a REAL array which is used for working space. Its length is IW.

IW is an INTEGER variable which must be set to the length of W. It must be at least equal to

IWR = 2*M*N + 5*N*N + 4*M + 8*N + 4*IC + 3.

IFALL is an INTEGER variable which on exit contains information about the solution that has been found by the package:

IFALL = -1 Erroneous input data.

IFALL = 0 Regular solution. Required accuracy obtained.

IFALL = 1 Singular solution. Required accuracy obtained.

IFALL = 2 Machine accuracy limit reached or maximum number of function evaluations reached. The best approximation to the solution is returned.

IFALL = 3 The feasible region is empty.

V. GENERAL INFORMATION

Use of COMMON: None.

Workspace: Provided by the user; see arguments W and IW.

Other subroutines: MLA1QS, S2LA1Q, FEASI, MMLPA, LINSYS, BFGS, ADDCOL, DELCOL, UTTRNS, UTRNS, RSOLV, RTSOLV, HACCUM, LIMIT.

Input/Output: None.

Restrictions: $IW \geq IWR$, $N \geq 1$, $M \geq 1$, $L \geq 0$, $LEQ \geq 0$, $LEQ \leq L$, $LEQ \leq N$, $IC \geq L$, $DX > 0$, $EPS \geq 0$, $MAXF > 0$.

Date: March 1981.

VI. EXAMPLES

Example 1 [3, Example 2]:

Minimize

$$F(\underset{\sim}{x}) = \max_{1 \leq i \leq 3} f_i(\underset{\sim}{x})$$

subject to

$$- 3x_1 - x_2 - 2.5 \geq 0,$$

where

$$f_1(\underset{\sim}{x}) = x_1^2 + x_2^2 + x_1 x_2 - 1,$$

$$f_2(\underset{\sim}{x}) = \sin(x_1),$$

$$f_3(\underset{\sim}{x}) = -\cos(x_2).$$

For this example,

$$N = 2$$

$$M = 3$$

$$LEQ = 0$$

$$L = 1$$

$$IC = 1$$

$$IW = 67$$

The starting point is

$$\underset{\sim}{x}^0 = \begin{bmatrix} -2 \\ -1 \end{bmatrix}.$$

The function has a minimum at

$$\underset{\sim}{x} = \begin{bmatrix} -25/28 \\ 5/28 \end{bmatrix}.$$

The extremely fast final convergence is due to the fact that the solution is determined by the linear constraint and the quadratic function. The problem is singular at the solution, because the number of constraints and functions being active at the solution is less than n+1. This means that the Haar condition is not satisfied [1]. Output data shows that 2 switches to Stage 2 were made before the solution was found.

```
      PROGRAM T1RMML(OUTPUT,TAPE1=OUTPUT)
C
C    J.HALD - EXAMPLE 1.
C
      DIMENSION X(2),W(67),C(1),DC(1,2)
      COMMON NCOUNT
      EXTERNAL FDF
      NCOUNT=1
      N=2
      M=3
      L=1
      LEQ=0
      IC=1
      C(1)=-2.5E0
      DC(1,1)=-3.E0
      DC(1,2)=-1.E0
      X(1)=-2.E0
      X(2)=-1.E0
      DX=0.1
      EPS=1.0E-10
      MAXF=200
      KEQS=3
      IW=67
      WRITE(1,100)
  100 FORMAT(" PROGRAM T1RMML (J.HALD - EXAMPLE 1)"//
     1 13X,"X(1)",10X,"X(2)",12X,"F(1)",10X,"F(2)",10X,"F(3)")
      CALL SECOND(TM1)
      CALL MMLA1Q(FDF,N,M,L,LEQ,C,DC,IC,X,DX,EPS,MAXF,KEQS,W,IW,IFALL)
      CALL SECOND(TM2)
      CPU=TM2-TM1
      WRITE(1,200) CPU
  200 FORMAT("0CPU TIME:",F6.3," SECONDS")
      WRITE(1,300) IFALL,EPS,MAXF,KEQS
  300 FORMAT("0IFALL  =",I5/" DXNORM= ",E13.6/
     1 " F.EVAL.=",I5/" NSHIFT =",I5)
      WRITE(1,400) (X(I),I=1,N)
  400 FORMAT("0SOLUTION:",2(/F20.10))
      WRITE(1,500) (W(I),I=1,M)
  500 FORMAT("0FUNCTION VALUES:",3(/F20.10)//)
      STOP
      END
C
C
      SUBROUTINE FDF(N,M,X,DF,F)
      DIMENSION X(N),F(M),DF(M,N)
      COMMON NCOUNT
      F(1)=X(1)**2+X(2)**2+X(1)*X(2)-1.E0
      F(2)=SIN(X(1))
      F(3)=-COS(X(2))
      DF(1,1)=2.E0*X(1)+X(2)
      DF(1,2)=2.E0*X(2)+X(1)
      DF(2,1)=COS(X(1))
      DF(2,2)=0.E0
      DF(3,1)=0.E0
      DF(3,2)=SIN(X(2))
      WRITE(1,100) NCOUNT,(X(I),I=1,N),(F(I),I=1,M)
  100 FORMAT(1X,I5,2(F13.8,1X),1X,3(1X,F13.8))
      NCOUNT=NCOUNT+1
      RETURN
      END
```

PROGRAM T1RMML (J.HALD - EXAMPLE 1)

	X(1)	X(2)	F(1)	F(2)	F(3)
1	-2.00000000	-1.00000000	6.00000000	-.90929743	-.54030231
2	-1.92191312	-.93753050	5.37456563	-.93898918	-.59178049
3	-1.76529118	-.81315378	4.21292520	-.98114543	-.68721077
4	-1.45001295	-.56698384	2.24614214	-.99271455	-.84352476
5	-.74458980	-.29292939	-.14166616	-.67767022	-.95740210
6	-.98954829	.46864486	-.26491290	-.83577804	-.89218119
7	-.92630273	.27890820	-.32252689	-.79940412	-.96135659
8	-.86305718	.08917154	-.32414088	-.75983364	-.99602685
9	-.89285714	.17857143	-.33035714	-.77886689	-.98409845
10	-.89285714	.17857143	-.33035714	-.77886689	-.98409845

CPU TIME: .068 SECONDS

IFALL = 1
DXNORM= .519212E-14
F.EVAL.= 10
NSHIFT = 2

SOLUTION:
-.8928571429
.1785714286

FUNCTION VALUES:
-.3303571429
-.7788668934
-.9840984453

Example 2

This is the antenna array problem of [3, Example 7] and [5]. The problem is to minimize the side lobes in the radiation pattern from a 15-element linear antenna array subject to constraints on the element positions. Mathematically, the problem is to minimize with respect to $\underset{\sim}{x}$ the objective function

$$F(\underset{\sim}{x}) = \max_{1 \le i \le 163} |f_i(\underset{\sim}{x})|,$$

where

$$f_i(\underset{\sim}{x}) = \frac{1}{15} + \frac{2}{15} \sum_{j=1}^{7} \cos(2\pi\ x_j \sin \theta_i),$$

$$\theta_i = \frac{\pi}{180} (8.5 + i\ 0.5)\ ,\ i = 1,2,\ldots,163,$$

$$x_7 = 3.5,$$

subject to linear constraints

$$\begin{aligned}
-x_4 + x_6 - 1.0 &= 0,\\
x_1 - 0.4 &\ge 0,\\
-x_1 + x_2 - 0.4 &\ge 0,\\
-x_2 + x_3 - 0.4 &\ge 0,\\
-x_3 + x_4 - 0.4 &\ge 0,\\
-x_4 + x_5 - 0.4 &\ge 0,\\
-x_5 + x_6 - 0.4 &\ge 0,\\
-x_6 + 3.1 &\ge 0.
\end{aligned}$$

The absolute value operation in $F(\underset{\sim}{x})$ is removed by introducing $-f_i$, $i=1,2,\ldots,163$, as extra functions, resulting in

$$F(\underset{\sim}{x}) = \max_{1 \le i \le 326} f_i(\underset{\sim}{x}),$$

$$f_{i+163} = - f_i \;, \quad i = 1,2,\ldots,163.$$

For this example,

$$N = 6$$

$$M = 326$$

$$LEQ = 1$$

$$L = 8$$

$$IC = 8$$

$$IW = 5479$$

The starting point is

$$\underset{\sim}{x}^0 = \underset{\sim}{0}.$$

The value of IFALL shows that the required accuracy EPS = 10^{-20} has not been reached due to roundoff errors (IFALL=2). No shifts to Stage 2 have taken place before the solution was found (NSHIFT=0).

```
      PROGRAM T2RMML(OUTPUT,TAPE6=OUTPUT)
C
C    J.HALD - EXAMPLE 2.
C
      DIMENSION AB(8,6),BB(8),X(6),W(5479)
      COMMON ICOUNT
      EXTERNAL FUN
      N=6
      M=326
      L=8
      LEQ=1
      DO 10 I=1,L
      DO 10 J=1,N
   10 AB(I,J)=0.E0
      AB(1,4)=-1.E0
      AB(1,6)=1.E0
      DO 20 I=1,N
      AB(I+1,I)=1.E0
   20 AB(I+2,I)=-1.E0
      IAB=8
      BB(1)=-1.E0
      DO 30 I=1,N
   30 BB(I+1)=-.4E0
      BB(8)=3.1E0
      DO 40 J=1,N
   40 X(J)=0.E0
      DX=.2E0
      EPS=1.E-20
      MAXF=100
      KEQS=3
      IW=5479
      WRITE(6,49)
   49 FORMAT(" PROGRAM T2RMML (J.HALD - EXAMPLE 2)"//)
      WRITE(6,50)
   50 FORMAT(" ",11X,"X(1)",6X,"X(2)",6X,"X(3)",6X,"X(4)",6X,
     1 "X(5)",6X,"X(6)",6X,"FMAX")
      ICOUNT=0
      CALL SECOND(TID0)
      CALL MMLA1Q(FUN,N,M,L,LEQ,BB,AB,IAB,X,DX,EPS,MAXF,KEQS,
     1 W,IW,IFALL)
      CALL SECOND(TID1)
      XCPU=TID1-TID0
      WRITE(6,55) XCPU
   55 FORMAT("0CPU TIME:",F6.3," SECONDS")
      WRITE(6,56) IFALL,EPS,MAXF,KEQS
   56 FORMAT("0IFALL  =",I5/" DXNORM =",F11.6/" F.EVAL.=",I5/
     1 " NSHIFT =",I5)
      FMAX=0.E0
      DO 60 I=1,M
   60 FMAX=AMAX1(FMAX,ABS(W(I)))
      WRITE(6,70) (X(I),I=1,N),FMAX
   70 FORMAT("0SOLUTION:",6(/F20.10)/
     1 "0FMAX AT THE SOLUTION:"/F20.10//)
      STOP
      END
C
```

```
C
      SUBROUTINE FUN(N,M,X,A,B)
      DIMENSION X(N),A(M,N),B(M),XX(7)
      COMMON ICOUNT
      PI=4.E0*ATAN(1.E0)
      MH=M/2
      N1=N+1
      DO 10 I=1,N
   10 XX(I)=X(I)
      XX(N1)=3.5E0
      THET0=8.5E0
      FMAX=0.E0
      DO 30 I=1,MH
      IMH=I+MH
      THET=PI/180.E0*(THET0+I*.5E0)
      SINTH=SIN(THET)
      B(I)=1.E0/15.E0
      DO 20 J=1,N1
      U=2.E0*PI*XX(J)*SINTH
      B(I)=B(I)+2.E0/15.E0*COS(U)
      IF(J.EQ.N1) GOTO 20
      A(I,J)=-4.E0*PI/15.E0*SINTH*SIN(U)
      A(IMH,J)=-A(I,J)
   20 CONTINUE
      B(IMH)=-B(I)
   30 FMAX=AMAX1(FMAX,ABS(B(I)))
      ICOUNT=ICOUNT+1
      WRITE(6,40)ICOUNT,(X(I),I=1,N),FMAX
   40 FORMAT(" ",I6,7(F10.6))
      RETURN
      END
```

PROGRAM T2RMML (J.HALD - EXAMPLE 2)

	X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	FMAX
1	.400000	.800000	1.200000	1.600000	2.000000	2.600000	.189267
2	.400000	.815742	1.215742	1.714063	2.116120	2.714063	.110957
3	.400000	.819770	1.219770	1.693954	2.093954	2.693954	.101873
4	.400000	.819839	1.219839	1.693985	2.093985	2.693985	.101831
5	.400000	.819839	1.219839	1.693985	2.093985	2.693985	.101831
6	.400000	.819839	1.219839	1.693985	2.093985	2.693985	.101831

```
CPU TIME: 2.226 SECONDS

IFALL  =    2
DXNORM =     .000000
F.EVAL.=    6
NSHIFT =    0

SOLUTION:
         .4000000000
         .8198390735
        1.2198390735
        1.6939853086
        2.0939853086
        2.6939853086

FMAX AT THE SOLUTION:
         .1018308888
```

VII. REFERENCES

[1] J. Hald and K. Madsen, "Combined LP and quasi-Newton methods for minimax optimization", Mathematical Programming, vol. 20, 1981, pp. 49-62.

[2] J. Hald, "Linear programming using a reduced gradient technique", Institute for Numerical Analysis, Technical Univ. of Denmark, Report NI-81-2, 1981.

[3] K. Madsen and H. Schjaer-Jacobsen, "Linearly constrained minimax optimization", Mathematical Programming, vol. 14, 1978, pp. 208-223.

[4] J. Hald, "Numerical optimization of antenna arrays, Vol. I: Second order algorithms for minimax optimization", Ph.D. Dissertation, Institute for Numerical Analysis, Technical Univ. of Denmark, 1981.

[5] K. Madsen and H. Schjaer-Jacobsen, "FORTRAN subroutines for nonlinear minimax optimization subject to linear constraints", Institute for Numerical Analysis, Technical Univ. of Denmark, Report NI-77-07, 1977.

APPENDIX

LISTING OF THE MMLA1Q PACKAGE

Subroutine	Number of Lines (source text)	Number of Words (compiled code)	Listing from Page
MMLA1Q	59	301	22
MLA1QS	232	1335	22
S2LA1Q	260	1273	26
FEASI	228	1275	30
MMLPA	280	1454	34
LINSYS	92	315	38
BFGS	42	203	39
ADDCOL	92	332	40
DELCOL	53	212	42
UTTRNS	35	114	42
UTRNS	32	125	43
RSOLV	21	72	43
RTSOLV	19	67	44
HACCUM	54	241	44
LIMIT	17	52	45

```
      SUBROUTINE MMLA1Q (FDF,N,M,L,LEQ,C,DC,IC,X,DX,EPS,MAXF,KEQS,W,IW,I
     1FALL)
C
C     MMLA1Q MINIMIZES THE MAXIMUM VALUE OF A SET OF NONLINEAR
C     FUNCTIONS SUBJECT TO LINEAR EQUALITY AND INEQUALITY CONSTRAINTS.
C     DERIVATIVES REQUIRED.
C
C     FOR A PROGRAM DESCRIPTION SEE:
C     J. HALD: "MMLA1Q, A FORTRAN SUBROUTINE FOR LINEARLY CONSTRAINED
C     MINIMAX OPTIMIZATION", REPORT NO. NI-81-1, INSTITUTE FOR NUMERICAL
C     ANALYSIS, TECHNICAL UNIVERSITY OF DENMARK, DK-2800 LYNGBY, DENMARK.
C     MARCH 1981.
C
C     THE SUBROUTINES: MMLPA,FEASI,S2LA1Q,BFGS,ADDCOL,DELCOL,HACCUM,
C     UTRNS,UTTRNS,RSOLV,RTSOLV,LIMIT,LINSYS   MUST BE AVAILABLE.
C
      DIMENSION C(IC), DC(IC,N), X(N), W(IW)
      EXTERNAL FDF
C
C        CHECK INPUT QUANTITIES
C
      IWR=2*M*N+5*N*N+4*M+8*N+4*IC+3
      IFALL=-1
      IF (IW.LT.IWR.OR.N.LT.1.OR.M.LT.1.OR.L.LT.0.OR.LEQ.LT.0.OR.LEQ.GT.
     1L.OR.LEQ.GT.N.OR.IC.LT.L.OR.DX.LE.0.E0.OR.EPS.LT.0.E0.OR.MAXF.LE.0
     2) RETURN
C
C        SPLIT UP THE WORK AREA
C
      N1=N+1
      NN=N+N
      NF=1
      NF1=NF+M
      NDF=NF1+M
      NDF1=NDF+M*N
      NX1=NDF1+M*N
      NB=NX1+N
      NU=NB+N*N
      NR=NU+N*N
      NA=NU
      NCL=NA+NN*NN
      NWL=NCL+IC
      NWL1=NWL+IC
      NXX=NWL1+IC
      NW=NXX+N
      NW1=NW+N
      NW2=NW1+N
      NWM=NW2+N
      NAS=NWM+M
      NKS=NAS+N1
      NKS0=NKS+N1
      NKSTC=NKS0+N1
      NKSTF=NKSTC+IC
      CALL MLA1QS (FDF,N,M,L,LEQ,C,DC,IC,X,DX,EPS,MAXF,KEQS,N1,NN,W(NF),
     1W(NF1),W(NDF),W(NDF1),W(NX1),W(NB),W(NU),W(NR),W(NA),W(NCL),W(NWL)
     2,W(NWL1),W(NXX),W(NW),W(NW1),W(NW2),W(NWM),W(NAS),W(NKS),W(NKS0),W
     3(NKSTC),W(NKSTF),IFALL)
      RETURN
      END
C
C
      SUBROUTINE MLA1QS (FDF,N,M,L,LEQ,C,DC,IC,X,DX,EPS,MAXF,KEQS,N1,NN,
     1F,F1,DF,DF1,X1,B,U,R,A,CLOC,WL,WL1,XX,W,W1,W2,WM,ASET,KSET,KSET0,K
     2STATC,KSTATF,IFALL)
      DIMENSION C(IC), DC(IC,N), X(N), F(M), F1(M), WM(M), DF(M,N), DF1(
```

```
     1M,N), XX(NN), X1(N), U(N,N), R(N,N), B(N,N), A(NN,NN), CLOC(IC), W
     2L(IC), WL1(IC), W(N), W1(N), W2(N), ASET(N1)
      INTEGER KSET(N1),KSET0(N1),KSTATC(IC),KSTATF(M)
      LOGICAL DIV4,ACCUM,SHIFT
      EXTERNAL FDF
C
C       SEPS IS AN EXPRESSION FOR THE MACHINE ACCURACY
C
      SEPS=16.E0**(-12)
C
C       SET SOME CONSTANTS
C
      DIV4=.FALSE.
      LI=L-LEQ
C
C       INITIALIZE
C
      KEQSET=0
      NCALL=0
      NSHIFT=0
      NSTEP=0
      FMMREF=1.E73
      DX0=DX
      DO 2 I=1,N
      DO 1 J=1,N
      B(I,J)=0.E0
    1 CONTINUE
      B(I,I)=1.E0
    2 CONTINUE
C
C       FIND A FEASIBLE POINT
C
      IF (L.EQ.0) GO TO 8
      DO 4 I=1,L
      T=C(I)
      DO 3 J=1,N
      T=T+DC(I,J)*X(J)
    3 CONTINUE
      CLOC(I)=T
    4 CONTINUE
      NACT=0
      CALL FEASI (CLOC,DC,IC,LEQ,LI,N,XX,NACT,KSET,ASET,U,R,W1,W2,WL,WL1
     1,W,KSTATC,IFALL,ACCUM,SEPS)
      IF (IFALL.NE.0) RETURN
      DO 5 I=1,N
      X(I)=X(I)+XX(I)
    5 CONTINUE
      DO 7 I=1,L
      T=C(I)
      DO 6 J=1,N
      T=T+DC(I,J)*X(J)
    6 CONTINUE
      CLOC(I)=T
    7 CONTINUE
C
C       CALCULATE FUNCTION VALUES IN THE FIRST FEASIBLE POINT
C
    8 CALL FDF (N,M,X,DF,F)
      FMM0=F(1)
      DO 9 I=1,M
      FMM0=AMAX1(FMM0,F(I))
    9 CONTINUE
      NACT0=0
      NCALL=1
C
```

```
C        ITERATIVE LOOP STARTS HERE
C
   10 NACT=NACT0
      IF (NACT.EQ.0) GO TO 12
      DO 11 I=1,NACT
      KSET(I)=KSET0(I)
   11 CONTINUE
C
C        SOLVE THE LINEAR SUBPROBLEMS
C
   12 CALL MMLPA (F,DF,CLOC,DC,M,N,N1,IC,LEQ,LI,DX,XXN,XX,NACT,KSET,ASET
     1,U,R,W1,W2,F1,WM,WL,WL1,KSTATF,KSTATC,W,SEPS,ACCUM,FMM,IFALL)
      IF (FMM.GE.FMM0) GO TO 39
C
C        CALCULATE FUNCTION VALUES IN THE NEW POINT
C
      DO 13 I=1,N
      X1(I)=X(I)+XX(I)
   13 CONTINUE
      CALL FDF (N,M,X1,DF1,F1)
      NCALL=NCALL+1
      FMM1=F1(1)
      DO 14 I=1,M
      FMM1=AMAX1(FMM1,F1(I))
   14 CONTINUE
C
C        REVISE THE STEP LENGTH
C
      IF ((FMM0-FMM1).GT.(FMM0-FMM)/4.E0) GO TO 15
      DX=XXN/4.E0
      DIV4=.TRUE.
      GO TO 17
   15 IF (DIV4) GO TO 16
      IF ((FMM0-FMM1).GT.(FMM0-FMM)*0.75E0) DX=XXN*2.E0
   16 DIV4=.FALSE.
C
C        UPDATE THE HESSIAN APPROXIMATION
C
   17 DO 18 J=1,N
      W(J)=0.E0
      W1(J)=0.E0
   18 CONTINUE
      DO 20 I=1,NACT
      K=KSET(I)
      IF (K.LE.L) GO TO 20
      KK=K-L
      T=-ASET(I)
      DO 19 J=1,N
      W1(J)=W1(J)+T*DF1(KK,J)
      W(J)=W(J)+T*DF(KK,J)
   19 CONTINUE
   20 CONTINUE
      DO 21 I=1,N
      W2(I)=W1(I)-W(I)
   21 CONTINUE
      CALL BFGS (B,N,W2,XX,W,SEPS)
C
C        TEST IF THE NEW POINT IS ACCEPTABLE
C
      IF ((FMM0-FMM1).LE.0.01E0*(FMM0-FMM)) GO TO 31
C
C        COMPARE THE NEW ACTIVE SET WITH THE PRECEDING
C
      IF (NACT0.NE.NACT) GO TO 24
      DO 23 I=1,NACT
```

```
      K=KSET(I)
      DO 22 J=1,NACT
      IF (K.EQ.KSET0(J)) GO TO 23
   22 CONTINUE
      GO TO 24
   23 CONTINUE
      KEQSET=KEQSET+1
      GO TO 25
   24 KEQSET=1
C
C        INTRODUCE THE NEW POINT
C
   25 NSTEP=NSTEP+1
      XN=0.E0
      FMM0=FMM1
      NACT0=NACT
      DO 26 I=1,N
      X(I)=X1(I)
      XN=XN+X(I)**2
      DO 26 J=1,M
   26 DF(J,I)=DF1(J,I)
      XN=SQRT(XN)
      DO 27 I=1,M
      F(I)=F1(I)
   27 CONTINUE
      DO 28 I=1,NACT0
      KSET0(I)=KSET(I)
   28 CONTINUE
      IF (L.EQ.0) GO TO 31
      DO 30 I=1,L
      T=C(I)
      DO 29 J=1,N
      T=T+DC(I,J)*X(J)
   29 CONTINUE
      CLOC(I)=T
   30 CONTINUE
   31 IF (NCALL.GT.MAXF) GO TO 39
C
C        TEST OF CONVERGENCE CRITERION
C
      IF (XXN.LE.EPS*XN) GO TO 40
      IF (XXN.LE.SEPS*XN) GO TO 39
C
C        TEST FOR SWITCH TO STAGE-2
C
      SHIFT=FMM0.LE.FMMREF.AND.KEQSET.GE.KEQS.AND.NSTEP.GE.N
      IF (.NOT.SHIFT) GO TO 10
      IF (NACT.EQ.N1) GO TO 37
C
C        TEST FOR POSITIVE DEFINITENESS OF THE HESSIAN APPROX.
C        IN A RELEVANT DIRECTION.
C
      DO 33 I=1,NACT
      K=KSET(I)
      IF (K.GT.L) GO TO 33
      T=ASET(I)
      DO 32 J=1,N
      W1(J)=W1(J)+T*DC(K,J)
   32 CONTINUE
   33 CONTINUE
      DO 35 I=1,N
      T=0.E0
      DO 34 J=1,N
      T=T+B(I,J)*W1(J)
   34 CONTINUE
```

```
      W(I)=T
   35 CONTINUE
      T=0.E0
      DO 36 I=1,N
      T=T+W(I)*W1(I)
   36 CONTINUE
      IF (T.LE.0.E0) GO TO 10
C
C        SHIFT TO STAGE-2
C
   37 NSHIFT=NSHIFT+1
      FMMREF=FMM0-10.E0*SEPS*ABS(FMM0)
      XXNMAX=AMAX1(DX0,2.E0*DX)
      CALL S2LA1Q (FDF,N,M,L,LEQ,C,CLOC,DC,IC,X,XXNMAX,B,NACT,KSET,ASET,
     1N1,KSTATF,KSTATC,A,XX,NN,F,DF,X1,F1,DF1,W1,W2,EPS,MAXF,NCALL,XXN,N
     2STEP,SEPS,IFALL)
      IF (IFALL.LT.3) GO TO 40
      FMM0=-1.E73
      DO 38 I=1,M
      FMM0=AMAX1(FMM0,F(I))
   38 CONTINUE
      DX=AMAX1(DX,XXN/2.E0)
      KEQSET=1
      GO TO 10
C
C        RETURN
C
   39 IFALL=2
   40 MAXF=NCALL
      KEQS=NSHIFT
      EPS=XXN
      RETURN
      END
C
C
      SUBROUTINE S2LA1Q (FDF,N,M,L,LEQ,C,CLOC,DC,IC,X,XXNMAX,B,NACT,KSET
     1,ASET,N1,KSTATF,KSTATC,DZ,ZZ,NN,F,DF,X1,F1,DF1,W,W1,EPS,MAXF,NCALL
     2,XXN,NSTEP,SEPS,IFALL)
C
C        STAGE-2 (QUASI-NEWTON) ALGORITHM FOR LINEARLY CONSTRAINED
C        MINIMAX OPTIMIZATION.
C
      DIMENSION C(IC), CLOC(IC), DC(IC,N), X(N), B(N,N), ASET(N1), DZ(NN
     1,NN), ZZ(NN), F(M), DF(M,N), X1(N), F1(M), DF1(M,N), W(N), W1(N)
      INTEGER KSET(N1),KSTATF(M),KSTATC(IC)
      EXTERNAL FDF
C
C        INITIALIZE
C
      LI=L-LEQ
      LE1=LEQ+1
      IFALL=0
      SSEPS=SQRT(SEPS)
      KK0=KSET(NACT)-L
      NACT1=NACT-1
      NZ=N+NACT1
      NSTEP2=0
      XXN=0.E0
      DO 1 I=1,M
      KSTATF(I)=0
    1 CONTINUE
      IF (L.EQ.0) GO TO 30
      DO 2 I=1,L
      KSTATC(I)=0
    2 CONTINUE
```

```
    3 DO 4 I=1,NACT
      K=KSET(I)
      IF (K.LE.L) KSTATC(K)=1
      IF (K.GT.L) KSTATF(K-L)=1
    4 CONTINUE
C
C        ITERATIVE LOOP STARTS HERE
C
C        SET UP THE ITERATION MATRIX AND THE RIGHTHAND SIDE
C
    5 DO 7 I=1,N
      DO 6 J=1,N
      DZ(I,J)=B(I,J)
    6 CONTINUE
      ZZ(I)=-DF(KK0,I)
    7 CONTINUE
      IF (NACT.EQ.1) GO TO 15
      DO 14 J=1,NACT1
      K=KSET(J)
      JN=J+N
      IF (K.GT.L) GO TO 9
      ZZ(JN)=-CLOC(K)
      DO 8 I=1,N
      DZ(I,JN)=DC(K,I)
      DZ(JN,I)=DZ(I,JN)
    8 CONTINUE
      GO TO 11
    9 KK=K-L
      ZZ(JN)=F(KK)-F(KK0)
      DO 10 I=1,N
      DZ(I,JN)=DF(KK0,I)-DF(KK,I)
      DZ(JN,I)=DZ(I,JN)
   10 CONTINUE
   11 DO 12 I=N1,NZ
      DZ(I,JN)=0.E0
   12 CONTINUE
      T=ASET(J)
      DO 13 I=1,N
      ZZ(I)=ZZ(I)-T*DZ(JN,I)
   13 CONTINUE
   14 CONTINUE
   15 RES0=0.E0
      DO 16 I=1,NZ
      RES0=RES0+ZZ(I)**2
   16 CONTINUE
      RES0=SQRT(RES0)
C
C        CALCULATE THE QUASI-NEWTON STEP
C
      CALL LINSYS (DZ,ZZ,NN,NZ,K,SEPS)
      IF (K.EQ.NZ) GO TO 17
      IFALL=3
      RETURN
C
C        CONTROL STEP LENGTH
C
   17 XXN1=0.E0
      ALFA=1.E0
      DO 18 I=1,N
      XXN1=XXN1+ZZ(I)**2
   18 CONTINUE
      XXN1=SQRT(XXN1)
      IF (XXN1.GT.XXNMAX) ALFA=XXNMAX/XXN1
C
C        WILL OTHER CONSTRAINTS OR FUNCTIONS BECOME ACTIVE ?
```

```
C
      STEP=1.E73
      IF (LI.EQ.0) GO TO 21
      DO 20 I=LE1,L
      IF (KSTATC(I).NE.0) GO TO 20
      T=0.E0
      DO 19 J=1,N
      T=T+ZZ(J)*DC(I,J)
   19 CONTINUE
      IF (T.GE.0.E0) GO TO 20
      T=-CLOC(I)/T
      IF (T.GT.STEP) GO TO 20
      STEP=T
   20 CONTINUE
   21 T0=0.E0
      DO 22 I=1,N
      T0=T0+ZZ(I)*DF(KK0,I)
   22 CONTINUE
      F0=F(KK0)
      DO 24 I=1,M
      IF (KSTATF(I).NE.0) GO TO 24
      T=0.E0
      DO 23 J=1,N
      T=T+ZZ(J)*DF(I,J)
   23 CONTINUE
      T=T0-T
      IF (T.GE.0.E0) GO TO 24
      T=(F(I)-F0)/T
      IF (T.GT.STEP) GO TO 24
      STEP=T
   24 CONTINUE
      IF (STEP.GT.ALFA) GO TO 25
      IFALL=4
      ALFA=STEP
C
C        SCALE THE STEP
C
   25 DO 26 I=1,NZ
      ZZ(I)=ALFA*ZZ(I)
   26 CONTINUE
      XXN1=ABS(ALFA)*XXN1
C
C       CALCULATE FUNCTION VALUES AND RESIDUALS IN THE NEW POINT
C
      XN1=0.E0
      DO 27 I=1,N
      X1(I)=X(I)+ZZ(I)
      XN1=XN1+X1(I)**2
   27 CONTINUE
      XN1=SQRT(XN1)
      NCALL=NCALL+1
      CALL FDF (N,M,X1,DF1,F1)
      DASET0=0.E0
      IF (NACT.EQ.1) GO TO 29
      DO 28 I=N1,NZ
      IF (KSET(I-N).GT.L) DASET0=DASET0-ZZ(I)
   28 CONTINUE
   29 RES=0.E0
      T=ASET(NACT)+DASET0
      DO 30 I=1,N
      W(I)=-T*DF(KK0,I)
      W1(I)=-T*DF1(KK0,I)
   30 CONTINUE
      IF (NACT.EQ.1) GO TO 35
      DO 34 J=1,NACT1
```

```
      K=KSET(J)
      JN=J+N
      T=ASET(J)+ZZ(JN)
      IF (K.GT.L) GO TO 32
      S=C(K)
      DO 31 I=1,N
      SS=DC(K,I)
      W(I)=W(I)+T*SS
      W1(I)=W1(I)+T*SS
      S=S+SS*X1(I)
   31 CONTINUE
      RES=RES+S**2
      GO TO 34
   32 KK=K-L
      DO 33 I=1,N
      W(I)=W(I)-T*DF(KK,I)
      W1(I)=W1(I)-T*DF1(KK,I)
   33 CONTINUE
      RES=RES+(F1(KK0)-F1(KK))**2
   34 CONTINUE
   35 DO 36 I=1,N
      RES=RES+W1(I)**2
   36 CONTINUE
      RES=SQRT(RES)
C
C        UPDATE THE HESSIAN APPROXIMATION
C
      DO 37 I=1,N
      W1(I)=W1(I)-W(I)
   37 CONTINUE
      CALL BFGS (B,N,W1,ZZ,W,SEPS)
C
C        TEST IF THE RESIDUAL HAS DECREASED
C
      IF (NSTEP2.EQ.0) GO TO 39
      IF (RES.LE.0.999E0*RES0) GO TO 39
      IF (IFALL.EQ.3) RETURN
C
C        IF NO - TEST FOR MACHINE ACCURACY
C
      IF (XXN1.GT.SSEPS*(XXNMAX+XN1).OR.NSTEP2.LT.2) GO TO 38
      IFALL=2
      RETURN
   38 IFALL=5
      RETURN
C
C        IF YES - INTRODUCE THE NEW POINT
C
   39 NSTEP2=NSTEP2+1
      NSTEP=NSTEP+1
      XN=0.E0
      DO 40 I=1,N
      X(I)=X1(I)
      DO 40 J=1,M
   40 DF(J,I)=DF1(J,I)
      XN=XN1
      XXN=XXN1
      FMAX=-1.E73
      DO 41 I=1,M
      T=F1(I)
      FMAX=AMAX1(T,FMAX)
      F(I)=T
   41 CONTINUE
      ASET(NACT)=ASET(NACT)+DASET0
      IF (ASET(NACT).GT.0.E0) IFALL=6
```

```
      IF (NACT.EQ.1) GO TO 43
      DO 42 I=1,NACT1
      IN=I+N
      ASET(I)=ASET(I)+ZZ(IN)
      IF (KSET(I).GT.LEQ.AND.ASET(I).GT.0.E0) IFALL=6
   42 CONTINUE
   43 IF (L.EQ.0) GO TO 47
      DO 45 J=1,L
      T=C(J)
      DO 44 I=1,N
      T=T+DC(J,I)*X(I)
   44 CONTINUE
      CLOC(J)=T
   45 CONTINUE
C
C        TEST IF THE ACTIVE SET IS COMPLETE
C
      T=FMAX+RES
      DO 46 I=1,M
      IF (F(I).LE.T) GO TO 46
      IFALL=7
      RETURN
   46 CONTINUE
C
C        TEST CONVERGENCE CRITERION
C
   47 IF (XXN.GT.EPS*XN) GO TO 48
      IF (NACT.LT.N1) IFALL=1
      RETURN
   48 IF (XXN.GT.SEPS*XN.AND.NCALL.LT.MAXF) GO TO 49
      IFALL=2
      RETURN
   49 IF (IFALL.GT.2) RETURN
      GO TO 5
      END
C
C
      SUBROUTINE FEASI (C,DC,IC,LE,LI,N,X,NACT,KSET,ASET,U,R,DL,RIGHT,CU
     1P,DLDC,W,KSTAT,IFALL,ACCUM,SEPS)
C
C        THE SUBROUTINE FINDS A FEASIBLE POINT FOR A SET OF LINEAR
C        EQUALITY AND INEQUALITY CONSTRAINTS.
C
      DIMENSION C(IC), DC(IC,N), X(N), ASET(N), U(N,N), R(N,N), DL(N), R
     1IGHT(N), CUP(IC), DLDC(IC), W(N)
      INTEGER KSET(N),KSTAT(IC)
      LOGICAL ACCUM,OBJECT
C
C        INITIALIZE
C
      EPS=(N+10)*SEPS
      ACCUM=.FALSE.
      NACTIN=NACT
      NACT=0
      LE1=LE+1
      LELI=LE+LI
      DO 1 I=1,N
      X(I)=0.E0
    1 CONTINUE
      IFALL=0
      IF (LELI.EQ.0) RETURN
      DO 2 I=1,LELI
      KSTAT(I)=0
    2 CONTINUE
C
```

```
C
C        MAKE ACTIVE THE EQUALITY CONSTRAINTS PLUS OTHER CONSTRAINTS
C        AS DEFINED IN KSET
C
      IF (LE.EQ.0) GO TO 5
      IF (LE.GT.N) GO TO 41
      DO 4 I=1,LE
      RIGHT(I)=-C(I)
      DO 3 J=1,N
      R(J,I)=DC(I,J)
    3 CONTINUE
      KSET(I)=I
      KSTAT(I)=1
    4 CONTINUE
      CALL ADDCOL (U,R,N,NACT,LE,RIGHT,W,ACCUM,.FALSE.,EPS)
      IF (NACT.LT.LE) GO TO 41
    5 IF (NACTIN.LT.1) GO TO 8
      DO 7 K=1,NACTIN
      KK=KSET(K)
      IF (KK.LT.LE1.OR.KK.GT.LELI) GO TO 7
      NACT1=NACT+1
      IF (NACT1.GT.N) GO TO 8
      DO 6 I=1,N
      R(I,NACT1)=DC(KK,I)
    6 CONTINUE
      CALL UTTRNS (U,N,NACT,ACCUM,R(1,NACT1),W)
      CALL ADDCOL (U,R,N,NACT,1,RIGHT,W,ACCUM,.FALSE.,EPS)
      IF (NACT.LT.NACT1) GO TO 7
      RIGHT(NACT1)=-C(KK)
      KSET(NACT1)=KK
      KSTAT(KK)=1
    7 CONTINUE
    8 CALL RTSOLV (R,N,NACT,RIGHT,X)
      IF (NACT.EQ.N) GO TO 10
      NACT1=NACT+1
      DO 9 I=NACT1,N
      X(I)=0.E0
    9 CONTINUE
   10 CALL UTRNS (U,N,NACT,ACCUM,X,W)
C
C        UPDATE THE CONSTRAINTS
C
      IF (LI.EQ.0) RETURN
      DO 12 I=LE1,LELI
      T=C(I)
      DO 11 J=1,N
      T=T+DC(I,J)*X(J)
   11 CONTINUE
      CUP(I)=T
   12 CONTINUE
C
C        INITIALIZE INEQUALITY CONSTRAINT LOOP
C
      DO 13 I=LE1,LELI
      IF (CUP(I).LT.0.E0.AND.KSTAT(I).EQ.0) KSTAT(I)=-1
   13 CONTINUE
C
C        ACTIVATE VIOLATED INEQUALITY CONSTRAINTS ONE BY ONE
C        USE THE STRONGEST VIOLATED AS OBJECTIVE CONSTRAINT
C
   14 FMIN=1.E73
      DO 15 I=LE1,LELI
      IF (KSTAT(I).NE.-1) GO TO 15
      IF (CUP(I).GE.FMIN) GO TO 15
      FMIN=CUP(I)
      NEW=I
```

```
   15 CONTINUE
      IF (FMIN.GE.0.E0) RETURN
      DO 16 I=1,N
      RIGHT(I)=DC(NEW,I)
   16 CONTINUE
      CALL UTTRNS (U,N,NACT,ACCUM,RIGHT,W)
      KSTAT(NEW)=1
C
C        CALCULATE MULTIPLIERS FOR THE NEW ACTIVE CONSTRAINT AND DROP
C        CONSTRAINTS WITH POSITIVE MULTIPLIERS IN ORDER TO ACHIEVE
C        THE DIRECTION OF STEEPEST INCREMENT
C
   17 IF (NACT.EQ.LE) GO TO 22
      CALL RSOLV (R,N,NACT,RIGHT,ASET)
      AMAX=-1.E73
      DO 18 I=LE1,NACT
      IF (ASET(I).LT.AMAX) GO TO 18
      K=I
      AMAX=ASET(I)
   18 CONTINUE
      IF (AMAX.LT.0.E0) GO TO 22
      KSTAT(KSET(K))=0
      DO 19 I=LE1,LELI
      IF (KSTAT(I).EQ.-2) KSTAT(I)=0
   19 CONTINUE
      IF (ACCUM) GO TO 20
      ACCUM=.TRUE.
      CALL HACCUM (U,N,NACT,W)
   20 CALL DELCOL (K,U,R,N,NACT,RIGHT,.TRUE.)
      IF (K.GT.NACT) GO TO 17
      DO 21 I=K,NACT
      KSET(I)=KSET(I+1)
   21 CONTINUE
      GO TO 17
C
C        CALCULATE THE PROJECTED GRADIENT
C
   22 T=0.E0
      DLN2=0.E0
      IF (NACT.EQ.0) GO TO 24
      DO 23 I=1,NACT
      T=T+RIGHT(I)**2
      DL(I)=0.E0
   23 CONTINUE
   24 NACT1=NACT+1
      IF (NACT.EQ.N) GO TO 26
      DO 25 I=NACT1,N
      DLN2=DLN2+RIGHT(I)**2
      DL(I)=RIGHT(I)
   25 CONTINUE
   26 T=T+DLN2
      IF (T.GT.0.E0.AND.DLN2.GT.EPS*EPS*T) GO TO 28
      S=(N+1)*ABS(C(NEW))
      DO 27 I=1,N
      S=S+ABS(DC(NEW,I)*X(I))*(N+3-I)
   27 CONTINUE
      IF (CUP(NEW).LT.-EPS*S) GO TO 41
      KSTAT(NEW)=0
      GO TO 14
   28 CALL UTRNS (U,N,NACT,ACCUM,DL,W)
C
C        PROJECT GRADIENTS ON THE PROJECTED GRADIENT
C
      DO 30 I=LE1,LELI
      T=0.E0
```

```
      DO 29 J=1,N
      T=T+DL(J)*DC(I,J)
   29 CONTINUE
      DLDC(I)=T
   30 CONTINUE
C
C        CALCULATE STEP LENGTH "ANES" TO MAKE THE OBJECTIVE CONSTRAINT
C        EQUAL ZERO, AND CALCULATE THE STEP LENGTH "AMIN" TO THE
C        NEAREST INACTIVE CONSTRAINT UNDER CONSIDERATION
C
      ANES=-CUP(NEW)/DLN2
   31 AMIN=1.E73
      DO 32 I=LE1,LELI
      IF (KSTAT(I).NE.0) GO TO 32
      T=DLDC(I)
      IF (T.GE.0.E0) GO TO 32
      T=-CUP(I)/T
      IF (T.GT.AMIN) GO TO 32
      AMIN=T
      K=I
   32 CONTINUE
C
C        WILL THE OBJECTIVE CONSTRAINT GET ACTIVE ?
C        IF NOT, MAKE ACTIVE THE CLOSEST
C
      OBJECT=ANES.LE.AMIN
      ALFA=AMIN1(AMIN,ANES)
      NACT1=NACT+1
      IF (OBJECT) GO TO 35
      DO 33 I=1,N
      R(I,NACT1)=DC(K,I)
   33 CONTINUE
      CALL UTTRNS (U,N,NACT,ACCUM,R(1,NACT1),W)
      CALL ADDCOL (U,R,N,NACT,1,RIGHT,W,ACCUM,.TRUE.,EPS)
      IF (NACT1.EQ.NACT) GO TO 34
      KSTAT(K)=-2
      GO TO 31
   34 KSTAT(K)=1
      KSET(NACT)=K
C
C        TAKE THE STEP
C
   35 IF (ALFA.EQ.0.E0) GO TO 38
      DO 36 I=1,N
      X(I)=X(I)+ALFA*DL(I)
   36 CONTINUE
      DO 37 I=LE1,LELI
      T=CUP(I)+ALFA*DLDC(I)
      IF (KSTAT(I).EQ.-1.AND.T.GE.0.E0) KSTAT(I)=0
      CUP(I)=T
   37 CONTINUE
   38 IF (.NOT.OBJECT) GO TO 17
C
C        ACTIVATE THE OBJECTIVE CONSTRAINT
C
      DO 39 I=1,N
      R(I,NACT1)=RIGHT(I)
   39 CONTINUE
      CALL ADDCOL (U,R,N,NACT,1,RIGHT,W,ACCUM,.FALSE.,EPS)
      IF (NACT.EQ.NACT1) GO TO 40
      KSTAT(NEW)=0
      GO TO 14
   40 KSET(NACT)=NEW
      GO TO 14
C
```

```
C        NO FEASIBLE POINTS
C
   41 IFALL=3
      RETURN
      END
C
C
      SUBROUTINE MMLPA (F,DF,C,DC,M,N,N1,IC,LE,LI,XNMAX,XN,X,NACT,KSET,A
     1SET,U,R,DL,RIGHT,FUP,DLDF,CUP,DLDC,KSTATF,KSTATC,W,SEPS,ACCUM,FMAX
     2,IFALL)
C
C        THE SUBROUTINE SOLVES A LINEARLY CONSTRAINED
C        LINEAR MINIMAX PROBLEM.
C        THE STARTING POINT MUST BE FEASIBLE.
C
      DIMENSION F(M), DF(M,N), C(IC), DC(IC,N), X(N), ASET(N1), U(N,N),
     1R(N,N), DL(N), RIGHT(N), FUP(M), DLDF(M), CUP(IC), DLDC(IC), W(N)
      INTEGER KSET(N1),KSTATF(M),KSTATC(IC)
      LOGICAL ACCUM
C
C        INITIALIZE
C
      LE1=LE+1
      LELI=LE+LI
      XNMAX2=XNMAX**2
      XN2=0.E0
      EPS=N*SEPS
      ACCUM=.FALSE.
      IFALL=0
      DO 1 I=1,N
      X(I)=0.E0
    1 CONTINUE
      FMAX=-1.E73
      DO 2 I=1,M
      KSTATF(I)=0
      T=F(I)
      IF (T.LE.FMAX) GO TO 2
      FMAX=T
      KSET0=I
    2 FUP(I)=T
      IF (LELI.EQ.0) GO TO 4
      DO 3 I=1,LELI
      KSTATC(I)=0
      CUP(I)=C(I)
    3 CONTINUE
C
C        ACTIVATE INITIAL ACTIVE SET
C
    4 NACTIN=NACT
      NACT=0
      IF (LE.EQ.0) GO TO 7
      DO 6 I=1,LE
      DO 5 J=1,N
      R(J,I)=DC(I,J)
    5 CONTINUE
      KSET(I)=I
      KSTATC(I)=1
    6 CONTINUE
      CALL ADDCOL (U,R,N,NACT,LE,RIGHT,W,ACCUM,.FALSE.,EPS)
      IF (NACT.EQ.LE) GO TO 7
      XN=0.E0
      IFALL=3
      RETURN
    7 IF (NACTIN.LT.LE1) GO TO 10
      DO 9 K=1,NACTIN
```

```
      KK=KSET(K)
      IF (KK.LT.LE1.OR.KK.GT.LELI) GO TO 9
      NACT1=NACT+1
      IF (NACT1.GT.N) GO TO 10
      DO 8 I=1,N
      R(I,NACT1)=DC(KK,I)
    8 CONTINUE
      CALL UTTRNS (U,N,NACT,ACCUM,R(1,NACT1),W)
      CALL ADDCOL (U,R,N,NACT,1,RIGHT,W,ACCUM,.FALSE.,EPS)
      IF (NACT.LT.NACT1) GO TO 9
      EPS=EPS+SEPS
      KSET(NACT1)=KK
      KSTATC(KK)=1
    9 CONTINUE
C
C        TRANSFORM OBJECTIVE FUNCTION GRADIENT
C
   10 KSTATF(KSET0)=1
      DO 11 J=1,N
      RIGHT(J)=-DF(KSET0,J)
   11 CONTINUE
      KSET0=KSET0+LELI
      CALL UTTRNS (U,N,NACT,ACCUM,RIGHT,W)
C
C        ITERATIVE LOOP
C
C        CALCULATE MULTIPLIERS AND FIND THE LARGEST
C
   12 ASET0=-1.E0
      IF (NACT.EQ.0) GO TO 24
      CALL RSOLV (R,N,NACT,RIGHT,ASET)
      IF (NACT.EQ.LE) GO TO 24
      AMAX=-1.E73
      DO 13 I=LE1,NACT
      IF (KSET(I).GT.LELI) ASET0=ASET0-ASET(I)
      IF (ASET(I).LE.AMAX) GO TO 13
      K=I
      AMAX=ASET(I)
   13 CONTINUE
      IF (AMAX.LT.0.E0.AND.ASET0.LT.0.E0) GO TO 24
      IF (AMAX.GT.ASET0) GO TO 18
C
C        CHANGE OBJECTIVE FUNCTION
C
      DO 14 I=LE1,NACT
      IF (KSET(I).LE.LELI) GO TO 14
      K=I
      GO TO 15
   14 CONTINUE
   15 DO 17 I=1,K
      T=R(I,K)
      IF (K.EQ.NACT) GO TO 17
      K1=K+1
      DO 16 J=K1,NACT
      IF (KSET(J).GT.LELI) R(I,J)=R(I,J)-T
   16 CONTINUE
   17 RIGHT(I)=RIGHT(I)+T
      KK=KSET0
      KSET0=KSET(K)
      KSET(K)=KK
C
C        DELETE ACTIVE CONSTRAINT NUMBER K
C
   18 KK=KSET(K)
      IF (KK.GT.LELI) KSTATF(KK-LELI)=0
```

```
      IF (KK.LE.LELI) KSTATC(KK)=0
      IF (ACCUM) GO TO 19
      ACCUM=.TRUE.
      CALL HACCUM (U,N,NACT,W)
   19 CALL DELCOL (K,U,R,N,NACT,RIGHT,.TRUE.)
      EPS=EPS+SEPS
      IF (K.GT.NACT) GO TO 21
      DO 20 I=K,NACT
      KSET(I)=KSET(I+1)
   20 CONTINUE
C
C        DELETE LINEAR DEPENDENCE LABELS
C
   21 DO 22 I=1,M
      IF (KSTATF(I).EQ.-2) KSTATF(I)=0
   22 CONTINUE
      IF (LI.EQ.0) GO TO 12
      DO 23 I=LE1,LELI
      IF (KSTATC(I).EQ.-2) KSTATC(I)=0
   23 CONTINUE
      GO TO 12
C
C        IS THERE AN UNBOUNDED SOLUTION ?
C
   24 IF (NACT.EQ.N) GO TO 49
C
C        CALCULATE THE PROJECTED GRADIENT
C
      K=NACT+1
      T=0.E0
      DLN2=0.E0
      DO 25 I=K,N
      DLN2=DLN2+RIGHT(I)**2
      DL(I)=RIGHT(I)
   25 CONTINUE
      IF (K.EQ.1) GO TO 27
      DO 26 I=1,NACT
      T=T+RIGHT(I)**2
      DL(I)=0.E0
   26 CONTINUE
   27 T=T+DLN2
      IF (T.GT.0.E0.AND.DLN2.GT.EPS*EPS*T) GO TO 28
      IFALL=2
      GO TO 49
   28 CALL UTRNS (U,N,NACT,ACCUM,DL,W)
C
C        PROJECT GRADIENTS ON THE PROJECTED GRADIENT
C
      DO 30 I=1,M
      T=0.E0
      DO 29 J=1,N
      T=T+DL(J)*DF(I,J)
   29 CONTINUE
      DLDF(I)=T
   30 CONTINUE
      IF (LELI.EQ.0) GO TO 33
      DO 32 I=1,LELI
      T=0.E0
      DO 31 J=1,N
      T=T+DL(J)*DC(I,J)
   31 CONTINUE
      DLDC(I)=T
   32 CONTINUE
C
C        CALCULATE STEP LENGTH
```

```
C
   33 SMINC=1.E73
      IF (LI.EQ.0) GO TO 35
      DO 34 I=LE1,LELI
      IF (KSTATC(I).NE.0) GO TO 34
      T=DLDC(I)
      IF (T.GE.0.E0) GO TO 34
      T=-CUP(I)/T
      IF (T.GT.SMINC) GO TO 34
      NEWC=I
      SMINC=T
   34 CONTINUE
   35 SMINF=1.E73
      K=KSET0-LELI
      T0=DLDF(K)
      F0=FUP(K)
      DO 36 I=1,M
      IF (KSTATF(I).NE.0) GO TO 36
      T=T0-DLDF(I)
      IF (T.GE.0.E0) GO TO 36
      T=(FUP(I)-F0)/T
      IF (T.GT.SMINF) GO TO 36
      SMINF=T
      NEWF=I
   36 CONTINUE
      STEP=AMIN1(SMINF,SMINC)
C
C        IN CASE THE STEP IS TOO LONG REDUCE AND RETURN
C
      S=STEP
      CALL LIMIT (XNMAX2,X,XN2,DL,DLN2,S,N)
      IF (S.EQ.STEP) GO TO 37
      IFALL=1
      STEP=S
      GO TO 44
C
C        INCLUDE THE NEW FUNCTION/CONSTRAINT
C
   37 NACT1=NACT+1
      KK0=KSET0-LELI
      IF (SMINF.LT.SMINC) GO TO 40
      DO 38 I=1,N
      R(I,NACT1)=DC(NEWC,I)
   38 CONTINUE
      CALL UTTRNS (U,N,NACT,ACCUM,R(1,NACT1),W)
      CALL ADDCOL (U,R,N,NACT,1,RIGHT,W,ACCUM,.TRUE.,EPS)
      IF (NACT.EQ.NACT1) GO TO 39
      KSTATC(NEWC)=-2
      GO TO 33
   39 KSTATC(NEWC)=1
      KSET(NACT)=NEWC
      GO TO 43
   40 DO 41 I=1,N
      R(I,NACT1)=DF(KK0,I)-DF(NEWF,I)
   41 CONTINUE
      CALL UTTRNS (U,N,NACT,ACCUM,R(1,NACT1),W)
      CALL ADDCOL (U,R,N,NACT,1,RIGHT,W,ACCUM,.TRUE.,EPS)
      IF (NACT.EQ.NACT1) GO TO 42
      KSTATF(NEWF)=-2
      GO TO 33
   42 KSTATF(NEWF)=1
      KSET(NACT)=NEWF+LELI
   43 EPS=EPS+SEPS
      IF (STEP.EQ.0.E0) GO TO 12
C
```

```
C        TAKE THE STEP AND UPDATE LINEAR FUNCTIONS
C
   44 FMAX=-1.E73
      XN2=0.E0
      DO 45 I=1,N
      X(I)=X(I)+STEP*DL(I)
      XN2=XN2+X(I)**2
   45 CONTINUE
      DO 46 I=1,M
      T=FUP(I)+STEP*DLDF(I)
      IF (T.GT.FMAX) FMAX=T
      FUP(I)=T
   46 CONTINUE
      IF (LELI.EQ.0) GO TO 48
      DO 47 I=1,LELI
      CUP(I)=CUP(I)+STEP*DLDC(I)
   47 CONTINUE
   48 IF (IFALL.EQ.0) GO TO 12
C
C        RETURN
C
   49 XN=SQRT(XN2)
      NACT=NACT+1
      KSET(NACT)=KSET0
      ASET(NACT)=ASET0
      RETURN
      END
C
C
      SUBROUTINE LINSYS (A,B,IDIM,N,NR,EPS)
C
C        THE SUBROUTINE SOLVES A SYSTEM OF LINEAR EQUATIONS
C        USING GAUSSIAN ELIMINATION.
C
      DIMENSION A(IDIM,IDIM), B(N)
      NR=0
C
C        A IS CONSIDERED TO BE OF RANK K-1 IF THE ABSOLUTE VALUE
C        OF THE K'TH PIVOT IS LESS THAN K*EPS.
C
      IF (N-1) 12,1,2
    1 IF (ABS(A(1,1)).LT.1.E-50) RETURN
      NR=1
      B(1)=B(1)/A(1,1)
      RETURN
C
C        EQUILIBRATION IN THE INFINITY NORM
C
    2 DO 4 I=1,N
      AM=ABS(A(I,1))
      DO 3 J=2,N
      S=ABS(A(I,J))
      IF (AM.LT.S) AM=S
    3 CONTINUE
      IF (AM.LT.1.E-50) AM=1.E0
      B(I)=B(I)/AM
      DO 4 J=1,N
    4 A(I,J)=A(I,J)/AM
C
C        ELIMINATION
C
      N1=N-1
      DO 9 K=1,N1
      NR=K-1
C
```

```
C        FIND PIVOTAL ROW
C
      AM=ABS(A(K,K))
      I0=K
      K1=K+1
      DO 5 I=K1,N
      S=ABS(A(I,K))
      IF (S.LE.AM) GO TO 5
      AM=S
      I0=I
    5 CONTINUE
      IF (AM.LT.2*K*EPS) RETURN
      IF (I0.EQ.K) GO TO 7
C
C        INTERCHANGE EQUATIONS K AND I0
C
      DO 6 J=K,N
      S=A(K,J)
      A(K,J)=A(I0,J)
      A(I0,J)=S
    6 CONTINUE
      S=B(K)
      B(K)=B(I0)
      B(I0)=S
C
C        STORE PIVOT IN AM AND ELIMINATE IN ROWS K+1 TO N.
C
    7 AM=A(K,K)
      DO 9 I=K1,N
      S=A(I,K)/AM
      DO 8 J=K1,N
      A(I,J)=A(I,J)-S*A(K,J)
    8 CONTINUE
    9 B(I)=B(I)-S*B(K)
      NR=N1
      IF (ABS(A(N,N)).LT.2*N*EPS) RETURN
C
C        A HAS FULL RANK
C
      NR=N
C
C        BACK SUBSTITUTION
C
      B(N)=B(N)/A(N,N)
      K=N
      DO 11 I=2,N
      K1=K
      K=K-1
      S=B(K)
      DO 10 J=K1,N
      S=S-A(K,J)*B(J)
   10 CONTINUE
      B(K)=S/A(K,K)
   11 CONTINUE
   12 RETURN
      END
C
C
      SUBROUTINE BFGS (B,N,Y,XX,W,SEPS)
C
C        UPDATES A HESSIAN APPROXIMATION USING BFGS-FORMULA.
C
      DIMENSION B(N,N), Y(N), XX(N), W(N)
      EPS=(N+10)*SEPS
      DO 2 I=1,N
```

```
      T=0.E0
      DO 1 J=1,N
      T=T+B(I,J)*XX(J)
    1 CONTINUE
      W(I)=T
    2 CONTINUE
      YXX=0.E0
      WXX=0.E0
      YN=0.E0
      XXN=0.E0
      WN=0.E0
      DO 3 I=1,N
      YN=YN+Y(I)**2
      XXN=XXN+XX(I)**2
      WN=WN+W(I)**2
      YXX=YXX+Y(I)*XX(I)
      WXX=WXX+W(I)*XX(I)
    3 CONTINUE
      YN=SQRT(YN)
      XXN=SQRT(XXN)
      WN=SQRT(WN)
      IF (YN.EQ.0.E0.OR.WN.EQ.0.E0.OR.XXN.EQ.0.E0) RETURN
      IF (ABS(YXX).LT.EPS*YN*XXN) RETURN
      IF (ABS(WXX).LT.EPS*WN*XXN) RETURN
      DO 4 I=1,N
      B(I,I)=B(I,I)+Y(I)**2/YXX-W(I)**2/WXX
    4 CONTINUE
      IF (N.EQ.1) RETURN
      DO 5 I=2,N
      I1=I-1
      DO 5 J=1,I1
      B(I,J)=B(I,J)+Y(I)*Y(J)/YXX-W(I)*W(J)/WXX
    5 B(J,I)=B(I,J)
      RETURN
      END
C
C
      SUBROUTINE ADDCOL (U,R,N,KCOL,KNEW,RIGHT,W,ACCUM,LRIGHT,EPS)
C
C        UPDATES HOUSEHOLDER FACTORIZATION.
C        THE NEW COLUMNS MUST HAVE BEEN TRANSFORMED
C        AS RIGHTHAND SIDES.
C
      DIMENSION U(N,N), R(N,N), RIGHT(N), W(N)
      LOGICAL ACCUM,LRIGHT
      K1=KCOL+1
      K2=KCOL+KNEW
C
C        COLUMN LOOP STARTS HERE
C
      DO 17 K=K1,K2
      S=0.E0
      T=0.E0
      IF (K.EQ.1) GO TO 2
      KK=K-1
      DO 1 I=1,KK
      T=T+R(I,K)**2
    1 CONTINUE
    2 DO 3 I=K,N
      S=S+R(I,K)**2
    3 CONTINUE
      T=T+S
      T=SQRT(T)
      S=SQRT(S)
C
```

```
C         RETURN IF THE NEW COLUMN DEPENDS LINEARLY ON THE
C         PRECEDING COLUMNS
C
        IF (T.EQ.0.E0) RETURN
        IF (S.LT.T*EPS) RETURN
C
C         PERFORM HOUSEHOLDER TRANSFORMATION
C
      TT=R(K,K)
      T=ABS(TT)
      ALFA=SQRT(S*(S+T))
      BETA=-SIGN(S,TT)
      R(K,K)=BETA
      W(K)=(TT-BETA)/ALFA
      IF (K.EQ.N) GO TO 8
      KK=K+1
      DO 4 I=KK,N
      W(I)=R(I,K)/ALFA
    4 CONTINUE
C
C        TRANSFORM THE REMAINING COLUMNS
C
      IF (K.EQ.K2) GO TO 8
      DO 7 J=KK,K2
      T=0.E0
      DO 5 I=K,N
      T=T+W(I)*R(I,J)
    5 CONTINUE
      DO 6 I=K,N
      R(I,J)=R(I,J)-T*W(I)
    6 CONTINUE
    7 CONTINUE
C
C         TRANSFORM THE RIGHTHAND SIDE
C
    8 IF (.NOT.LRIGHT) GO TO 11
      T=0.E0
      DO 9 I=K,N
      T=T+W(I)*RIGHT(I)
    9 CONTINUE
      DO 10 I=K,N
      RIGHT(I)=RIGHT(I)-T*W(I)
   10 CONTINUE
C
C         ACCUMULATE THE TRANSFORMATIONS IN U
C         U MUST HAVE BEEN INITIALIZED
C
   11 IF (ACCUM) GO TO 13
      DO 12 I=K,N
      U(I,K)=W(I)
   12 CONTINUE
      GO TO 17
   13 DO 16 I=1,N
      T=0.E0
      DO 14 J=K,N
      T=T+U(I,J)*W(J)
   14 CONTINUE
      DO 15 J=K,N
      U(I,J)=U(I,J)-T*W(J)
   15 CONTINUE
   16 CONTINUE
   17 KCOL=KCOL+1
      RETURN
      END
C
```

```
C
      SUBROUTINE DELCOL (K,U,R,N,KCOL,RIGHT,LRIGHT)
C
C       DELETES COLUMN NUMBER K IN THE FACTORIZED MATRIX.
C       K MUST SATISFY      1.LE.K.LE.KCOL
C       U MUST HAVE BEEN ACCUMULATED.
C
      DIMENSION U(N,N), R(N,N), RIGHT(N)
      LOGICAL LRIGHT
C
C       DELETE COLUMN NUMBER K
C
      KCOL=KCOL-1
      IF (K.GT.KCOL) RETURN
      DO 1 J=K,KCOL
      J1=J+1
      DO 1 I=1,J1
    1 R(I,J)=R(I,J1)
C
C       TRANSFORM TO UPPER TRIANGULAR FORM
C       USING STANDARD GIVENS TRANSFORMATIONS
C
      DO 6 KK=K,KCOL
      K1=KK+1
      X=R(KK,KK)
      Y=R(K1,KK)
      A=SQRT(X*X+Y*Y)
      C=X/A
      S=Y/A
      R(KK,KK)=C*X+S*Y
      IF (KK.EQ.KCOL) GO TO 3
      DO 2 J=K1,KCOL
      X=R(KK,J)
      Y=R(K1,J)
      R(KK,J)=C*X+S*Y
      R(K1,J)=C*Y-S*X
    2 CONTINUE
    3 IF (.NOT.LRIGHT) GO TO 4
      X=RIGHT(KK)
      Y=RIGHT(K1)
      RIGHT(KK)=C*X+S*Y
      RIGHT(K1)=C*Y-S*X
C
C       ACCUMULATE THE TRANSFORMATIONS
C
    4 DO 5 I=1,N
      X=U(I,KK)
      Y=U(I,K1)
      U(I,KK)=C*X+S*Y
      U(I,K1)=C*Y-S*X
    5 CONTINUE
    6 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE UTTRNS (U,N,KCOL,ACCUM,R,W)
C
C       TRANSFORM THE VECTOR R AS A RIGHTHAND SIDE.
C
      DIMENSION U(N,N), R(N), W(N)
      LOGICAL ACCUM
C
C       IF THE TRANSFORMATIONS HAVE BEEN ACCUMULATED
C       DO SIMPLE MATRIX-MULTIPLICATION
```

```
C        ELSE TRANSFORM RIGHTHAND SIDES
C
      IF (ACCUM) GO TO 4
      IF (KCOL.EQ.0) RETURN
      DO 3 K=1,KCOL
      T=0.E0
      DO 1 I=K,N
      T=T+R(I)*U(I,K)
    1 CONTINUE
      DO 2 I=K,N
      R(I)=R(I)-T*U(I,K)
    2 CONTINUE
    3 CONTINUE
      RETURN
    4 DO 5 I=1,N
      W(I)=R(I)
    5 CONTINUE
      DO 7 K=1,N
      T=0.E0
      DO 6 I=1,N
      T=T+U(I,K)*W(I)
    6 CONTINUE
      R(K)=T
    7 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE UTRNS (U,N,KCOL,ACCUM,R,W)
C
C        TRANSFORM THE VECTOR R OPPOSITE A RIGHTHAND SIDE.
C
      DIMENSION U(N,N), R(N), W(N)
      LOGICAL ACCUM
      K1=KCOL+1
      IF (ACCUM) GO TO 4
      IF (KCOL.EQ.0) RETURN
      DO 3 KK=1,KCOL
      K=K1-KK
      T=0.E0
      DO 1 J=K,N
      T=T+U(J,K)*R(J)
    1 CONTINUE
      DO 2 J=K,N
      R(J)=R(J)-T*U(J,K)
    2 CONTINUE
    3 CONTINUE
      RETURN
    4 DO 5 I=1,N
      W(I)=R(I)
    5 CONTINUE
      DO 7 I=1,N
      T=0.E0
      DO 6 J=1,N
      T=T+U(I,J)*W(J)
    6 CONTINUE
      R(I)=T
    7 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE RSOLV (R,N,KCOL,RIGHT,X)
C
C        PERFORM BACK SUBSTITUTION ON RIGHT.
```

```
C
      DIMENSION R(N,N), RIGHT(N), X(N)
C
C       CALCULATE ALFA USING BACK SUBSTITUTION ON R
C
      K=KCOL
      K1=K+1
    1 IF (K.EQ.0) RETURN
      T=RIGHT(K)
      IF (K1.GT.KCOL) GO TO 3
      DO 2 J=K1,KCOL
      T=T-X(J)*R(K,J)
    2 CONTINUE
    3 X(K)=T/R(K,K)
      K1=K
      K=K-1
      GO TO 1
      END
C
C
      SUBROUTINE RTSOLV (R,N,KCOL,RIGHT,X)
C
C       PERFORM BACK SUBSTITUTION ON RIGHT USING THE
C       TRANSPOSED TRIANGULAR MATRIX.
C
      DIMENSION R(N,N), RIGHT(N), X(N)
      IF (KCOL.EQ.0) RETURN
      X(1)=RIGHT(1)/R(1,1)
      IF (KCOL.EQ.1) RETURN
      DO 2 I=2,KCOL
      I1=I-1
      T=RIGHT(I)
      DO 1 J=1,I1
      T=T-X(J)*R(J,I)
    1 CONTINUE
      X(I)=T/R(I,I)
    2 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE HACCUM (U,N,KCOL,W)
C
C       ACCUMULATES HOUSEHOLDER VECTORS STORED IN LOWER TRIANGLE
C       OF THE FIRST KCOL COLUMNS OF U IN AN ORTHONORMAL MATRIX U.
C       THE HOUSEHOLDER VECTORS MUST HAVE TWO NORM EQUAL TO TWO.
C       KCOL.GE.1 .
C
      DIMENSION U(N,N), W(N)
C
C       INITIALIZE USING LAST TRANSFORMATION
C
      K1=KCOL+1
      DO 1 I=KCOL,N
      W(I)=U(I,KCOL)
    1 CONTINUE
      DO 2 I=KCOL,N
      U(I,I)=1.E0-W(I)**2
    2 CONTINUE
      IF (KCOL.EQ.N) GO TO 4
      DO 3 I=K1,N
      I1=I-1
      T=W(I)
      DO 3 J=KCOL,I1
      S=-T*W(J)
```

```
      U(I,J)=S
    3 U(J,I)=S
    4 IF (KCOL.EQ.1) RETURN
C
C       ACCUMULATE REMAINING TRANSFORMATIONS
C
      DO 10 KK=2,KCOL
      K=K1-KK
      DO 5 I=K,N
      W(I)=U(I,K)
    5 CONTINUE
      T=W(K)
      KP1=K+1
      U(K,K)=1.E0-T**2
      DO 6 I=KP1,N
      U(I,K)=-T*W(I)
    6 CONTINUE
      DO 9 L=KP1,N
      S=0.E0
      DO 7 I=KP1,N
      S=S+W(I)*U(I,L)
    7 CONTINUE
      U(K,L)=-T*S
      DO 8 I=KP1,N
      U(I,L)=U(I,L)-S*W(I)
    8 CONTINUE
    9 CONTINUE
   10 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE LIMIT (XNMAX2,X,XN2,P,PN2,ALFA,N)
C
C       LIMIT THE STEP LENGTH ALFA.
C
      DIMENSION X(N), P(N)
      XTP=0.E0
      DO 1 I=1,N
      XTP=XTP+X(I)*P(I)
    1 CONTINUE
      B=XTP/PN2
      T=SQRT(B*B+(XNMAX2-XN2)/PN2)
      AP=T-B
      AM=-T-B
      IF (ALFA.GT.AP) ALFA=AP
      IF (ALFA.LT.AM) ALFA=AM
      RETURN
      END
```

SOC-281

MMLA1Q - A FORTRAN PACKAGE FOR LINEARLY CONSTRAINED MINIMAX OPTIMIZATION

J. Hald (Adapted and Edited by J.W. Bandler and W.M. Zuberek)

December 1981, No. of Pages: 45

Revised:

Key Words: Minimax optimization, constrained optimization, nonlinear programming, computer-aided design, optimization program

Abstract: This report provides a user-oriented description of a program package written in Fortran IV for linearly constrained minimax optimization. The new subroutine MMLA1Q is very similar to MINLA1, which was presented by Madsen and Schjaer-Jacobsen, the main difference being that MMLA1Q accumulates and uses approximate second-order information as described by Hald and Madsen. Both routines require first-order partial derivatives of the nonlinear functions defining the minimax problem. The list of parameters is described herein, and a listing of the complete program package including the linear programming part is given. Instead of the revised simplex algorithm used in MINLA1, a reduced gradient algorithm has been developed. Finally, a couple of simple examples illustrate the use of the program. The program and documentation have been adapted for the CDC 170/730 system.

Description: Contains Fortran listing, user's manual. Source deck or magnetic tape available for $150.00. The listing contains 1544 lines, of which 358 are comments.

Related Work: SOC-218, SOC-280.

Price: $ 30.00.