

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

$$F(\underline{x}) = \max_{1 \leq i \leq m} f_i(\underline{x}) \quad (1)$$

subject to the constraints

$$\underline{c}_i^T \underline{x} + c_i = 0, \quad i = 1, 2, \dots, \ell_{eq}, \quad (2)$$

$$\underline{c}_i^T \underline{x} + c_i \geq 0, \quad i = \ell_{eq} + 1, \dots, \ell,$$

where  $\underline{c}_i^T$  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,  $\tilde{x}^*$ , 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  $\tilde{x}^*$  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  $f_1(\underline{x})$



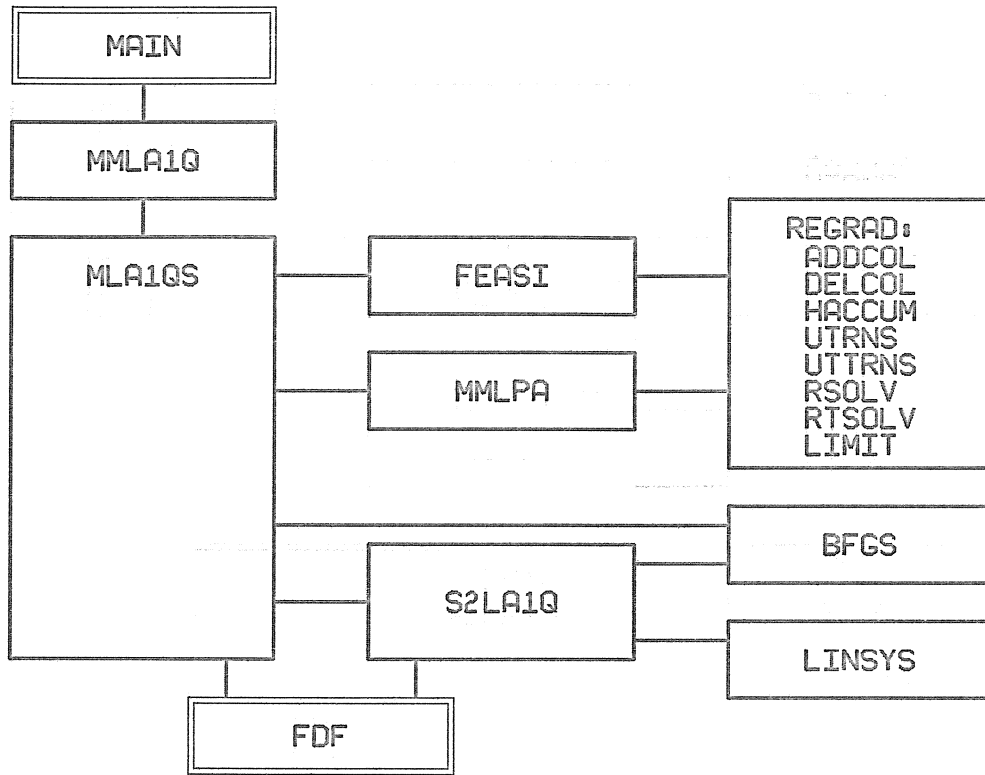


Fig. 1 Structure of the MMLA1Q package for nonlinear minimax optimization subject to linear equality and inequality constraints.

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

$$F(I) = f_I(\underline{x}), \quad I = 1, \dots, M,$$

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

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.,  $C(I) = c_I$ ,  $I = 1, \dots, L$ . 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  $\underline{x}$  in the linear constraints (2), i.e.,

$$(DC(I,1),DC(I,2),\dots,DC(I,N)) = \underline{c}'_I^T, \quad I = 1,\dots,L.$$

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,  $X(I) = x_I^0, I=1,\dots,N$ . On exit 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  $\underline{x}^0$  and the solution. But if more curvature is present, this value may be too large. Normally  $DX = 0.1 * \|\underline{x}^0\|$  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  $\|h_{\sim k}\| \leq EPS * \|\underline{x}_{\sim k}\|$ , where  $h_{\sim k}$  is the change given by the algorithm to the kth approximation  $\underline{x}_{\sim k}$ . 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. On exit 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. On exit 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. On exit 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, UTRNS, 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

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

subject to

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

where

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

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

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

For this example,

$$N = 2$$

$$M = 3$$

$$LEQ = 0$$

$$L = 1$$

$$IC = 1$$

$$IW = 67$$

The starting point is

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

The function has a minimum at

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

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("CPU TIME:",F6.3," SECONDS")
  WRITE(1,300) IFALL, EPS, MAXF, KEQS
300 FORMAT("0IFALL = ", I5/" DKNORM= ", 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, MD)
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, MD)
100 FORMAT(1X, I5, 2(F13.8, 1X), 1X, 3(1X, F13.8))
  NCOUNT=NCOUNT+1
  RETURN
  END
```

PROGRAM TIRML (J.HALD - EXAMPLE 1) END

	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: 0.068 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  $\underline{x}$  the objective function

$$F(\underline{x}) = \max_{1 \leq i \leq 163} |f_i(\underline{x})|,$$

where

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

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

$$x_7 = 3.5,$$

subject to linear constraints

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

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

$$F(\underline{x}) = \max_{1 \leq i \leq 326} f_i(\underline{x}),$$

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

For this example,

$$N = 6$$

$$M = 326$$

$$LEQ = 1$$

$$L = 8$$

$$IC = 8$$

$$IW = 5479$$

The starting point is

$$\tilde{x}^0 = \tilde{0}.$$

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).

```
C
C
C
PROGRAM T2RMML(OUTPUT,TAPE6=OUTPUT)
J.HALD - EXAMPLE 2.
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("CPU TIME:",F6.3," SECONDS")
WRITE(6,56) IFALL, EPS, MAXF, KEQS
56 FORMAT("IFALL =",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("SOLUTION:",6(/F20.10)/
1 "OFMAX AT THE SOLUTION:"/F20.10//)
STOP
END
C
C
C
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
```

C

```
SUBROUTINE FUNC(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
```

00000570  
00000580  
00000590  
00000600  
00000610  
00000620  
00000630  
00000640  
00000650  
00000660  
00000670  
00000680  
00000690  
00000700  
00000710  
00000720  
00000730  
00000740  
00000750  
00000760  
00000770  
00000780  
00000790  
00000800  
00000810  
00000820  
00000830  
00000840  
00000850  
00000860  
00000870

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

<u>Subroutine</u>	<u>Number of Lines</u> (source text)	<u>Number of Words</u> (compiled code)	<u>Listing from Page</u>
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,IFALL) 000010
C 000020
C 000030
C MMLA1Q MINIMIZES THE MAXIMUM VALUE OF A SET OF NONLINEAR 000040
C FUNCTIONS SUBJECT TO LINEAR EQUALITY AND INEQUALITY CONSTRAINTS. 000050
C DERIVATIVES REQUIRED. 000060
C 000070
C FOR A PROGRAM DESCRIPTION SEE: 000080
C J. HALD: "MMLA1Q, A FORTRAN SUBROUTINE FOR LINEARLY CONSTRAINED 000090
C MINIMAX OPTIMIZATION", REPORT NO. NI-81-1, INSTITUTE FOR NUMERICAL 000100
C ANALYSIS, TECHNICAL UNIVERSITY OF DENMARK, DK-2800 LYNGBY, DENMARK. 000110
C MARCH 1981. 000120
C 000130
C THE SUBROUTINES: MMLPA,FEASI,S2LA1Q,BFGS,ADDCOL,DELCOL,HACCUM, 000140
C UTRNS,UTTRNS,RSOLV,RTSOLV,LIMIT,LINSYS MUST BE AVAILABLE. 000150
C 000160
C DIMENSION C(IC), DC(IC,N), X(N), W(IW) 000170
C EXTERNAL FDF 000180
C 000190
C CHECK INPUT QUANTITIES 000200
C 000210
C IWR=2*M*N+5*N*N+4*M+8*N+4*IC+3 000220
C IFALL=-1 000230
C IF (IW.LT.IWR.OR.N.LT.1.OR.M.LT.1.OR.L.LT.0.OR.LEQ.LT.0.OR.LEQ.GT. 000240
C 1L.OR.LEQ.GT.N.OR.IC.LT.L.OR.DX.LE.0.E0.OR.EPS.LT.0.E0.OR.MAXF.LE.0 000250
C 2) RETURN 000260
C 000270
C SPLIT UP THE WORK AREA 000280
C 000290
C N1=N+1 000300
C NN=N+N 000310
C NF=1 000320
C NF1=NF+M 000330
C NDF=NF1+M 000340
C NDF1=NDF+M*N 000350
C NX1=NDF1+M*N 000360
C NB=NX1+N 000370
C NU=NB+N*N 000380
C NR=NU+N*N 000390
C NA=NU 000400
C NCL=NA+NN*N 000410
C NWL=NCL+IC 000420
C NWL1=NWL+IC 000430
C NXX=NWL1+IC 000440
C NW=NXX+N 000450
C NW1=NW+N 000460
C NW2=NW1+N 000470
C NWM=NW2+N 000480
C NAS=NWM+M 000490
C NKS=NAS+N1 000500
C NKS0=NKS+N1 000510
C NKSTC=NKS0+N1 000520
C NKSTF=NKSTC+IC 000530
C CALL MMLA1QS (FDF,N,M,L,LEQ,C,DC,IC,X,DX,EPS,MAXF,KEQS,N1,NN,W(NF), 000540
C 1W(NF1),W(NDF),W(NDF1),W(NX1),W(NB),W(NU),W(NR),W(NA),W(NCL),W(NWL) 000550
C 2,W(NWL1),W(NXX),W(NW),W(NW1),W(NW2),W(NW1),W(NAS),W(NKS),W(NKS0),W 000560
C 3(NKSTC),W(NKSTF),IFALL) 000570
C RETURN 000580
C END 000590
C 000600
C 000610
C SUBROUTINE MMLA1QS (FDF,N,M,L,LEQ,C,DC,IC,X,DX,EPS,MAXF,KEQS,N1,NN, 000620
C 1F,F1,DF,DF1,X1,B,U,R,A,CLOC,WL,WL1,XX,W,W1,W2,WM,ASET,KSET,KSET0,K 000630
C 2STATC,KSTATF,IFALL) 000640
C DIMENSION C(IC), DC(IC,N), X(N), F(MD), F1(MD), WM(MD), DF(M,N), DF1(
000650

```



```

1M,N), XX(NN), X1(N), U(N,N), R(N,N), B(N,N), A(NN,NN), CLOC(IC), W 000660
2L(IC), WL1(IC), W(N), W1(N), W2(N), ASET(N1) 000670
INTEGER KSET(N1), KSET0(N1), KSTATC(IC), KSTATF(M) 000680
LOGICAL DIV4, ACCUM, SHIFT 000690
EXTERNAL FDF 000700
C
C SEPS IS AN EXPRESSION FOR THE MACHINE ACCURACY 000710
C 000720
SEPS=16.E0**(-12) 000730
C 000740
C SET SOME CONSTANTS 000750
C 000760
DIV4=.FALSE. 000770
LI=L-LEQ 000780
C 000790
C INITIALIZE 000800
C 000810
KEQSET=0 000820
NCALL=0 000830
NSHIFT=0 000840
NSTEP=0 000850
FMMREF=1.E73 000860
DX0=DX 000880
DO 2 I=1,N 000890
DO 1 J=1,N 000900
B(I,J)=0.E0 000910
1 CONTINUE 000920
B(I,I)=1.E0 000930
2 CONTINUE 000940
C 000950
C FIND A FEASIBLE POINT 000960
C 000970
IF (L.EQ.0) GO TO 8 000980
DO 4 I=1,L 000990
T=C(I) 001000
DO 3 J=1,N 001010
T=T+DC(I,J)*X(J) 001020
3 CONTINUE 001030
CLOC(I)=T 001040
4 CONTINUE 001050
NACT=0 001060
CALL FEAS1 (CLOC, DC, IC, LEQ, LI, N, XX, NACT, KSET, ASET, U, R, W1, W2, WL, WL1
1, W, KSTATC, IFALL, ACCUM, SEPS) 001070
IF (IFALL.NE.0) RETURN 001080
DO 5 I=1,N 001090
X(I)=X(I)+XX(I) 001100
5 CONTINUE 001110
DO 7 I=1,L 001120
T=C(I) 001130
DO 6 J=1,N 001140
T=T+DC(I,J)*X(J) 001150
6 CONTINUE 001160
CLOC(I)=T 001170
7 CONTINUE 001180
C 001190
C 001200
C CALCULATE FUNCTION VALUES IN THE FIRST FEASIBLE POINT 001210
C 001220
8 CALL FDF (N, M, X, DF, F) 001230
FMM0=F(1) 001240
DO 9 I=1, M 001250
FMM0=AMAX1(FMM0, F(I)) 001260
9 CONTINUE 001270
NACT0=0 001280
NCALL=1 001290
C 001300

```

```
C          ITERATIVE LOOP STARTS HERE          001310
C
10 NACT=NACT0                                001320
   IF (NACT.EQ.0) GO TO 12                    001330
   DO 11 I=1,NACT                              001340
   KSET(I)=KSET0(I)                            001350
11 CONTINUE                                    001360
C
C          SOLVE THE LINEAR SUBPROBLEMS        001370
C
12 CALL MMLPA (F,DF,CLOC,DC,M,N,N1,IC,LEQ,LI,DX,XXN,XX,NACT,KSET,ASET 001380
   1,U,R,W1,W2,F1,WM,WL,WL1,KSTATF,KSTATC,W,SEPS,ACCUM,FMM,IFALL) 001390
   IF (FMM.GE.FMM0) GO TO 39                   001400
C
C          CALCULATE FUNCTION VALUES IN THE NEW POINT
C
   DO 13 I=1,N                                  001440
   X1(I)=X(I)+XX(I)                             001450
13 CONTINUE                                    001460
   CALL FDF (N,M,X1,DF1,F1)                    001470
   NCALL=NCALL+1                               001480
   FMM1=F1(1)                                   001490
   DO 14 I=1,M                                  001500
   FMM1=AMAX1(FMM1,F1(I))                      001510
14 CONTINUE                                    001520
C
C          REVISE THE STEP LENGTH              001530
C
   IF ((FMM0-FMM1).GT.(FMM0-FMM0/4.E0)) GO TO 15 001540
   DX=XXN/4.E0                                  001550
   DIV4=.TRUE.                                  001560
   GO TO 17                                      001570
15 IF (DIV4) GO TO 16                          001580
   IF ((FMM0-FMM1).GT.(FMM0-FMM0*.75E0)) DX=XXN*2.E0 001590
16 DIV4=.FALSE.                                001600
C
C          UPDATE THE HESSIAN APPROXIMATION    001610
C
17 DO 18 J=1,N                                  001620
   W(J)=0.E0                                    001630
   W1(J)=0.E0                                   001640
18 CONTINUE                                    001650
   DO 20 I=1,NACT                              001660
   K=KSET(I)                                    001670
   IF (K.LE.L) GO TO 20                        001680
   KK=K-L                                       001690
   T=-ASET(I)                                  001700
   DO 19 J=1,N                                  001710
   W1(J)=W1(J)+T*DF1(KK,J)                    001720
   W(J)=W(J)+T*DF(KK,J)                      001730
19 CONTINUE                                    001740
20 CONTINUE                                    001750
   DO 21 I=1,N                                  001760
   W2(I)=W1(I)-W(I)                            001770
21 CONTINUE                                    001780
   CALL BFGS (B,N,W2,XX,W,SEPS)                001790
C
C          TEST IF THE NEW POINT IS ACCEPTABLE
C
   IF ((FMM0-FMM1).LE.0.01E0*(FMM0-FMM0)) GO TO 31 001800
C
C          COMPARE THE NEW ACTIVE SET WITH THE PRECEDING
C
   IF (NACT0.NE.NACT) GO TO 24                001810
   DO 23 I=1,NACT                              001820
   001830
   001840
   001850
   001860
   001870
   001880
   001890
   001900
   001910
   001920
   001930
   001940
   001950
```

```

      K=KSET(I)
      DO 22 J=1,NACT
      IF (K.EQ.KSET(J)) GO TO 23
22  CONTINUE
      GO TO 24
23  CONTINUE
      KEQSET=KEQSET+1
      GO TO 25
24  KEQSET=1
C
C      INTRODUCE THE NEWPOINT
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 (XN.LE.EPS*XN) GO TO 40
      IF (XN.LE.SEPS*XN) GO TO 39
C
C      TEST FOR SWITCH TO STAGE-2
C
      SHIFT=FMM0.LE.FMMREF.AND.KEQSET.GE.KEQSHAND.NSTEP.GE.NACT0
      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
FMM0=FM0-10.E0*SEPS*ABS(FM0)
XXNMAX=AMAX1(DX,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
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)
KKO=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                                003260
  K=KSET(I)                                     003270
  IF (K.LE.L) KSTATC(K)=1                       003280
  IF (K.GT.L) KSTATF(K-L)=1                     003290
4 CONTINUE                                       003300
C                                               003310
C   ITERATIVE LOOP STARTS HERE AND             003320
C                                               003330
C   SET UP THE ITERATION MATRIX AND THE        003340
C   RIGHTHAND SIDE                             003350
5 DO 7 I=1, N                                    003360
  DO 6 J=1, N                                    003370
  DZ(I, J)=B(I, J)                              003380
6 CONTINUE                                       003390
  ZZ(I)=-DF(KK0, I)                             003400
7 CONTINUE                                       003410
  IF (NACT.EQ.1) GO TO 15                       003420
  DO 14 J=1, NACT1                              003430
  K=KSET(J)                                      003440
  JN=J+N                                         003450
  IF (K.GT.L) GO TO 9                            003460
  ZZ(JN)=-CLOC(K)                               003470
  DO 8 I=1, N                                    003480
  DZ(I, JN)=DC(K, I)                           003490
  DZ(JN, I)=DZ(I, JN)                          003500
8 CONTINUE                                       003510
  GO TO 11                                       003520
9 KK=K-L                                         003530
  ZZ(JN)=F(KK)-F(KK0)                          003540
  DO 10 I=1, N                                   003550
  DZ(I, JN)=DF(KK0, I)-DF(KK, I)               003560
  DZ(JN, I)=DZ(I, JN)                          003570
10 CONTINUE                                     003580
11 DO 12 I=N1, NZ                               003590
  DZ(I, JN)=0.E0                                003600
12 CONTINUE                                     003610
  T=ASET(J)                                     003620
  DO 13 I=1, N                                   003630
  ZZ(I)=ZZ(I)-T*DZ(JN, I)                      003640
13 CONTINUE                                     003650
14 CONTINUE                                     003660
15 RES0=0.E0                                     003670
  DO 16 I=1, NZ                                  003680
  RES0=RES0+ZZ(I)**2                            003690
16 CONTINUE                                     003700
  RES0=SQRT(RES0)                               003710
C                                               003720
C   CALCULATE THE QUASI-NEWTON STEP           003730
C                                               003740
C   CALL LINSYS (DZ, ZZ, NN, NZ, K, SEPS)     003750
C   IF (K.EQ.NZ) GO TO 17                       003760
C   IFALL=3                                     003770
C   RETURN                                      003780
C                                               003790
C   CONTROL STEP LENGTH                       003800
C                                               003810
17 XXN1=0.E0                                     003820
  ALFA=1.E0                                     003830
  DO 18 I=1, N                                   003840
  XXN1=XXN1+ZZ(I)**2                            003850
18 CONTINUE                                     003860
  XXN1=SQRT(XXN1)                               003870
  IF (XXN1.GT.XXNMAX) ALFA=XXNMAX/XXN1         003880
C                                               003890
C   WILL OTHER CONSTRAINTS OR FUNCTIONS      003900

```

```
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
XXN1=0.E0
DO 27 I=1,N
X1(I)=X(I)+ZZ(I)
XXN1=XXN1+X1(I)**2
27 CONTINUE
XXN1=SQRT(XXN1)
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) 004560
JN=J+N 004570
T=ASET(J)+ZZ(JN) 004580
IF (K.GT.L) GO TO 32 004590
S=C(K) 004600
DO 31 I=1,N 004610
SS=DC(K,I) 004620
W(I)=W(I)+T*SS 004630
W1(I)=W1(I)+T*SS 004640
S=S+SS*X1(I) 004650
31 CONTINUE 004660
RES=RES+S**2 004670
GO TO 34 004680
32 KK=K-L 004690
DO 33 I=1,N 004700
W(I)=W(I)-T*DF(KK,I) 004710
W1(I)=W1(I)-T*DF1(KK,I) 004720
33 CONTINUE 004730
RES=RES+(F1(KK0)-F1(KK))**2 004740
34 CONTINUE 004750
35 DO 36 I=1,N 004760
RES=RES+W1(I)**2 004770
36 CONTINUE 004780
RES=SQRT(RES) 004790
004800
C UPDATE THE HESSIAN APPROXIMATION 004810
C 004820
C DO 37 I=1,N 004830
C W1(I)=W1(I)-W(I) 004840
37 CONTINUE 004850
CALL BFCS (B,N,W1,ZZ,W,SEPS) 004860
004870
C TEST IF THE RESIDUAL HAS DECREASED 004880
C 004890
C IF (NSTEP2.EQ.0) GO TO 39 004900
C IF (RES.LE.0.999E0*RES0) GO TO 39 004910
C IF (IFALL.EQ.3) RETURN 004920
004930
C IF NO - TEST FOR MACHINE ACCURACY 004940
C 004950
C IF (XXN1.GT.SSEPS*(XXNMAX+XXN1).OR.NSTEP2.LT.2) GO TO 38 004960
C IFALL=2 004970
C RETURN 004980
38 IFALL=5 004990
C RETURN 005000
005010
C IF YES - INTRODUCE THE NEW POINT 005020
C 005030
39 NSTEP2=NSTEP2+1 005040
NSTEP=NSTEP+1 005050
XN=0.E0 005060
DO 40 I=1,N 005070
X(I)=X1(I) 005080
DO 40 J=1,M 005090
40 DF(J,I)=DF1(J,I) 005100
XN=XN1 005110
XXN=XXN1 005120
FMAX=-1.E73 005130
DO 41 I=1,M 005140
T=F1(I) 005150
FMAX=AMAX1(T,FMAX) 005160
F(I)=T 005170
41 CONTINUE 005180
ASET(NACT)=ASET(NACT)+DASET0 005190
IF (ASET(NACT).GT.0.E0) IFALL=6 005200

```

```

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*XXN) GO TO 48
IF (NACT.LT.N1) IFALL=1
RETURN
48 IF (XXN.GT.SEPS*XXN.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,V,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

```

005210  
005220  
005230  
005240  
005250  
005260  
005270  
005280  
005290  
005300  
005310  
005320  
005330  
005340  
005350  
005360  
005370  
005380  
005390  
005400  
005410  
005420  
005430  
005440  
005450  
005460  
005470  
005480  
005490  
005500  
005510  
005520  
005530  
005540  
005550  
005560  
005570  
005580  
005590  
005600  
005610  
005620  
005630  
005640  
005650  
005660  
005670  
005680  
005690  
005700  
005710  
005720  
005730  
005740  
005750  
005760  
005770  
005780  
005790  
005800  
005810  
005820  
005830  
005840  
005850





15	CONTINUE	006510
	IF (FMIN.GE.0.E0) RETURN	006520
	DO 16 I=1,N	006530
	RIGHT(I)=DC(NEW,I)	006540
16	CONTINUE	006550
	CALL UTRNS (U,N,NACT,ACCUM,RIGHT,W)	006560
	KSTAT(NEW)=1	006570
C		006580
C	CALCULATE MULTIPLIERS FOR THE NEW ACTIVE CONSTRAINT AND DROP	006590
C	CONSTRAINTS WITH POSITIVE MULTIPLIERS IN ORDER TO ACHIEVE	006600
C	THE DIRECTION OF STEEPEST INCREMENT	006610
C		006620
17	IF (NACT.EQ.LE) GO TO 22	006630
	CALL RSOLV (R,N,NACT,ASET)	006640
	AMAX=-1.E73	006650
	DO 18 I=LE1,NACT	006660
	IF (ASET(I).LT.AMAX) GO TO 18)	006670
	K=I	006680
	AMAX=ASET(I)	006690
18	CONTINUE	006700
	IF (AMAX.LT.0.E0) GO TO 22	006710
	KSTAT(KSET(K))=0	006720
	DO 19 I=LE1,LELI	006730
	IF (KSTAT(I).EQ.-2) (KSTAT(I)=0)	006740
19	CONTINUE	006750
	IF (ACCUM) GO TO 20)	006760
	ACCUM=.TRUE.	006770
	CALL HACCUM (U,N,NACT,W)	006780
20	CALL DELCOL (K,U,R,N,NACT,RIGHT,.TRUE.)	006790
	IF (K.GT.NACT) GO TO 17	006800
	DO 21 I=K,NACT	006810
	KSET(I)=KSET(I+1)	006820
21	CONTINUE	006830
	GO TO 17	006840
C		006850
C	CALCULATE THE PROJECTED GRADIENT	006860
C		006870
22	T=0.E0	006880
	DLN2=0.E0	006890
	IF (NACT.EQ.0) GO TO 24	006900
	DO 23 I=1,NACT	006910
	T=T+RIGHT(I)**2	006920
	DL(I)=0.E0	006930
23	CONTINUE	006940
24	NACT1=NACT+1	006950
	IF (NACT.EQ.N) GO TO 26	006960
	DO 25 I=NACT1,N	006970
	DLN2=DLN2+RIGHT(I)**2	006980
	DL(I)=RIGHT(I)	006990
25	CONTINUE	007000
26	T=T+DLN2	007010
	IF (T.GT.0.E0.AND.DLN2.GT.EPS*EPS*T) GO TO 28	007020
	S=(N+1)*ABS(C(NEW))	007030
	DO 27 I=1,N	007040
	S=S+ABS(DC(NEW,I)*K(I))*(N+3-I)	007050
27	CONTINUE	007060
	IF (CUP(NEW).LT.-EPS*S) GO TO 41	007070
	KSTAT(NEW)=0	007080
	GO TO 14	007090
28	CALL UTRNS (U,N,NACT,ACCUM,DL,W)	007100
C		007110
C	PROJECT GRADIENTS ON THE PROJECTED GRADIENT	007120
C		007130
	DO 30 I=LE1,LELI	007140
	T=0.E0	007150

```

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 CONSIDERATIONS
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
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
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
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

```

007160  
007170  
007180  
007190  
007200  
007210  
007220  
007230  
007240  
007250  
007260  
007270  
007280  
007290  
007300  
007310  
007320  
007330  
007340  
007350  
007360  
007370  
007380  
007390  
007400  
007410  
007420  
007430  
007440  
007450  
007460  
007470  
007480  
007490  
007500  
007510  
007520  
007530  
007540  
007550  
007560  
007570  
007580  
007590  
007600  
007610  
007620  
007630  
007640  
007650  
007660  
007670  
007680  
007690  
007700  
007710  
007720  
007730  
007740  
007750  
007760  
007770  
007780  
007790  
007800

```

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
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)
C      INTEGER KSET(N1),KSTATF(M),KSTATC(IC)
C      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
   KSET=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

```

```

007810
007820
007830
007840
007850
007860
007870
007880
007890
007900
007910
007920
007930
007940
007950
007960
007970
007980
007990
008000
008010
008020
008030
008040
008050
008060
008070
008080
008090
008100
008110
008120
008130
008140
008150
008160
008170
008180
008190
008200
008210
008220
008230
008240
008250
008260
008270
008280
008290
008300
008310
008320
008330
008340
008350
008360
008370
008380
008390
008400
008410
008420
008430
008440
008450

```

	KK=KSET(K)	008460
	IF (KK.LT.LE1.OR.KK.GT.LELI) GO TO 9	008470
	NACT1=NACT+1	008480
	IF (NACT1.GT.N) GO TO 10	008490
	DO 8 I=1,N	008500
	R(I,NACT1)=DC(KK,I)	008510
8	CONTINUE	008520
	CALL UTRRNS (U,N,NACT,ACCUM,R(1,NACT1),W)	008530
	CALL ADDCOL (U,R,N,NACT,1,RIGHT,W,ACCUM, FALSE.,EPS)	008540
	IF (NACT.LT.NACT1) GO TO 9	008550
	EPS=EPS+SEPS	008560
	KSET(NACT1)=KK	008570
	KSTATC(KK)=1	008580
9	CONTINUE	008590
C		008600
C	TRANSFORM OBJECTIVE FUNCTION GRADIENT	008610
C		008620
10	KSTATF(KSET0)=1	008630
	DO 11 J=1,N	008640
	RIGHT(J)=-DF(KSET0,J)	008650
11	CONTINUE	008660
	KSET0=KSET0+LELI	008670
	CALL UTRRNS (U,N,NACT,ACCUM,RIGHT,W)	008680
C		008690
C	ITERATIVE LOOP	008700
C		008710
C	CALCULATE MULTIPLIERS AND FIND THE LARGEST	008720
C		008730
12	ASET0=-1.E0	008740
	IF (NACT.EQ.0) GO TO 24	008750
	CALL RSOLV (R,N,NACT,RIGHT,ASET)	008760
	IF (NACT.EQ.LE) GO TO 24	008770
	AMAX=-1.E73	008780
	DO 13 I=LE1,NACT	008790
	IF (KSET(I).GT.LELI) ASET0=ASET0-ASET(I)	008800
	IF (ASET(I).LE.AMAX) GO TO 13	008810
	K=I	008820
	AMAX=ASET(I)	008830
13	CONTINUE	008840
	IF (AMAX.LT.0.E0.AND.ASET0.LT.0.E0) GO TO 24	008850
	IF (AMAX.GT.ASET0) GO TO 18	008860
C		008870
C	CHANGE OBJECTIVE FUNCTION	008880
C		008890
	DO 14 I=LE1,NACT	008900
	IF (KSET(I).LE.LELI) GO TO 14	008910
	K=I	008920
	GO TO 15	008930
14	CONTINUE	008940
15	DO 17 I=1,K	008950
	T=R(I,K)	008960
	IF (K.EQ.NACT) GO TO 17	008970
	K1=K+1	008980
	DO 16 J=K1,NACT	008990
	IF (KSET(J).GT.LELI) R(I,J)=R(I,J)-T	009000
16	CONTINUE	009010
17	RIGHT(I)=RIGHT(I)+T	009020
	KK=KSET0	009030
	KSET0=KSET(K)	009040
	KSET(K)=KK	009050
C		009060
C	DELETE ACTIVE CONSTRAINT NUMBER K	009070
C		009080
18	KK=KSET(K)	009090
	IF (KK.GT.LELI) KSTATF(KK-LELI)=0	009100

```
IF (KK.LE.LELI) KSTATC(KK)=0 009110
IF (ACCUM) GO TO 19 009120
ACCUM=.TRUE. 009130
CALL HACCU (U,N,NACT,W) 009140
19 CALL DELCOL (K,U,R,N,NACT,RIGHT,.TRUE.) 009150
EPS=EPS+SEPS 009160
IF (K.GT.NACT) GO TO 21 009170
DO 20 I=K,NACT 009180
KSET(I)=KSET(I+1) 009190
20 CONTINUE 009200
C 009210
C DELETE LINEAR DEPENDENCE LABELS 009220
C 009230
21 DO 22 I=1,M 009240
IF (KSTATF(I).EQ.-2) KSTATF(I)=0 009250
22 CONTINUE 009260
IF (LI.EQ.0) GO TO 12 009270
DO 23 I=LE1,LELI 009280
IF (KSTATC(I).EQ.-2) KSTATC(I)=0 009290
23 CONTINUE 009300
GO TO 12 009310
C 009320
C IS THERE AN UNBOUNDED SOLUTION ? 009330
C 009340
24 IF (NACT.EQ.N) GO TO 49 009350
C 009360
C CALCULATE THE PROJECTED GRADIENTS 009370
C 009380
K=NACT+1 009390
T=0.E0 009400
DLN2=0.E0 009410
DO 25 I=K,N 009420
DLN2=DLN2+RIGHT(I)**2 009430
DL(I)=RIGHT(I) 009440
25 CONTINUE 009450
IF (K.EQ.1) GO TO 27 009460
DO 26 I=1,NACT 009470
T=T+RIGHT(I)**2 009480
DL(I)=0.E0 009490
26 CONTINUE 009500
27 T=T+DLN2 009510
IF (T.GT.0.E0.AND.DLN2.GT.EPS*EPS*T) GO TO 28 009520
IFALL=2 009530
GO TO 49 009540
28 CALL UTRNS (U,N,NACT,ACCUM,DL,W) 009550
C 009560
C PROJECT GRADIENTS ON THE PROJECTED GRADIENTS 009570
C 009580
DO 30 I=1,M 009590
T=0.E0 009600
DO 29 J=1,N 009610
T=T+DL(J)*DF(I,J) 009620
29 CONTINUE 009630
DLDF(I)=T 009640
30 CONTINUE 009650
IF (LELI.EQ.0) GO TO 33 009660
DO 32 I=1,LELI 009670
T=0.E0 009680
DO 31 J=1,N 009690
T=T+DL(J)*DC(I,J) 009700
31 CONTINUE 009710
DLDC(I)=T 009720
32 CONTINUE 009730
C 009740
C CALCULATE STEP LENGTH 009750
```

```
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=-GUP(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 NO
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,R01,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,R01,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
009760
009770
009780
009790
009800
009810
009820
009830
009840
009850
009860
009870
009880
009890
009900
009910
009920
009930
009940
009950
009960
009970
009980
009990
010000
010010
010020
010030
010040
010050
010060
010070
010080
010090
010100
010110
010120
010130
010140
010150
010160
010170
010180
010190
010200
010210
010220
010230
010240
010250
010260
010270
010280
010290
010300
010310
010320
010330
010340
010350
010360
010370
010380
010390
010400
```

```
C      TAKE THE STEP AND UPDATE LINEAR FUNCTIONS 010410
C
44 FMAX=-1.E73 010420
   XN2=0.E0 010430
   DO 45 I=1,N 010440
   X(I)=X(I)+STEP*DL(I) 010450
   XN2=XN2+X(I)**2 010460
45 CONTINUE 010470
   DO 46 I=1,M 010480
   T=FUP(I)+STEP*DLDF(I) 010490
   IF (T.GT.FMAX) FMAX=T 010500
   FUP(I)=T 010510
46 CONTINUE 010520
   IF (LELI.EQ.0) GO TO 48 010530
   DO 47 I=1,LELI 010540
   CUP(I)=CUP(I)+STEP*DLDC(I) 010550
47 CONTINUE 010560
48 IF (IFALL.EQ.0) GO TO 12 010570
C
C      RETURN 010580
C
C
49 XN=SQRT(XN2) 010590
   NACT=NACT+1 010600
   KSET(NACT)=KSET0 010610
   ASET(NACT)=ASET0 010620
   RETURN 010630
   END 010640
C
C
C      SUBROUTINE LINSYS (A,B, IDIM,N,NR, EPS) 010650
C
C      THE SUBROUTINE SOLVES A SYSTEM OF LINEAR EQUATIONS 010660
C      USING GAUSSIAN ELIMINATION. 010670
C
C      DIMENSION A(IDIM, IDIM), B(N) 010680
C      NR=0 010690
C
C      A IS CONSIDERED TO BE OF RANK K-1 IF THE ABSOLUTE VALUE 010700
C      OF THE K'TH PIVOT IS LESS THAN K*EPS. 010710
C
C      IF (N-1) 12,1,2 010720
C      1 IF (ABS(A(1,1)).LT.1.E-50) RETURN 010730
C      NR=1 010740
C      B(1)=B(1)/A(1,1) 010750
C      RETURN 010760
C
C      EQUILIBRATION IN THE INFINITY NORM 010770
C
C      2 DO 4 I=1,N 010780
C      AM=ABS(A(I,1)) 010790
C      DO 3 J=2,N 010800
C      S=ABS(A(I,J)) 010810
C      IF (AM.LT.S) AM=S 010820
C      3 CONTINUE 010830
C      IF (AM.LT.1.E-50) AM=1.E0 010840
C      B(I)=B(I)/AM 010850
C      DO 4 J=1,N 010860
C      4 A(I,J)=A(I,J)/AM 010870
C
C      ELIMINATION 010880
C
C      N1=N-1 010890
C      DO 9 K=1,N1 010900
C      NR=K-1 010910
C
C      9
```



```
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=NR+1
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
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 21
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)
011710
011720
011730
011740
011750
011760
011770
011780
011790
011800
011810
011820
011830
011840
011850
011860
011870
011880
011890
011900
011910
011920
011930
011940
011950
011960
011970
011980
011990
012000
012010
012020
012030
012040
012050
012060
012070
012080
012090
012100
012110
012120
012130
012140
012150
012160
012170
012180
012190
012200
012210
012220
012230
012240
012250
012260
012270
012280
012290
012300
012310
012320
012330
012340
012350

```

```
C      RETURN IF THE NEW COLUMN DEPENDS LINEARLY ON THE PRECEDING COLUMNS 012360
C      012370
C      012380
IF (T.EQ.O.EO) RETURN 012390
IF (S.LT.T*EPS) RETURN 012400
C      012410
C      PERFORM HOUSEHOLDER TRANSFORMATION 012420
C      012430
TT=R(K,K) 012440
T=ABS(TT) 012450
ALFA=SQRT(S*(S+T)) 012460
BETA=-SIGN(S,TT) 012470
R(K,K)=BETA 012480
W(K)=(TT-BETA)/ALFA 012490
IF (K.EQ.N) GO TO 8 012500
KK=K+1 012510
DO 4 I=KK,N 012520
W(I)=R(I,K)/ALFA 012530
4 CONTINUE 012540
C      012550
C      TRANSFORM THE REMAINING COLUMNS 012560
C      012570
IF (K.EQ.K2) GO TO 8 012580
DO 7 J=KK,K2 012590
T=O.EO 012600
DO 5 I=K,N 012610
T=T+W(I)*R(I,J) 012620
5 CONTINUE 012630
DO 6 I=K,N 012640
R(I,J)=R(I,J)-T*W(I) 012650
6 CONTINUE 012660
7 CONTINUE 012670
C      012680
C      TRANSFORM THE RIGHTHAND SIDE 012690
C      012700
8 IF (.NOT.LRIGHT) GO TO 11 012710
T=O.EO 012720
DO 9 I=K,N 012730
T=T+W(I)*RIGHT(I) 012740
9 CONTINUE 012750
DO 10 I=K,N 012760
RIGHT(I)=RIGHT(I)-T*W(I) 012770
10 CONTINUE 012780
C      012790
C      ACCUMULATE THE TRANSFORMATIONS IN U 012800
C      U MUST HAVE BEEN INITIALIZED. 012810
C      012820
11 IF (ACCUM) GO TO 13 012830
DO 12 I=K,N 012840
U(I,K)=W(I) 012850
12 CONTINUE 012860
GO TO 17 012870
13 DO 16 I=1,N 012880
T=O.EO 012890
DO 14 J=K,N 012900
T=T+U(I,J)*W(J) 012910
14 CONTINUE 012920
DO 15 J=K,N 012930
U(I,J)=U(I,J)-T*W(J) 012940
15 CONTINUE 012950
16 CONTINUE 012960
17 KCOL=KCOL+1 012970
RETURN 012980
END 012990
C      013000
```

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

C	ELSE TRANSFORM RIGHTHAND SIDES	013660
C	IF (ACCUM) GO TO 4	013670
	IF (KCOL.EQ.0) RETURN	013680
	DO 3 K=1, KCOL	013690
	T=0.E0	013700
	DO 1 I=K, N	013710
	T=T+R(I)*U(I, K)	013720
1	CONTINUE	013730
	DO 2 I=K, N	013740
	R(I)=R(I)-T*U(I, K)	013750
2	CONTINUE	013760
3	CONTINUE	013770
	RETURN	013780
4	DO 5 I=1, N	013790
	W(I)=R(I)	013800
5	CONTINUE	013810
	DO 7 K=1, N	013820
	T=0.E0	013830
	DO 6 I=1, N	013840
	T=T+U(I, K)*W(I)	013850
6	CONTINUE	013860
	R(K)=T	013870
7	CONTINUE	013880
	RETURN	013890
	END	013900
C		013910
C		013920
	SUBROUTINE UTRNS (U, N, KCOL, ACCUM, R, W)	013930
C		013940
C	TRANSFORM THE VECTOR R OPPOSITE A RIGHTHAND SIDE.	013950
C		013960
C		013970
	DIMENSION U(N, N), R(N), W(N)	013980
	LOGICAL ACCUM	013990
	K1=KCOL+1	014000
	IF (ACCUM) GO TO 4	014010
	IF (KCOL.EQ.0) RETURN	014020
	DO 3 KK=1, KCOL	014030
	K=K1-KK	014040
	T=0.E0	014050
	DO 1 J=K, N	014060
	T=T+U(J, K)*R(J)	014070
1	CONTINUE	014080
	DO 2 J=K, N	014090
	R(J)=R(J)-T*U(J, K)	014100
2	CONTINUE	014110
3	CONTINUE	014120
	RETURN	014130
4	DO 5 I=1, N	014140
	W(I)=R(I)	014150
5	CONTINUE	014160
	DO 7 I=1, N	014170
	T=0.E0	014180
	DO 6 J=1, N	014190
	T=T+U(I, J)*W(J)	014200
6	CONTINUE	014210
	R(I)=T	014220
7	CONTINUE	014230
	RETURN	014240
	END	014250
C		014260
C		014270
	SUBROUTINE RSOLV (R, N, KCOL, RIGHT, X)	014280
C		014290
C	PERFORM BACK SUBSTITUTION ON RIGHT.	014300

C		014310
C	DIMENSION R(N,N), RIGHT(N), X(N)	014320
C		014330
C	CALCULATE ALFA USING BACK SUBSTITUTION ON R	014340
C		014350
	K=KCOL	014360
	K1=K+1	014370
	1 IF (K.EQ.0) RETURN	014380
	T=RIGHT(K)	014390
	IF (K1.GT.KCOL) GO TO 3	014400
	DO 2 J=K1,KCOL	014410
	T=T-X(J)*R(K,J)	014420
	2 CONTINUE	014430
	3 X(K)=T/R(K,K)	014440
	K1=K	014450
	K=K-1	014460
	GO TO 1	014470
	END	014480
C		014490
C		014500
C	SUBROUTINE RTSOLV (R,N,KCOL,RIGHT,X)	014510
C		014520
C	PERFORM BACK SUBSTITUTION ON RIGHT USING THE	014530
C	TRANSPOSED TRIANGULAR MATRIX.	014540
C		014550
	DIMENSION R(N,N), RIGHT(N), X(N)	014560
	IF (KCOL.EQ.0) RETURN	014570
	X(1)=RIGHT(1)/R(1,1)	014580
	IF (KCOL.EQ.1) RETURN	014590
	DO 2 I=2,KCOL	014600
	I1=I-1	014610
	T=RIGHT(I)	014620
	DO 1 J=1,I1	014630
	T=T-X(J)*R(J,I)	014640
	1 CONTINUE	014650
	X(I)=T/R(I,I)	014660
	2 CONTINUE	014670
	RETURN	014680
	END	014690
C		014700
C		014710
C	SUBROUTINE HACCUM (U,N,KCOL,W)	014720
C		014730
C	ACCUMULATES HOUSEHOLDER VECTORS STORED IN LOWER TRIANGLE	014740
C	OF THE FIRST KCOL COLUMNS OF U IN AN ORTHONORMAL MATRIX U.	014750
C	THE HOUSEHOLDER VECTORS MUST HAVE TWO NORM EQUAL TO TWO.	014760
C	KCOL.GE.1	014770
C		014780
	DIMENSION U(N,N), W(N)	014790
C		014800
C	INITIALIZE USING LAST TRANSFORMATION	014810
C		014820
	K1=KCOL+1	014830
	DO 1 I=KCOL,N	014840
	W(I)=U(I,KCOL)	014850
	1 CONTINUE	014860
	DO 2 I=KCOL,N	014870
	U(I,I)=1.E0-W(I)**2	014880
	2 CONTINUE	014890
	IF (KCOL.EQ.N) GO TO 4	014900
	DO 3 I=K1,N	014910
	I1=I-1	014920
	T=W(I)	014930
	DO 3 J=KCOL,I1	014940
	S=-T*W(J)	014950

```
      U(I,J)=S      014960
3  U(J,I)=S      014970
4  IF (KCOL.EQ.1) RETURN      014980
C      ACCUMULATE REMAINING TRANSFORMATIONS      014990
C      015000
C      DO 10 KK=2,KCOL      015010
      K=K1-KK      015020
      DO 5 I=K,N      015030
      W(I)=U(I,K)      015040
5  CONTINUE      015050
      T=W(K)      015070
      KP1=K+1      015080
      U(K,K)=1.E0-T**2      015090
      DO 6 I=KP1,N      015100
      U(I,K)=-T*W(I)      015110
6  CONTINUE      015120
      DO 9 L=KP1,N      015130
      S=0.E0      015140
      DO 7 I=KP1,N      015150
      S=S+W(I)*U(I,L)      015160
7  CONTINUE      015170
      U(K,L)=-T*S      015180
      DO 8 I=KP1,N      015190
      U(I,L)=U(I,L)-S*W(I)      015200
8  CONTINUE      015210
9  CONTINUE      015220
10 CONTINUE      015230
      RETURN      015240
      END      015250
C      015260
C      015270
C      SUBROUTINE LIMIT (XNMAX2,X,XN2,P,PN2,ALFA,N)      015280
C      015290
C      LIMIT THE STEP LENGTH ALFA.      015300
C      015310
C      DIMENSION X(N), P(N)      015320
      XTP=0.E0      015330
      DO 1 I=1,N      015340
      XTP=XTP+X(I)*P(I)      015350
1  CONTINUE      015360
      B=XTP/PN2      015370
      T=SQRT(B*B+(XNMAX2-XN2)/PN2)      015380
      AP=T-B      015390
      AM=-T-B      015400
      IF (ALFA.GT.AP) ALFA=AP      015410
      IF (ALFA.LT.AM) ALFA=AM      015420
      RETURN      015430
      END      015440
```

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.