AAEC/E408

AAEC/E408

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

SUPERFIT - AN INTERACTIVE PROGRAM FOR FUNCTION EVALUATION AND LEAST-SQUARES FITTING

by

B.E. CLANCY

March 1977

ISBN 0 642 99740 3

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

SUPERFIT - AN INTERACTIVE PROGRAM FOR FUNCTION EVALUATION AND LEAST-SQUARES FITTING

by

B.E. CLANCY

ABSTRACT

An interactive program package is described which provides a tool for evaluating a functional form defined by the user, for plotting it over a user-specified range, for comparing the form with experimental data specified by the user, and for carrying out least-squares fits of the functional form to the experimental data points.

National Library of Australia card number and ISBN 0 642 99740 3

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA-INIS-12(INIS: Manual for Indexing) and IAEA-INIS-13(INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

DIAGRAMS; FUNCTIONS; LEAST SQUARES FIT; MATHEMATICS; PLOTTERS; S CODES

CONTENTS

1. INTRODUCTION

It is not uncommon to be faced with one of the following tasks:

(i) To evaluate a particular function $y = f(x)$ for a range of values of a single independent variable x, with $f(x)$ having a specified analytic form.

(ii) To plot the function $f(x)$ for the given range.

(iii) To compare the function values with a set of experimental values of y, often to see simply whether or not the functional form is a reasonable representation of the experimental values.

(iv) To determine values of coefficients in the function $f(x)$ for which the functional form best fits the experimental data - the goodness of the fit being often measured by a least-squares criterion.

Provided time, a desk calculator and graph paper are available, the first three of these tasks can be carried out by conventional methods and, for the fourth task, direct techniques are available if the functional form $f(x)$ is linear in the coefficients or can be made so by a suitable transformation. For a complicated function, or one with many values of x for which the function has to be evaluated, the first three tasks become exceedingly tedious. It then becomes tempting to use a digital computer for the function evaluations and to use some sort of graph plotting system rather than pencil and paper. For the fourth task, a computer will be necessary for most non-linear least-squares fitting jobs.

In most installations having the necessary computing hardware, subprograms will probably be available for all these tasks. However, to carry out any of the four tasks, some sort of program to use the subprograms has to be written before the tasks can be carried out. This program will usually have the functional form $f(x)$ coded into it explicitly and, if the form $f(x)$ is to be changed, the program will need to be modified.

In a normal batch processing computer system, this program writing and modification may take several runs on the computer, and what should be a simple job can easily take more than a day to carry out.

SUPERFIT was conceived and developed in an attempt to remove these difficulties by providing, in one package, a set of routines for carrying out the tasks listed above while leaving the user with the responsibility of specifying only the functional form $y = f(x)$, the set of x

values to be evaluated, the initial values of any coefficients in the form, and the experimental data values (if any) with which it is to be compared.

2. STRUCTURE OF THE PROGRAM

With any program able to carry out the tasks described in Section 1, a user will most likely want to modify his ideas - say about the form of the function f(x) or typical coefficient values - after he has some results. In designing SUPERFIT, it was decided, therefore, that it should operate in an interactive fashion and allow the user to do all his modifications in the one session with the program. The program is thus designed to operate from any terminal attached *via* the AAEC Dataway system to the IBM360 computer but it should be transportable to other terminal systems. The program has four segments.

2.1 Control Segment

The control segment converses with the user, allowing him to enter data and the commands for processing these data and then to edit, plot or print the data and the results of his calculations.

Part of the user-supplied data must be a description of the functional form f(x) which is to be evaluated. This description takes the form of a block of FORTRAN statements which assumes that the variable X has some numerical value and defines a corresponding value for the variable Y. As a simple example, the sequence could consist of one line

```
Y = SIN (B*X)+(COS (X*2.0))**2                    (1)
```

which is the FORTRAN equivalent of the statement:

$$y = \sin(bx) + \cos^2 (2x) \quad ,$$

where b is some coefficient. The control segment surrounds the user-supplied statements with a shell of additional FORTRAN statements so that the whole block forms a complete FORTRAN program.

2.2 Compile Segment

The second program segment compiles the total user program and converts it into executable code. The FOREX compiler of Robinson [1968] was chosen to perform this process. The compile segment also executes the compiled program when it is free of errors and stores the results of the calculations in locations accessible to the control segment.

2.3 Plot Segment

The third segment plots the results of the calculations. For

terminals which have an interactive graphics capability, the resulting plot is displayed immediately on the display screen. For terminals without this capability (*e.g.* teletypes), the plot will, at the user's request, be transmitted to the CALCOMP for pen and paper plotting. The XYPLOT package of Trimble [AAEC/E report in preparation] is used for this task.

2.4 Fitting Segment

The shell of coding which the control segment wraps around the user-supplied FORTRAN statements includes a COMMON area which contains locations for twenty coefficients denoted by the variable names A to T inclusive. The program adheres to the standard FORTRAN convention that I,J,K,L,M and N can have only integer values, whereas the other coefficients can have the full range of real values. During his conversation with the control segment, the user may define values for all these variables and refer to them in his function definition statements. In the statement example given in Section 2.1, the variable B is a coefficient over which the user has control. In a typical situation, the user might tentatively vary this and other coefficients in an attempt to make the function give a better fit to his experimental data points.

The fitting segment varies specified coefficients in such a way as to seek the best fit to the data points; the goodness of fit is determined by a least-squares criterion. Only those coefficients which take real values are allowed to vary in this way. The fitting segment treats the integer coefficients I,J,L,M and N as fixed. The fitting segment must handle linear and non-linear functions and makes use of the Harwell Scientific Library subroutine VA06A developed by Powell [1970]. If we denote by x_i, y_i the set of experimental data values to be fitted by $f(x)$, then the error in the fit can be written as:

$$E^2 = \sum_i w_i (y_i - f(x_i))^2 \quad , \qquad (2)$$

where the w_i are weights associated with the individual data points. Here, strictly speaking, the functional form depends on the coefficients a,b,...t, so that

$$y = f(x;a,b,c,.....);$$

these coefficients are to be varied until

$$\frac{\partial E^2}{\partial a} = 0 = \frac{\partial E^2}{\partial b} = \frac{\partial E^2}{\partial c} = ...etc.$$

In describing the functional form, the user must then also describe, by FORTRAN statements, the derivatives of the function with respect to each of the varying coefficients. These derivatives are denoted by the FORTRAN variable names

DYDA, DYDB, DYDC, ... *etc.*,

which are held in the common area defined by the program shell produced by the control segment. This shell includes the coding necessary to perform the sum indicated in Equation (2) and the sums associated with the equations

$$\frac{\partial}{\partial a} E^2 = 2\sum_i w_i(y_i - f(x_i)) \frac{\partial}{\partial a} f(x_i)$$

$$\frac{\partial}{\partial b} E^2 = 2\sum_i w_i(y_i - f(x_i)) \frac{\partial}{\partial b} f(x_i) \qquad \ldots\ etc.$$

Unless the user specifies differently, the weights w_i are set to unity, and an unweighted fit is thus produced. However, the user may include in his coding a statement which defines a weight for the point (x,y) and this weight is coded as the FORTRAN variable WEIGHT. If, for instance, the weight is to be set equal to the fitted value y_i, the user's coding would include, *after* the statements defining y as a function of x, the statement

WEIGHT = Y .

With the starting values $a_o, b_o, c_o, \ldots$ of the variable coefficients, an initial pass is made through the total program produced by the user and the control segment. As a result, values of E^2 and its derivatives with respect to any variable coefficients are available in the common area.

In the fitting segment, a sequence of variables $u_1, u_2 \ldots$ and a function $G(u_1, u_2 \ldots)$ are constructed, related to the user's coefficients and the error function E^2 by the equations

$$a = a_o + \alpha_1 u_1$$

$$b = b_o + \alpha_2 u_2 \qquad etc.,$$

$$G(u_1 u_2 \ldots) = \Lambda \cdot E^2(a,b,c, \ldots)$$

with $\alpha_1 \alpha_2 \ldots$ and Λ chosen so that

$$G(o,o,o,\ldots) = 1$$

and $$\frac{\partial G}{\partial u_m}(o,o,o,\ldots) = 1 \qquad \text{for all } m.$$

The VA06A subroutine is then given the task of changing the variables u_m so as to make all the derivatives $\partial G/\partial u_m$ less than 10^{-4} in magnitude . The evaluations of G and its derivatives are performed by executing the total user's program with appropriate values of the user's coefficients. The task of reducing the magnitude of the derivatives is terminated if more than sixty passes have been made through the user's program.

After this process (called a fitting iteration) is completed, those values of the user's coefficients which gave the smallest derivatives of G and hence of E^2 are used as starting values for the next fitting iteration. If this process is repeated sufficiently often, loss of significance in evaluating the error function and/or its derivatives often results. Within the VA06A subroutine, it will appear that the E^2 function cannot be reduced by changing the coefficients in a fashion which should reduce E^2 and the subroutine then transmits an error message to the user. This inability to reduce the error E^2 would also be present if the user had made certain errors in coding the function or its derivatives, for instance, if the derivative had been coded with the wrong sign. The error message suggests that the user check his function coding but, at the end of a series of fitting iterations, it usually means that the best possible fit has been achieved with the function form chosen.

3. PROGRAM USE AND DATA INPUT

The program is available as a catalogued procedure, requires 120K bytes of core storage and is invoked by the job control language statement:

```
// EXEC SUPERFIT
```

The procedure has three steps, the first (PLT) prepares for plotting on the CALCOMP and the second (GO) interacts with the user. When the GO step commences, the user should log on at a terminal using the link task $.READY and wait for instructions. The first is a request from the program to know whether the user's terminal has a graphics capability, after which the program enters the input phase of the control segment. This is indicated by the program typing INPUT? at the user's terminal.

At this stage, the user must type in a keyword to indicate his wishes - a list of the 29 allowable keywords and their function will have been typed at the terminal by the program on initial entry to the control segment. The keywords (only the first four characters of which are significant) fall into three classes: data entry, data output and command.

The third step cancels all printed output for the job. It is not executed if, during the GO step, the user has requested output on the line printer.

3.1 Data Entry Keywords

Each of these keywords instructs the program to accept data of a particular kind and the program responds by typing instructions to the user as to how the particular data should be entered. The individual keywords and their significance are now described.

POINT is used to initiate the entry of the user's 'experimental' data points if any. A maximum of 200 point pairs can be entered for any one case. The number of points is first requested, then the set of x values and finally the set of y values. These are read with the SCAN subroutine of Bennett & Pollard [1967] as is all other data entered.

CURVE signals to the program that the user wishes to enter a set of x values (again 200 is the maximum number) at which he wants his function evaluated. If he gets a plot of his results, it will include a curve joining the points thus determined.

COEF prepares the code to accept values for any of the allowable 20 coefficients. The code types instructions for this entry to the user who might respond by typing the line

```
A = 1.2   N = 5   B = 14   C = 1   T = 3.   Z
```

the Z being a signal that no more coefficient values follow. Unless the user specifies otherwise, all coefficients are initially set to zero.

F signals the code to accept the sequence of FORTRAN statements which define his function y = f(x), the derivatives of y with respect to any coefficients and the weight associated with the point (x,y). Up to 150 statements can be inserted and the user should signal that he is finished entering them by typing END after his last statement has been entered.

PAPE is used to prepare the code to accept information about the form of plot to be produced. With instructions from the code, the user may specify linear or logarithmic scales for his graph axes and define a heading for his plot as well as for the x and y axes.

VARY is the keyword used when a fit to the data points is to be made. When requested by the code, the user types in the names of the coefficients (any of A,B,C,... *etc.* but not I,J,K,L,M or N which are integers) which the least-squares fitting routine can vary. In some situations, it may be necessary to restrict variations in the coefficients which the code may make during any one interaction. In such a case, the absolute value of the allowable variation should be entered immediately following the coefficient letter. Such a situation may arise when the coefficient appears in an exponent - say e^{bx} - when too large a variation in the coefficient b is likely to cause arithmetic overflow in calculating the exponential function. The effect of the VARY keyword is cumulative so that the sequence of entries:

```
VARY
A B 0.1 C Z
VARY
D
```

is the same as the single entry:

```
VARY
A B 0.1 C Z
```

Once a coefficient is allowed to vary, it remains variable unless its allowable variation is set to zero or it is reset to be a fixed coefficient by the use of the next keyword.

FIX prepares the program to accept a string of coefficient letters (again terminated by Z) which are to be treated as constants.

CORRECT is a keyword which lets the user modify the data point pairs (x,y) which he has previously entered. Under instruction by the code, he may correct or delete any points or add new ones.

EDIT is the keyword which lets the user change or add to the sequence of statements he has entered to define his function. A full description of the edit procedure is given in Section 3.4. At this point we simply remark that the facilities are a

subset of those available with the $LOGON program of Backstrom [1976].

3.2 Data Output Keywords

These keywords instruct the program to list any data items which have been entered from the terminal or been produced by the user's function coding. The keywords are formed from the corresponding data entry keywords by preceding them with ? or P. Those beginning with ? produce a listing at the user's terminal whereas those beginning with P produce a listing on the IBM360 line printer. The specific ? keywords are:

?F produces a list of the statements defining the user's function.

?PAPE produces a listing which describes the plot scales, the plot heading and the labels on the x and y axes.

?VARY produces a listing of those coefficients which have been nominated as variable, together with the derivative of the error function E^2 with respect to those coefficients.

?COEF produces a listing of the values of those coefficients which are non-zero.

?CURV produces a tabulation of the user-defined function at the x values entered with the CURVE keyword.

?POINT produces a tabulation of the data values entered after the POINT keyword, together with the y values calculated from the user's function and the difference between the calculated and experimental y values.

The corresponding information is listed on the line printer as job output if the ? prefix is replaced by a P.

The entry of the prefix ? or P alone produces a complete listing as if all the ? or P prefixed keywords had been entered successively.

HELP is the last keyword in this particular class. It produces a list of the keywords and their uses at the user's terminal.

3.3 Command Keywords

These keywords transfer control from the conversational segment of the program to one of its other segments and thus carry out one or other of the tasks described in the Introduction.

GO causes the user's program to be compiled and then executed if there are no compilation errors. (The compilation phase is

actually bypassed if the user's coding has been successfully compiled following a previous GO statement.) After execution of the program, the user's function f(x) will have been evaluated at any of the x values entered with a POINT or CURVE keyword and the error function E^2 evaluated according to Equation (2). If experimental data values have been entered, the value of E^2 is typed back at the terminal before the code re-enters the conversational mode by typing the message INPUT?. If the terminal has an interactive graphics capability, the code will automatically send to the terminal a plot of the experimental and calculated data points and the curve through those points calculated for the x values is entered with the CURVE keyword. If the user has defined weights for his fit, these will be interpreted as inverses of the variances on the experimental data points and these points will be plotted with one standard deviation error bars.

PLOT is the keyword which sends the equivalent graphics information to the CALCOMP for pen and paper plotting. The information is normally spooled in the Q output queue which allows the user, if he wishes, to inspect it using one of the interactive graphics terminals.

FIT is the keyword used to begin a sequence of fitting interactions with the VA06A subroutine as described in Section 2. The user must, when the program requests it, specify the number of iterations to be performed. At the end of each iteration, the least value of the error function achieved so far is typed back at the user's terminal so that he may monitor the progress of the fitting. When the sequence is finished, whether by completion of the requested number of iterations, by the inability of the VA06A routine to reduce the error function or by use of the interrupt procedure described in Section 3.5, the program produces the same output as that from the GO keyword.

SUSPEND is a keyword which suspends operation of the program and the terminal, thus freeing the terminal for someone else to use. The user may reactivate and carry on the program with no loss of the stored information by logging on again at any terminal.

STOP is the final keyword and this closes down the terminal and

terminates execution of the program. Unless the user has requested printed output, the catalogued procedure then causes HASP output for the job to be cancelled. The user should note that if the program detects that only a few seconds of the user's job time remain at any stage of entry to the conversational control segment, it automatically processes the keywords P and STOP, thus giving the user sufficient output to enable him to carry on with his job at another session.

3.4 EDIT Facility

EDIT is the keyword which lets the user modify the sequence of function statements entered earlier. After detecting the keyword, the program attaches a line number to each statement and lists the statements - each preceded by its line number - at the user's terminal. Editing can then begin.

Two modes of editing are available to the user - input mode and command mode - with the program initially in command mode. An immediate carriage return changes the mode from command to input and *vice versa*. The simpler input mode is described first.

Input Mode

In this mode, lines containing up to 72 characters may be entered sequentially. Line numbers are generated automatically and appear with a trailing blank at the left hand edge of the page. After the user types in his next statement, it is appended to the sequence of statements already defined and a new line number is generated. The process continues until the user changes to command mode.

Command Mode

Readiness of this mode is signified to the user by the appearance of a prompting hyphen at the left hand edge of the page and the user must respond with some command. At this stage the program will respond normally to any of the SUPERFIT keywords (except STOP and F) described earlier, first leaving the EDIT phase and then processing the keyword. If the user types any of the commands E, F, END or STOP, the edit phase terminates and returns to the control segment of SUPERFIT.

In addition, the following command words are recognised in this mode:

LIST (alias L), RENUMBER (R), CLEAR (CL), DUPLICATE (D), MOVE (M), INSERT (I) and CHANGE (C).

In the following description, the symbols m,n are any numbers and usually

line numbers while the double quotes "" are used to identify the command string but they should not be typed at the terminal.

LIST:

"L" on its own lists the entire sequence of statements. The single line m, or the lines in the range m to n, may also be listed by the commands

"L m" or "L m n"

RENUMBER:

"R" on its own renumbers the present sequence of statements in steps of ten starting from ten. The step size may be changed, however, by using the command "R n".

CLEAR:

"CL" on its own is not recognised. Line number m or all lines in the range m to n inclusive may be removed from the sequence by the commands

"CL m" or "CL m n"

DUPLICATE:

The command "D m n" inserts a copy of the line numbered m into the sequence at position n. If a line numbered n already exists it will be replaced by the duplicated line.

MOVE:

The command "M m n" has the same result as "D m n" except that the line originally numbered m is removed from the sequence.

INSERT:

The command "I m n" allows statements to be inserted (as if in input mode) beginning with line number m and continuing at lines m+n, m+2n, *etc.* If the increment n is omitted, it is assumed by the code to have a value of 1.

CHANGE:

The command "C m /ABC/DEFG" causes the first appearance of the string of characters ABC in line m to be deleted and the string DEFG to be inserted in their place. Instead of the slash /, any non-blank character can be used to delimit the strings which can be of any length. After the command is executed, the changed line is typed back at the terminal for user verification.

The user may respond in command mode by typing a line beginning with a sequence number m and trailing blank. If nothing else follows,

this is treated as a command to clear line number n. If a string follows the trailing blank, it is inserted in the sequence of statements at a position determined by the line number. If line number n is already present, it is replaced by the new line.

3.5 Interruptions to Boring Monologues

At various stages, the conversation with the program may have the appearance of a monologue with SUPERFIT doing all the talking (or typing). For example, the user who knows the various keywords will find it tedious when, at program start time, SUPERFIT begins by typing the list of keywords at his terminal. If he has made an error in his function coding and the compiler segment has detected it, the user may resent having to wait while the whole program with error messages is typed at his terminal, particularly if he is aware of the nature of the error. The user who has given the ?POINT or ?CURVE keyword may only be interested in one or two values whereas the program will, if left alone, type all of them at this terminal.

A final example is the situation where the user has requested a large number of fitting iterations but can see, from the first few values of E^2 typed at his terminal, that the fitting process has proceeded far enough for his present purposes. All of these monologues and a few others can be interrupted if the user presses the ? key at his terminal. This usually brings the program back to a true conversational mode and it will signal this by typing INPUT? at the user's terminal and waiting for a keyword. The exceptions to this occur when the program is responding to the ?POINT or ?CURVE keywords. The ? character then simply suspends the list typing and the program types CONT.... at the terminal and waits to accept an integer number. The user's response as read by SCAN is treated as the number of items in the list to be skipped over from where the interruption occurred and the program continues to process the previous keyword. A negative number is treated as a request to go backwards in the list before restarting the keyword processing. This provides the user with the opportunity of examining a few data values if he so wishes. If the number of items to be skipped (or backspaced) could cause the keyword processing to restart out of its range, the processing is terminated.

The interrupt facility can be very useful if the user's function coding or the coefficient values which he has entered are such as to cause arithmetic overflows or divide checks since ten of these errors

cause his job to be terminated. The FORTRAN messages signalling these errors are sent directly to the terminal; if they appear, the user should try to interrupt immediately and examine the data he has entered to see what is going wrong. The interrupt attempt may not be successful since its presence is only checked at certain stages of the program, but the user may be lucky enough to save his job from aborting.

4. FUNCTIONS AND SUBROUTINES AVAILABLE TO THE USER

When writing the sequence of statements which define the functional form y = f(x), the user may refer to any of the following mathematical functions:

(i) the direct and inverse trigonometric functions SIN(X),COS(X), TAN(X),COTAN(X),ARSIN(X),ARCOS(X),ATAN(X),

(ii) the hyperbolic functions SINH(X),COSH(X),TANH(X),

(iii) the exponential and logarithmic functions EXP(X),ALOG(X),ALOG10(X),

(iv) the uniform random number function RAND(X) , for which 0<RAND<1 , and

(v) the arithmetic functions ABS(X),SQRT(X),

as well as the arithmetic operations + - * / and ** of standard FORTRAN.

The user's access to subroutines is very restricted. He has access to the free input subroutine SCAN and to the linear equation-solving subroutine SID. He also has access to the FORTRAN input/output subroutines and he may include in his coding any normal input output statements. In this context he should note that in SUPERFIT, FORTRAN logical units 5 and 6 are reserved for direct input and output at the user's terminal. Finally, the user has access to a simplified form of the AEBUF subroutine which allows him to do a limited amount of in-core writing. The subroutine is called in the fashion:

CALL XYBUF (BUF, NCH, FMT, VBLE)

where VBLE is the single variable to be written,

FMT is the name of dimensioned array containing the appropriate FORMAT for writing,

BUF is the name of the array to which the variable is to be written, and

NCH (which is returned by the subroutine) will be the number of characters in BUF occupied by the writing of VBLE.

This subroutine was made available to the user's program mainly to allow the construction of a suitable plot heading from within the program; an example of its use in this fashion is given in Section 6.3.

5. STATISTICS OF THE FITTING PROCESS

Part of the shell which the control segment wraps around the user's function statements is a section of coding which calculates the matrix of elements

$$(VARMT)_{m,n} = \sum_{i=1}^{NPOINT} w_i \frac{\partial y_i}{\partial v_m} \frac{\partial y_i}{\partial v_n} ,$$

where y_i is the i^{th} fitted y value,

w_i is the weight attached to this value, and

$v_1, v_2, v_3, \ldots$ are the coefficients a,b,c,... which the user has decreed will be variables for the fitting process.

After a sequence of fitting iterations, the control segment inverts this matrix and stores the diagonal element of the inverse in an array labelled VARNCE. If, in his function statements, the user has included a definition of the weights which sets the w_i equal to the inverse of the variances of the point y variables, the elements of the VARNCE array will be the squares of the standard error in the estimates of the coefficients v_m which the fit has produced. The inverted matrix is printed when the user enters the command PCOEF.

6. EXAMPLES OF SUPERFIT USE

Several examples of runs with SUPERFIT are given below, the listings being those obtained at a DECWRITER terminal. To make the action clear, the user's input is typed in lower case and the responses from SUPERFIT are in upper case.

6.1 Tabulating and Plotting a Simple Function

This is the simplest use for SUPERFIT where the user wishes to tabulate the function

$$y = e^{-(x^2/t)} / \sqrt{4t}$$

for x in the range 0 to 1,

and t having the values 0.3 and 0.2.

The terminal listing includes a $LOGON job to submit SUPERFIT, use of the $#CONSOLE to monitor the progress of the job and the initiation of the $.READY link task. In entering his function, the user erred by

providing a mixed mode statement. After his first GO command, the compiler segment detected the error, and typed the full program listing at the terminal. For this example, it is simpler to retype the function than to go through the edit procedure and this was done by the user. The rest of the run was uneventful. The plot produced is shown in Figure 1.

EXAMPLE 1

```
$#CONSOLE
ID:
=$LOGON
JOBNAME: BECFIT

TABS 0,
0000 //BECFIT   JOB ('********/DOOPHICI',N1),B.E.CLANCY,
0010 //       MSGLEVEL=(2,0),CLASS=A,TIME=2
0020 /*ROUTE  PRINT PHYS
0030 /*JOBPARM L=1,K=0
0040 // EXEC SUPERFIT,SYSOUT=F
-SUBMIT
 BECFIT SUBMITTED AT 14.50.06
-END
END:LOGON
=D BECFIT
 BECFIT    GO          JTU=000M00.2S TTG=001M59.2S RC=120K,UUC=004K
=END
END:CONSOLE
$.READY
ID:
JOBNAME: BECFIT
OK

JOB 283 BECFIT   NOW READY (251)


IS PLOTTING POSSIBLE AT YOUR TERMINAL?
ANSWER YES OR NO
no
-DATA ENTRY KEYWORDS ARE:-
F     TO ENTER USERS CODING
PAPE TO ENTER XYPAPE ARGUMENTS
POIN TO ENTER POINT COORDINATES
CURV TO ENTER CURVE X-COORDINATES
COEF TO ENTER COEFFICIENT VALUES
VARY TO ALLOW COEFFICIENTS TO VARY
FIX  TO STOP THEM VARYING
-DATA ENQUIRY KEYWORDS
 TO FIND PRESENT VALUES OF DATA
    PRECEDE KEYWORD BY ? (E.G.  ?F)
 TO PRINT PRESENT VALUES OF DATA
    PRECEDE KEYWORD BY P
 ? OR P ALONE GIVES ALL INFORMATION
-COMMAND KEYWORDS ARE:-
EDIT  TO MODIFY USERS CODING
GO    TO COMPILE AND RUN USERS CODE
FIT   TO TRY LEAST SQUARES FIT TO POINTS
CORR  TO CORRECT OR DELETE POINT DATA ENTERED
PLOT  TO GET CALCOMP PLOT
HELP  WHICH WRITES THIS MESSAGE
```

```
SUSPEND TO SUSPEND TERMINAL SESSION
STOP  TO TERMINATE JOB
- TO ACCESS JOB DATA BLOCKS HELD IN LIBRARY
SAVE   (WITH A JOBNAME  , IF DESIRED)
LOAD       "
DELETE     "
?SAVE TO LIST THOSE DATA BLOCKS ALREADY SAVED

    INPUT?

*Scan data procedure ignbores all lines startins with asterisk
*
*Example 1.   Evaluate, tabulate and plot a simple function.
curv
ENTER NUMBER OF POINTS FOR CURVE (200 MAXIMUM)
101
ENTER X-COORDINATES OF POINTS FOR CURVE
0.0(0.01)1.0

    INPUT?
f
ENTER FORTRAN STATEMENTS TO DEFINE
      Y=F(X)
      DYDA,DYDB, ....  ETC. IF YOU WANT TO FIT
      WEIGHT IF YOU WANT A WEIGHTED FIT
AVOID STATEMENT NUMBERS 9996-9999
NO MIXED MODE STATEMENTS PLEASE
YOU MAY USE ANY COEFFICIENTS (A-T), THEY ARE IN COMMON
END   IN COLUMNS 1-3 TO FINISH
      y=exp(-x*x/t)/sqrt(4*t)
end

    INPUT?
so
      COMMON A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,XVALUE(400),YVALUE(
     1400),NVALUE,DESQDC(14),DYDA,DYDB,DYDC,DYDD,DYDE,DYDF,DYDG,DYDH,DYD
     20,DYDP,DYDQ,DYDR,DYDS,DYDT,NPOINT,YPOINT(200),ERRRSQ,VARMT(14,14),
     3HEADER(18),XLABEL(18),YLABEL(18),NHEAD,NXLABL,NYLABL,XPOINT(201)
      DIMENSION DYDCOF(14),COEFFT(20),PVAR(200)
      EQUIVALENCE (DYDCOF(1),DYDA),(COEFFT(1),A),(XPOINT(201),PVAR(1))
      DO 9996 ICOF=1,14
      DESQDC(ICOF)=0.
      DYDCOF(ICOF)=0.
      DO 9996 JCOF=1,14
 9996 VARMT(JCOF,ICOF)=0.
      ERRRSQ=0.
      DO 9999 IVALUE=1,NVALUE
      Y=0.
      X=XVALUE(IVALUE)
      WEIGHT=1.
C ----USERS CODING BEGINS ----------------------------------------
      Y=EXP(-X*X/T)/SQRT(4*T)
C ----USERS CODING ENDS------------------------------------------
EVERITY 4
             MIXED MODE EXPRESSION
      IF(IVALUE.GT.NPOINT) GO TO 9999
      ERROR=Y-YPOINT(IVALUE)
      ERROW=ERROR*WEIGHT
      ERRRSQ=ERRRSQ+ERROR*ERROW
      DO 9998 ICOF=1,14
      IF(DYDCOF(ICOF).EQ.0.) GO TO 9998
      DESQDC(ICOF)=DESQDC(ICOF)+ERROW*DYDCOF(ICOF)*2.0
      DO 9997 JCOF=1,14
```

```
 9997 VARMT(JCOF,ICOF)=VARMT(JCOF,ICOF)+WEIGHT*DYDCOF(JCOF)*DYDCOF(ICOF)
 9998 CONTINUE
      PVAR(IVALUE)=WEIGHT
 9999 YVALUE(IVALUE)=Y
      END
-------------------------------
-------------------------------
COMPILATION ERROR, RE-ENTER Y=F(X)

    INPUT?

*Simplest way to correct this error is to re-enter function.
f
ENTER FORTRAN STATEMENTS TO DEFINE
      Y=F(X)
      DYDA,DYDB, ....  ETC. IF YOU WANT TO FIT
      WEIGHT IF YOU WANT A WEIGHTED FIT
AVOID STATEMENT NUMBERS 9996-9999
NO MIXED MODE STATEMENTS PLEASE
YOU MAY USE ANY COEFFICIENTS (A-T), THEY ARE IN COMMON
END   IN COLUMNS 1-3 TO FINISH
      y=exp(-x*x/t)/sqrt(4.0*t)
end

    INPUT?
coef
ENTER COEFFICIENT (SINGLE LETTER) & VALUE (E.G.  A=1.0)
ENTER LETTER Z TO FINISH
t= 0.3   z

    INPUT?
so

    INPUT?
*Compilation was successful.
*tabulate part of function at terminal
?curv
? WILL INTERRUPT TYPING OF
CURVE COORDINATES
     0.0              9.12871E-01
     1.00000E-02      9.12567E-01
     2.00000E-02      9.11655E-01
     3.00000E-02      9.10137E-01
     4.00000E-02      9.08016E-01
     5.00000E-02      9.05296E-01
CONT... +90  to see what values are like at far end.
     9.60000E-01      4.22919E-02
     9.70000E-01      3.96568E-02
     9.80000E-01      3.71611E-02
     9.90000E-01      3.47992E-02
     1.00000E+00      3.25658E-02

     INPUT?

*Print the tabulation on the line printer/
?curv
```

```
    INPUT?
* and print the coefficient value (t) for convenience.
pcoef

    INPUT?
*Plot the function after setting up plot paper to suit purposes.



pape
X-SCALE LOGARITHMIC? ENTER YES OR NO
no
Y-SCALE LOGARITHMIC?
no
ENTER NEW PLOT HEADER (OR IMMEDIATE C.R.)
y=exp(-x*x/t)/sqrt(4*t)
ENTER NEW X-AXIS LABEL (OR IMMEDIATE C.R.)
 x
ENTER NEW Y-AXIS LABEL (OR IMMEDIATE C.R.)
 y  with t+0.3

    INPUT?
plot

    INPUT?
*now change value of t
coef
ENTER COEFFICIENT (SINGLE LETTER) & VALUE (E.G.  A=1.0
ENTER LETTER Z TO FINISH
t = 0.2    z

    INPUT?
go

    INPUT?
*  and tabulate this at printer
pcurv

    INPUT?
pcoef

    INPUT?
*now all finished
stop
END OF RUN
```

6.2 Fitting the Sum of Two Exponentials to a Set of Data

Output for this example starts where SUPERFIT first began to interact with the user. Liberal use has been made of the monologue interruption feature described in Section 3.5. In entering his data values, the user made two mistakes and these were corrected before proceeding further. Two fits were made, one unweighted, the second weighting the values by 1/y - the normal procedure if the variance of the data is due solely to counting statistics. The plot of the fitted function (shown in Figure 2) seems reasonable, but the value of E^2 is still too high for a satisfactory fit.

EXAMPLE 2

```
JOB 301 BECFIT   NOW READY (251)


IS PLOTTING POSSIBLE AT YOUR TERMINAL?
ANSWER YES OR NO
no
-DATA ENTRY KEYWORDS ARE:-
F     TO ENTER USERS CODING
PAPE TO ENTER XYPAPE ARGUMENTS

     INPUT?
* A ? was used to interrupt the listing of keywords
* Example 2.  Fitting data points with two exponentials
point
ENTER NUMBER OF POINTS TO BE FITTED (200 MAXIMUM)
19
ENTER X-COORDINATES OF POINTS
20(20)380
ENTER Y-COORDINATES OF POINTS
53408 47585 441960 37960 35550 33348 30670 27979 27797
26657 25288 23683 211971 21131 20306 19382 18300 15993 15302

     INPUT?
* The third and thirteenth y-values were entered wronsly
*       and have to be corrected
corr
HOW MANY POINTS TO BE CORRECTED?
2
ENTER POINT NUMBER
3
OLD X,Y ARE  6.00000E+01  4.41960E+05
ENTER NEW X & Y
60 41960
ENTER POINT NUMBER
13
OLD X,Y ARE  2.60000E+02  2.11971E+05
ENTER NEW X & Y
260 21971
HOW MANY POINTS TO BE DELETED?
0
```

```
HOW MANY POINTS TO BE ADDED?
0

     INPUT?
f
ENTER FORTRAN STATEMENTS TO DEFINE
       Y=F(X)
       DYDA,DYDB, ....  ETC. IF YOU WANT TO FIT
       WEIGHT IF YOU WANT A WEIGHTED FIT
AVOID STATEMENT NUMBERS 9996-9999
NO MIXED MODE STATEMENTS PLEASE
YOU MAY USE ANY COEFFICIENTS (A-T), THEY ARE IN COMMON
END    IN COLUMNS 1-3 TO FINISH
       dyda=exp(b*x)
       dydb=a*x*dyda
       dydc=exp(d*x)
       dydd=c*x*dydc
       y=a*dyda+c*dydc
end

     INPUT?

coef
ENTER COEFFICIENT (SINGLE LETTER) & VALUE (E.G.  A=1.0)
ENTER LETTER Z TO FINISH
a=50000 b=-0.004 c=2000 d=-1.3e-4      z

     INPUT?
curv
ENTER NUMBER OF POINTS FOR CURVE (200 MAXIMUM)
20
ENTER X-COORDINATES OF POINTS FOR CURVE
0(20)380

     INPUT?
vary
ENTER NEW (AND/OR OLD) VARYING COEFFICIENT LETTERS
FOLLOW EACH BY MAX. VARIATION IF DESIRED
ENTER Z TO FINISH
a  b  5.e-4 c  d 1.e-5
*    better not allow the exponents to vary too wildly to start.
z

     INPUT?
go
ERRRSQ =     1.19630E+08

     INPUT?
*  Compiled correctly , so now try a fit.
fit
HOW MANY ITERATIONS?
7  will do for now
? WILL INTERRUPT
ERRRSQ =     1.19630E+08
ERRRSQ =     2.55163E+07
ERRRSQ =     2.29654E+07
ERRRSQ =     6.79296E+06
```

```
ERRRSQ =     5.57790E+06
ERRRSQ =     5.49850E+06
VA06A IN TROUBLE-MAY BE ROUNDOFF
   BUT CHECK YOUR FUNCTION CODING
ERRRSQ =     5.49830E+06

     INPUT?
?coef
NON-ZERO COEFFICIENTS (A-T) FOLLOW
A =      1.93616E+04
VARNCE=    7.421E+00
B =     -2.49645E-02
VARNCE=    6.773E-11
C =      4.42424E+04
VARNCE=    7.379E+00
D =     -2.66678E-03
VARNCE=    5.127E-14

     INPUT?

*variances are meaninsless with all weishts set to unity
*so no modify prosramme (user's function)


edit
-0010          DYDA=EXP(B*X)
-0020          DYDB=A*X*DYDA
-0030          DYDC=EXP(D*X)
-0040          DYDD=C*X*DYDC
-0050          Y=A*DYDA+C*DYDC
-
-0060          weisht=1.0/y
-0070
-so
EDIT COMPLETE
ERRRSQ =     2.34444E+02

     INPUT?
*now refit with the weishts in the function definition
fit
HOW MANY ITERATIONS?
5
? WILL INTERRUPT
ERRRSQ =     2.34444E+02
ERRRSQ =     2.30539E+02
VA06A IN TROUBLE-MAY BE ROUNDOFF
   BUT CHECK YOUR FUNCTION CODING
ERRRSQ =     2.30539E+02

     INPUT?
*try one more time
fit
HOW MANY ITERATIONS?
2
? WILL INTERRUPT
ERRRSQ =     2.30539E+02
```

```
VA06A IN TROUBLE-MAY BE ROUNDOFF
   BUT CHECK YOUR FUNCTION CODING
ERRRSQ =     2.30539E+02

     INPUT?
?coef
NON-ZERO COEFFICIENTS (A-T) FOLLOW
A =       1.93825E+04
VARNCE=    3.929E+05
B =      -2.73718E-02
VARNCE=    2.804E-06
C =       4.49740E+04
VARNCE=    1.542E+05
D =      -2.72699E-03
VARNCE=    9.931E-10

     INPUT?

page
X-SCALE LOGARITHMIC? ENTER YES OR NO
no
Y-SCALE LOGARITHMIC?
yes
ENTER NEW PLOT HEADER (OR IMMEDIATE C.R.)
fit to double exponential
ENTER NEW X-AXIS LABEL (OR IMMEDIATE C.R.)
time (seconds)
ENTER NEW Y-AXIS LABEL (OR IMMEDIATE C.R.)
counts

     INPUT?
plot
ERRRSQ =     2.30539E+02

     INPUT?
*print everything
p

     INPUT?
stop
END OF RUN
```

6.3 A Fitting Problem Showing Use of XYBUF Subroutine

In this example, the square-wave function

$y = 1 \quad 0<x<1$

$y = 0 \quad x = 0$

$y = 0 \quad x = 1$

is approximated by a discrete sequence of point pairs entered as data after the POINT keyword. The user's function is the partial Fourier series

$$y = \sum_{\ell=1}^{N} a_\ell \sin [(2\ell - 1)\pi x] \quad .$$

Instead of referring directly to each of the COMMON variables A,B,C,... which are the coefficients in the sum, use is made of the array COEFFT which is equivalent to these variables, care being taken not to reference the non-varying integer variables I,J,...N when stepping through this array. On the first pass through the user's coding, when IVALUE is one, the subroutine XYBUF is called. As coded, this has the effect of writing the coefficient N onto the array HEADER using the format defined in the array FMT. SUPERFIT uses the first NHEAD characters of the array HEADER as a title for each plot. The XYBUF subroutine call in the example thus automatically provides the appropriate heading for the plots following the function evaluations made with different values of N. In similar fashion, use can be made of the arrays XLABEL and YLABEL from which NXLABL and NYLABL characters are used to label to x and y axes of plots. The plots produced in this example are given as Figures 3 and 4.

EXAMPLE 3

```
JOB 330 BECFIT    NOW READY (253)


IS PLOTTING POSSIBLE AT YOUR TERMINAL?
ANSWER YES OR NO
no
-DATA ENTRY KEYWORDS ARE:-
F     TO ENTER USERS CODING
PAPE TO ENTER XYPAPE ARGUMENTS

     INPUT?
*     Example 3.     "Fourier" fit  to square wave.
point
ENTER NUMBER OF POINTS TO BE FITTED (200 MAXIMUM)
441
ENTER NUMBER OF POINTS TO BE FITTED (200 MAXIMUM)
41
ENTER X-COORDINATES OF POINTS
0(0.025)1
ENTER Y-COORDINATES OF POINTS
0 39*1 0

     INPUT?
f
ENTER FORTRAN STATEMENTS TO DEFINE
       Y=F(X)
       dimension fmt(8)
       data fmt /'(i4,14h odd harmonics)'/
       data pi /3.141593/
       if(ivalue.ne.1) go to 1
       call xybuf(header,nhead,fmt,n)
```

```
      1 do 10 ii=1,n
        ei=ii+ii-1
        sinang=sin(ei*x*pi)
        jj=ii
        if(jj.gt.8)  jj=ii+6
c       this jumps the  jj  index over the fixed coefficients.
        dydcof(ii)=sinang
     10 y=y+sinang*coefft(jj)
end

     INPUT?
coefft
ENTER COEFFICIENT (SINGLE LETTER) & VALUE (E.G.  A=1.0)
ENTER LETTER Z TO FINISH
n=12 a=1.     z

     INPUT?

go
ERRRSQ =     8.09658E+00

     INPUT?
*  Compiles  correctly
vary
ENTER NEW (AND/OR OLD) VARYING COEFFICIENT LETTERS
FOLLOW EACH BY MAX. VARIATION IF DESIRED
ENTER Z TO FINISH
a b c d e f g h o p q r        z

     INPUT?
fit
HOW MANY ITERATIONS?
3
? WILL INTERRUPT
ERRRSQ =     8.09658E+00
ERRRSQ =     6.21663E-02
VA06A IN TROUBLE-MAY BE ROUNDOFF
   BUT CHECK YOUR FUNCTION CODING
ERRRSQ =     6.21660E-02

     INPUT?
curv
ENTER NUMBER OF POINTS FOR CURVE (200 MAXIMUM)
200
ENTER X-COORDINATES OF POINTS FOR CURVE
0(0.005)1

     INPUT?
go
ERRRSQ =     6.21660E-02

     INPUT?
plot
ERRRSQ =     6.21660E-02

     INPUT?
```

```
curv
ENTER NUMBER OF POINTS FOR CURVE (200 MAXIMUM)
* Fitting loop goes faster if no curve points have to be evaluated.
0

     INPUT?
fix
ENTER COEFFICIENT LETTERS TO BE FIXED
ENTER Z TO FINISH
o p q r      z

     INPUT?
coef
ENTER COEFFICIENT (SINGLE LETTER) & VALUE (E.G.  A=1.0)
ENTER LETTER Z TO FINISH
n=8     z

     INPUT?

fit
HOW MANY ITERATIONS?
4
ERRRSQ =     2.74932E-01

     INPUT?
fit
HOW MANY ITERATIONS?
4
? WILL INTERRUPT
ERRRSQ =     2.74932E-01
VA06A IN TROUBLE-MAY BE ROUNDOFF
   BUT CHECK YOUR FUNCTION CODING
ERRRSQ =     2.74932E-01

     INPUT?
curv
ENTER NUMBER OF POINTS FOR CURVE (200 MAXIMUM)
200
ENTER X-COORDINATES OF POINTS FOR CURVE
0(0.005)1

     INPUT?
go
ERRRSQ =     2.74932E-01

     INPUT?
plot
ERRRSQ =     2.74932E-01

     INPUT?
p      print everything

     INPUT?
stop
END OF RUN
```

7. THE SUPERFIT COMMON AREA

This area supplies the necessary connection between the user's FOREX program and the rest of the SUPERFIT program, allowing them to pass values of variables to one another. The meanings of many of the variables have been described earlier in this report and, for most applications, the user need not concern himself with the area. In any event, much of the area's structure is self-evident from an example of the program listing given in Section 6.1, but the following description may be useful in special cases.

The NPOINT x,y values of the data points entered by the user are held in the arrays XPOINT and YPOINT which are common. The NLINES x values entered by the user after the CURV keyword are held in an array XLINES; neither the XLINES array nor the variable NLINES are accessible to the FOREX program. When the user enters either of the keywords GO or FIT, the NPOINT numbers from the array XPOINT are moved to the array XVALUE and then followed by the NLINES numbers from the array XLINES. The array XVALUE then contains NVALUE (=NPOINT + NLINES) numbers which are the different x coordinates for which the user's function is to be evaluated, the results being stored in the array YVALUE. After execution of the user's FOREX program, the calculated y values are extracted from this array and stored in private arrays.

8. ACCESSING DATA AND PROGRAMS FROM DISC

8.1 Reading Point Data from Data Sets

A user with a large number of data values could find it tedious if he had to enter these values into SUPERFIT by typing them in at a terminal when they were already stored on magnetic tape or disc. Such data can be made available to SUPERFIT by judicious design of the statements which (nominally) define the user's function.

The user's function printed below provides an example of the way in which data on disc may be accessed. The disc data set was previously written as one record with the following layout:

N_1, N_2,

E_1, E_2, ... E_{185}

σ_1, σ_2, ... σ_{185}

in which N_1, N_2 were identifying numbers and the E_i and σ_i were neutron energy and cross section values. Before the compilation and execution of the user's coding, it was necessary to define 185 dummy values of E,

σ in the XPOINT, YPOINT arrays. On the first pass through the user's code (when the variable IVALUE was set to 1), the correct values were read into these arrays for use with a more practical user's function.

```
      COMMON A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,XVALUE(400),YVALUE(
     1400),NVALUE,DESQDC(14),DYDA,DYDB,DYDC,DYDD,DYDE,DYDF,DYDG,DYDH,DYD
     20,DYDP,DYDQ,DYDR,DYDS,DYDT,NPOINT,YPOINT(200),ERRSQ,VARMT(14,14),
     3HEADER(18),XLABEL(18),YLABEL(18),NHEAD,NXLABL,NYLABL,XPOINT(201)
      DIMENSION DYDCOF(14),COEFFT(20),PVAR(200)
      EQUIVALENCE (DYDCOF(1),DYDA),(COEFFT(1),A),(XPOINT(201),PVAR(1))
      DO 9996 ICOF=1,14
      DESQDC(ICOF)=0.
      DYDCOF(ICOF)=0.
      DO 9996 JCOF=1,14
 9996 VARMT(JCOF,ICOF)=0.
      ERRSQ=0.
      DO 9999 IVALUE=1,NVALUE
      Y=0.
      X=XVALUE(IVALUE)
      WEIGHT=1.
C ----USERS CODING BEGINS ----------------------------
C     READ E, SIGMA VALUES FROM DISC
      IF(IVALUE.GT.1) GO TO 4
      READ(15,END=2,ERR=2)N1,N2,
     &(XPOINT(II),II=1,185),
     &(YPOINT(II),II=1,185)
      REWIND 15
      GO TO 4
C     NOW EXIT FROM THIS AND DEFINE THE REAL USER'S FUNCTION
    2 WRITE(6,3)
    3 FORMAT(' ERROR WHEN READING, STOP EXIT TAKEN')
      STOP
    4 CONTINUE
C ----USERS CODING ENDS-------------------------------
      IF(IVALUE.GT.NPOINT) GO TO 9999
      ERROR=Y-YPOINT(IVALUE)
      ERROR=ERROR*WEIGHT
      ERRSQ=ERRSQ+ERROR*ERROR
      DO 9998 ICOF=1,14
      IF(DYDCOF(ICOF).EQ.0.) GO TO 9998
      DESQDC(ICOF)=DESQDC(ICOF)+ERROR*DYDCOF(ICOF)*2.0
      DO 9997 JCOF=1,14
 9997 VARMT(JCOF,ICOF)=VARMT(JCOF,ICOF)+WEIGHT*DYDCOF(JCOF)*DYDCOF(ICOF)
 9998 CONTINUE
      PVAR(IVALUE)=WEIGHT
 9999 YVALUE(IVALUE)=Y
      END
```

8.2 Library of SUPERFIT Data Blocks

A facility is available by which a user may save his data (including those statements defining the user's function) in a SUPERFIT library; four special command keywords are recognised for this purpose. The user may, at any stage, enter the keyword SAVE and follow it with up to eight characters as an identifying label. On recognition of the keyword, SUPERFIT collects the user's function statements, point data values, curve x-coordinates, plot labelling information and current values of the coefficients A to T inclusive and writes these on magnetic disc as a labelled entry in its library. During the same run or any subsequent one, the user may recover this information by entering the keyword LOAD (followed by the identifying label). Library entries can be removed by entering the keyword DELETE and the library entry label. The user who wishes to know what library entries have been saved may enter the keywords ?SAV. In all cases, the label can be omitted if the user wishes; in this event, SUPERFIT uses the user's job name as an identifying label. When saving or deleting a library entry, the identi-

fying label must begin with the user's initials but he may load library entries saved by any other user.

9. REFERENCES

Backstrom, R.P. [1976] - $LOGON - an interactive job submission facility for the IBM360 computer. AAEC unpublished report.

Bennett, N.W. & Pollard, J.P. [1967] - SCAN - a free input subroutine for the IBM360. AAEC/TM399.

Powell, M.J.D. [1970] - a Fortran subroutine for unconstrained minimisation requiring first derivatives of the object function. AERE-R6469.

Robinson, G.S. [1968] - FOREX - A Fortran compilation subroutine for the IBM360. AAEC/E190.

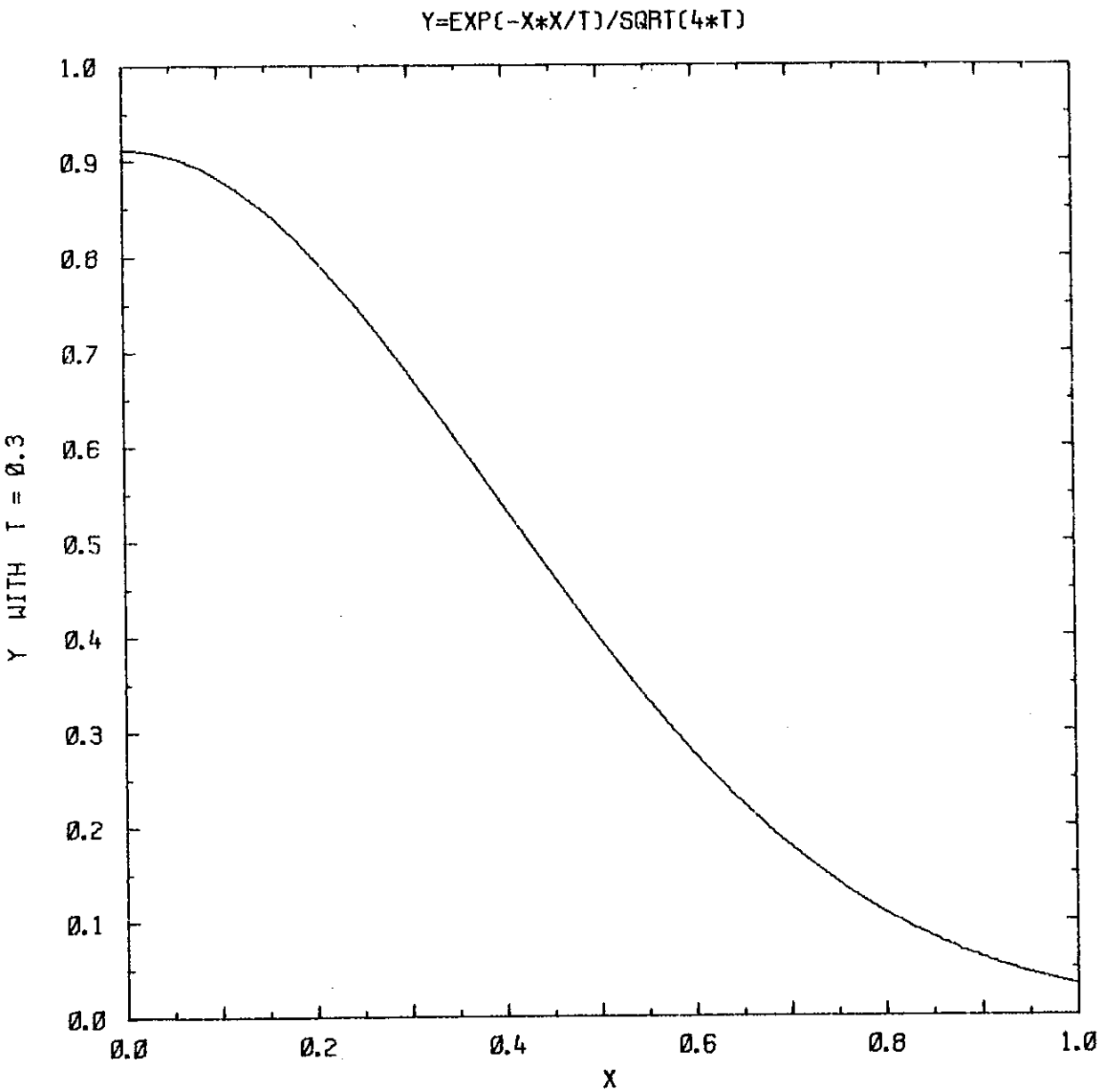

FIGURE 1. PLOT OF A SIMPLE FUNCTION

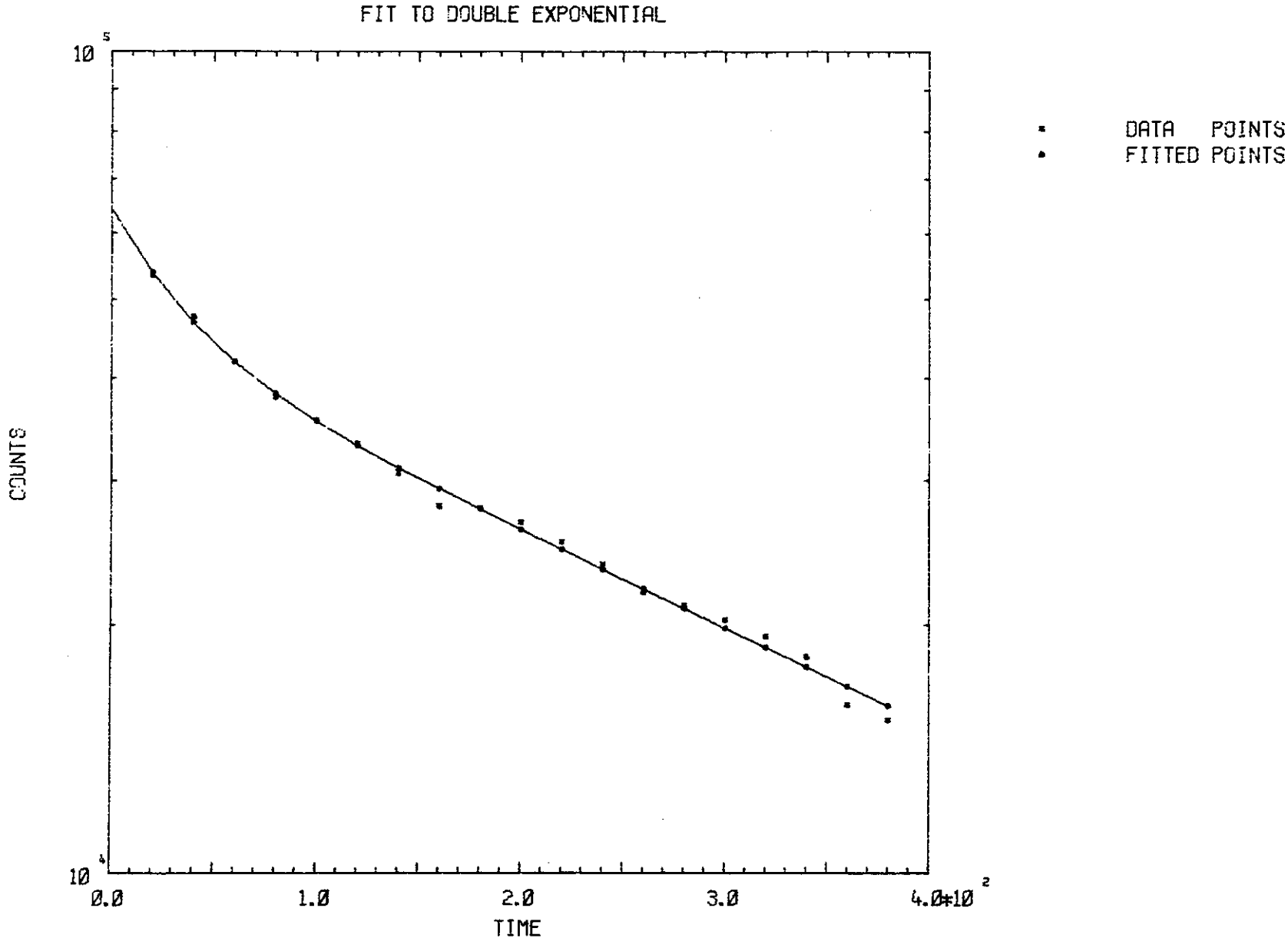

FIGURE 2. PLOT OF A FITTED FUNCTION

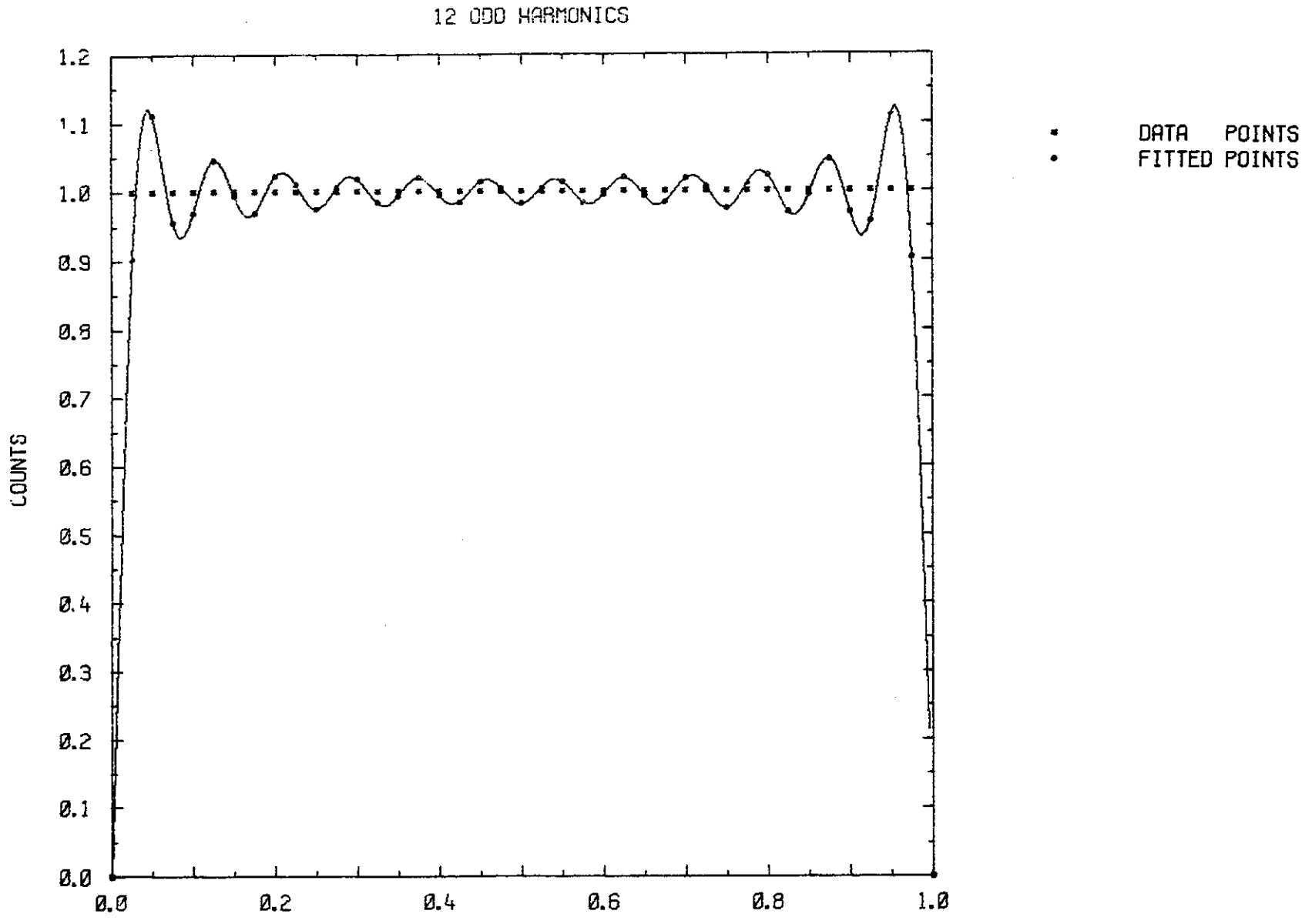

FIGURE 3. PLOT OF A FOURIER FIT TO A SQUARE-WAVE FUNCTION, N=12

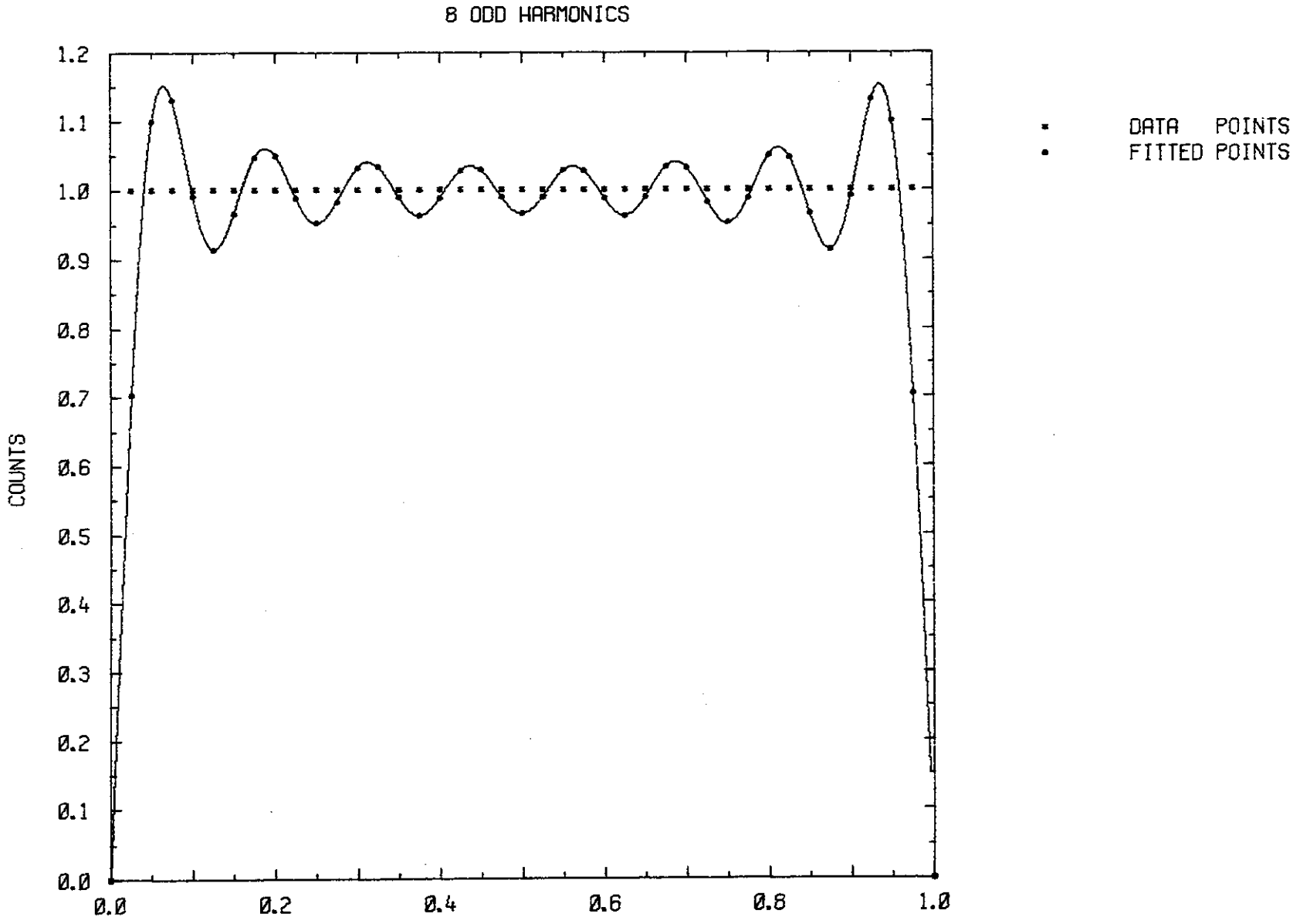

FIGURE 4. PLOT OF A FOURIER FIT TO A SQUARE-WAVE FUNCTION, N=8