No. SOC-88

SQUARE-AN IMPLEMENTATION OF THE MASSARA-FIDLER ALGORITHM FOR LEAST-SQUARES PROBLEMS

W. S. Ishak

May 1975

SQUARE-AN IMPLEMENTATION OF THE MASSARA-FIDLER ALGORITHM FOR LEAST-SQUARES PROBLEMS

Waguih S. Ishak

Abstract SQUARE is a computer program primarily for solving least-squares problems. Its main features include Marquardt-Levenberg damped least-squares and Massara-Fidler new algorithm for controlling the damping factor λ. The program can solve different optimization problems such as: minimization of functions in the form of sum of squares, data fitting problems and system modelling problems, in the least-squares sense. The program has been used on a CDC-6400 computer and several examples are shown. A FORTRAN IV listing is included.

This work was supported by the National Research Council of Canada.

The author is with the Group on Simulation, Optimization and Control and Department of Electrical Engineering, McMaster University, Hamilton, Ontario, Canada, L8S 4L7.

SQUARE - AN IMPLEMENTATION OF THE MASSARA - FIDLER ALGORITHM FOR LEAST-SQUARES PROBLEMS

Waguih S. Ishak

(I) Introduction: Generalized Least Squares Methods:

Most algorithms for the least-squares estimation of non-linear parameters have centered about either using Taylor series approximation or various modifications of the method of steepest-descent. The Taylor series method usually results in a divergent process and the steepest-descent approach suffers from slow convergence after the first few iterations.

Different algorithms have been implemented and their use in computer-aided design of circuits and networks is well established in the literature [1], [2], [3]. The method of least-squares is applicable when the objective function U can be expressed as the sum of n squared residuals in the form:

$$U = \sum_{i=1}^{n} e_i^2(x_1, x_2, \ldots, x_k) \qquad n \geq k \tag{1}$$

where $\underset{\sim}{x} = [x_1 \; x_2 \; \ldots \; x_k]^T$ is the network variable vector. The Gauss' least-squares algorithms proceed by obtaining the correction vector $\Delta\underset{\sim}{x}$, that minimizes the objective function U, which takes the form:

$$\Delta\underset{\sim}{x} = - (\underset{\sim}{J}^T\underset{\sim}{J})^{-1}\underset{\sim}{J}^T\underset{\sim}{e} \tag{2}$$

where $\underset{\sim}{e} = [e_1\ e_2\ \dots\ e_n]^T$ and $\underset{\sim}{J}$ = Jacobian matrix defined as:

$$\underset{\sim}{J} = \begin{bmatrix} \frac{\partial e_1}{\partial x_1} & \frac{\partial e_1}{\partial x_2} & \cdots & \frac{\partial e_1}{\partial x_k} \\ \frac{\partial e_2}{\partial x_1} & \frac{\partial e_2}{\partial x_2} & \cdots & \frac{\partial e_2}{\partial x_k} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_n}{\partial x_1} & \frac{\partial e_n}{\partial x_2} & \cdots & \frac{\partial e_n}{\partial x_k} \end{bmatrix} \qquad (3)$$

Very often, the predicted correction given by equation (2) is large enough to cause eratic moves as the optimization proceeds. This eratic behavior is explained by the fact that large steps $\Delta x_i (i=1,2, \dots, k)$ invalidate the Taylor series approximation. A useful scheme to overcome the above difficulty is to accept the predicted step of eqn. (2) if the objective function is decreased, otherwise the step length is successively halved until the objective is decreased. The problem with this step-halving scheme is that usually premature termination occurs as a result of insistance on an objective decrease [4].

(II) Damping: Levenberg - Marquardt Algorithms:

Levenberg, and later Marquardt, suggested a restriction on the size of the step size to validate the Taylor series approximation.

The Levenberg algorithm actually minimizes both the objective function defined in eqn. (1) together with an additional correction vector which is proportional to the square of the step-size. This leads to a predicted correction of:

$$\Delta \underset{\sim}{x} = - (\underset{\sim}{J}^T \underset{\sim}{J} + \lambda \underset{\sim}{I})^{-1} \underset{\sim}{J}^T \underset{\sim}{e} \qquad (4)$$

with the positive weighting factor λ, termed the Levenberg parameter, and $\underset{\sim}{I}$ the unit matrix.

Different algorithms have been implemented for the choice of the parameter λ which becomes the central problem in the use of eqn. (4). Almost all these algorithms agree with the fact that relatively large values of λ are required in the early stages of an optimization problem because the direction given in eqn. (4) will, then, be biased towards the steepest-descent. As the minimization proceeds small values of λ should be used to accelerate the convergence.

A recent algorithm has been reported by Massara and Fidler [4] in which the parameter λ has been used very effciently to control the minimization process. In their paper, Massara and Fidler related the fact that for small error values it was found that inaccurate $\Delta \underset{\sim}{x}$ prediction results in divergent iteration to the fact that the determinant of the matrix $(\underset{\sim}{J}^T \underset{\sim}{J} + \lambda \underset{\sim}{I})$ becomes very small and may approach zero. This near singularity makes the inverse required in eqn. (4) subject to large errors.

Another important fact to be considered is that a condition for the step $\underset{\sim}{\Delta x}$ to be downhill is that $(\underset{\sim}{J}^T\underset{\sim}{J} + \lambda\underset{\sim}{I})$ should be positive definite. Massara and Fidler suggested using the parameter λ to control the matrix $(\underset{\sim}{J}^T\underset{\sim}{J} + \lambda\underset{\sim}{I})$ to be a non-singular, positive definite matrix through the minimization process. Except for this controlling nature of λ, the algorithm proceeds in the same fashion as Marquardt algorithm [5]. Some initial value for λ is chosen, 10 say, and λ is halved (doubled) following convergent (divergent) iterations. Massara and Fidler, in their algorithm followed Bown's idea [1] for restricting the number of step-halvings to only 4 successive reductions. This prevents premature termination or slow convergence at the possible expense of error-reduction condition.

The following section describes, in detail, the algorithm reported by Massara and Fidler together with some modifications and suggestions for future extensions.

(III) Description of the Algorithm:

(1) Purpose:

The algorithm is a modified damping method for the minimization of functions in the form of sum of squares. i.e. it minimizes the function:

$$U = \sum_{i=1}^{n} e_i^2(x_1, x_2, \ldots, x_k) \quad , \quad n \geq k$$

The minimization is done using a search method with a predicted step size given by:

$$\underset{\sim}{\Delta x} = - (\underset{\sim}{J}^T\underset{\sim}{J} + \lambda\underset{\sim}{I})^{-1}\underset{\sim}{J}^T\underset{\sim}{e}$$

where the parameters are explained before.

<u>(2) The Algorithm:</u>

(i) Set $r=1$. Set $\lambda=10$.

(ii) Calculate e_i, $i=1,2, \dots, n$ and $\frac{\partial e_i}{\partial x_j}$, $i=1,2, \dots, n$ and $j=1,2, \dots, k$. at the starting point $\underset{\sim}{x}^o$.

(iii) Formulate the Jacobian matrix $\underset{\sim}{J}$ and formulate the matrix $\underset{\sim}{A}$ given by: $\underset{\sim}{A} = (\underset{\sim}{J}^T\underset{\sim}{J} + \lambda\underset{\sim}{I})$.

(iv) Check for the matrix $\underset{\sim}{A}$. If $\underset{\sim}{A}$ is not positive definite or $|\underset{\sim}{A}| < 10^{-7}$, set $\lambda=10\lambda$ and go to (i).

(v) Calculate the step correction $\underset{\sim}{\Delta x}^r = - (\underset{\sim}{J}^T\underset{\sim}{J} + \lambda\underset{\sim}{I})^{-1}\underset{\sim}{J}^T\underset{\sim}{e}$. Calculate $\underset{\sim}{x}^{r+1} = \underset{\sim}{x}^r + \underset{\sim}{\Delta x}^r$.

(vi) Calculate $U(\underset{\sim}{x}^{r+1})$. If $U(\underset{\sim}{x}^{r+1}) < U(\underset{\sim}{x}^r)$ go to (viii).

(vii) Perform a step-halving operation and set $\lambda=2\lambda$. If the number of step-halvings is 4, set $\lambda=\frac{\lambda}{2}$ and $r=r+1$. Go to (iii).

(viii) Check for: $|\underset{\sim}{\Delta x}| < \varepsilon$ where ε is a small number. If yes stop.

(ix) If $r \geq r_{max}$, where r_{max} is the maximum allowable number of iterations, stop.

(x) Set $r=r+1$ and go to (iii).

(3) Comments:

The initial setting of the parameter λ to be 10 is very arbitrary. Other values can be used and acceleration of convergence may be achieved by choosing $\lambda=0.1$ say as an initial setting. This can occur in well conditioned problems as shall be discussed later.

In step (v), a system of linear equations in $\Delta \underset{\sim}{x}$ will be solved in the form:

$$(\underset{\sim}{J}^T\underset{\sim}{J} + \lambda \underset{\sim}{I})\Delta \underset{\sim}{x} = - \underset{\sim}{J}^T\underset{\sim}{e}$$

This can be made use of in the fact that a subroutine can be written to check for the positive-definiteness of the matrix $\underset{\sim}{A} = (\underset{\sim}{J}^T\underset{\sim}{J} +\lambda \underset{\sim}{I})$ and which solves the system of equations in $\Delta \underset{\sim}{x}$ at the same time.

(IV) Implementation of the Algorithm:

A computer program has been written in FORTRAN IV to implement the above described algorithm. All input date is entered through the argument of SUBROUTINE SQUARE, hence the program can be incorporated into other programs for computer-aided design purposes.

(1) The Argument List:

```
CALL SQUARE (USER,N,K,X,E,A,V,VV,G,AA,DR,EPS,MAX,IPRINT,
INPUT,ALAMDA,LLL,MMM)
```

The arguments are as follows:

USER	the identifier of the user subroutine.
N	an integer set to the number of residuals $e_2^!$s.
K	an integer set to the number of variables $(K \leq N)$.
X	a real array of K elements in which the current values of the solution are stored.
E	a real array of N elements in which the current values of the residuals are stored.
A	a two suffix array of KxK elements in which the current values of the matrix $(\underset{\sim}{J}^T\underset{\sim}{J} + \lambda\underset{\sim}{I})$ are stored.
V	a real array of length K is used to store the current values of the matrix $\underset{\sim}{J}^T\underset{\sim}{e}$. This array also stores the values of the step predictions $\Delta\underset{\sim}{x}$ as an output of subroutine POSMAT.
W	a real array of length K is used to temporarily store the current values of the step prediciton $\Delta\underset{\sim}{x}$.
G	a real array of length K is used to store the current values of the gradients of the objective U.
AA	a real array of K(K+1)/2 elements which stores the elements of the matrix $(\underset{\sim}{J}^T\underset{\sim}{J} + \lambda\underset{\sim}{I})$ up to the diagonal elements.
DR	a two suffix array of size NxK in which the current values of the gradients of the residuals $e_i^!$s with respect to the vector $\underset{\sim}{x}$, are stored. i.e. DR(I,J) contains the value of $\frac{\partial e_i}{\partial x_j}$.
EPS	a real array of size K in which tolerance values are stored to be used in the minimization process.

IPRINT	an integer controlling output printing to be set: to the required value to print every IPRINT iteration
INPUT	an integer set to 1 if the input data is to be printed and set to 0 if the input data is to be suppressed
ALAMDA	a real number which sets an initial value for the parameter λ. ALAMDA can take any value between 0.0001 and 20.0. For values less than 0.0001 (greater than 20) the program will automatically set ALAMDA to 0.0001(20). AlAMDA should be high (within the above specified range) if no infromation are present for the problem. For these cases a suitable ALAMDA will be 10.
LLL,MMM	Two working arrays of size K.

(2) The User Subroutine:

The user must provide a subroutine as follows:

```
SUBROUTINE UUU(N,K,X,E,DR)
DIMENSION  X(1), E(1), DR(N,1)
```

where UUU is an identifier chosen by the user. This subroutine should use the variables $\underset{\sim}{x}$ supplied in X, the number of variables supplied in N, and the number of functions (residuals) supplied in K to evaluate the residuals and their partial derivatives, with respect to $\underset{\sim}{x}$, and place them in arrays E, DR, respectively. It should be noted that UUU must be passed to SQUARE as its first argument and it must appear in an EXTERNAL statement in the program that calls SQUARE as will be shown in the examples.

(3) Other Subroutines:

Other subroutines are called by SQUARE. These are as follows:

GRDCHK	checks the derivatives at the starting point [7].
GRADES	evaluates the gradients of the formulated objective function.
POSMAT	solves for $\Delta \underset{\sim}{x}$ and checks for the positive-definitions of $(\underset{\sim}{J}^T \underset{\sim}{J} + \lambda \underset{\sim}{I})$ [8].
DETERM	checks for singularity of $(\underset{\sim}{J}^T \underset{\sim}{J} + \lambda \underset{\sim}{I})$ [8].
OUTPUT	outputs the optimal solution or the solution obtained at the last iteration.

A flow chart which describes the construction of the program is shown in Figure 1.

(V) Illustrative Examples:

Different examples were tried out to test the program. The chosen examples were as follows:

Example 1:

To minimize the function [9]:

$$U = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

which is the familiar Rosenbrock function. This function is in the form of a sum of squares with:

$$U = \sum_{i=1}^{2} e_i^2 \quad , \quad e_1 = 10(x_2 - x_1^2) \quad , \quad e_2 = 1 - x_1 .$$

Different starting points were used (including the standard starting point $\underset{\sim}{x}^o = [-1.2 \ 1.0]^T$). The algorithm was also tested for different starting values of the parameter λ. Figures 2,3 and 4 give a typical calling program, a typical user subroutine and the printout of input data and optimal solution, respectively. Table 1 gives a comparison between the number of function evaluations and the obtained optimal solution for the Rosenbrock function using different starting value for the parameter λ.

Table 1
Results of Rosenbrock Function

λ_{start}	λ_{final}	x_1	x_2	U	No. of functions
0.001	.5000E-3	1.0000000	1.0000000	.454383E-27	31
0.01	.6250E-3	1.0000000	1.0000000	.113595E-25	28
0.05	.7812E-3	1.0000000	1.0000000	.962738E-24	22
1.0	.4882E-3	1.0000000	1.0000000	.201948E-27	23
10.0	.6103E-3	1.0000000	1.0000000	.408945E-26	26

Example 2 [10] :

To minimize the function: $U = 2(x_1-5)^2 + (x_2-6)^2$. Different starting values were tried and Fig. 5 shows the obtained results for this problem using the standard starting point $\underset{\sim}{x}^o = [8.0\ 9.0]^T$. Table 2 shows the effect on the different starting values for λ on the number of function evaluations.

Table 2
Results for the function $U = 10(x_1-5)^2 + (x_2-6)^2$

λ_{start}	λ_{final}	x_1	x_2	U	No. of functions
.0001	.2500E-4	5.0000000	6.0000000	.136517E-24	4
.001	.1250E-3	5.0000000	6.0000000	.807793E-27	5
.01	.1250E-3	5.0000000	6.0000000	.211670E-18	5
0.1	.3125E-2	5.0000000	6.0000000	.572422E-20	7
1.0	.7812E-2	5.0000000	6.0000000	.557924E-17	9
10.0	.9765E-2	5.0000000	6.0000000	.317305E-16	12

Example 3:

It is required to find the best fitting of the data $y(t_1)$, $y(t_2)$, ..., $y(t_{10})$ by a function of the form [11]

$$f(t) = a + bt + c\ \exp(-\tfrac{1}{2}(t-d)^2/e^2)$$

where a,b,c,d and e are parameters to be determined. The problem can be posed as that of choosing the parameters a,b,c,d and e so as to minimize:

$$\sum_{i=1}^{10} \{ e_i(a,b,c,d,e) \}^2$$

where

$$e_i(a,b,c,d,e) = a + bt_i + c \exp(-\tfrac{1}{2}(t_i-d)^2/e^2) - y(t_i).$$

Thus the problem has 10 residuals e_i, 5 variables, a,b,c,d and e, and $t_1, t_2, \ldots, t_{10}$ and $y(t_1), y(t_2), \ldots, y(t_{10})$ are given data. The partial derivatives for these residuals with respect to the variable parameters are obtained as follows:

$$\frac{\partial e_i}{\partial a} = 1 \quad , \quad \frac{\partial e_i}{\partial b} = t_i \quad , \quad \frac{\partial e_i}{\partial c} = \exp(-\tfrac{1}{2}(t_i-d)^2/e^2)$$

$$\frac{\partial e_i}{\partial d} = \frac{c(t_i-d)}{e^2} \exp(-\tfrac{1}{2}(t_i-d)^2/e^2) \quad , \quad \frac{\partial e_i}{\partial e} = \frac{c(t_i-d)^2}{e^3} \exp(-\tfrac{1}{2}(t_i-d)^2/e^2)$$

A typical calling program, a user subroutine and a print out of the optimal solution are shown in Figures 6,7 and 8, respectively. Table 3 gives a comparison between $y(t_i)$ and $f(t_i)$ for $t_1, t_2, \ldots, t_{10}$.

Table 3
Results for the data fitting example

t_i	-0.5	0.0	0.5	1.0	2.0
$y(t_i)$	4.35000	4.93000	5.48000	6.00000	6.95000
$f(t_i)$	4.35256	4.93047	5.47974	5.99987	6.94526
t_i	5.0	10.0	20.0	40.0	50.0
$y(t_i)$	9.17500	12.59000	22.01000	42.00200	52.00100
$f(t_i)$	9.18045	12.5858	22.01180	42.00320	51.99990

Example 4 [10] :

To minimize $U = (x_1+10x_2)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4 + 5(x_3-x_4)^2$

Figure 9 gives a print out for the results of this example.

Example 5:

Find a second-order model of a fourth-order system when the input to the system is an impulse, in the least-square sense [7] .

The transfer function of a fourth-order system is

$$G(s) = \frac{(s+4)}{(s+1)(s^2+4s+8)(s+5)}$$

and the transfer function of the second-order model is

$$H(s) = \frac{x_3}{(s+x_1)^2 + x_2^2}$$

The problem is therefore equivalent to finding the optimum point $\underset{\sim}{x}$ such that the function:

$$F(\underset{\sim}{x},t) = \frac{x_3}{x_2} \exp(-x_1 t) \sin x_2 t$$

best approximates the function:

$$S(t) = \frac{3}{20} \exp(-t) + \frac{1}{52} \exp(-5t) - \frac{\exp(-2t)}{65} (3\sin 2t + 11 \cos 2t)$$

in the <u>least-square</u> sense.

The problem was discretized into 51 uniformly spaced points in the interval 0 to 10 seconds and the function to be minimized is given by:

$$U = \sum_{i=1}^{51} e_i^2(\underset{\sim}{x})$$

where $e_i(\underset{\sim}{x}) = F(\underset{\sim}{x},t_i) - S(t_i)$

The following results were obtained:

$$U = 4.3682967 \times 10^{-4}$$

$$\underset{\sim}{x} = \begin{bmatrix} 1.0164706 \\ 0.7892702 \\ 0.1614000 \end{bmatrix}$$

The number of function evaluations were 13(28) using a starting value for λ of 0.001(1.0). Fig. 10 shows the obtained results for this system modelling example.

(VI) Comments on the Obtained Results:

Looking at the obtained results for the test examples used it can be concluded that the algorithm works very well for minimizing functions in the form of a sum of squares. The control on the parameter λ at the start of the minimization procedure is not that critical and different starting values ranging from 0.0001 up to 20.0 can be used.

The program written can be used in its form now for curve fitting problems especially for statistical analysis and modelling problems. It should be noted that the results of example 5 are different from those obtained using the minimax sense which is true when the number of sampling points is larger than the number of residuals.

(VII) Acknowledgements:

The author is greatly indebted to the encouragement and useful suggestions from Dr. J.W. Bandler during the course of this work.

(VIII) References:

[1] Massara, R.E. and Fidler, J.K., "Efficient Damping Method for Least-Squares Algorithms", Elect. Lett., Vol. 11, No. 2, Jan. 75, pp. -3-34.

[2] Bandler, J.W., "Optimization Methods for Computer-Aided Design", IEEE Trans., MTT-17, 1969, pp. 533-552.

[3] Temes, G.C. in Kuo, F.C. and Magnuson, W.G. (eds.), "Computer Oriented Circuit Design", Prentice Hall, 1969.

[4] Bown, G.C.S. and Geiger, G.V., "Design and Optimization of Circuits by Computer", Proc. IEEE, Vol. 118, No. 5, 1971, pp. 649-661.

[5] Levenberg, K., "A Method For the Solution of Certain Non-Linear Problems in Least Squares", Quart. J. Appl. Math., Vol. 2, 1944, pp. 164-168.

[6] Marquardt, D.W., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters", SIAM J., Vol. 11, No. 2, June 1963, pp. 431-441.

[7] Bandler, J.W., Charalambous, C., Chen, J., "MINOPT - An Optimization Program Based on Recent Minimax Results", G-SOC.No.70, McMaster University, Dec. 1974.

[8] "McMaster University Program Library Index", Publication No. 5.1, April 1974.

[9] Rosenbrock, H.H., "An Automatic Method for Finding the Greatest or Least Value of a Function", Compt. J., Vol. 3, pp. 175-184, 1960.

[10] Himmelblau, "Applied Nonlinear Programming", McGraw-Hill Book Comp.,1972.

[11] Fletcher, R., "A Modified Marquardt Subroutine for Nonlinear Least-Squares", UKAEA Research Group Report, AERER. 6799, 1971.

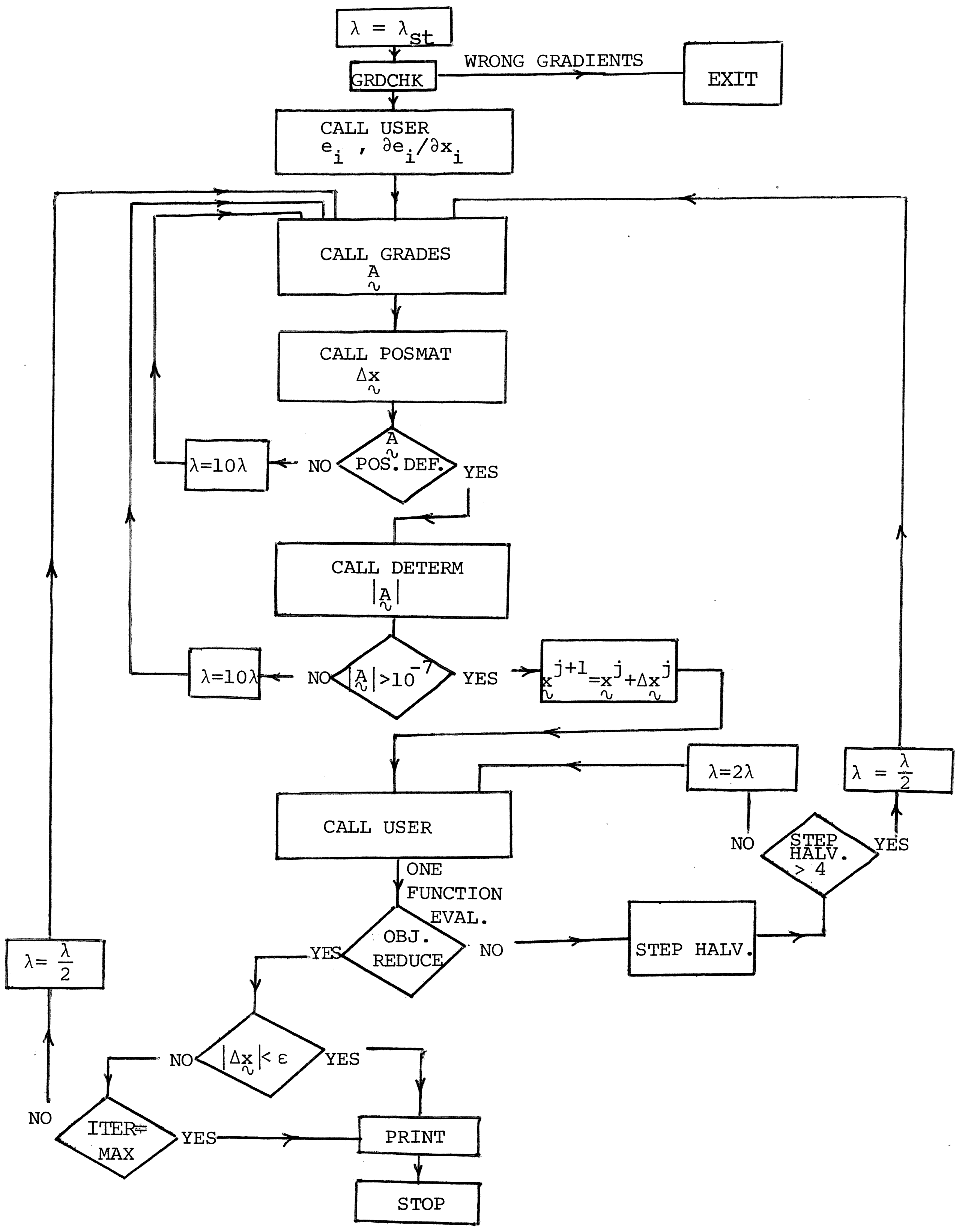

Fig. 1 Flow chart for SQUARE.

```
 DIMENSION X(2),E(2),A(2,2),AA(3),EPS(2),G(2),V(2),DR(2,2),VV(2),
1LLL(2),MMM(2)
 EXTERNAL USER
 ALAMDA=1.0
 X(1)=-1.2
 X(2)=1.0
 EPS(1)=1.E-8
 EPS(2)=1.E-8
 N=2
 K=2
 MAX=200
 IPRINT=4
 INPUT=1
 CALL SQUARE(USER,N,K,X,E,A,V,VV,G,AA,DR,EPS,MAX,IPRINT,INPUT,
1ALAMDA,LLL,MMM)
 STOP
 END


                         CDTOT    0017
```

Fig. 2   The main program for Rosenbrock's problem.

```
      SUBROUTINE USER(N,K,X,E,DR)
      DIMENSION X(1),E(1),DR(N,1)
      S1=X(2)-X(1)*X(1)
      E(1)=10.*S1
      E(2)=1.-X(1)
      DR(1,1)=-20.*X(1)
      DR(1,2)=10.
      DR(2,1)=-1.0
      DR(2,2)=0.0
      RETURN
      END


                                    CDTOT   0011
```

Fig. 3 The user subroutine for Rosenbrock's problem.

```
INPUT DATA
----------

NUMBER OF INDEPENDENT VARIABLES........................K =   2
MAXIMUM NUMBER OF ALLOWABLE ITERATIONS..............MAX = 200
INTERMEDIATE PRINTOUT AT EVERY IPRINT ITERATIONS..IPRINT =   4
STARTING VALUES FOR VECTOR X(I).....................X( 1)=  -.120000E+01
                                                    X( 2)=   .100000E+01

TEST QUANTITIES TO BE USED........................EPS( 1)=   .100000E-07
                                                  EPS( 2)=   .100000E-07

STARTING VALUE FOR THE DAMPING FACTOR LAMDA..............=   .100000E+01




THE OPTIMAL SOLUTION OBTAINED USING MASSARA-FIDLER ALGORITHM .
---------------------------------------------------------------

 THE MINIMUM SUM OF SQUARES (OBJECTIVE FUNCTION) =     .14462029E-22

 X( 1)=     .10000000E+01                  G( 1)=    -.33395509E-11

 X( 2)=     .10000000E+01                  G( 2)=    -.21316282E-11

 THE NUMBER OF ITERATIONS PERFORMED..............=     15

 THE NUMBER OF FUNCTION EVALUATIONS..............=     22

 FINAL VALUE OF PARAMETER LAMDA..................=     .976563E-03
```

Fig. 4   Results obtained for Rosenbrock's problem.

```
INPUT DATA
----------

NUMBER OF INDEPENDENT VARIABLES.........................K =   2
MAXIMUM NUMBER OF ALLOWABLE ITERATIONS...............MAX = 100
INTERMEDIATE PRINTOUT AT EVERY IPRINT ITERATIONS..IPRINT =   1
STARTING VALUES FOR VECTOR X(I)....................X( 1)=   .800000E+01
                                                   X( 2)=   .900000E+01

TEST QUANTITIES TO BE USED.......................EPS( 1)=   .100000E-05
                                                 EPS( 2)=   .100000E-05

STARTING VALUE FOR THE DAMPING FACTOR LAMDA.............=   .100000E+01




THE OPTIMAL SOLUTION OBTAINED USING MASSARA-FIDLER ALGORITHM .
--------------------------------------------------------------

 THE MINIMUM SUM OF SQUARES (OBJECTIVE FUNCTION) =     .55792472E-17

 X( 1)=     .50000000E+01                  G( 1)=     .68212103E-12

 X( 2)=     .60000000E+01                  G( 2)=     .47240860E-08

 THE NUMBER OF ITERATIONS PERFORMED..............=      7

 THE NUMBER OF FUNCTION EVALUATIONS..............=      9

 FINAL VALUE OF PARAMETER LAMDA..................=     .781250E-02
```

Fig. 5 Results for Example 2.

```
      DIMENSION X(5),E(10),V(5),G(5),DR(10,5),A(5,5),AA(15),EPS(5),VV(5)
     1,LLL(5),MMM(5)
      COMMON/AA/T(10),Y(10)
      EXTERNAL USER
      ALAMDA=0.01
      READ(5,10) (T(I),I=1,10)
      READ(5,10) (Y(I),I=1,10)
   10 FORMAT(10F7.3)
      X(1)=1.0
      X(2)=1.0
      X(3)=0.0
      X(4)=5.0
      X(5)=3.0
      EPS(1)=1.E-6
      EPS(2)=1.E-6
      EPS(3)=1.E-6
      EPS(4)=1.E-6
      EPS(5)=1.E-6
      N=10
      K=5
      MAX=200
      IPRINT=4
      INPUT=1
      CALL SQUARE(USER,N,K,X,E,A,V,VV,G,AA,DR,EPS,MAX,IPRINT,INPUT,
     1ALAMDA,LLL,MMM)
      STOP
      END
```

CDTOT 0027

Fig. 6 The main program for the data fitting problem.

```
      SUBROUTINE USER(N,K,X,E,DR)
      DIMENSION X(1),E(1),DR(N,1)
      DIMENSION EX(10)
      COMMON/AA/T(10),Y(10)
      IF(X(5).EQ.0.0) GO TO 3
      DO 1 I=1,N
      SS=T(I)-X(4)
      EX(I)=EXP(-0.5*(SS)/X(5))**2)
    1 E(I)=X(1)+X(2)*T(I)+X(3)*EX(I)-Y(I)
      DO 2 I=1,N
      DR(I,1)=1.0
      DR(I,2)=T(I)
      DR(I,3)=EX(I)
      DR(I,4)=X(3)*SS*EX(I)/X(5)**2
    2 DR(I,5)=X(3)*SS**2*EX(I)/X(5)**2
      GO TO 5
    3 WRITE(6,4)
    4 FORMAT(1H1,2X,*OVERFLOW WILL OCCUR - X(5) IS ZERO *,/)
    5 RETURN
      END
```

CDTOT 0020

Fig. 7 The user subroutine for the data fitting problem.

```
INPUT DATA
----------

NUMBER OF INDEPENDENT VARIABLES........................K =   5
MAXIMUM NUMBER OF ALLOWABLE ITERATIONS..............MAX = 200
INTERMEDIATE PRINTOUT AT EVERY IPRINT ITERATIONS..IPRINT =   4
STARTING VALUES FOR VECTOR X(I)....................X( 1)=   .100000E+01
                                                   X( 2)=   .100000E+01
                                                   X( 3)=  0.
                                                   X( 4)=   .500000E+01
                                                   X( 5)=   .300000E+01

TEST QUANTITIES TO BE USED.......................EPS( 1)=   .100000E-05
                                                 EPS( 2)=   .100000E-05
                                                 EPS( 3)=   .100000E-05
                                                 EPS( 4)=   .100000E-05
                                                 EPS( 5)=   .100000E-05

STARTING VALUE FOR THE DAMPING FACTOR LAMDA.............=   .100000E-01


THE OPTIMAL SOLUTION OBTAINED USING MASSARA-FIDLER ALGORITHM .
--------------------------------------------------------------

 THE MINIMUM SUM OF SQUARES (OBJECTIVE FUNCTION) =      .83562756E-04

 X( 1)=      .20166960E+01                G( 1)=    -.52741096E-05

 X( 2)=      .99966309E+00                G( 2)=    -.36034865E-04

 X( 3)=      .29827640E+01                G( 3)=    -.26135666E-05

 X( 4)=      .10640235E+01                G( 4)=    -.11279473E-05

 X( 5)=      .49182090E+01                G( 5)=    -.76475244E-05

 THE NUMBER OF ITERATIONS PERFORMED.............=      50

 THE NUMBER OF FUNCTION EVALUATIONS.............=      53

 FINAL VALUE OF PARAMETER LAMDA.................=      .177636E-16
```

Fig. 8   Results obtained for the data fitting problem.

```
INPUT DATA
----------

NUMBER OF INDEPENDENT VARIABLES.........................K =   4
MAXIMUM NUMBER OF ALLOWABLE ITERATIONS...............MAX = 100
INTERMEDIATE PRINTOUT AT EVERY IPRINT ITERATIONS..IPRINT =   1
STARTING VALUES FOR VECTOR X(I)...................X( 1)=   .100000E+01
                                                  X( 2)=   .100000E+01
                                                  X( 3)=   .100000E+01
                                                  X( 4)=   .100000E+01

TEST QUANTITIES TO BE USED......................EPS( 1)=   .100000E-05
                                                EPS( 2)=   .100000E-05
                                                EPS( 3)=   .100000E-05
                                                EPS( 4)=   .100000E-05

STARTING VALUE FOR THE DAMPING FACTOR LAMDA.............=   .100000E+01


THE OPTIMAL SOLUTION OBTAINED USING MASSARA-FIDLER ALGORITHM .
---------------------------------------------------------------

 THE MINIMUM SUM OF SQUARES (OBJECTIVE FUNCTION) =     .70898857E-13

 X( 1)=     .42968113E-03               G( 1)=     .43811715E-09

 X( 2)=    -.42968111E-04               G( 2)=    -.43082790E-10

 X( 3)=     .21413591E-03               G( 3)=     .21711217E-09

 X( 4)=     .21413597E-03               G( 4)=     .21949612E-09

 THE NUMBER OF ITERATIONS PERFORMED.............=      75

 THE NUMBER OF FUNCTION EVALUATIONS.............=      77

 FINAL VALUE OF PARAMETER LAMDA.................=     .264698E-03
```

Fig. 9   Results for Example 3.

```
INPUT DATA
----------

NUMBER OF INDEPENDENT VARIABLES.........................K =    3
MAXIMUM NUMBER OF ALLOWABLE ITERATIONS...............MAX = 200
INTERMEDIATE PRINTOUT AT EVERY IPRINT ITERATIONS..IPRINT =    1
STARTING VALUES FOR VECTOR X(I)......................X( 1)=   .100000E+01
                                                     X( 2)=   .100000E+01
                                                     X( 3)=   .100000E+01

TEST QUANTITIES TO BE USED.........................EPS( 1)=   .100000E-05
                                                   EPS( 2)=   .100000E-05
                                                   EPS( 3)=   .100000E-05

STARTING VALUE FOR THE DAMPING FACTOR LAMDA...............=   .100000E+01


THE OPTIMAL SOLUTION OBTAINED USING MASSARA-FIDLER ALGORITHM .
--------------------------------------------------------------

 THE MINIMUM SUM OF SQUARES (OBJECTIVE FUNCTION) =      .43682967E-03

 X( 1)=      .10164707E+01                  G( 1)=    -.80055246E-11

 X( 2)=      .78927021E+00                  G( 2)=    -.27904939E-09

 X( 3)=      .16140009E+00                  G( 3)=    -.52957023E-12

 THE NUMBER OF ITERATIONS PERFORMED..............=      16

 THE NUMBER OF FUNCTION EVALUATIONS..............=      18

 FINAL VALUE OF PARAMETER LAMDA..................=      .152588E-04
```

Fig. 10   Results obtained for the system modelling problem.

APPENDIX I

FORTRAN listing for * SQUARE *

```
      SUBROUTINE SQUARE (USER,N,K,X,E,A,V,VV,G,AA,DR,EPS,MAX,IPRINT,INPU
     1T,ALAMDA,LLL,MMM)
C
C     THIS IS THE ORGANIZING SUBROUTINE FOR THE PACKAGE * SQUARE *. THE
C     PACKAGE MINIMIZES A FUNCTION IN THE FORM OF SUM OF SQUARES USING
C     THE MODIFIED MARQUARDT-LEVENBERG DAMPING TECHNIQUE. THE WAY IN
C     WHICH THE DAMPING FACTOR (LAMDA) VARIES IS THE SAME AS THAT  DES-
C     CRIBED BY MASSARA-FIDLER (ELECTRONIC LETTERS, VOL.11, PP 33). THE
C     FACTOR LAMDA IS USED TO ....
C     1) BIAS THE SEARCH DIRECTION TOWARDS THE STEEPEST DESCENT AT THE
C       START OF THE MINIMIZATION PROCESS...,
C     2) BIAS THE SEARCH DIRECTION TOWARDS THE DIRECTION PREDICTED BY
C        GAUSS APPROXIMATION AS THE MINIMIZATION PROCESS GOES...,
C     3) RESTRICT THE MATRIX OF COEFFICIENTS TO BE POSITIVE DEFINITE...,
C     4) RESTRICT THE COEFFICIENT MATRIX TO BE NON-SINGULAR.
C
      DIMENSION X(1), E(1), A(K,1), AA(1), EPS(1), G(1), V(1), DR(N,1),
     1VV(1), LLL(1), MMM(1)
      EXTERNAL USER
      IHALV=0
      IT=0
      IFC=0
      IF (ALAMDA.LT.0.001) ALAMDA=0.001
      IF (ALAMDA.GT.20.) ALAMDA=20.0
      CALL GRDCHK (USER,N,K,X,E,A,V,G,DR,ALAMDA)
      IF (INPUT.EQ.0) GO TO 1
      WRITE (6,23)
      WRITE (6,24) K
      WRITE (6,25) MAX
      WRITE (6,26) IPRINT
      WRITE (6,27) (I,X(I),I=1,K)
      WRITE (6,28) (I,EPS(I),I=1,K)
      WRITE (6,29) ALAMDA
    1 CALL USER (N,K,X,E,DR)
      IFC=IFC+1
      USUM1=0.0
      DO 2 I=1,N
      USUM1=USUM1+E(I)*E(I)
    2 CONTINUE
      WRITE (6,30)
    3 CALL GRADES (N,K,X,E,A,V,G,DR,ALAMDA)
      ISTEP=0
      L=0
      DO 4 J=1,K
      DO 4 JJ=1,J
      L=L+1
    4 AA(L)=A(J,JJ)
      CALL POSMAT (AA,K,IERR,V)
      IF (IERR.EQ.0) GO TO 5
      ALAMDA=ALAMDA*10.
      GO TO 3
    5 DO 6 I=1,K
      II=I+1
      DO 6 III=II,K
    6 A(I,III)=A(III,I)
      CALL DETERM (A,K,DET,LLL,MMM)
      IF (DET.GT.1.E-6) GO TO 7
      ALAMDA=ALAMDA*10.
      GO TO 3
```

```
7     DO 8 I=1,K
      X(I)=X(I)+V(I)
8     CONTINUE
9     CALL USER (N,K,X,E,DR)
      IFC=IFC+1
      USUM2=0.0
      DO 10 I=1,N
      USUM2=USUM2+E(I)*E(I)
10    CONTINUE
      IF (ISTEP.EQ.1) GO TO 12
      DO 11 I=1,K
      VV(I)=V(I)
11    CONTINUE
12    CALL GRADES (N,K,X,E,A,V,G,DR,ALAMDA)
      IF (IT.GE.MAX) GO TO 20
      IF (MOD(IT,IPRINT).NE.0) GO TO 13
      WRITE (6,31) IT,IFC,ALAMDA,USUM2,((X(I),G(I)),I=1,K)
13    DIFF=USUM2-USUM1
      IF (DIFF) 14,14,17
14    IHALV=0
      IT=IT+1
      USUM1=USUM2
      DO 15 I=1,K
      ERROR=ABS(VV(I))-EPS(I)
      IF (ERROR) 15,15,16
15    CONTINUE
      GO TO 21
16    ALAMDA=ALAMDA*0.5
      GO TO 3
17    IF (IHALV.EQ.0) NN=1
      IF (IHALV.EQ.1) NN=NN+1
      IHALV=1
      IF (NN.LT.4) GO TO 18
      IHALV=0
      ALAMDA=ALAMDA*0.5
      GO TO 3
18    DO 19 I=1,K
      VV(I)=0.5*VV(I)
      X(I)=X(I)-VV(I)
19    CONTINUE
      ALAMDA=ALAMDA*2.0
      ISTEP=1
      GO TO 9
20    IIT=1
      IT=IT-1
      CALL OUTPUT (K,X,USUM2,IT,IFC,IIT,G,ALAMDA)
      GO TO 22
21    IIT=2
      IT=IT-1
      CALL OUTPUT (K,X,USUM2,IT,IFC,IIT,G,ALAMDA)
      GO TO 22
22    RETURN
C
C
23    FORMAT (1H1,*INPUT DATA*,/,1H ,10(*-*),/)
24    FORMAT (1H0,*NUMBER OF INDEPENDENT VARIABLES*,24(*.*),*K =*,I4,/)
25    FORMAT (1H ,*MAXIMUM NUMBER OF ALLOWABLE ITERATIONS*,15(*.*),*MAX
     1=*,I4,/)
```

```
26    FORMAT (1H ,*INTERMEDIATE PRINTOUT AT EVERY IPRINT ITERATIONS*,2(*
     1.*),*IPRINT =*,I4,/)
27    FORMAT (1H ,*STARTING VALUES FOR VECTOR X(I)*,21(*.*),80(*X(*,I2,*
     1)=*,E14.6,//,53X),/)
28    FORMAT (1H ,*TEST QUANTITIES TO BE USED*,24(*.*),80(*EPS(*,I2,*)=*
     1,,E14.6,//,51X),/)
29    FORMAT (1H ,*STARTING VALUE FOR THE DAMPING FACTOR LAMDA*,14(*.*),
     1*=*,E14.6,/)
30    FORMAT (1H1,1X,*THE DETAILED RESULTS OF THE ALGORITHM.*,/,2X,38(*-
     1*),//,96(*-*),/,1X,*ITER*,10X,*FUNC*,10X,*LAMDA*,10X,*OBJECTIVE*,1
     21X,*VARIABLE*,12X,*GRADIENT*,/,2X,*NO.*,10X,*EVAL*,25X,*FUNCTION*,
     314X,*X(I)*,17X,*G(I)*,/,96(*-*),/)
31    FORMAT (1H ,I4,10X,I4,5X,E14.6,3X,E14.6,6X,80(E14.6,7X,E14.6,/,61X
     1))
      END


                              CDTOT   0132
      SUBROUTINE GRDCHK (USER,N,K,X,E,A,V,G,DR,ALAMDA)
C
C     THIS IS THE SUBROUTINE WHICH CHECKS FOR THE GRADIENTS OF THE
C     SUPPLIED FUNCTIONS AT THE STARTING POINT USING NUMERICAL
C     PURTURBATION. IT IS A MODIFIED VERSION OF GRDCHK OF * MINOPT *
C
      DIMENSION X(1), G(1), E(1), A(K,1), V(1), DR(N,1)
      IC=0
      CALL USER (N,K,X,E,DR)
      CALL GRADES (N,K,X,E,A,V,G,DR,ALAMDA)
      WRITE (6,5)
      WRITE (6,6)
      DO 3 I=1,K
      DX=1.E-4*X(I)
      IF (ABS(DX).LT.1.E-10) DX=1.E-10
      X(I)=X(I)+DX
      CALL USER (N,K,X,E,DR)
      F2=0.0
      DO 1 II=1,N
      F2=F2+E(II)*E(II)
1     CONTINUE
      X(I)=X(I)-2.*DX
      CALL USER (N,K,X,E,DR)
      F1=0.0
      DO 2 KK=1,N
      F1=F1+E(KK)*E(KK)
2     CONTINUE
      Y=0.5*(F2-F1)/DX
      X(I)=X(I)+DX
      IF (ABS(Y).LT.1.E-14) Y=1.E-14
      IF (ABS(G(I)).LT.1.E-14) G(I)=1.E-14
      YP=ABS((Y-G(I))/Y)*100.0
      WRITE (6,7) G(I),Y,YP
      IF (YP.GT.10.0) IC=1
3     CONTINUE
      IF (IC.EQ.1) GO TO 4
      WRITE (6,8)
      RETURN
4     WRITE (6,9)
      CALL EXIT
C
C
5     FORMAT (1H1,3X,*GRADIENTS CHECK AT STARTING POINT*,/,1H ,3X,32(*-*
     1))
6     FORMAT (1H0,3X,67(*-*),/,5X,*ANALYTICAL GRADIENTS*,5X,*NUMERICAL G
     1RADIENTS*,5X,*PERCENTAGE ERROR*,/,4X,67(*-*),/)
7     FORMAT (1H ,6X,E14.6,10X,E14.6,8X,E14.6)
8     FORMAT (1H0,3X,67(*-*),//,1H0,1X,*YOUR GRADIENTS ARE O.K.*)
9     FORMAT (1H0,*YOUR PROGRAM HAS BEEN TERMINATED BECAUSE YOUR GRADIEN
     1TS ARE NOT CORRECT*,/,1H ,*WOULD YOU PLEASE CHECK IT AGAIN*)
      END
```

CDTOT 0051

```
      SUBROUTINE GRADES (N,K,X,E,A,V,G,DR,ALAMDA)
C
C     THIS IS THE SUBROUTINE WHICH FORMULATES THE MATRIX OF COEFFICIENTS
C     IT ALSO EVALUATES THE GRADIENTS OF THE FORMULATED OBJECTIVE
C
      DIMENSION X(1), E(1), A(K,1), V(1), G(1), DR(N,1)
      FF=FLOAT(N)
      FN=1.0/FF
      DO 1 I=1,K
      DO 1 J=1,I
      A(I,J)=0.0
      DO 1 II=1,N
      A(I,J)=A(I,J)+DR(II,I)*DR(II,J)
      IF (I.EQ.J) A(I,J)=A(I,J)+FN*ALAMDA
1     CONTINUE
      DO 2 I=1,K
      V(I)=0.0
      DO 2 J=1,N
2     V(I)=V(I)+E(J)*DR(J,I)
      DO 3 I=1,K
      G(I)=2.*V(I)
      V(I)=-V(I)
3     CONTINUE
      RETURN
      END
```

CDTOT   0025

```
      SUBROUTINE POSMAT (A,N,IERR,X)
C
C     THIS IS THE SUBROUTINE WHICH CHECKS FOR THE POSITIVE DEFINITENESS
C     OF THE MATRIX OF COEFFICIENTS. FOR REFERENCE SEE  MCMASTER UNIV.
C     PROGRAM LIBRARY INDEX
C
      DIMENSION A(1), X(1)
      IERR=N
      DO 5 I=1,N
      II=I*(I-1)/2
      IN=I-1
      IP=II+I
      IF (I.EQ.1) GO TO 2
      DO 1 K=1,IN
      IK=II+K
      A(IP)=A(IP)-A(IK)**2
1     CONTINUE
2     IF (A(IP).LE.0.) RETURN
      A(IP)=SQRT(A(IP))
      IF (I.EQ.N) GO TO 6
      IM=I+1
      DO 4 J=IM,N
      JJ=J*(J-1)/2
      IJ=JJ+I
      IF (I.EQ.1) GO TO 4
      DO 3 K=1,IN
      IK=II+K
      JK=JJ+K
      A(IJ)=A(IJ)-A(IK)*A(JK)
3     CONTINUE
4     A(IJ)=A(IJ)/A(IP)
5     CONTINUE
6     DO 9 I=1,N
      II=I*(I-1)/2
      IM=I+1
      IP=II+I
      A(IP)=1./A(IP)
      IF (I.EQ.N) GO TO 10
      DO 8 J=IM,N
      J1=J-1
      JJ=J*J1/2
      AA=0.0
      DO 7 K=I,J1
      KJ=JJ+K
      KK=K*(K-1)/2
      IK=KK+I
      AA=AA+A(IK)*A(KJ)
7     CONTINUE
      IJ=JJ+I
      IP=J+JJ
      A(IJ)=-AA/A(IP)
8     CONTINUE
9     CONTINUE
10    IERR=0
      I=N
      DO 12 II=1,N
      AA=0.0
      IJ=(I*(I+1))/2
      J=I
      DO 11 JJ=1,I
      AA=AA+A(IJ)*X(J)
      J=J-1
      IJ=IJ-1
11    CONTINUE
      X(I)=AA
      I=I-1
12    CONTINUE
      DO 14 I=1,N
      AA=0.0
      IJ=(I*(I+1))/2
      DO 13 J=I,N
      AA=AA+A(IJ)*X(J)
      IJ=IJ+J
13    CONTINUE
      X(I)=AA
14    CONTINUE
      RETURN
      END
```

CDTOT   0078

```
      SUBROUTINE DETERM (A,N,D,L,M)
C
C     THIS IS THE SUBROUTINE WHICH CHECKS FOR THE SINGULARITY OF THE
C     MATRIX OF COEFFICIENTS. FOR REFERENCE SEE  MCMASTER UNIVERSITY
C     PROGRAM LIBRARY INDEX
C
      DIMENSION A(1), L(1), M(1)
      D=1.0
      NK=-N
      DO 18 K=1,N
      NK=NK+N
      L(K)=K
      M(K)=K
      KK=NK+K
      BIGA=A(KK)
      DO 2 J=K,N
      IZ=N*(J-1)
      DO 2 I=K,N
      IJ=IZ+I
      IF (ABS(BIGA)-ABS(A(IJ))) 1,2,2
1     BIGA=A(IJ)
      L(K)=I
      M(K)=J
2     CONTINUE
      J=L(K)
      IF (J-K) 5,5,3
3     KI=K-N
      DO 4 I=1,N
      KI=KI+N
      HOLD=-A(KI)
      JI=KI-K+J
      A(KI)=A(JI)
      A(JI)=HOLD
4     CONTINUE
5     I=M(K)
      IF (I-K) 8,8,6
6     JP=N*(I-1)
      DO 7 J=1,N
      JK=NK+J
      JI=JP+J
      HOLD=-A(JK)
      A(JK)=A(JI)
      A(JI)=HOLD
7     CONTINUE
8     IF (BIGA) 10,9,10
9     D=0.0
      RETURN
10    DO 12 I=1,N
      IF (I-K) 11,12,11
11    IK=NK+I
      A(IK)=A(IK)/(-BIGA)
12    CONTINUE
      DO 15 I=1,N
      IK=NK+I
      HOLD=A(IK)
      IJ=I-N
      DO 15 J=1,N
      IJ=IJ+N
      IF (I-K) 13,15,13
```

```
13     IF (J-K) 14,15,14
14     KJ=IJ-I+K
       A(IJ)=HOLD*A(KJ)+A(IJ)
15     CONTINUE
       KJ=K-N
       DO 17 J=1,N
       KJ=KJ+N
       IF (J-K) 16,17,16
16     A(KJ)=A(KJ)/BIGA
17     CONTINUE
       D=D*BIGA
       A(KK)=1.0/BIGA
18     CONTINUE
       K=N
19     K=K-1
       IF (K) 26,26,20
20     I=L(K)
       IF (I-K) 23,23,21
21     JQ=N*(K-1)
       JR=N*(I-1)
       DO 22 J=1,N
       JK=JQ+J
       HOLD=A(JK)
       JI=JR+J
       A(JK)=-A(JI)
       A(JI)=HOLD
22     CONTINUE
23     J=M(K)
       IF (J-K) 19,19,24
24     KI=K-N
       DO 25 I=1,N
       KI=KI+N
       HOLD=A(KI)
       J1=K1-K+J
       A(KI)=-A(JI)
       A(JI)=HOLD
25     CONTINUE
       GO TO 19
26     RETURN
       END
```

CDTOT   0099

```
      SUBROUTINE OUTPUT (K,X,USUM2,IT,IFC,IIT,G,ALAMDA)
C
C     THIS IS THE SUBROUTINE WHICH OUTPUTS THE RESULTS OF THE PACKAGE
C
      DIMENSION X(1), G(1)
      IF (IIT.EQ.1) WRITE (6,2)
      IF (IIT.EQ.2) WRITE (6,3)
      WRITE (6,4) USUM2
      DO 1 I=1,K
      WRITE (6,5) I,X(I),I,G(I)
1     CONTINUE
      WRITE (6,6) IT
      WRITE (6,7) IFC
      WRITE (6,8) ALAMDA
      RETURN
C
C
2     FORMAT (1H1,15X,25HRESULTS AT LAST ITERATION,/,16X,25(*-*))
3     FORMAT (1H1,10X,*THE OPTIMAL SOLUTION OBTAINED USING MASSARA-FIDLE
     1R ALGORITHM .*,/,11X,62(*-*),/)
4     FORMAT (1H0,11X,*THE MINIMUM SUM OF SQUARES (OBJECTIVE FUNCTION) =
     1*,2X,E16.8,/)
5     FORMAT (1H0,11X,*X(*,I2,*)=*,2X,E16.8,19X,*G(*,I2,*)=*,2X,E16.8,/)
6     FORMAT (1H0,11X,*THE NUMBER OF ITERATIONS PERFORMED..............=
     1*,2X,I5,/)
7     FORMAT (1H0,11X,*THE NUMBER OF FUNCTION EVALUATIONS..............=
     1*,2X,I5,/)
8     FORMAT (1H0,11X,*FINAL VALUE OF PARAMETER LAMDA..................=
     1*,2X,E14.6)
      END
```

CDTOT 0030