



Munich Personal RePEc Archive

NLINLS: a Differential Evolution based nonlinear least squares Fortran 77 program

Mishra, SK

North-Eastern Hill University, Shillong (India)

25 August 2007

Online at <https://mpra.ub.uni-muenchen.de/4949/>

MPRA Paper No. 4949, posted 18 Sep 2007 UTC

NLINLS: A Differential Evolution based Nonlinear Least Squares Fortran 77 Program

SK Mishra
Dept. of Economics
NEHU, Shillong

Introduction: Curve fitting or estimation by nonlinear least squares is a difficult task. There are two types of algorithm that are often used for this purpose: those that need evaluation of derivatives and the others that do not. In the first category we have Gauss-Newton, Levenberg-Marquardt and Fletcher-Powell, Fletcher-Reeves, Rosen, Quasi-Newton, etc while in the second category we have Powell, Rosenbrock, Hooke-Jeeves, Nelder-Mead and Box, etc (see Kuester and Mize, 1973). All of them are very prone to be caught into local minimum trap. Recently, some methods of global optimization have been developed that are based on some sort of stochastic process. The method of Differential Evolution (Storn and Price, 1995) is perhaps the most promising among them. In developing this program we have used DE as an optimizer of the nonlinear least square function. The DE is an evolutionary, population-based, algorithm; a development on the Genetic Algorithm. The DE as a method of optimization has been well tested on many extremely difficult multi-modal problems of nonlinear optimization (see at <http://www1.webng.com/economics> or http://www.freewebs.com/nehu_economics for its applications including estimation of Sato's nested CES production function (Mishra, 2006), Zellner-Revankar production function (Mishra, 2007-a) and joint production functions (Mishra, 2007-b), etc).

Fortran Compiler, NLINLS and User's Data: The program is written in Fortran 77 (<http://www1.webng.com/economics> or http://www.freewebs.com/nehu_economics for **nlins.txt version to directly copy-paste in a Fortran editor**). To run the program one has to use a Fortran 77 compiler. FORCE-2.0 is a powerful compiler that may be downloaded from <http://www.guilherme.tk/> (there going to **Downloads** and clicking on **UK** against Force208.exe) free and installed on the user's computer. Any other Fortran compiler may also be used.

Hougen Model Data				
Sl No.	X ₁	X ₂	X ₃	Y
1	470	300	10	8.55
2	285	80	10	3.79
3	470	300	120	4.82
4	470	80	120	0.02
5	470	80	10	2.75
6	100	190	10	14.39
7	100	80	65	2.54
8	470	190	65	4.35
9	100	300	54	13.00
10	100	300	120	8.50
11	100	80	120	0.05
12	285	300	10	11.32
13	285	190	120	3.13

Then the program NLINLS may be copy-pasted from the source file (that may be in the text file such as NLINLS.txt) to the FORCE editor and saved as NLINLS.f. However, if you have obtained this program as NLINLS.f then after installing the Fortran compiler you may directly double click it. It will be taken up by FORCE automatically.

The first step in using the program is to make a text file (for example, HOUGEN.TXT in which data, note - only data, the colored portion - are to be stored in the scheme exemplified in the left panel). The sequence of columns is sl no., x₁, x₂, ..., y.

Please note that the data file and the NLINLS.f should be saved in the same folder (directory) to avoid typing long paths afterwards. However, this is only a suggestion.

The user of NLINLS program has to specify the parameters, bounds (limits) on them as well as the function to be fitted to data. This is done in the last subroutine of the program: REGFUNC. It has been specifically mentioned in the REGFUNC as to what is to be modified in the program and what is not to be altered. Only three sets of changes are needed: (1) defining the set of parameters; $p1 = p(1)$, $p2 = p(2)$, etc depending on the problem; (2) defining the set of variables, $x1 = \text{datum}(i, 1)$, $x2 = \text{datum}(i, 2)$, ..., $y = \text{datum}(i, \text{last})$, for example, in the Hougen problem $x1 = \text{datum}(i, 1)$, $x2 = \text{datum}(i, 2)$, $x3 = \text{datum}(i, 3)$, $y = \text{datum}(i, 4)$; and finally, (3) defining y_x in terms of p and x , such that in the Hougen problem the function is defined as $y_x = (p1 * x2 - x3 / p5) / (1.d0 + p2 * x1 + p3 * x2 + p4 * x3)$. Almost never one would require changing FORMAT(I5, 2F25.12), but one may change it if (at all) needed.

Once the changes in the program are made, it may be saved and compiled or even run directly. If no error has been committed in changing/redefining as advised above, the program will run. While it runs, the program asks for several inputs from the user. The program issues clear instructions to be (normally) followed.

1. “If help is needed” : The user (who has read this help document) will not possibly gain anything by feeding a non-zero number. So he/she should type 0 and strike the **Enter** key.
2. “Feed the name of input file ...” : As has been advised above, the user is ready with the input data file, for example, in *mydata.txt* named file. He/she may type it and strike the **Enter** key.
3. “Number of observations and variables ...”: The user should type the number of variables (e.g. 13 in the Hougen problem above) and the number of variables (including y) in the problem (e.g. 4 in the Hougen problem above). Then strike the **Enter** key. His/her input from the keyboard will be 13, 4 **Enter**.
4. “No. of parameters to be estimated” : It is the number m in $p1, p2, \dots, P_m$. For the Hougen problem it is 5. Input 5 and **Enter**.
5. “Would you specify any limits ...”: If the user has to specify some bounds (limits) on the values of parameter, he/she should type an integer number other than zero (0), e.g. 2, 3 or 4, etc and **Enter**. But if he/she has not to specify any bounds, he/she should type 0 and **Enter**.
6. If the user has chosen to provide bounds on the parameter, then he/she will be asked to provide those (lower and upper limits).
7. Specify the [lower upper, lower upper ...” : The user may specify them. For example, the lower upper bound on the parameters of the Hougen problem above may be given as
-10 10, -10 10, -10 10, -10 10, -10 10 **Enter**
8. “Option: Would you minimize ...” : If the user wants to minimize sum of squares, he/she should type 0 and **Enter**. If he/she wants to maximize R^2 then

type any non-zero integer and strike the **Enter** key. Both amount to the same, but display different results.

Once these inputs are accepted by the computer, the program enters into DE optimization. The DE algorithm needs a number of inputs. These are:

1. “Population size and number of iterations ...” : Population size should not be less than ten times the number of parameters to be estimated. For example, in the Hougen problem above, 5 parameters are there. So population size should not be less than 50. Unless the number of parameters is more than 10, population size = 100 is good enough. *In the main program the parameter IPRES may be 0 or 1. If IPRES=0 then population size should be at least 10 times M (no. of parameters) else it should be 20 times M or more.* Number of Iterations = 10000 is enough for moderately sized problems.
2. “Cross-over probability ... and two scale parameters” : It depends on the complexity of the problem. If the problem appears to be simple, (0.9, 0.5, 0) is alright and very effective. In some cases (0.9, 0.9 0.01) performs better. In some other cases (0.5, 0.9, 0.1) has worked. In case of complicated problems one may run the program several times with these alternatives 0.9, 0.5, 0 **Enter**.
3. “Accuracy for convergence” : For most problems 0.0001 would be all right.
4. “Random number seed.” : any 4-digit odd integer such as 5781 **Enter**.
5. “Name of output file” : Such as *myresults.txt* **Enter**.

And the program goes in for computation. All intermediate results pop up on the screen. All intermediate and final results including expected values of y are stored in the output file specified by the user (e.g. *myresults.txt*). Note that it is automatically saved in the same folder (directory) where NLINLS.f and data file are already saved. Then one may enter into that folder and double click the output file (e.g. *myresults.txt*) to view the results. One may also open it by any other text editor.

In case of any difficulty or further help, the user is most welcome to contact me at mishrsknehu@yahoo.com.

References

- Kuester, JL and Mize, JH (1973) *Optimization Techniques with Fortran*, McGraw-Hill Book Co., New York.
- Mishra, SK (2006) “A Note on Numerical Estimation of Sato’s Two-Level CES Production Function” , SSRN at <http://www.ssrn.com/author=353253>
- Mishra, SK (2007-a) "Estimation of Zellner-Revankar Production Function Revisited." *Economics Bulletin*, 3 (14), pp. 1-7.
- Mishra, SK (2007-b) “Least Squares Estimation of Joint Production Functions by the Differential Evolution method of Global Optimization” *Social Science Research Network (SSRN)* at <http://www.ssrn.com/author=353253>
- Storn, R and Price, K (1995) "Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces": *Technical Report*, International Computer Science Institute, Berkley.

NLINLS

PROGRAM NLINLS

PARAMETER (MAXN=1000, MAXM=50) ! MAX NUMBER OF OBSERVATIONS =1000
AND MAX NUMBER OF INDEPENDENT VARIABLES=50. THIS MAY BE INCREASED
OR DECREASED IF NEEDED. USUALLY ONE DOES NOT NEED CHANGING THEM.

THE LEAST SQUARES OBJECTIVE FUNCTION IN THIS PROGRAM IS OPTIMIZED
BY THE "DIFFERENTIAL EVOLUTION ALGORITHM" OF GLOBAL OPTIMIZATION
THIS METHOD WAS PROPOSED BY R. STORN AND K. PRICE IN 1995. REF --
"DIFFERENTIAL EVOLUTION - A SIMPLE AND EFFICIENT ADAPTIVE SCHEME
FOR GLOBAL OPTIMIZATION OVER CONTINUOUS SPACES" : TECHNICAL REPORT
INTERNATIONAL COMPUTER SCIENCE INSTITUTE, BERKLEY, 1995.

////////////////////////////////////
PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)

PARAMETER(NMAX=500,MMAX=50) ! MAXIMUM DIMENSION PARAMETERS FOR DE
PARAMETER(IPRES=1) ! THIS SPECIFICATION IS FOR DIFFICULT FUNCTIONS
PARAMETER(IPRES=0) ! THIS SPECIFICATION IS FOR GENERAL FUNCTIONS
PARAMETER(IPRINT=500) ! FOR WATCHING INTERMEDIATE RESULTS
IT PRINTS THE INTERMEDIATE RESULTS AFTER EACH IPRINT ITERATION.
IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
COMMON /RNDM/IU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
COMMON /REGDAT/DATUM(MAXN,MAXM),PARLM,NDP,NDV,MINOPT,NLIMITS,FOUT
NDP = NO. OF DATA POINTS, NDV = NO. OF VARIABLES (INCLUDING Y)
FOUT IS THE NAME OF OUTPUT FILE TO STORE FINAL RESULTS
COMMON /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS, TITLE
CHARACTER *80 FTIT ! TITLE: NONLINEAR LEAST SQUARES PROBLEM
CHARACTER *80 FOUT ! THE OUTPUT FILE NAME

THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING -----
(1) FTIT= TITLE; (2) NO. OF VARIABLES IN THE FUNCTION (M);
(3) N=POPULATION SIZE (SUGGESTED 10 TIMES OF NO. OF VARIABLES, M,
FOR SMALLER PROBLEMS N=100 WORKS VERY WELL);
(4) PCROS = PROB. OF CROSS-OVER (SUGGESTED : ABOUT 0.85 TO .99);
(5) FACT = SCALE (SUGGESTED 0.5 TO .95 OR 1, ETC);
(6) ITER = MAXIMUM NUMBER OF ITERATIONS PERMITTED (5000 OR MORE)
(7) RANDOM NUMBER SEED (4 DIGITS INTEGER)

DIMENSION PARLM(MAXM,2) ! LOWER AND UPPER LIMITS OF PARAMETERS
DIMENSION X(NMAX,MMAX),Y(NMAX,MMAX),FV(NMAX),A(MMAX),IR(4)

HELP=99.D0
99 WRITE(*,*) ' ' ,
WRITE(*,*) ' ' , NONLINEAR LEAST SQUARES PROGRAM'
WRITE(*,*) ' ' ,
WRITE(*,*) ' ' , WRITTEN BY PROF. SK MISHRA'
WRITE(*,*) ' ' , DEPT. OF ECONOMICS, NEHU, SHILLONG'
WRITE(*,*) ' ' ,
WRITE(*,*) '-----'
&-----'

FTIT='NONLINEAR REGRESSION PROBLEM'
----- PROVISION OF HELP -----

IF (HELP.EQ.99.D0) THEN
WRITE(*,*) 'IF ANY HELP IS NEEDED FEED 1 (ONE) OR ANY OTHER NUMBER'
READ(*,*) HELP
ELSE
GO TO 999
ENDIF
IF(HELP.EQ.1.D0) THEN
CALL HELPFIL()'
STOP
ENDIF

HELP=0.D0

```

C -----
999 KF=1 ! THIS PROGRAM WILL SOLVE ONE PROBLEM AT A TIME
   NRD=0 ! NRD = 0 DATA FROM INPUT FILE WILL BE READ
   NQ=0
   CALL REGFUNC(A,F,M,NQ,NRD)
C -----
C ----- SPECIFY PARAMETERS OF DE OPTIMIZATION -----
   WRITE(*,*) ' '
   WRITE(*,*) '*****'
   WRITE(*,*) ' '
   WRITE(*,*) 'NOW SPECIFY THE PARAMETERS OF DIFFERENTIAL EVOLUTION
&OPTIMIZATION'
   WRITE(*,*) 'PLEASE CONSIDER THE SUGGESTIONS, ALSO READ HELP FILE'
   WRITE(*,*) ' '
   WRITE(*,*) 'POPULATION SIZE [NPOP] AND NO. OF ITERATIONS [ITER] ?'
   WRITE(*,*) 'SUGGESTED : NPOP => 100 OR => 10 TIMES M (THE NUMBER OF
& PARAMETERS) IF IPRES=0 ELSE 20 TIMES M;'
   WRITE(*,*) 'ITER 10000 OR LARGER'
   READ(*,*) NPOP,ITER ! POPULATION SIZE AND NO. OF ITERATIONS
   N=NPOP ! RENAMED FOR CONVENIENCE
C (PLEASE DO NOT CONFUSE IT TO BE AS NO. OF OBSERVATIONS IN DATA)
   WRITE(*,*) ' '
   WRITE(*,*) 'CROSSOVER PROBABILITY [PCROS], AND TWO SCALE PARAMETERS
& [FACT AND FACT1] ?'
   WRITE(*,*) 'SUGGESTED: PCROS BETWEEN [0.5 - 0.9]; FACT BETWEEN [0.5
& - 0.9]'
   WRITE(*,*) 'AND FACT1 BETWEEN [0.0 - 0.01] - LIMITS ARE INCLUSIVE'
   READ(*,*) PCROS,FACT,FACT1
   WRITE(*,*) ' '
C EPS DETERMINES ACCURACY FOR TERMINATION. IF EPS= 0, ALL ITERATIONS
C WOULD BE UNDERGONE EVEN IF NO IMPROVEMENT IN RESULTS IS THERE.
C ULTIMATELY "DID NOT CONVERGE" IS REOORTED.
   WRITE(*,*) 'ACCURACY FOR CONVERGENCE - DEPENDS ON THE PROBLEM'
   WRITE(*,*) '0.0001 IS OK; 0.0000001 IS OFTEN MORE THAN ENOUGH'
   READ(*,*) EPS
   WRITE(*,*) ' '
   WRITE(*,*) 'RANDOM NUMBER SEED ?'
   WRITE(*,*) 'A 4-DIGIT POSITIVE ODD (NOTE, ODD) INTEGER [E.G. 5471]'
   READ(*,*) IU !SEED OF RANDOM NUMBER (4-DIGIT ODD NATURAL NUMBER)
   WRITE(*,*) ' '
   WRITE(*,*) 'NAME OF OUTPUT FILE IN WHICH RESULTS WILL BE STORED'
   READ(*,*) FOUT ! (IT IS THE THE GENERAL NAME OF THE OUTPUT FILE)
   WRITE(*,*) ' '
   OPEN(11, FILE=FOUT) ! OPENS THE OUTPUT FILE TO STORE RESULTS'
   ICQ=0
   NFCALL=0 ! INITIALIZE COUNTER FOR FUNCTION CALLS
   GBEST=1.D30 ! TO BE USED FOR TERMINATION CRITERION
C INITIALIZATION : GENERATE X(N,M) RANDOMLY
   DO I=1,N
   DO J=1,M
   CALL RANDOM(RAND) ! GENERATES INITION X WITHIN
C X(I,J)=(RAND-0.5D0)*2000 ! GENERATES INITION X WITHIN
C X(I,J)=(RAND-0.5D0)*10 ! GENERATES INITION X WITHIN
C RANDOM NUMBERS BETWEEN -10 AND 10 (BOTH EXCLUSIVE)
   ENDDO
   ENDDO
   WRITE(*,*) ' '
   WRITE(*,*) 'COMPUTING. PLEASE WAIT. LARGER PROBLEMS TAKE MORE TIME'
   WRITE(*,*) '*****'
   WRITE(*,*) ' '
   WRITE(*,*) 'PROBLEM=',KF,' ',FTIT

```

NLINLS

```

WRITE(*,*) ' '
WRITE(11,*) '***** INTERMEDIATE RESULTS *****'
WRITE(*,*) '***** INTERMEDIATE RESULTS *****'
IPCOUNT=0
DO 100 ITR=1,ITER ! ITERATION BEGINS

CALL RANDOM(RAND)
RX1=RAND
CALL RANDOM(RAND)
RX2=RAND
C -----
C EVALUATE ALL X FOR THE GIVEN FUNCTION
DO I=1,N
DO J=1,M
A(J)=X(I,J)
ENDDO
CALL SETFUNC(A,M,F)
C STORE FUNCTION VALUES IN FV VECTOR
FV(I)=F
ENDDO
C -----
C FIND THE FITTEST (BEST) INDIVIDUAL AT THIS ITERATION
      FBEST=FV(1)
      KB=1
      DO IB=2,N
        IF(FV(IB).LT.FBEST) THEN
          FBEST=FV(IB)
          KB=IB
        ENDIF
      ENDDO
C BEST FITNESS VALUE = FBEST : INDIVIDUAL X(KB)
C -----
C GENERATE OFFSPRINGS
CALL RANDOM(RAND)
C -----
NL=N
IF(IPRES.GT.0) THEN
IF(RAND.GT.0.9D0) THEN
NL=N
ELSE
NL=INT(0.5*N)
ENDIF
ENDIF
C -----
ICQ=ICQ+1
IF(ICQ.GE.50) ICQ=0
IF(ICQ.EQ.0.AND.NL.LT.N) THEN
DO IN=NL+1,N
DO J=1,M
CALL RANDOM(RAND)
X(I,J)=(RAND-0.5D0)*10.D0
ENDDO
ENDDO
ENDIF
C -----
DO I=1,N ! I LOOP BEGINS
C INITIALIZE CHILDREN IDENTICAL TO PARENTS; THEY WILL CHANGE LATER
DO J=1,M
Y(I,J)=X(I,J)
ENDDO

```

```

C                                     NLINLS
C 20  SELECT RANDOMLY THREE OTHER INDIVIDUALS
      DO IRI=1,4  ! IRI LOOP BEGINS
        IR(IRI)=0

        CALL RANDOM(RAND)
C -----
      NL=N
      IF(IPRES.GT.0) THEN
        IF(RAND.GT.0.10D0) THEN
          NL=N
        ELSE
          NL=INT(0.5*N)
        ENDIF
      ENDIF
C -----
      IF(IRI.LT.4) THEN
        IRJ=INT(RAND*NL)+1
      ELSE
        IF(NL.LT.N) THEN
          IRJ=INT(RAND*(N-NL))+NL
        ELSE
          IRJ=INT(RAND*NL)+1
        ENDIF
      ENDIF
C  CHECK THAT THESE THREE INDIVIDUALS ARE DISTICT AND OTHER THAN I
      IF(IRI.EQ.1.AND.IRJ.NE.I) THEN
        IR(IRI)=IRJ
      ENDIF
      IF(IRI.EQ.2.AND.IRJ.NE.I.AND.IRJ.NE.IR(1)) THEN
        IR(IRI)=IRJ
      ENDIF
      IF(IRI.EQ.3.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2)) THEN
        IR(IRI)=IRJ
      ENDIF
      IF(IRI.EQ.4.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2).AND.
* IRJ.NE.IR(3)) THEN
        IR(IRI)=IRJ
      ENDIF

      ENDDO  ! IRI LOOP ENDS
C  CHECK IF ALL THE THREE IR ARE POSITIVE (INTEGERS)
      DO IX=1,4
        IF(IR(IX).LE.0) THEN
          GOTO 20  ! IF NOT THEN REGENERATE
        ENDIF
      ENDDO
C  THREE RANDOMLY CHOSEN INDIVIDUALS DIFFERENT FROM I AND DIFFERENT
C  FROM EACH OTHER ARE IR(1),IR(2) AND IR(3)
C  ===== RANDOMIZATION OF NCROSS =====
C  RANDOMIZES NCROSS
      NCROSS=0
      CALL RANDOM(RAND)
      IF(RAND.GT.RX1.AND.RAND.LT.RX2) NCROSS=1
      ! IF RX1=>1, SCHEME 2 NEVER IMPLEMENTED
      IF(RAND.GT.RX1.AND.RAND.GT.RX2) NCROSS=2
      ! IF RX2=>1, SCHEME 3 NEVER IMPLEMENTED

C ----- SCHEME 1 -----
C  NO CROSS OVER, ONLY REPLACEMENT THAT IS PROBABILISTIC
      IF(NCROSS.LE.0) THEN
        DO J=1,M  ! J LOOP BEGINS
          CALL RANDOM(RAND)
          IF(RAND.LE.PCROS) THEN ! REPLACE IF RAND < PCROS

```



```

                                NLINLS
      A(J)=X(IR(1),J)+(X(IR(2),J)-X(IR(3),J))*FACT +
* (X(IR(3),J)-0.5D0*X(IR(4),J)-.5D0*X(IR(1),J))*FACT1! CANDIDATE CHILD
      ENDIF
      ENDDO ! J LOOP ENDS
      ENDIF

C ----- SCHEME 2 -----
C THE STANDARD CROSSOVER SCHEME
C CROSSOVER SCHEME (EXPONENTIAL) SUGGESTED BY KENNETH PRICE IN HIS
C PERSONAL LETTER TO THE AUTHOR (DATED SEPTEMBER 29, 2006)
      IF(NCROSS.EQ.1) THEN
        CALL RANDOM(RAND)
1       JR=INT(RAND*M)+1
        J=JR
2       A(J)=X(IR(1),J)+FACT*(X(IR(2),J)-X(IR(3),J))+
* (X(IR(3),J)-0.5D0*X(IR(4),J)-.5D0*X(IR(1),J))*FACT1
3       J=J+1
        IF(J.GT.M) J=1
4       IF(J.EQ.JR) GOTO 10
5       CALL RANDOM(RAND)
        IF(PCROS.LE.RAND) GOTO 2
6       A(J)=X(I,J)
7       J=J+1
        IF(J.GT.M) J=1
8       IF (J.EQ.JR) GOTO 10
9       GOTO 6
10      CONTINUE
      ENDIF

C ----- SCHEME 3 -----
C ESPECIALLY SUITABLE TO NON-DECOMPOSABLE (NON-SEPERABLE) FUNCTIONS
C CROSSOVER SCHEME (NEW) SUGGESTED BY KENNETH PRICE IN HIS
C PERSONAL LETTER TO THE AUTHOR (DATED OCTOBER 18, 2006)
      IF(NCROSS.GE.2) THEN
        CALL RANDOM(RAND)
        IF(RAND.LE.PCROS) THEN
          CALL NORMAL(RN)
          DO J=1,M
C       A(J)=X(I,J)+(X(IR(1),J)+ X(IR(2),J)-2*X(I,J))*(RN-0.5D0)*2
          A(J)=X(I,J)+(X(IR(1),J)+ X(IR(2),J)-2*X(I,J))*RN*FACT1+
* (X(IR(3),J)-0.5D0*X(IR(4),J)-.5D0*X(IR(1),J))*FACT1
          ENDDO
        ELSE
          DO J=1,M
C       A(J)=X(I,J)+(X(IR(1),J)- X(IR(2),J))+
* (X(IR(3),J)-0.5D0*X(IR(4),J)-.5D0*X(IR(1),J))*FACT1! FACT ASSUMED TO BE 1
          ENDDO
        ENDIF
      ENDIF

C -----
      CALL SETFUNC(A,M,F) ! EVALUATE THE OFFSPRING
      IF(F.LT.FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
        FV(I)=F
        DO J=1,M
          Y(I,J)=A(J)
        ENDDO
      ENDIF
      ENDDO ! I LOOP ENDS
      DO I=1,N
        DO J=1,M
          X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
C       BETTER CHILDREN
        ENDDO
      ENDDO

```

NLINLS

```

IPCOUNT=IPCOUNT+1
IF(IPCOUNT.EQ.IPRINT) THEN
DO J=1,M
A(J)=X(KB,J)
ENDDO
WRITE(*,*)'ESTIMATED PARAMETERS UP TO NOW'
WRITE(*,*)(X(KB,J),J=1,M)
WRITE(11,*)'ESTIMATED PARAMETERS UP TO NOW'
WRITE(11,*)(X(KB,J),J=1,M)
IF(MINOPT.EQ.0) THEN
WRITE(*,101) FBEST
WRITE(11,101) FBEST
ELSE
WRITE(*,102) -FBEST
WRITE(11,102) -FBEST
ENDIF
WRITE(*,*)'-----'
WRITE(11,*)'-----'
101 FORMAT(1X,'SUM OF SQUARED ERRORS UP TO NOW = ',F25.16)
102 FORMAT(1X,'R_SQUARE UP TO NOW = ',F25.16)
C   WRITE(*,*)'TOTAL NUMBER OF FUNCTION CALLS = ',NFCALL
IF(DABS(FBEST-GBEST).LT.EPS) THEN
WRITE(*,*)'COMPUTATION OVER. RESULTS STORED IN THE FOLLOWING FILE'
WRITE(*,*) FOUT
WRITE(*,*) ' '
C   -----
WRITE(11,*) ' '
WRITE(11,*) '*****          FINAL RESULTS          *****'
WRITE(11,*) '-----'
WRITE(11,*) ' '
NQ=1
NRD=1
CALL REGFUNC (A,F,M,NQ,NRD)
C   -----
CLOSE(11) ! CLOSSES THE OUTPUT FILE
WRITE(*,*)'THANK YOU'
STOP
ELSE
GBEST=FBEST
ENDIF
IPCOUNT=0
ENDIF
C   -----
100 ENDDO ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
C   -----
WRITE(*,*)'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
WRITE(*,*)'INCREASE POPULATION SIZE, PCROS, OR SCALE FACTOR(FACT)'
CLOSE(11) ! CLOSSES THE OUTPUT FILE
END
C   -----
SUBROUTINE NORMAL(R)
C   PROGRAM TO GENERATE N(0,1) FROM RECTANGULAR RANDOM NUMBERS
C   IT USES BOX-MULLER VARIATE TRANSFORMATION FOR THIS PURPOSE.
C   -----
C   ----- BOX-MULLER METHOD BY GEP BOX AND ME MULLER (1958) -----
C   BOX, G. E. P. AND MULLER, M. E. "A NOTE ON THE GENERATION OF
C   RANDOM NORMAL DEVIATES." ANN. MATH. STAT. 29, 610-611, 1958.
C   IF U1 AND U2 ARE UNIFORMLY DISTRIBUTED RANDOM NUMBERS (0,1),
C   THEN X=[(-2*LN(U1))**.5]*(COS(2*PI*U2) IS N(0,1)
C   ALSO, X=[(-2*LN(U1))**.5]*(SIN(2*PI*U2) IS N(0,1)
C   PI = 4*ARCTAN(1.0)= 3.1415926535897932384626433832795
C   2*PI = 6.283185307179586476925286766559
C   -----

```

```

                                NLINLS
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /RNDM/IU,IV

```

```

C -----
CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
U1=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
U2=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
R=DSQRT(-2.D0*DLOG(U1))
R=R*DCOS(U2*6.283185307179586476925286766559D00)
C R=R*DCOS(U2*6.28318530718D00)
RETURN
END

```

```

C -----
C RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
SUBROUTINE RANDOM(RAND)
DOUBLE PRECISION RAND
COMMON /RNDM/IU,IV
IV=IU*65539
IF(IV.LT.0) THEN
IV=IV+2147483647+1
ENDIF
RAND=IV
IU=IV
RAND=RAND*0.4656613D-09
RETURN
END

```

```

C -----
SUBROUTINE SETFUNC(A,M,F)
C SET THE FUNCTION FOR GLOBAL OPTIMIZATION PROGRAM
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /RNDM/IU,IV
COMMON /KFF/KF,NFCALL,FTIT
CHARACTER *80 FTIT
DIMENSION A(*)
NFCALL=NFCALL+1 ! INCREMENT TO NUMBER OF FUNCTION CALLS
C KF IS THE CODE OF THE TEST FUNCTION

```

```

C -----
NQ=0
NRD=1
CALL REGFUNC(A,F,M,NQ,NRD)
RETURN
END

```

```

C -----
SUBROUTINE HELPFILE()
WRITE(*,*) 'HERE IS SOME HELP INFORMATION FOR YOU !'
WRITE(*,*) '-----'
WRITE(*,*) 'FIRST FEED YOUR DATA AS FOLLOWS IN SOME TEXT FILE, FOR
& EXAMPLE HOUGEN.TXT'
WRITE(*,*) ' '
WRITE(*,*) ' 1 470.0 300.0 10.0 8.55'
WRITE(*,*) ' 2 285.0 80.0 10.0 3.79'
WRITE(*,*) ' 3 470.0 300.0 120.0 4.82'
WRITE(*,*) ' 4 470.0 80.0 120.0 0.02'
WRITE(*,*) ' .. ..'
WRITE(*,*) '13 285.0 190.0 120.0 3.13'
WRITE(*,*) ' '
WRITE(*,*) 'NOTE THAT SL NO. IS THE FIRST COLUMN AND Y IS THE LAST'
WRITE(*,*) 'THEN IN SUBROUTINE REGFUNC (THE LAST SUBROUTINE OF THIS
& PROGRAM'
WRITE(*,*) 'DEFINE PARAMETER NAMES (P1, P2, P3, ETC), AND VARIABLE
& NAMES (X1, X2,..., Y);'
WRITE(*,*) 'CHANGE THE FUNCTION (YX=AS PER YOUR SPECIFICATION),E.G.
& HOUGEN FUNCTION IS'

```

```

                                NLINLS
WRITE(*,*) 'SPECIFIED AS: YX=(P1*X2-X3/P5)/(1.D0+P2*X1+P3*X2+P4*X3)'
WRITE(*,*) 'MODIFY FORMAT SPECIFICATION (ALMOST NEVER NEEDED)'
WRITE(*,*) 'SAVE THE PROGRAM AND RUN. FOLLOW THE INSTRUCTIONS. IF
& YOU ARE DONE RUN THE PROGRAM AGAIN. THANK YOU'
RETURN
END

C -----
SUBROUTINE REGFUNC(P,F,M,NQ,NRD)
PARAMETER(MAXN=1000, MAXM=50)
C MAXN=MAX NUMBER OF OBSRVATIONS, MMAX= MAX NUMBER OF PARAMETERS
C MAXN=MAX NO. OF VARIABLES. UNLESS YOUR OBSERVATIONS EXCEED 1000 OR
C NUMBER OF EXPLANATORY VARIABLES EXCEED 50 DO NOT CHANGE IT.
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
COMMON /REGDAT/DATUM(MAXN,MAXM), PARLIM,NDP,NDV,MINOPT,NLIMITS,FOUT
C CHARACTER *80 FIN, FOUT ! NAME OF INPUT AND OUTPUT FILES
C NAME OF INPUT AND OUTPUT FILES MAXIMUM 80 CHARACTERS LONG
DIMENSION PARLIM(MAXM,2)! LOWER AND UPPER LIMITS OF PARAMETERS
DIMENSION P(*)! PARAMETERS

C -----
C NOW READ DATA FROM THE FILE ONLY ONCE (IN THE BEGINNING)
IF(NRD.EQ.0) THEN
WRITE(*,*) 'FEED THE NAME OF INPUT FILE IN WHICH DATA ARE STORED'
READ(*,*) FIN ! (IT IS THE THE GENERAL NAME OF THE INPUT FILE)
WRITE(*,*) ' '
WRITE(*,*) 'NUMBER OF OBSERVATIONS AND VARIABLES (INCLUDING Y): X1,
& X2,...,Y ?'
WRITE(*,*) 'SUPPOSE Y=F(X1,X2,X3), THEN WE HAVE 4 VARIABLES IN ALL'
READ(*,*) NDP,NDV!NO. OF OBSERVATIONS AND VARIABLES INCLUDING Y
WRITE(*,*) ' '
WRITE(*,*) 'NO. OF PARAMETERS TO BE ESTIMATED ?'
READ(*,*) M ! (NO. OF PARAMETERS TO BE ESTIMATED)
WRITE(*,*) ' '
WRITE(*,*) 'WOULD YOU SPECIFY ANY LIMITS TO THE PARAMETERS ? IF NO
& THEN FEED 0 (ZERO) ELSE FEED A NONZERO INTEGER'
READ(*,*) NLIMITS
WRITE(*,*) ' '
IF(NLIMITS.NE.0) THEN
WRITE(*,*) 'SPECIFY THE [LOWER UPPER, LOWER UPPER, LOWER UPPER,...]
& LIMITS OF PARAMETERS FROM THE FIRST TO THE LAST'
READ(*,*) (PARLIM(J,1),PARLIM(J,2),J=1,M)
WRITE(*,*) ' '
ENDIF
OPEN(7, FILE=FIN)!OPENS INPUT DATA FILE TO READ DATA (SL_NO.,X,Y)
DO I=1,NDP
READ(7,*) NSL,(DATUM(I,J),J=1,NDV) ! READS SL NUMBER, X AND Y
ENDDO
CLOSE(7) ! CLOSSES THE INPUT FILE
IF(NSL.NE.NDP) WRITE(*,*) 'MISMATCH IN NUMBER OF OBSERVATIONS'
WRITE(*,*) 'OPTION: WOULD YOU MINIMIZE THE SUM OF SQUARED ERRORS
&(FEED 0)'
WRITE(*,*) 'OR MAXIMIZE R_SQUARE (FEED ANY NON-ZERO INTEGER)'
READ(*,*) MINOPT
WRITE(*,*) ' '
NRD=1 ! ONCE NRD = 1 DATA FROM INPUT FILE WOULD NOT BE READ AGAIN
ENDIF

C -----
C N=NDP ! RENAMING THE PARAMETER JUST FOR CONVENIENCE
SY=0.D0 ! INITIALIZE SUM OF Y
SSY=0.D0 ! INITIALIZE SUM OF SQUARED Y
SSE=0.D0 ! INITIALIZE SUM OF SQUARED ERRORS
C CHECKING THE LIMITS OF PARAMETERS AND BRINGING WITHIN LIMITS IF
C ANY OF THEM CROSS THE LIMITS - IT IS REQUIRED IN SOME CASES

```

NLINLS

```

IF(NLIMITS.NE.0) THEN
DO J=1,M
IF(P(J).LT.PARLIM(J,1) .OR. P(J).GT.PARLIM(J,2)) THEN
CALL RANDOM(RAND)
P(J)=PARLIM(J,1)+ (PARLIM(J,2)-PARLIM(J,1))*RAND
ENDIF
ENDDO
ENDIF
C
C //////////////////////////////////////
C ----- HERE ONWARDS (UNLESS OTHERWISE INSTRUCTED) THE USER IS
C ----- PERMITTED TO SPECIFY THE REGRESSION PROBLEM -----
C //////////////////////////////////////
C
C PLEASE REDEFINE PARAMETERS P1=P(1), P2=P(2), ..., PM=P(M)
C
P1=P(1)
P2=P(2)
P3=P(3)
P4=P(4)
P5=P(5)
C
C REDEFINE VARIABLES X1=DATUM(I,1), X2=DATUM(I,2), X3=DATUM(I,3)... ,
C FINALLY Y=DATUM(I, NDV). REMEMBER THAT Y IS THE LAST COLUMN DATA
DO I=1,N
X1=DATUM(I,1)
X2=DATUM(I,2)
X3=DATUM(I,3)
Y=DATUM(I,4)
C
C NOW DEFINE YOUR REGRESSION FUNCTION YX IN TERMS OF X AND P
C YX IS THE EXPECTED VALUE OF Y IN TERMS OF X AND P. FOR EXAMPLE
C WE DEFINE THE HOUGEN FUNCTION AS GIVEN BELOW
C
C YX=(P1*X2-X3/P5)/(1.D0+P2*X1+P3*X2+P4*X3) ! HOUGEN FUNCTION
1 FORMAT(I5, 2F25.12) ! THIS IS THE FORMAT OF OUTPUT. IF THE INTEGER
C PART OF Y DOES NOT EXCEED 11 DIGITS AND THE FRACTION PART DOES NOT
C EXCEED 12 DIGITS THEN THIS FORMAT NEED NOT BE CHANGED AT ALL.
C
C ----- JURISDICTION OF THE USER ENDS HERE -----
C //////////////////////////////////////
C --- ----- PLEASE DO NOT CHANGE ANYTHING BELOW THIS -----
C //////////////////////////////////////
C IF(NQ.NE.0) WRITE(11,1)I,Y,YX !STORE IN OUTPUT
C WRITES SL NO. Y AND EXPECTED Y (THAT IS YX) IN THE OUTPUT FILE
ER=(Y-YX) ! ESTIMATED ERROR
SSE=SSE+ER**2 ! SUM UP THE SQUARED ERROR
SY=SY+Y ! SUM OF Y
SSY=SSY+Y**2 ! SUM OF SQUARED Y
ENDDO
RMS2=SSE/N! MEAN SQUARED ERROR (SQUARED 'ROOT MEAN SQUARE' OR RMS)
V=SSY/N-(SY/N)**2 ! VARIANCE OF Y
RSQ=1.D0-RMS2/V
IF(MINOPT.EQ.0) THEN
F=SSE ! THE MINIMAND FUNCTION IS THE SUM OF SQUARED ERRORS
ELSE
F = -RSQ ! MINIMAND FUNCTION IS THE NEGATIVE OF R_SQUARE
C MINIMIZATION OF NEGATIVE OF R_SQUARE MAXIMIZES THE R_SQUARE
ENDIF
IF(NQ.NE.0) THEN
WRITE(11,*)'[THE ORDER OF ABOVE OUTPUT IS: SL NO., Y, EXPECTED Y] '
WRITE(11,*)' '
WRITE(11,*)'ESTIMATED PARAMETER ARE AS FOLLOWS '

```

```

                                NLINLS
WRITE(11,*)(P(J),J=1,M) ! THESE ARE M PARAMETERS

WRITE(11,*) 'SUM OF SQUARED ERRORS = ', SSE
WRITE(11,*) 'R-SQUARE = ', RSQ
WRITE(11,*) 'RMS = ', DSQRT(RMS2)
ENDIF
RETURN
END

```