FNFIT: AN EAST-TO-USE, ARBITRARY FUNCTION-TO-DATA FITTING ROUTINE (A USER'S MANUAL)

J. Terrence Klopcic

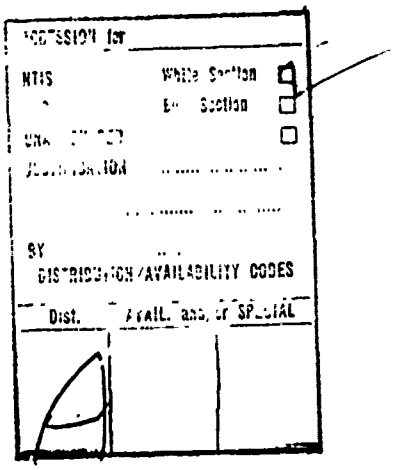
Ballistic Research Laboratories
Aberdeen Proving Ground, Maryland

August 1974

Destroy this report when it is no longer needed.
Do not return it to the originator.

Secondary distribution of this report by originating or sponsoring activity is prohibited.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22151.



The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD/A-000 655

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BRL MEMORANDUM REPORT NO. 2402	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FNFIT: AN EASY-TO-USE, ARBITRARY FUNCTION-TO-DATA FITTING ROUTINE (A USER'S MANUAL)		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J. Terrence Klopcic		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS USA Ballistic Research Laboratories Aberdeen Proving Ground, Maryland 21005		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS RDT&E 1W062117AD51
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Materiel Command 5001 Eisenhower Avenue Alexandria, Virginia 22304		12. REPORT DATE AUGUST 1974 13. NUMBER OF PAGES 46
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Function Fitting
Least Squares Analysis
Data Fitting

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The program FNFIT uses both steepest gradient, and chi-square minimization searches to fit a user-chosen, arbitrary (including nonlinear) function to a set of input data points. Options are included to allow control of the search for particularly pathological functions, making the program quite versatile. However, the program can also be run without the options, providing an easy-to-learn-and-use tool for the pedestrian.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

I. INTRODUCTION

A most common technique in data analysis involves fitting a function, the form of which is dictated by theoretical considerations, to experimental data. The parameters of the function, which are adjusted to provide the best fit to the data, are related to the physical quantities which the experiment sought to measure; hence, the values of these parameters become the desired results of the experiment. In particular, most of the experiments conducted at this Laboratory, including recent work on energy and rate dependence of solid state x-ray dosimeters, response of ferroelectric dosimeters, yield outputs which vary with changes in experimental conditions. By a theoretical analysis of the mechanisms involved in the interaction of radiation with these dosimeters, their expected responses have been described as functions of the experimental conditions. From the parameters of these functions, the energy, dose and rate dependence, time dependence, electrical characteristics, have been determined. In some cases the failure of the expected function to satisfactorily fit the data has resulted in the discovery of other mechanisms involved in the response, and in the subsequent generation of a more complete functional description.

Because of the broad application of this kind of analysis, data fitting routines are commonplace; most, however, involve fitting a particular functional form, such as a power series or Fourier series. For experiments whose functional behaviour is not such a series, the subsequent extraction of the desired quantities may be very difficult.

FNFIT is designed to allow the analyst to input his own functional form, and reap those parameters which are directly applicable to the experiment. The program was intended to be quick and easy-to-use for the analyst who has only occasional need to fit a simple function, yet versatile enough to handle complicated functions, and functions which change erratically with small changes in parameters. The user supplied function, G (x; A(1), A(2)...) can contain up to 20 parameters, A(J), and up to 1000 data points, [X(I), Y(I)], can be input.

Five or six sets (the sixth being optional) of cards are read in to run the program. These sets input the following:

EXEC A. The function, G.
EXEC B. Assemble and execute instructions
SET A. Control information
SET B. Starting values for the parameters
SET C. Limits on the parameters (Optional)
SET D. The Data

These card sets are described, along with some "convenience" options, are noted in the following section, ("PROGRAM INPUT").

The program, at the end of the run, outputs the parameters, the data points and function value at each point, and the derivative of the function at each point. Various outputs are available at the beginning of each iteration.

II. PROGRAM INPUT

A. Blank (Free-Field) Format

Since FNFIT makes use of the BLANK FORMAT feature of the 1108, a note of description is in order. In BLANK format, the computer, directed by a READ statement, begins to scan a record (card), interpreting whatever it finds as the type of variable sought by the statement; the field ends at a comma or at the end of the record. Blanks are usually ignored, except an entirely blank field is read as integer zero.

For example:
```
READ (5,9) A, B, J
    9 FORMAT ( )
```
would read
```
9.0       , 6.7E-1,7
```
or
```
9.0       , 6.7-1
7
```

identically, giving the values

A = 9.0, B = .67, and J = 7

The advantages of using BLANK Format are:

1. Numbers need not be carefully placed in fields on the cards. One merely punches the numbers, separated by commas.

2. The same read and format statement can read data on one or several cards, decided by the user at data input time. This is useful in FNFIT.

One precaution to remember is that the end of a card acts like a comma (unless a comma is in column 80). Hence the data

```
9.0         , 6.7E-1,
7
```

would set A = 9.0, B = .67, J = 0, and

read 7 on next READ statement.

B. Program Operation

The six card sets, described in the introduction, are constructed as follows. (Control cards (beginning with @) are for the EXEC VIII operating system for the UNIVAC 1108).

EXEC A

The first set of cards must cause Subroutine FCN to be compiled with the function G inserted where indicated [See Listing - Appendix A]. The exact commands depend on the file names used, etc. At the APG-EA UNIVAC 1108 facility, this requires only an executive card,

```
@ADD AMUCK*FNFIT.DO-G
```

followed by FORTRAN statements that construct the user supplied function, G(x; A(1), A(2), ...). For example

```
G = A(1) + A(2) * X ** 2 + X ** A(3)
```

or

```
G = 0.
```

```
      DO  10  I = 1, 4
 10   G = A(I) * SIN (I * A(5) * X) + G
```

The @ADD command enters the instructions from element DO-G into the run stream.

EXEC B

The next instructions must make an absolute element which includes the newly compiled FCN, and execute the program. Again, exact commands depend on the system which in our case requires only one card -

```
@ADD AMUCK*FNFIT.DO-FNFIT
```

The program is now running. The following sets insert data for a particular fit using the user-supplied-function.

SET A

(1) "Gab cards" - i.e. Any cards which do not start with five integers separated by commas, will be printed at the top of any output. This allows printing the G-function or messages.

The first cards starting with five integers will be read as the following control card.

(2) "Control card" (FORMAT ( ))

MXSTP, MAXIT, IPRT, ICHK, NHOLD

where

MXSTP: Maximum number of steepest gradient steps

MAXIT: Maximum number of least square iterations.

IPRT: OUTPUT control

= 0; Output only at end

= 1; Gives parameters and χ^2 after every step/iteration

= 3; Gives parameters, χ^2,,and fit to function after every step/iteration

= 4, Acts like 3, but includes $\partial G/\partial X$

= 2; Acts like 3 if χ^2 has improved, otherwise like 1.

ICHK: COOLIT control

= 1; Check parameters for limits on minima, maxima and variation (i.e. Call COOLIT each iteration)

= 0; No check

NHOLD: Number of parameters NOT allowed to vary.

If NHOLD is not 0; then the next cards are
(3),... IHOLD(J) (FORMAT ( )),
the indicies of the NHOLD parameters which are to be held fixed.
For example, if A(3) is to be a fixed value, then NHOLD = 1, and the following card is

3

SET B

(1) NPARAM (FORMAT ( )) - the total number of parameters - A(J) - in G(x; A(1), ...)

Then NPARAM cards, (for J = 1, 2, ... NPARAM)

(2),... A(J), MAG(J) (FORMAT ( ))

where A(J) = initial guess for parameter J

MAG(J) = "magnitude" of parameter J (i.e. the approximate power of 10)

e.g. $0 \leq A(1) \leq 100.$ means MAG(1) = 2

or $-0.01 \leq A(3) \leq 0.05$ means MAG(3) = -2

Option 1. If A(J) - the initial guess for a parameter - indicates the order of magnitude of parameter (J), then inputting "$" for MAG(J) will automatically set MAG(J) = log A(J). Clearly, to use this option,

$A(J) \neq 0$.

Option 2. @EOF in place of card (1) on any search after the first, will compute starting values, A(J), from previous magnitudes. Eliminates need to respecify A(J)s.

EXAMPLE: For G = A(1) * X + A(2), input might be:

```
2
0.0, 1
1.0E-2, $
```

Subsequent fit

```
@EOF
```

SET C

If ICHK=1 - set C is used.

(COOLIT CONTROL SET)

Subroutine COOLIT limits allowed parameter values

(1),... "Coolit Cards" (FORMAT ( ))

J1, L1, DUM (FORMAT ( ))

where

J1: Subscript of cooled parameter

L1: Kind of limit

= 1 - limit size of change

= 2 - limit minimum value

= 3 - limit maximum value

DUM: Value of the limit

LAST) LAST CARD MUST BE @EOF

Option: Inputting "$" for a coolit card automatically limits all parameters to change by less than 1/2 of their "magnitude" (from SET B)

EXAMPLE:

```
2, 1, 0.5
6, 2, 0.0
$
@EOF
```

This set of COOLIT Cards

(1) Limits parameter 2 to change by less than 0.5 in any iteration.

(2) Limits parameter 6 to values greater than or equal to 0.0.

(3) Limits all parameters to change by less than 1/2 their "magnitude".

SET D

1) WT - (FORMAT ( ))

where WT is the power of Y(I) by which to weigh each data pt.

(Note: WT = 0.0 means unweighed)

2),..... X(I), Y(I) - (FORMAT ( ))

These are the data points to be fit

Option: If WT = $, then the subsequent cards are

2),..... X(I), Y(I), W(I) - (FORMAT ( ))

where W(I) is the user-supplied weight for each data point.

(Note: Program normalizes Σ W(I) = 1.0)

Option: If WT = @EOF, the data and weight from the previous run are used.

Option: If the first X(I),... = @EOF, the data from the previous run are used.

LAST CARD) Following the data, the last card must be @EOF.

To end program, place (another)

@EOF

at end of deck.

III. HOW FNFIT WORKS

For statistical reasons[1], it is assumed that the function χ^2 is a measure of the "poorness of a fit". χ^2 is given by

$$\chi^2 = \sum_{I=1}^{NDATA} \frac{W(I)\ (Y(I)-F(X(I))^2}{NDATA-(NPARAM-NHOLD)}$$

where the data points, NDATA in number, are the set [X(I), Y(I)], F(x; A(1), A(2)..., A(NPARAM)) is the fitting function containing the parameters A(J), (NPARAM-NHOLD) is the true number of variable parameters, and W(I) is a factor allowing the data points (X(I), Y(I)) to be weighted.

(FNFIT causes $\sum_{I=1}^{NDATA} W(I) = 1.$)

χ^2 can be considered as a surface in the NPARAM + 1 dimensional space [χ^2, A(1), A(2)...,] The best fit is the lowest point on that surface (smallest χ^2), and the purpose of FNFIT is to find the NPARAM + 1 coordinates of that lowest point. To accomplish this, FNFIT uses two search techniques

1. Steepest Gradient ($\leq$ MXSTP steps)
2. χ^2 Minimization ($\leq$ MAXIT steps)

Consider the 3-D χ^2 surface of Figure 1. Suppose one starts at point I. A χ^2 minimization (according to Reference 1) is unstable here (convex surface). The best approach is to find the steepest gradient and step in that direction. However, once getting to IV, the surface is concave and the steepest gradient search becomes inefficient and inaccurate. Thus, having sensed that the gradient at IV was less steep than at III, FNFIT switches to a χ^2 minimization. In this phase of search, the program approximates χ^2 linearly, and solves the set of NPARAM linear equations to get the values of the parameters, A(J), for which $\frac{\partial \chi^2}{\partial A(J)} = 0$.

FNFIT consists of a main program (FNFIT), 10 subroutines, and 2 executive instruction elements (DO-G and DO-FNFIT). The specific function and operation of the more involved routines are discussed below.

FNFIT - Main Program

FNFIT serves three purposes. It acts as an executive, calling the various operations (such as the steepest descent search); it reads all data after the control card; and it performs the χ^2 minimization.

Possible confusion about the program can be reduced by distinguishing between two groupings of the parameters. The supplied function (in FCN via FN) is a function of X, containing parameters A(1), A(2)...A(NPARAM). However, NHOLD of them may be excluded by the user from being changed in value. The indicies of the excluded parameters are in an array IHOLD(1), IHOLD(2)...IHOLD(NHOLD). Furthermore, as described in the section on COOLIT, some parameters may be temporarily held constant

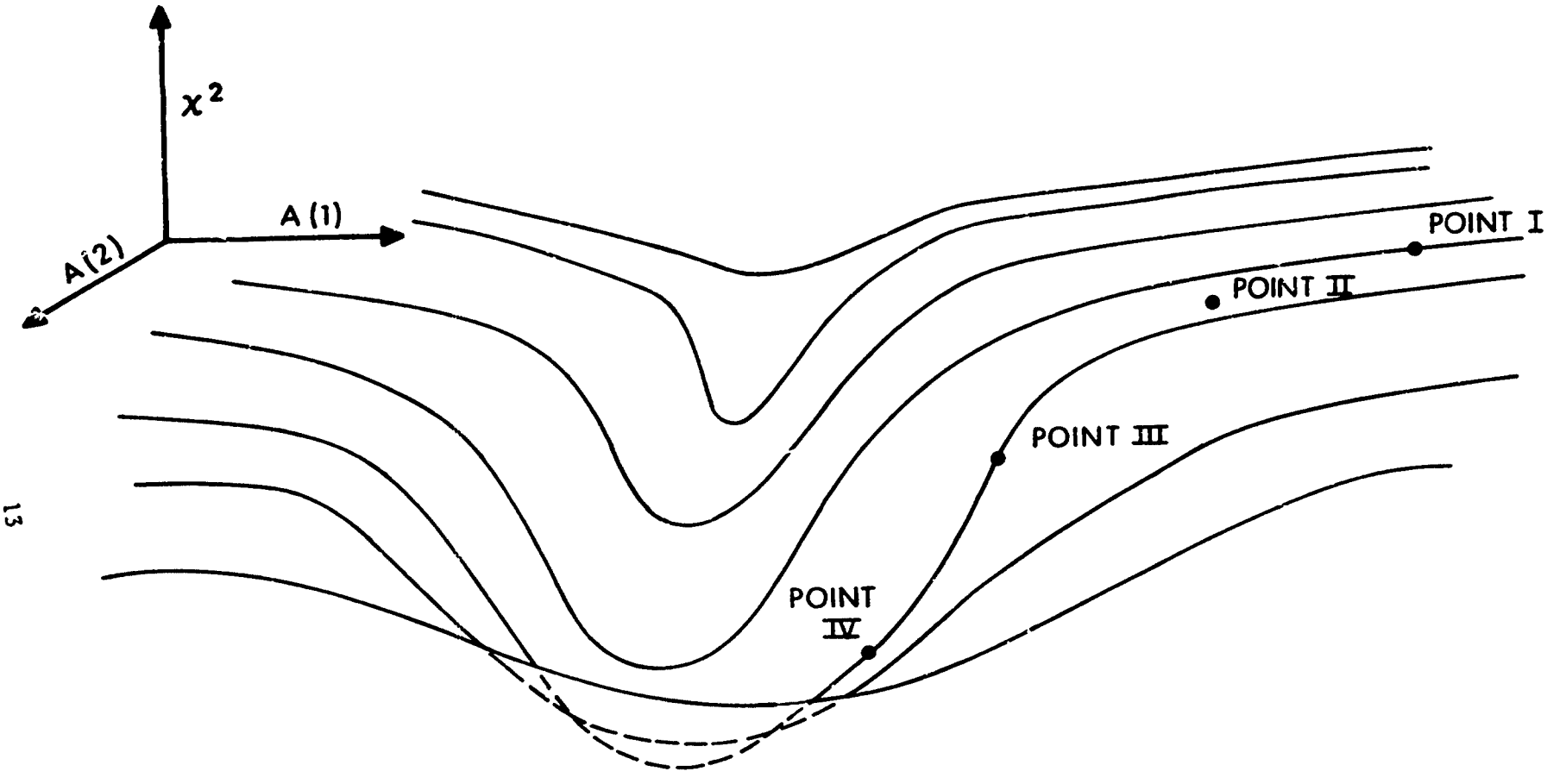


Figure 1 – Example of an Error Surface for NPARAM = 2

by the program. There are NT of these, and they go into IHOLD(IHOLD(NHOLD + 1),...IHOLD(NHOLD + NT).) The number actually being varied at any particular time is NV = NPARAM-NHOLD-NT. Before each iteration the indicies of the variable parameters are assembled, in ascending order, in the array KL(1), KL(2),...KL(NV). FNFIT uses this array to find the indicies of the parameters involved in that particular iteration of the χ^2 minimization search.

Having read the input data and having returned from GRAD (steepest gradient search), FNFIT begins the χ^2 iterative loop. The steps are as follows:

1) It checks the variable parameters, releasing one if appropriate (as described under COOLIT) and assuring that NV $\geq$ 1. If NV = 0, that iteration is skipped, and one parameter is released on the following iteration.

2) It then evaluates the fitting function - using the current values of the parameters - at all the X(I) (data point abscissae); it calculates χ^2, and outputs as directed by the control card.

3) Next, it sets up the KL matrix, and calculates the coefficients for the NV simultaneous linear equations

$$\frac{\partial \chi^2}{\partial A(J)} = 0. \quad \text{Recall, } \chi^2 \propto \sum_{I=1}^{N\ DATA} W(I)(Y(I)-F(X(I)))^2,$$

and F(Z; A(1), A(2),...) is being linearly approximated as

$$F(Z;\ A(1),\ A(2),\ldots) = F'(Z) + \left(\frac{dF(z)}{dA(1)}\right)' \cdot \Delta A(1) + \left(\frac{dF(z)}{dA(2)}\right)' \cdot \Delta A(2) + \ldots$$

where primed quantities refer to evaluation at the most recent values of A(J). Using this, the NV simultaneous equations can be written in matrix form as

$$\overline{\overline{AM}} \cdot \overline{\Delta A} = \overline{R}$$

where

$$AM_{JK} = \sum_{I=1}^{N\ DATA} W(I) \cdot \left(\frac{dF(X(I))}{dA(J)}\right)' \cdot \left(\frac{dF(X(I))}{dA(K)}\right)'$$

and

$$R_J = \sum_{I=1}^{N\ DATA} W(I) \cdot (Y(I)-F'(X(I))) \cdot \left(\frac{dF(X(I))}{dA(J)}\right)'$$

FNFIT gets the necessary derivatives from DFN (Z). It even checks to make sure that (dF/dA(J)) is not identically zero for some J (i.e., to make sure that F is a function of all A(J)s).

4) FNFIT then calls LSIMEQ from the 1108 System Library, and solves for the ΔA s.

5) At this point, the ΔA's are checked. If all are near zero (no parameters are changing), a fit has been found, and FNFIT exits.

6) Assuming no fit yet, FNFIT goes about updating the parameters. It first calls COOLIT to assure that no parameter is changed too much. Upon return, the allowed changes are added to the appropriate A(J)s. COOLIT is then recalled to assure that no parameter has exceeded its lower or upper absolute limits.

FNFIT then returns to step 1.

Upon exit, either via step 5, or by doing the maximum number of iterations, FNFIT restarts by looking for a new control card.

DGDK(X, J, DG) - Derivatives

DGDK takes the derivative of the function F(X; A(1), A(2)...) with respect to A(J) at X by fitting a 5th order polynomial to F evaluated at 5 values of A(J). For J = 0, the derivative is taken with respect to X. The five A(J) values are A(J), A(J) $\pm 10^{-6}$ C(J), and A(J) $\pm 10^{-2}$ C(J), where the C(J) are the "magnitudes" of the A(J) which are user supplied (refer to USERS MANUAL section). The big spread in the points is an attempt to fit both functions that change rapidly and

functions that change slowly.

Having fit F(X; A(1), A(2)...) by a_4 A(J)^4 + a_3 A(J)^3 + a_2 A(J)^2 + a_1 A(J) + a_0, the derivative is given by

$$DF/DA(J) = 4a_4A(J)^3 + 3a_3A(J)^2 + 2a_2A(J) + a_1$$

GRAD - Steepest Gradient Search

GRAD, like FNFIT, sets up the array, KL(J), of variable parameter indicies. Hence, A(J) refers to NV actually varied parameters.

The idea of GRAD is to detect the steepest gradient in the χ^2 surface, take a step down that gradient, and recompute. The size of the step, STPSZ, is chosen internally as 0.4 in the non-dimensionalized parameter space described below.

A difficulty in this kind of search is treating all parameters equally. A step size of 10m is huge for a parameter whose range is from -10^{-6} to 10^{-6}m, is negligible for one whose value is in the order of 10^6m, and is meaningless to one whose dimension is grams. Therefore, GRAD converts the problem from A-space (χ^2, A(1), A(2)...) to B-space (χ^2, B(1), B(2)...), where B(J) = A(J)/C(J). C(J) refers, as in DGDK, to the user supplied "magnitudes" of the A(J). If the C(J)s fairly well describe the magnitudes of the A(J)s, then a step of 1 in B-space has the same relative effect on all the parameters.

Having converted to B-space, GRAD proceeds as follows:

1) like FNFIT, it evaluates G and χ^2 and outputs as directed. GRAD exits here a) if MXSTP steps have been taken or b) if two successive steps have resulted in a worse χ^2. Upon <u>any</u> exit, the values of the A(J) are set to those that gave the smallest χ^2.

2) the next step is to evaluate the $\partial\chi^2/\partial B(J)$.

Since

$$\frac{\partial\chi^2}{\partial B(J)} = \frac{\partial\chi^2}{\partial A(J)} \cdot \frac{dA(J)}{dB(J)} = \frac{\partial\chi^2}{\partial A(J)} \cdot C(J)$$

GRAD can evaluate $\partial\chi^2/\partial B(J)$ by calling DGDK and evaluating $\partial\chi^2/\partial A(J)$ as does FNFIT. Also, the root-square, $(\sum(\partial\chi^2/\partial B(J))^2)^{1/2}$, is computed.

3) The step in B-space is taken by adding to each B(J) a change equal to $-(\partial\chi^2/\partial B(J))\cdot(STPSZ)/root\text{-}square$. Since the B(J)s are orthogonal, it is seen that this procedure takes a step in B-space of size STPSZ, with the largest components of the step in the B(J)s having the steepest gradient. The minus sign assures movement down the gradient.

4) GRAD now goes about checking the limits and second derivative of the new B(J)s. First COOLIT is called to check maxima and minima (since COOLIT works in A-space, the A array is first updated). Then, the most recent $\partial\chi^2/\partial B(J)$s are compared to the previous $\partial\chi^2/\partial B(J)$s. If any parameter has either exceeded a COOLIT limit or found a shallower gradient, that parameter is held fixed, and the program returns to step 3) and recalculates the B-step. This procedure is continued until a) an allowed step is taken, or until b) all parameters have found a shallower gradient and GRAD exits. If a step is taken, GRAD loops back to step 1.

COOLIT - Parameter Limiting Subroutine

As described in the USERS MANUAL, the user can input limits on the minimum or maximum that a parameter can attain, or limit the amount a parameter can change in any FNFIT iteration. (This latter limit keeps the search from making wild jumps because of linear approximations to touchy parameters.) When COOLIT is called, it checks the A(J)s, or DA(J)s, versus their limits. If any limits are exceeded, the A(J) (or DA(J)) is reduced to its allowed limit. Then, the errant A(J) is placed at the top of the list of parameters temporarily held constant (not changed in the succeeding iterations in FNFIT or GRAD).

The list is maintained in FNFIT by these rules.

1) If any parameter was added to the list during the past iteration no parameter is released.

2) If no parameter was added to the list during the past iteration then the parameter at the bottom of the list (most time on the list) is

released.

The result of this procedure is to allow the program to readjust to any "artificial" changes caused by imposing limits.

To avoid the possibility of a subtle kind of loop, the limit on the DA(J)s is reduced as a function of the iteration. This is not a forced convergence; it merely assures non-repetition.

IV. GENERAL CAUTIONS

SQ is a function defined on an NPARAM dimensional space. Depending on the form of G and on the data, there may exist any number of "local" minima, i.e., sets of values for the parameters for which any small change in any of the parameters produces an increase in SQ. It is assumed that the best fit, and hence the desired set of parameters, corresponds to only one of these local minima. The local minimum which the program seeks out depends on the starting set of parameters. There exists no completely satisfactory method to avoid the multiple minima problem; it is, in fact, another aspect of the generally unresolved problem of unambiguously defining a best fit. Two features of FNFIT aid in best fit selection: the output subroutine displays data versus F(X) for easy comparison by the user, and initial parameters are easily changed to "map out" local minima.

Another phenomenon of this kind of fitting program occurs when F(X) is a much stronger function of some parameters than others. In such a case, some parameters are varied much less than others. The hold-parameters-fixed and check-new-values-before-reiteration features of FNFIT are useful aids in assuring a suitable search on all parameters.

An error to be avoided in any function fitting problem is an attempt to use indeterminant parameters. For example, the function A(1)*(A(2)*X+A(3)) can never be uniquely fit, since A(2) and A(3) can compensate for any value of A(1). Such errors can be very subtle (e.g. a variable starting point for a Fourier Series is indeterminant.) A bad start can also fool the program into thinking that such an error

exists. For example, in fitting a function like $A_1-A_2e^{-A_3X}$ (See section V), a search that starts by making A_1 and A_2 very large might result in values for A_3 that are so small that the function is approximately $A_1 - A_2$, and acts indeterminant. Therefore, the size of changes (L1=1 option on page 10) should be held to the minimum required.

If the function does not closely fit the data, a possible symptom of indeterminancy in FNFIT is non-convergence. If the data does fit well, a probable sympton is the detection of a singular matrix by Subroutine LSIMEQ. If, during the matrix reduction-inversion process, the largest pivot element is smaller than EPS1(10^{-12}, as listed) LSIMEQ aborts and returns to FNFIT. FNFIT prints out the partially reduced matrix, the indicies of the previous pivot elements, and then looks for new data to begin a new fit.

V. A FITTING EXAMPLE

As an example of the operation of the program, twenty-five values of X(I), ($0 \leq X \leq 10.$), and the corresponding values of Y(I) from the relationship Y(I) = 1. - EXP (-X(I)) were fit with the following functions.

A. G = A(1) + A(2) * EXP (-A(3) * X) + A(4) * EXP (A(5) * X)

B. $G = \sum_{J=1}^{4} A(J) * SIN (J * A(5) * X)$

C. $G = \sum_{J=1}^{5} A(J) * P_{J-1} (X)$

where $P_L(X)$ is the Legendre polynomial

D. G = A(1) + A(2) * ALOG (A(3) * X + 1) + A(4) * ALOG (A(5) * (X**2) + 1)

Two attempts, with different starting values and limits, were made with case B.

The sets of cards, with running explanation of the function of each, are included in Appendix B.

The same set of starting parameters was used in all cases

(A(1) thru A(4) = 1.0, A(5) = 0.02), indicating the relative insensitivity of the search in this case to initial values. Each starting value also served as its magnitude ($ option in Set B). The maximum change in any iteration was limited to 1/2 magnitude ($ option in Set C). Also, parameters A(3) and A(5) were forced to stay positive in cases A and D, and also in case B for A(5).

The initial searches were limited to 200 iterations. However, cases A and B had not finished (i.e. exit was by maximum iterations rather than by non-changing values.) Case A is difficult in that A(4) = 0.0 is the exact solution; however, as A(4) approaches 0.0, A(5) becomes indeterminant. Case B was then continued for 600 iterations and did not improve; the search was in a very slowly damped oscillation about the final values.

The results are plotted in Figure 2.

The values of χ^2 for the cases were:

A: 2.139×10^{-7}

B: 6.903×10^{-3}

C: 6.692×10^{-4}

D: 4.931×10^{-7}

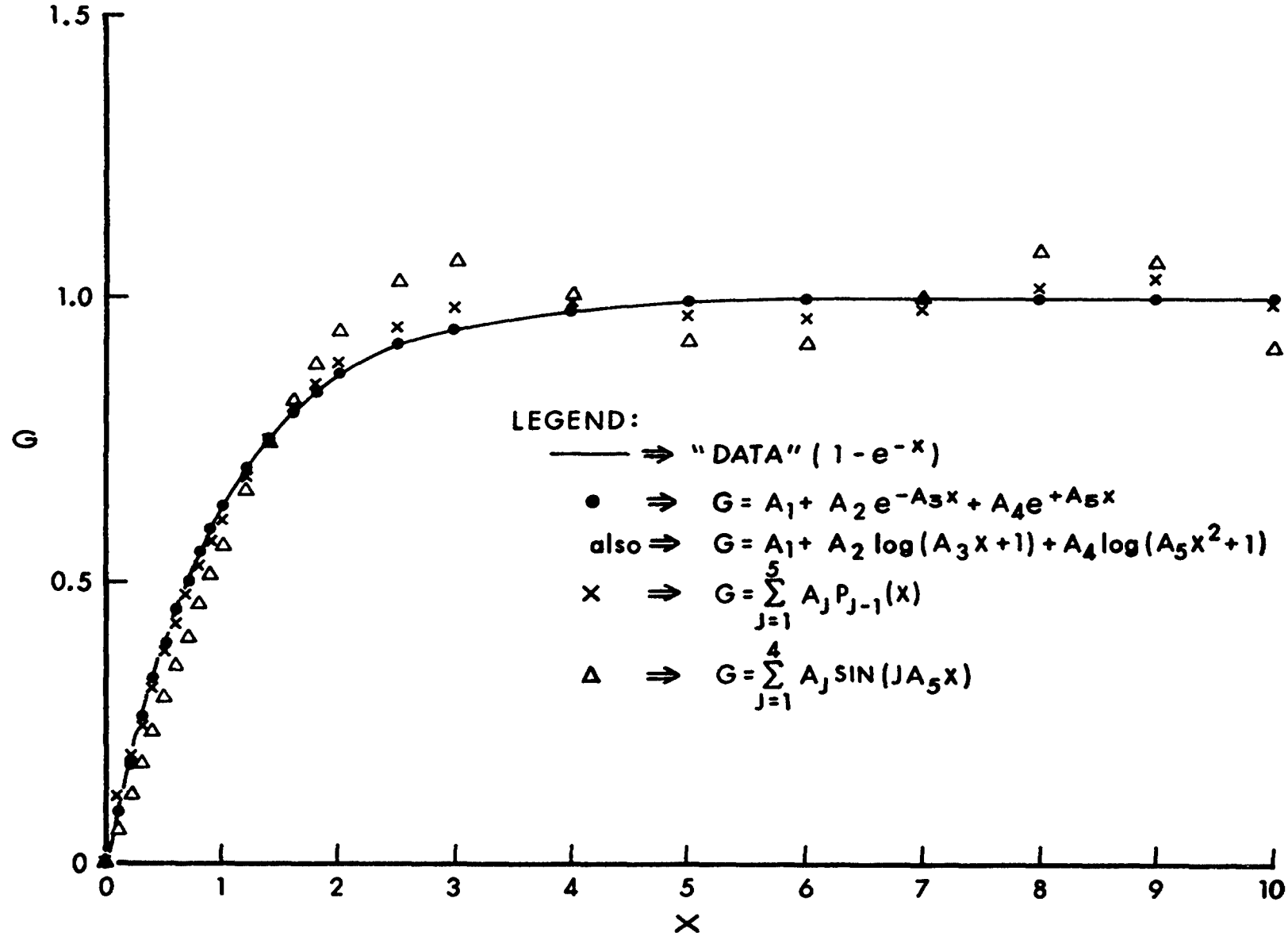


Figure 2 - Results of Fitting Example

VI. REFERENCE

1. Bevington, P. R., Data Reduction and Error Analysis for the Physical Sciences, (McGraw - Hill) 1969.

APPENDIX A

HEADING

```
      SUBROUTINE HEADING( IK )
      COMMON/PARAM/NPARAM, A(20), SQ
      INTEGER V(7)
      IF( IK .GT. 0 )  GO TO 40
      NNN = NPARAM
      IF( NNN .GT. 10 ) NNN=10
      ENCODE ( 579, V )  NNN
579   FORMAT ( '(5HOIT' , 'ER,5X,' , '2HSQ, ', 13, '(6X2HA' , '(,12,1' ,
     $ 'H) ) )'  )
40    WRITE ( 6, V )  ( I, I = 1, NNN )
      RETURN
      END
```

FN

```
      SUBROUTINE FN
      COMMON/FUNCT/ F(1000)
      COMMON/DATA/NDATA,X(1000),Y(1000),W(1000)
      DOUBLE PRECISION G, T
      DO 70 I=1, NDATA
      T = X(I)
      CALL FCN( T, 0, 0., G )
70    F(I) = G
      RETURN
      END
```

DFN

```
      SUBROUTINE DFN(Z)
      COMMON/PARAM/NPARAM, A(20), SQ
      COMMON/ DFUNCT/ DF(20), NV, KL(20)
      DOUBLE PRECISION  T
      T = Z
      DO 100  K = 1, NV
      J = KL(K)
      CALL DGDK( T, J, DG )
100   DF(J) = DG
      RETURN
      END
```

COOLIT

```
      SUBROUTINE COOLIT( ITER, KEY, MARK )
      COMMON/ HOLD / NHOLD, IHOLD(20)
      COMMON/PARAM/NPARAM, A(20), SQ
      COMMON/ COOL/ NT, MAXIT, DA(20), NDAM(3), JDAM(3,20), DAM(3,20)
      IF( MARK )  100
C     CHECK AMINS
      N2 = NDAM(2)
      IF(N2) 40,40
      DO 30  J2 = 1, N2
      K2 = JDAM(2, J2)
      IF( A(K2) .GE. DAM( 2, J2 ) )  GO TO 30
      A(K2) = DAM(2, J2)
      KEY = 1
      IF(NT) 28,28
      N1 = NHOLD + 1
      NN = NHOLD + NT
      DO 27  KK =N1,NN
 27   IF( IHOLD(KK) .EQ. K2 )  GO TO 30
 28   NT = NT+1
      IHOLD( NHOLD + NT ) = K2
 30   CONTINUE
C     CHECK AMAXS
 40   N3 = NDAM(3)
      IF( N3 .EQ. 0 )  RETURN
      DO 60  J3 = 1, N3
      K3 = JDAM(3, J3)
      IF( A(K3) .LE. DAM(3, J3) )  GO TO 60
      A(K3) = DAM(3, J3)
      KEY = 1
      IF(NT) 48, 48
      N1 = NHOLD + 1
      NN = NHOLD + NT
      DO 47  KK =N1, NN
 47   IF( IHOLD(KK) .EQ. K3 )  GO TO 60
 48   NT = NT+1
      IHOLD( NHOLD + NT ) = K3
 60   CONTINUE
      RETURN
C     CHECK DA S
 100  N1 = NDAM(1)
      IF( N1 .EQ. 0 )  RETURN
      PON = FLOAT( ITER ) / FLOAT( MAXIT )
      DO 130  J1 = 1, N1
      K1 = JDAM( 1, J1 )
      YUK = DAM(1, J1) * EXP( - PON )
      IF( ABS( DA(K1) ) .LT. YUK )  GO TO 130
      DA(K1) = SIGN( YUK, DA(K1) )
      KEY = 1
      IF(NT) 128,128
```

```
      N1 = NHOLD + 1
      NN = NHOLD + NT
      DO 127  KK =N1, NN
127   IF( IHOLD(KK) .EQ. K1 )  GO TO 130
128   NT = NT+1
      IHOLD( NHOLD + NT ) = K1
130   CONTINUE
      RETURN
      END
```

GAB

```
      SUBROUTINE GAB( MXSTP, MAXIT, IPRT, ICHK, NHOLD, K2K )
      DIMENSION GAB( 140 )
      NEW = 1
10    READ( 5, 9, ERR=25, END=30 )  MXSTP, MAXIT, IPRT, ICHK, NHOLD
9     FORMAT(  )
      IF( K2K ) 5, 5
      PRINT 89, (  GAB(M), M = 1, K2K )
89    FORMAT( '1', ( T20, 13A6,A2, / ) )
      PRINT 98
99    FORMAT( 13A6, A2 )
98    FORMAT ( // )
5     RETURN
25    CONTINUE
      IF( NEW ) 35, 35
      K2K = 14
      K1K = 1
      GO TO 40
35    K1K = K1K + 14
      K2K = K2K + 14
40    READ( 0, 99 )  ( GAB(M), M = K1K, K2K )
      NEW = 0
      GO TO 10
30    STOP
      END
```

DO-FNFIT

```
@MAP,IS  IT
LIB MISD*LIBRARY.
IN AMUCK*FNFIT.,TPF$.FCN
END
XQT
```

```
C       FN.FIT      MAIN PROGRAM
        COMMON/ COOL/ NT, MAXIT, DA(20), NDAM(3), JDAM(3,20), DAM(3,20)
        COMMON/DATA/NDATA,X(1000),Y(1000),W(1000)
        COMMON/ DFUNCT/ DF(20), NV, KL(20)
        COMMON/FUNCT/ F(1000)
        COMMON/ GRADC / MXSTP, IPRT, NNN, ICHK
        COMMON/ HOLD / NHOLD, IHOLD(20)
        COMMON/ MAG / C(20)
        COMMON/PARAM/NPARAM,A(20),SQ
        DATA  EPS1, EPS2 / 1.0E-12, 1.0E-5 /
        LOGICAL PRE, POST
        DIMENSION AM(22,22), IR(22), JC(22),RDA(22)
        DIMENSION MAG(20)
        DIMENSION  ISDFO(20)
        K2K = 0
  3     CALL GAB( MXSTP, MAXIT, IPRT, ICHK, NHOLD, K2K )
        IF( NHOLD) 4,4
        READ ( 5 , 9 )  ( IHOLD(K), K = 1, NHOLD )
  4     CONTINUE
        READ( 5, 9, ERR = 7777, END = 211 )  NNNN
        NPARAM = NNNN
        DO 17  J = 1, NPARAM
        READ( 5, 9, ERR = 18 )  A(J), MAG(J)
        C(J) = 10.**MAG(J)
        GO TO 17
  18    C(J) = ABS( A(J) )
  17    CONTINUE
  21    CONTINUE
        DO 11  I = 1,3
  11    NDAM(I) = 0
        PRE = .FALSE.
        POST = .FALSE.
        IF( ICHK ) 5,5
        DO 15  J = 1, 60
         READ( 5, 9, ERR=8888, END=15 )  J1, L1, DUM
        NDAM(L1) = NDAM(L1) + 1
        N1 = NDAM(L1)
        JDAM(L1, N1) = J1
        DAM(L1, N1) = DUM
         GO TO 13
C       IF COOLIT CARD IS $,*, ETC., ALL PARAMETER CHANGES ARE LIMITED
C       TO 0.5*MAGNITUDE OF THE PARAMETER
 8888 DO 14  J2= 1, NPARAM
        NDAM(1) = NDAM(1) + 1
        N1 = NDAM(1)
        JDAM( 1, N1 ) = J2
         DAM( 1, N1 ) = .5*C(J2)
```

```
 14   CONTINUE
 13   CONTINUE
 15   IF( NDAM(1) .GT. 0 )  PRE = .TRUE.
      IF( ( NDAM(2) .GT. 0 ) .OR. ( NDAM(3) .GT. 0 ) )  POST = .TRUE.
 5    CONTINUE
 9    FORMAT ( )
      CALL DATAIN
C
C     SET UP DGDK
      CALL DGDK( 0.0, -1, 0.0 )
C
C     PRINT HEADING INFO
      IF ( NHOLD .EQ. 0 )  GO TO 48
      PRINT 249,  ( IHOLD(L), L = 1, NHOLD)
 48   CONTINUE
      PRINT 609
      IF( NDAM(1) )  620, 620
      NDAM1 = NDAM(1)
      PRINT 619, ( JDAM(1,K), DAM(1,K), K = 1, NDAM1 )
 620  IF( NDAM(2) )  630, 630
      NDAM1 = NDAM(2)
      PRINT 629, ( JDAM(2,K), DAM(2,K), K = 1, NDAM1 )
 630  IF( NDAM(3) )  640, 640
      NDAM1 = NDAM(3)
      PRINT 639, ( JDAM(3,K), DAM(3,K), K = 1, NDAM1 )
 640  CONTINUE
 249  FORMAT ( 29HOPARAMETERS HELD CONSTANT...   , 20( I4, IH,  ) )
 609  FORMAT (  '0'  )
 619  FORMAT ( '  CHANGE LIMITED FOR PARAMETERS ... '
     $ 4( T37, 5( I4, ' (', 1PE9.2, ' ),' ) / ) )
 629  FORMAT ( ' MINIMUM LIMITED FOR PARAMETERS ... '
     $ 4( T37, 5( I4, ' (', 1PE9.2, ' ),' ) / ) )
 639  FORMAT ( ' MAXIMUM LIMITED FOR PARAMETERS ... '
     $ 4( T37, 5( I4, ' (', 1PE9.2, ' ),' ) / ) )
C
C     CALL GRADIENT SEARCH
      NNN = NDATA + NHOLD - NPARAM
      IF( MXSTP ) 40, 40
      CALL GRAD
 40   PRINT 299
 299  FORMAT ( / / ' CHISQ MINIMIZATION SEARCH '  )
      CALL HEADING(MXSTP)
      KEY = 0
      NT = 0
C
C     THE ITERATIVE LOOP STARTS HERE
C
      ITER = 0
      N1 = NHOLD + 1
 1002 CONTINUE
```

```
C
      CALL FN
C     FN FILLS F(I) AT ALL INPUT POINTS
C
C     NOW CALCULATE SQ
      SQ = 0.0
      DO 20  I - 1, NDATA
 20   SQ = SQ + W(I)*( Y(I) - F(I) )**2 / NNN
C
C     OUTPUT VARIABLES FROM PREVIOUS ITERATION
      CALL CUTPUT( ITER, IPRT )
C
 1001 CONTINUE
C
C     IF NO NEW PARAMETER IS BEING HELD THIS ITER., REMOVE THE OLDEST
      IF( ( KEY .NE. 0 ) .OR. ( NT .EQ. 0 ) )  GO TO 140
      NT = NT - 1
C
      NAH = NHOLD + NT
      DO 135  KK = N1, NAH
 135  IHOLD(KK) = IHOLD( KK+1 )
 140  CONTINUE
      NAH = NHOLD + NT
      NV = NPARAM - NAH
C
C     MAKE SURE THAT SOME PARAMETER IS ALLOWED TO VARY
      IF( NV ) 195,195,200
 195  ITER = ITER + 1
      PRINT 199, ITER
 199  FORMAT( 1X, I3, ' ALL VARIABLES HELD ' )
      KEY = 0
      GO TO 1001
 200  CONTINUE
C
      IF ( ITER .EQ. MAXIT )  GO TO 1010
C
C     NOW SET UP ARRAY, KL, OF VARIABLE PARAMETER INDICES, IN NUMERICAL
C     ORDER
      K = 1
      DO 210 II = 1,NPARAM
      IF( NAH .EQ. 0 )  GO TO 215
      DO 220 JJ = 1,NAH
 220  IF( IHOLD(JJ) .EQ. II )  GO TO 210
 215  KL(K) = II
      K = K+1
 210  CONTINUE
C
C     SET UP NV X NV+1 MATRIX OF COEF. , ONE DATA POINT AT A TIME
      NV1 = NV+1
      DO 50 I1 = 1,NV
      ISDFO(I1) = 0
```

```
      DO 50 I2 = 1, NV1
 50   AM(I1,I2) = 0.
      DO 100 I = 1,NDATA
      CALL DFN(X(I))
C     INTO / DEFUNCT/ APPEARETH ( DF( KL(J) ) , J = 1,NV ) EVALUATED AT X(I)
      DO 80 K1 = 1,NV
      K = KL(K1)
C
C     CHECK IF DF(SOME VARIABLE) IS I.D. = 0
      IF( ABS( DF(K) ) .LT. EPS1 )  ISDFO(K1) = ISDFO(K1) + 1
C
C     FILL UP COEFICIENT MATTRIX FOR DA
      AM(K1, NV1 ) = AM(K1, NV1 ) + W(I)*( Y(I) - F(I) )*DF(K)
      DO 60  K2 = 1,NV
      L = KL(K2)
      AM(K1,K2) = AM(K1,K2) + W(I)*DF(K)*DF(L)
 60   CONTINUE
 80   CONTINUE
 100  CONTINUE
C
C     FINISH CHECK FOR DF I.D. = 0 AND REDUCE MATRICES OF VARIABLES
C     ACCORDINGLY
      NVQ = NV
      DO 300  K = 1, NVQ
      IF( ISDFO(K) .NE. NDATA )  GO TO 300
      PRINT 599, KL(K), KL(K)
 599  FORMAT( '    DF/DA(', I2, ') IS I.D. = 0 NEXT ITER. A(', I2, ') WI
     $LL BE HELD. ' )
      NV1 = NV
      NV = NV-1
      IF(NV) 195, 195
      DO 340  L1 = K, NV
      KL(L1) = KL( L1+1 )
      DO 340  L2 = 1, NV1
 340  AM( L1, L2 ) = AM( L1+1, L2 )
      DO 350  L2 = K, NV1
      DO 350  L1 = 1, NV
 350  AM( L1, L2 ) = AM( L1, L2+1 )
 300  CONTINUE
C
C     NOW FIND SOLN TO DA(20)
      CALL LSIMEQ ( AM, 22, IR, JC, NV, EPS1,RDA, IERR1 )
      IF(IERR1  .EQ. -1 )  GO TO 7734
      DO 102 J = 1, NPARAM
 102  DA(J) = 0.0
      DO 108 K1 = 1, NV
      K2 = KL(K1)
 108  DA(K2) = RDA(K1)
C
C     QUIT IF NO PARAMETERS ARE CHANGING OR ALL ARE ZERO
      IF(  ( ITER .LT.   4  ) .OR. ( NT .GT. 0 ) )  GO TO 106
```

```
      DO 105  J = 1, NPARAM
      IF( ABS( A(J) ) .LT. EPS1 )  GO TO 105
      IF( ABS( DA(J)/A(J) ) .GT. EPS2 )  GO TO 106
 105  CONTINUE
      GO TO 1010
 106  CONTINUE
      KEY = 0
C
C     IF ANY PARAMETER CHANGE IS LIMITED, CALL COOLIT
      IF( PRE )  CALL COOLIT( ITER, KEY,  -1 )
C
C     NOW ADD THE DA S  TO THE PROPER A S
 110  DO 120 J = 1, NPARAM
 120  A(J) = A(J) + DA(J)
C
C     IF ANY PARAMETER IS LIMITED, CALL COOLIT
      IF( POST )  CALL COOLIT( ITER, KEY, 1 )
 1000 CONTINUE
      ITER = ITER + 1
      IF( ITER .LE. MAXIT )  GO TO 1002
C
C     THE ITERATIVE LOOP ENDS HERE.
C
 1010 CALL OUTPUT( ITER,4 )
      GO TO 3
 7734 PRINT 689,
 689  FORMAT ( ' LARGEST ELEMENT .LT. EPS1 IN LSIMEQ ' )
      DO 701  I = 1,NV
      PRINT 9,    ( AM(I),J), J = 1, NV1 )
 701  CONTINUE
      PRINT 797
 797  FORMAT ( ' VARIABLES THIS ITERATION ' )
      PRINT 799, ( KL(J), J = 1, NV )
      PRINT 798
 798  FORMAT( 'OINDETERMINATE**** PROBLEM MIGHT BE...',/,
     $ T20, '1)  PARAMETERS ARE NOT INDEPENDENT',/,
     $ T20, '2)  NUMBER PARAMETERS + 1 .GE. NUMBER DATA POINTS', /,
     $ T20, '3)  DATA NOT IN RIGHT RANGE TO DETERMINE ONE PARAMETER' )
C
 799  FORMAT (  / 1X 2015 )
      GO TO 1010
 211  DO 212  J = 1, NPARAM
 212  A(J) = C(J)
      GO TO 21
 7777 PRINT 9999
 9999 FORMAT( ' MESS-UP IN MAIN PROGRAM READ ' )
      STOP
      END
```

DO-G

```
@FOR,S  AMUCK*FNFIT.FCN,TPF$.FCN
-8
```

DATAIN

```
      SUBROUTINE DATAIN
      COMMON/DATA/NDATA,X(1000),Y(1000),W(1000)
      COMMON/ XTREME/ DELY10, DELX10
      READ( 5, 9, ERR=40, END=25 )  WT
      AWT = ABS( WT)
      DO 20  I = 1, 1000
      READ( 5, 9, ERR = 7734, END = 33 )  XX, YY
      X(I) = XX
      Y(I) = YY
      W(I) = 1.0
      NDATA = I
20    CONTINUE
33    CONTINUE
      IF( AWT .LT. 1.0E-8 )  GO TO 52
      DO 50  I = 1, NDATA
      T = ABS( Y(I) )
      IF( T .LT. 1.0E-12 )  GO TO 50
      W(I) = T**WT
50    CONTINUE
52    CONTINUE
      XMIN = X(1)
      XMAX = X(1)
      WORM = 0.0
      DO 60  I = 1, NDATA
      WORM = WORM + W(I)
      XMIN = AMIN1( XMIN, X(I) )
      XMAX = AMAX1( XMAX, X(I) )
60    CONTINUE
      DELX10 = ( XMAX - XMIN ) / ( 10. * NDATA )
      DO 30  I = 1, NDATA
30    W(I) = NDATA*W(I)/ WORM
25    RETURN
7734  PRINT 69
      STOP
9     FORMAT (  )
69    FORMAT( '  MESS-UP IN DATAIN READ ' )
40    CONTINUE
      DO 45  I = 1, 1000
      READ( 5, 9, ERR = 7734, END = 52 )  X(I) , Y(I) , W(I)
      NDATA = I
45    CONTINUE
      GO TO 7734
      END
```

DGDK

```
      SUBROUTINE DGDK( T, K5, DG )
      COMMON/PARAM/NPARAM,A(20),SQ
      COMMON/ XTREME/ DELY10, DELX10
      COMMON/ MAG / C(20)
      DIMENSION SGN(2)/ -1., 1. /
      DOUBLE PRECISION G, T, DEL
      DOUBLE PRECISION  P(2), Q(3), DT(21,3), D2(21,3), DNOM(21)
      DOUBLE PRECISION  D3(21,3 )
      IF(K5) 30, 40, 40
30    DT(1,1) = DELX10
      DO 32  J = 1, NPARAM
32    DT( J+1, 1 ) = 0.01*C(J)
      NP1 = NPARAM + 1
      DO 34  N = 1, NP1
      DT(N,2) = DT(N,1)*1.0E-4
      DO 36  II = 1,2
      D2(N,II) = DT(N,II)**2
      D3(N,II) = DT(N,II)**3
36    CONTINUE
      DNOM(N) = 0.5/ ( DT(N,1)*DT(N,2)*( D2(N,2) - D2(N,1) ) )
34    CONTINUE
      RETURN
40    CONTINUE
      N = K5 + 1
      DO 20  L = 1,2
      DO 10  K = 1,2
      DEL = DT(N,L)*SGN(K)
      CALL FCN( T, K5, DEL, G )
10    P(K) = G
20    Q(L) = P(2) - P(1)
      DG = ( Q(1)*D3(N,2) - Q(2)*D3(N,1) ) * DNOM(N)
      RETURN
      END
```

OUTPUT

```
      SUBROUTINE OUTPUT( ITER,KKEY )
      COMMON/PARAM/NPARAM,A(20),SQ
      COMMON/FUNCT/F(1000)
      COMMON/DATA/NDATA,X(1000),Y(1000),W(1000)
      DOUBLE PRECISION  T
      DIMENSION  DD(1000), BEST(20)
      LOGICAL  YEP
      YEP = .TRUE.
      IF( KKEY ) 15,15
```

```
      IF( KKEY .NE. 4 )  GO TO 18
      PRINT 695
695   FORMAT( ///, 10X, 'BEST FIT' / )
      SQ = SOFAR
      ITER = IBEST
      DO 20  J = 1, NPARAM
      A(J) = BEST(J)
20    CONTINUE
      CALL FN
      DO 25  I = 1, NDATA
      T = X(I)
      CALL DGDK( T, 0, D )
      DD(I) = D
25    CONTINUE
18    CONTINUE
      IF( NPARAM .GT. 10 )  GO TO 30
      PRINT 499, ITER,   SQ, ( A(I), I = 1, NPARAM )
35    CONTINUE
499   FORMAT ( 1X, I3, 1X, 11(1PE11.3)  )
15    IF( ( ITER .EQ. 0 ) .OR. ( SQ .LT. SOFAR ) )  GO TO 12
16    CONTINUE
      IF( KKEY .LE. 1 )  RETURN
      IF( KKEY .GE. 3 )  GO TO 6
      IF( YEP ) RETURN
6     PRINT 598
598   FORMAT( /,54X,'I',5X,'X(I)', 8X, 'Y(I)', 8X, 'F(I)', 8X, 'W(I)'  )
      IF(KKEY .NE. 4 )  GO TO 8
      PRINT 698
698   FORMAT ( '+' , T107, 'DF/DX(I)'  )
8     PRINT 697
697   FORMAT( // )
      IF(KKEY .EQ. 4 )  GO TO 9
      PRINT 599, ( I, X(I), Y(I), F(I), W(I),  I = 1, NDATA )
599   FORMAT ( / , 1000(50X,I4,4(2X1PE10.4) / )  )
      GO TO 7
9     CONTINUE
      PRINT 699, ( I, X(I), Y(I), F(I), W(I), DD(I), I = 1,NDATA )
699   FORMAT (  / , 1000(50X,I4,5(2X1PE10.4) / )  )
7     CONTINUE
      RETURN
12    SOFAR = SQ
      IBEST = ITER
      DO 14 J = 1, NPARAM
      BEST(J) = A(J)
14    CONTINUE
      YEP = .FALSE.
      GO TO 16
30    CONTINUE
      PRINT 499, ITER, SQ, (A(I), I = 1,10 )
      PRINT 799, (A(I), I = 11, NPARAM )
799   FORMAT( T15, 10(1PE11.3)  )
      GO TO 35
      END
```

FCN

```
      SUBROUTINE FCN( X, K5,TDA, G )
      COMMON/PARAM/NPARAM, A(20), SQ
      DOUBLE PRECISION G, X , TDA
      IF( K5 .EQ. 0 )  X = X +TDA
      IF( K5 .GT. 0 )   A(K5) =  A(K5) +TDA
      CONTINUE
C  INSERT FUNCTION (  G =  FUNCTION( X, A(1),A(2),... ) HERE BETWEEN
C     THE CONTINUES
      CONTINUE
      IF( K5 .EQ. 0 ) X = X - TDA
      IF( K5 .GT. 0 )   A(K5) =  A(K5) - TDA
      RETURN
      END
```

GRAD

```
      SUBROUTINE GRAD
      COMMON/DATA/NDATA,X(1000),Y(1000),W(1000)
      COMMON/ COOL/ NT, MAXIT, DA(20), NDAM(3), JDAM(3,20), DAM(3,20)
      COMMON/ DFUNCT/ DF(20), NV, KL(20)
      COMMON/FUNCT/ F(1000)
      COMMON/ GRADC / MXSTP, IPRT, NNN, ICHK
      COMMON/ HOLD / NHOLD, IHOLD(20)
      COMMON/PARAM/NPARAM,A(20),SQ
      COMMON/ MAG / C(20)
      DATA STPSZ / 0.4 /
      DIMENSION  B(20), D(20), DXDB(20), DXDB1(20), ISG(20)
      LOGICAL L1, L2, L3
C
C     SET UP ARRAY, KL, OF VARIABLE PARAMETER INDICES
      NV = NPARAM - NHOLD
      K = 1
      DO 100  J1 = 1, NPARAM
      DXDB1(J1) = 0.0
      ISG(J1) = 1
      IF( NHOLD ) 90, 90
      DO 80  J2 = 1, NHOLD
 80   IF( IHOLD(J2) .EQ. J1 )  GO TO 100
 90   KL(K) = J1
      K = K + 1
 100  CONTINUE
 110  CONTINUE
      NHOLD1 = NHOLD + 1
C
C     SET UP ARRAY, B(J), OF DIMENSIONLESS PARAMETERS
```

```
      DO 150  J = 1, NPARAM
      B(J) = A(J) / C(J)
 150  CONTINUE
      PRINT 299
 299  FORMAT ( // ' STEEPEST DESCENT SEARCH ' / )
      CALL HEADNG( 0 )
      MMM = MXSTP + 1
      L1 = .TRUE.
      L3 = .FALSE.
C
C     THE ITERATIVE LOOP STARTS HERE
C
      DO 1000  ISTP = 1, MMM
      ITER = ISTP - 1
      NT = 0
      NF = NHOLD
      DO 41  J = NHOLD1, NPARAM
 41   IHOLD(J) = 0
      CALL FN
      SQ = 0.
      DO 20  I = 1, NDATA
 20   SQ = SQ + W(I)*( Y(I) - F(I) )**2 / NNN
C
      CALL OUTPUT( ITER, IPRT )
C
      IF( SQ .LT. SOFAR )  L1 = .TRUE.
      IF( .NOT. L1 )  GO TO 56
      SOFAR = SQ
      DO 50  J = 1, NPARAM
 50   D(J) = A(J)
      GO TO 560
 56   CONTINUE
      DO 550  J = 1, NPARAM
      IF( ISG(J) .EQ. -1)  GO TO 555
 550  CONTINUE
      GO TO 560
 555  PRINT 399
 399  FORMAT( ' CHI-SQ INCREASED AND GRADIENT CHANGED SIGN.  IF NO FINAL
     $ FIT, CHECK USER SPECIFIED MAGNITUDES. ' )
 560  CONTINUE
      IF(L3)  GO TO 21
      IF( L1 .OR. L2 )  GO TO 25
 21   CONTINUE
      DO 35  J = 1, NPARAM
 35   A(J) = D(J)
      RETURN
 25   CONTINUE
      IF( ISTP .EQ. MMM ) RETURN
      L2 = L1
      L1 = .FALSE.
```

```
C
C       FIND THE PARTIAL DERIVATIVES OF CHISQ W.R.T. THE B(J) S
        DO 30  J = 1, NPARAM
 30     DXDB(J) = 0.
        DO 400  I = 1, NDATA
C       DFN FILLS DF(I)
        CALL DFN(X(I))
        DO 300  K1 = 1, NV
        K2 = KL(K1)
 300    DXDB(K2) = DXDB(K2) + W(I)*DF(K2)*C(K2)*( F(I) - Y(I) )
 400    CONTINUE
 401    CONTINUE
        DXB = 0.
        DO 500  K1 = 1, NV
        K2 = KL(K1)
        DDD = DXDB(K2)*DXDB1(K2)
        ISG(K2) = ISIGN( 1, DDD )
        IF( ABS( DDD ) .LT. 1.0E-24 )  ISG(K2) = 1
 500    DXB = DXB + DXDB(K2)**2
        DXB = SQRT(DXB )
C
C       NOW GET NEW COEFICIENTS
        DO 800  K1 = 1, NV
        K2 = KL(K1)
        B(K2) = B(K2) - DXDB(K2) * STPSZ / DXB
 800    CONTINUE
        DO 40  J = 1, NPARAM
        A(J) = B(J)*C(J)
 40     CONTINUE
        KEY = 0
C
C       HOLD ANY PARAMETER THAT EXCEEDS ITS 'COOLIT' LIMITS, OR WHO IS
C       ' IN THE VALLEY ' ( I.E. WHOSE DXDB HAS DECREASED )
        IF( ICHK ) 501, 501
        CALL COOLIT( ITER, KEY, 1 )
 501    NF = NHOLD + NT
        DO 505  K1 = 1, NV
        K2 = KL(K1)
        DO 502  M1 = NHOLD1, NF
 502    IF( IHOLD(M1) .EQ. K2 )  GO TO 505
        IF( ABS( DXDB(K2) ) .GE. ABS( DXDB1(K2) ) )  GO TO 504
        NT = NT + 1
        KEY = 1
        IHOLD( NHOLD + NT ) = K2
        GO TO 505
 504    CONTINUE
        DXDB1(K2) = DXDB(K2)
 505    CONTINUE
        IF( KEY .EQ. 0 )  GO TO 1000
        NF = NHOLD + NT
```

```
      IF( NPARAM - NF )  60, 60
      DO 42  J = NHOLD1, NF
      K = IHOLD(J)
      B(K) = A(K)/C(K)
42    DXDB(K) = 0.
      GO TO 401
60    L3 = .TRUE.
1000  CONTINUE
      RETURN
      END
```

APPENDIX B

The following cards do case A, and case B twice, as described in the main text.

```
@ASG,A AMUCK * FNFIT.
@ADD AMUCK * FNFIT.DO-G
     G=A(1) + A(2)*EXP(-A(3)*X) + A(4)*EXP(A(5)*X)
@ADD AMUCK * FNFIT.DO-FNFIT
G=A(1) + A(2)*EXP(-A(3)*X) + A(4)*EXP(A(5)*X)        - "GAB" Card
100, 200, 1, 1, 0                                    - Control
5                                                    - NPARAM
1.0, $
1.0, $                                               - initial
1.0, $                                                 values and
1.0, $                                                 "magnitudes"
2.0E-2, 1
$                                                    - COOLIT Option
3, 2, 1.0E-3                                         - Limit A(3)&A(5)
5, 2, 1.0E-3                                           Greater than 10^-3
@EOF                                                 - Ends COOLIT Data
0.0                                                  - WT (0.0 means
0.0, 0.0                                               unweighted)
0.1, 9.516E-2
0.2, 1.826E-1                                        - Data (X(I), Y(I))
0.3, 2.5918E-1
etc.
@EOF                                                 - Ends Data
@EOF                                                 - Ends Program
```

NOTE: Must end program because new case requires new absolute element to be compiled.

```
@ADD AMUCK * FNFIT.DO-G
     G = 0.0
     DO 10 J = 1, 4
10   G = G + A(J) * SIN (J*A(5)*X)
@ADD AMUCK * FNFIT.DO-FNFIT
```

```
THIS IS A FOUR SINE FIT                          - "GAB"
G = SUM (A(J).SIN(A(5).J.X))                       Cards
100, 200, 1, 1, 0                                - Control
5                                                - NPARAM
1.0, $
1.0, $                                           - initial
1.0, $                                             values
1.0, $                                             and
0.02, $                                            magnitudes
$                                                  COOLIT Option
5, 1, 0.005                                A(5)  - Small Change
5, 2, 0.001                                        A(5) ≥ 10^-3
@EOF                                               End COOLIT Data
0.0                                              - WT
0.0, 0.0
0.1, 9.516E-2                                    - DATA
etc.
@EOF
100, 400, 1, 1, 0                                - Control for
                                                   second run
$                                                - Instead of NPARAM
                                                   Repeats w/old "MAGS"
$                                                  COOLIT Data
5, 1, 0.005
5, 2, 0.001
@EOF
$ Footnote                                       - instead of weight
0.0, 0.0, 1.0                                      read in weights
0.1, 9.516E-2, 1.1
0.2, 1.826E-1, 1.3                               X(I), Y(I), W(I)
etc.
@EOF
@EOF                                               End Program
```

Footnote: If this card had been

@EOF

the remaining cards up to the final @EOF would be eliminated, and the previous values for the data would be used.

GLOSSARY OF INPUT TERMS

A(1), A(2), ...	Starting values for the parameters
DA(1)...	Change in a parameter in an iteration
DUM	Actual Value of a COOLIT limit
G	User supplied function
ICHK	COOLIT indicator
IHOLD(1), IHOLD(2)...	Indicies of parameters held constant
IPRT	OUTPUT indicator
J1	Index of parameter to be cooled
L1	Type of COOLIT limit
MAG	Magnitude (power of 10) of a parameter
MAXIT	Maximum number of FNFIT iterations
MXSTP	Maximum number of GRAD steps
NHOLD	Number of parameters held constant
NPARAM	Total number of parameters
W(1), W(2)...	Weights for data points, if input by user
WT	Power of Y(I) to weight data points, if computed by program
X(I) Y(I)	DATA to be fit