SLAC-95
UC-32, Mathematics
and Computers
(MISC)

PEG — AN INTERACTIVE LEAST SQUARES DATA-FITTING PROGRAM

by

Lyle B. Smith

February 1969

Technical Report

Prepared Under

Contract AT(04-3)-515

for the USAEC

San Francisco Operations Office

Printed in USA. Available from CFSTI, National Bureau of Standards,
U.S. Department of Commerce, Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.65.

ABSTRACT

An interactive program for data-fitting by the method of least squares is discussed in some detail. The program was implemented on an IBM 360/75 and 360/91 with an IBM 2250 Mod II display unit as the man-machine interface. First the least squares data-fitting problem is briefly discussed. That is followed by a short description of the direct search minimization method which was chosen to handle the nonlinear least squares problems. Finally, the displays utilized in the interactive data-fitting process are depicted by photographs of the 2250 screen during operation, and an extensive discussion of the use of the program is given.

The program was written in FORTRAN IV for ease of implementation and to achieve some degree of machine independence.

The interaction was designed to make the program very easy to use. It is hoped that very little (if any) reference to any write-up will be necessary when using the program.

TABLE OF CONTENTS

LIST OF FIGURES

I. THE LEAST SQUARES DATA-FITTING PROBLEM

There are various reasons why one might wish to approximate a given set of data points by a curve with a known functional form. Suppose we are given a set of points in a plane $\{(x_i, y_i)\}_{i=1}^{N}$. Some theoretical physical grounds may give us a function, f(x), which is supposed to represent the data. The problem is then to determine values for the coefficients occurring in f(x) such that the data is approximated by f(x) as well as possible in the least squares sense. A second problem is that of obtaining a "good" (in the least squares sense) approximation to the data points by some easily (efficiently) calculable function for use in some later computation. These problems are easily handled by the program to be described.

To handle the case involving a function peculiar to the data at hand, we employ user defined functions which are coded anew for each different function. A description of how this is accomplished is given in Section III. The case where only a good fit is desired can usually be handled by one of the built-in functions — orthogonal polynomials, spline functions, or Fourier approximations. A user function could also be coded and utilized in this latter case if so desired.

The least squares or L^2 norm has a long history of use in the kind of data-fitting problems which this program is designed to handle. There is statistical basis for the choice as well as historical precedent. In 1809 Gauss [1] first used the method of least squares to determine the coefficients of a function from experimentally determined points. A century later in 1900, Markov [2] used a minimum variance approach to coefficient determination which is identical to the least squares approach under certain assumptions on the errors in the data points. For more on the statistics involved, see any standard statistics text, for example, that by Wilks [3].

As an example of the type of problem that might easily be handled by this program, consider the data points tabulated in Table 1 and shown graphically in Fig. 1.1.

Table 1

Data Points

I	X(I)	Y(I)	Weight W(I)
1	.713	2.7	5.0000
2	.856	8.1	1.6670
3	.932	13.2	1.1110
4	.988	21.5	.6667
5	.994	22.4	.6250

Now suppose we wish to approximate these data by a polynomial in x^2 (only even powers) with the added restriction that the constant term be non-negative.[1] In other words, the fitting function is f(x) where

$$f(x) = a_0 + a_1 x^2 + \ldots + a_n x^{2n}, \quad \text{and } a_0 \geq 0 . \tag{I.1}$$

We need a FORTRAN function to evaluate (I.1) for any given $\{a_i\}$ at any value of x. The required form for coding such FORTRAN functions is specified in Section III. When this function has been coded and compiled, its object deck is included in the deck of cards necessary to run the interactive data-fitting program and the fitting can be accomplished at the 2250 console. The degree (number of terms to be used in (I.1)) and initial guesses for the coefficients $\{a_i\}$ will be requested on-line by the program and the least squares fit will then be computed and displayed. For example, Fig. 1.2 shows the fit to the data of Table 1 using n = 3 in (I.1).

[1] Data and function obtained from S. Howry (SLAC Computation Group). The problem arose during a problem involved with cross section estimations.

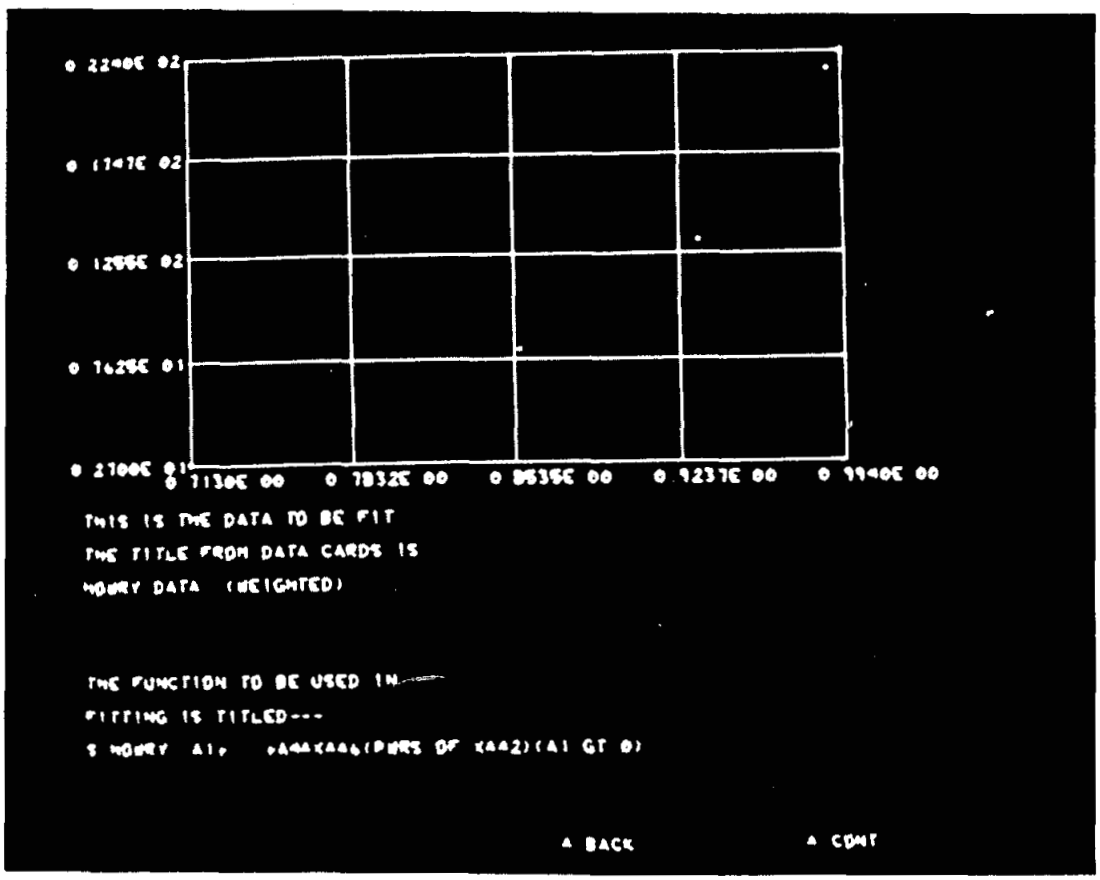

FIG. 1.1--Plot of Table 1 data.

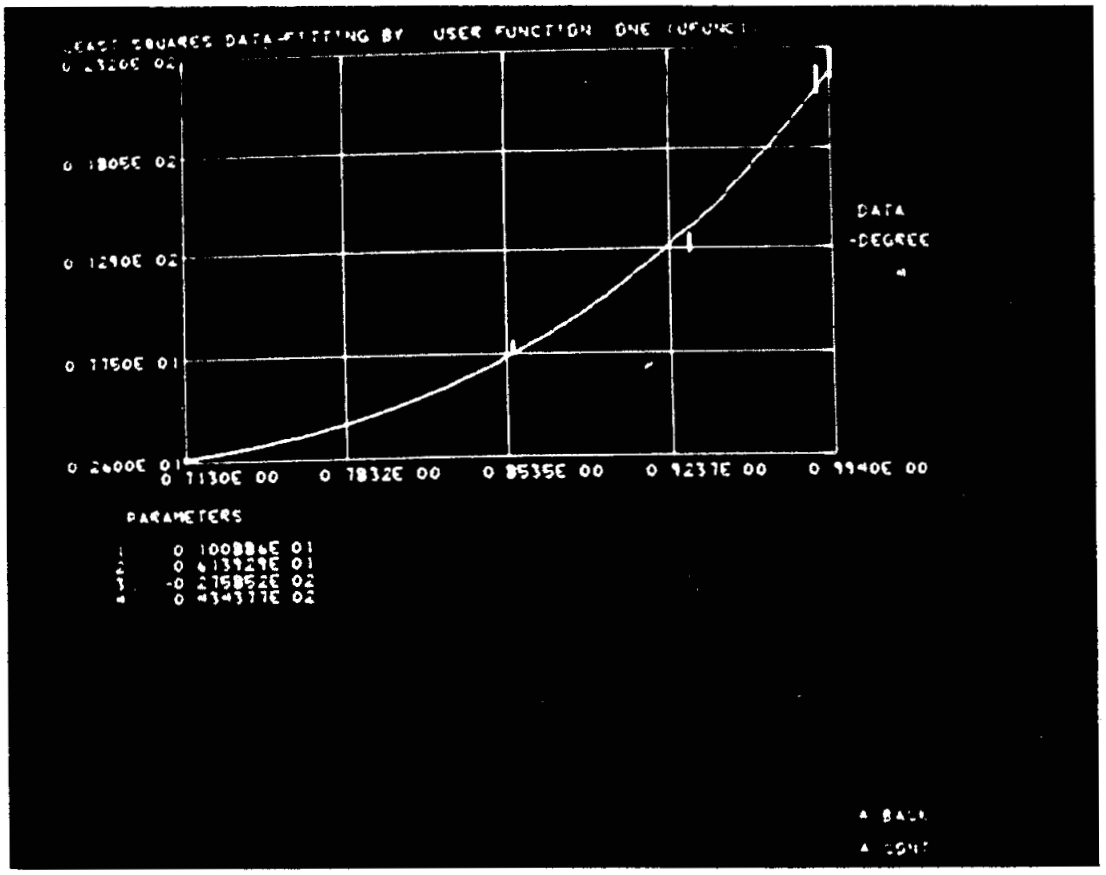

FIG. 1.2--Least squares fit to Table 1 data using even power polynomial.

II. THE LEAST SQUARES DATA-FITTING METHOD

The weighted least squares problem can be defined as follows:

Given: data points $\{(x_i, y_i)\}_{i=1}^{N}$

weights $\{w_i\}_{i=1}^{N}$

a function $f(x, a_1, a_2, \ldots, a_n)$

Determine: values of the coefficients $\{a_i\}_{i=1}^{n}$
which minimize the weighted sum
of squares of residuals, $S(a_1, a_2, \ldots, a_n)$

where

$$S(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{N} w_i^2 \left[f(x_i, a_1, \ldots, a_n) - y_i\right]^2 \quad . \tag{II.1}$$

If no weights are to be considered we simply set $w_i = 1$, for all i, in the above formulation.

Another formulation considers the error at each point, say σ_i. The weight in this case is $w_i = 1/\sigma_i$ so $S(a_1, a_2, \ldots, a_n)$ becomes (assume $\sigma_i > 0$, all i)

$$S(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{N} \left[\frac{f(x_i, a_1, a_2, \ldots, a_n) - y_i}{\sigma_i}\right]^2 \quad . \tag{II.2}$$

The minimization of $S(a_1, a_2, \ldots, a_n)$ is handled in different ways depending on the particular function being used as the approximating function. The built-in functions in this program include:

1. orthogonal polynomials
2. spline functions
3. Fourier approximations.

Arbitrary functions are allowed in the form of user defined functions. For the orthogonal polynomials and Fourier approximations the minimization of the sum of squares in (II. 1) is accomplished by essentially solving a set of simultaneous linear equations since the problem is linear in these cases.

A spline function is a piecewise continuous polynomial of degree m with the added property that all derivatives up to and including the $m-1^{st}$ are continuous at the "joints." The "joints" are those points where the polynomial pieces are joined together. If the joints are fixed, the minimization of (II. 1) is again a linear problem which can be solved directly by solving a set of simultaneous linear equations.

For spline functions with variable joints (the joints and the coefficients are adjusted in order to minimize the sum of squares) as well as user defined functions the problem becomes non-linear and an iterative method is needed to minimize (II. 1). There are various methods such as gradient methods, steepest descent, Newton-Raphson and others which could be used to accomplish the minimization. However, because of its simplicity of use and because of much successful usage of the method in the past, the author has chosen to use the "direct search" method to minimize (II. 1) for the non-linear problems. The method is described in an article by Hooke and Jeeves in the Journal of the ACM (1961), see [4].

The direct search minimization method can be simply described as a method which "feels around" for a smaller function value by adjusting each coefficient in turn. When a smaller value is found the point of search moves there. When no smaller value can be found by varying the coefficients by the current stepsize, the stepsize is reduced and the search continues. When the stepsize is reduced to some specified stopping criterion and no more reduction in the function value can be achieved, convergence is claimed.

A. Orthogonal Polynomials

The polynomial fitting provided by PEG is in the form of orthogonal polynomials as described by Forsythe [5]. These orthogonal polynomials are equivalent to the use of monomials, x^i, as approximating functions but their computational properties are much better.

The calculated polynomial approximation is given in terms of the orthogonal polynomials which are generated by a recurrence formula:

$$p_0(x) = 1$$

$$p_1(x) = xp_0(x) - \alpha_1 p_0(x)$$

$$p_2(x) = xp_1(x) - \alpha_2 p_1(x) - \beta_1 p_0(x)$$

$$\vdots$$

$$p_{i+1}(x) = xp_i(x) - \alpha_{i+1} p_i(x) - \beta_i p_{i-1}(x) \ .$$

The least squares polynomial of degree n, $p_n^*(x)$, is given as

$$p_n^*(x) = \sum_{i=0}^{n} S_i \, p_i(x) \quad .$$

The output from PEG includes a table of values for α_i, β_i, and S_i, $i = 0, 1, \ldots, n$. Using these values and the above equations, the approximating function, $p_n^*(x)$, can be evaluated for any value of the independent variable with the following consideration.

PEG scales the data to lie in the interval [-2, 2] before calculating the orthogonal polynomial least squares fit. Therefore a transformation on points in the original coordinates is necessary before using the given α_i, β_i, and S_i to evaluate the approximating function. Assuming the largest and smallest

abscissa values in the data are x_N and x_1 respectively, the transformation

$$x' = 4\left(\frac{x - x_1}{x_N - x_1}\right) - 2$$

will transform x into x' to be used with the coefficients output by PEG.

B. Spline Functions

A spline function of degree m is a piecewise polynomial with J joints, $\hat{x}_1 < \hat{x}_2 < \ldots \hat{x}_J$. The results of calculating a least squares spline by PEG are given in terms of the polynomial coefficients in each interval. There will be J+1 sets of polynomial coefficients given by PEG.

For $x < \hat{x}_1$ use the 1^{st} polynomial

For $\hat{x}_1 \leq x < \hat{x}_2$ use the 2^{nd} polynomial

.
.
.

For $\hat{x}_J < x$ use the $J + 1^{st}$ polynomial

A transformation of the data is made by PEG before calculating the spline function fit. PEG scales the independent variable values to be in the interval [-1, 1] so a reverse transformation is necessary when using the computed fit in further calculations. Assuming that x_1 is the smallest data value and that x_n is the largest data value the transformation necessary to evaluate the spline function a point x is:

$$x' = 2\left(\frac{x - x_1}{x_n - x_1}\right) - 1 \quad .$$

Evaluation of the appropriate polynomial segment at the point x' will give the ordinate value corresponding to the point x in the original coordinate system.

C. Fourier Approximation

The function used to calculate the Fourier approximation is

$$f_m(x) = \frac{a_0}{2} + \sum_{j=1}^{m} (a_j \cos jx + bj \sin jx) \quad .$$

If the data is given as N equally spaced points, an algorithm due to Goertzel and discussed by Ralston [6] is used to calculate the least squares fit. This algorithm assumes that the points are equally spaced on the interval $\left[0, \frac{2\pi(N-1)}{N}\right]$ therefore a transformation is necessary when evaluating the function for values of the independent variable in the original coordinate system. To evaluate the function at a point x, compute

$$x' = \frac{2\pi(N-1)}{N} \left(\frac{x - x_1}{x_N - x_1}\right)$$

and evaluate the function at x' using the coefficients given by PEG.

In the case of unequally spaced points or if weighting was used, the coefficients printed by PEG correspond to the original coordinate system and no transformation is needed when evaluating the function.

D. User Defined Functions

In the case of least squares fitting by a user defined function, $f(x, a_1, a_2, \ldots, a_m)$, the values of the parameters, $a_1, \ldots, a_m$, determined by PEG correspond to the function in the original coordinate system of the data. There is no automatic transformation or scaling of the data by PEG as there is in the three previous cases. However, if the data is transformed by a user selected option while on-line, the coefficients determined by PEG will correspond to the transformed coordinate system.

III. AN INTERACTIVE DATA-FITTING PROGRAM

A. The Interactive Program

This program has been implemented on an IBM 360/75 and later an IBM 360/91 with an IBM 2250 display unit (CRT), see Fig. 3.0. An overall view of the capabilities of the program or system is depicted by the flowchart in Fig. 3.1. Each block of the flowchart is numbered according to the numbering used for the paragraph describing the function of that block.

The design of this program is modular in the sense that additional capabilities can be easily added. For example the list of user functions could be expanded or the list of built-in functions could be augmented. Also the inclusion of more options as to what kind of minimization routine a user wants to use is easily accomplished. Additional display modes for examination of the results could also be added.

1. Introduction

The first display to appear on the CRT is a paragraph of introductory remarks. These remarks briefly summarize the flow of the program, give the built-in limits on the number of data points and number of parameters (or degree), and give some other general information. The display is shown in Fig. 3.2.

The display shown in Fig. 3.2 as well as several displays to be described later have a "*BRANCH OUT" displayed which, when selected by the lightpen, will transfer control to the branching display described in paragraph III.A.11. One of the options given by the branching display is *TUTORIAL DISPLAYS. If this tutorial option is selected by lightpen at that point, a choice of various tutorial displays will be presented for lightpen selection. The tutorial displays are described in detail in paragraph III.A.13. The *BRANCH OUT option gives an "escape-to-get-help" capability to the program.

FIG. 3.0--The IBM 2250 display console.

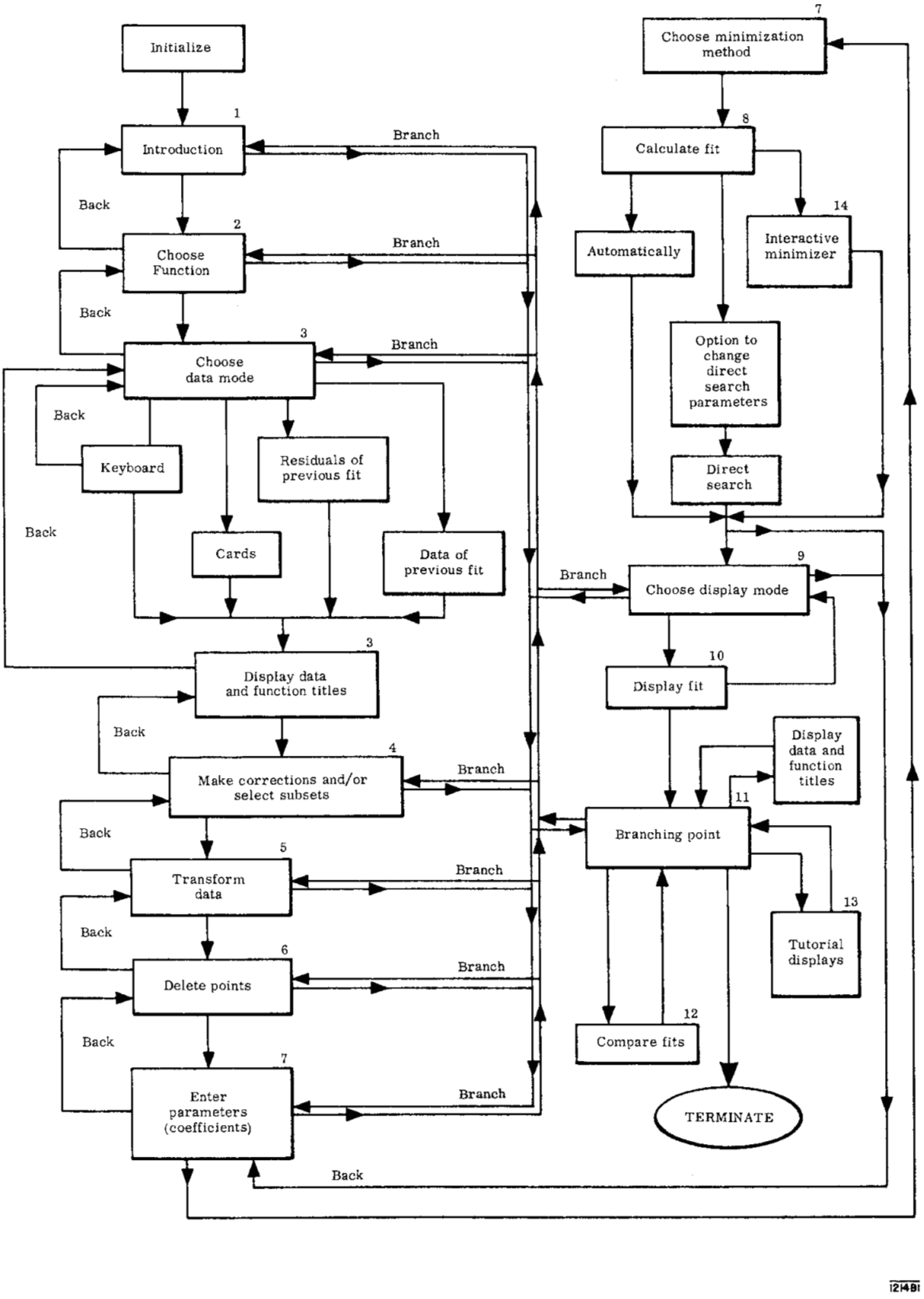

FIG. 3.1--Flow Chart — Interactive data-fitting program.

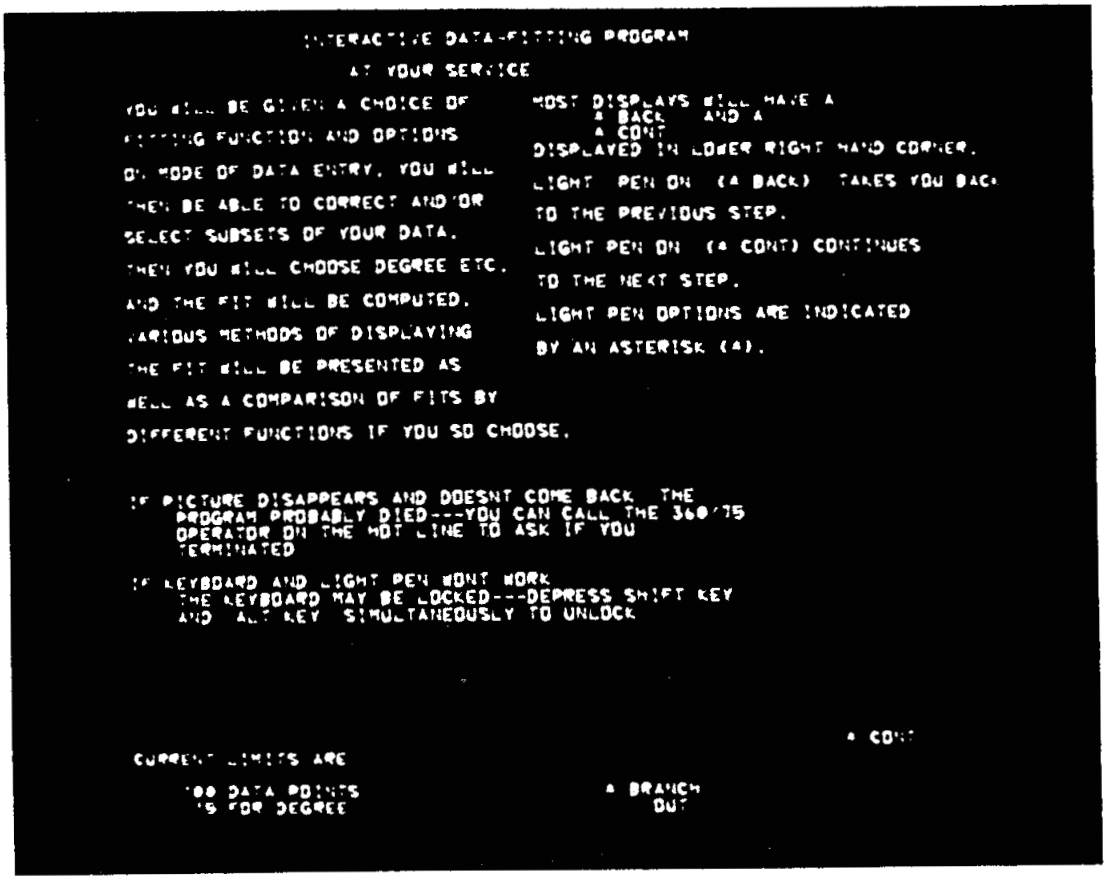

FIG. 3.2--Introductory display.

2. Choosing the Function

The first step in the data-fitting process is next to appear on the CRT. This is the display which allows user selection of a function with which to approximate the data. The choices as shown in Fig. 3.3 include several well known functions which are built into the system and a list of user functions which are to be supplied by the user. The construction and use of user defined functions are described in detail in Section III. B. A user function may be any function $f(x, a_1, \ldots, a_n)$ where x is the independent variable and $\{a_i\}_{i=1}^{n}$ are n parameters (n less than or equal to the limit specified in Fig. 3.2) to be determined by the least squares fitting process. The parameters $\{a_i\}_{i=1}^{n}$ may occur linearly or non-linearly.

This display also indirectly allows a user to display a mathematical description of the function he has chosen to use for data-fitting. Thus, for example, if he has forgotten exactly what a spline function is he can quickly refresh his memory by looking at the description stored in the computer. These descriptions are available through the *BRANCH OUT lightpen option.

3. Choosing the Data Mode

After choosing the desired function the next step is entering the data to be approximated by that function. There are two types of data sources in this system. The first is external data which consists of data from cards (submitted with the job) or data from some device such as a tape or a disk. Data entered through the keyboard is also considered to be of external type. The second type of data source is internal. The internal data is either the residuals of the immediately preceding data-fitting problem or the original data of the immediately preceding problem.

The display which allows the selection of data mode is shown in Fig. 3.4.

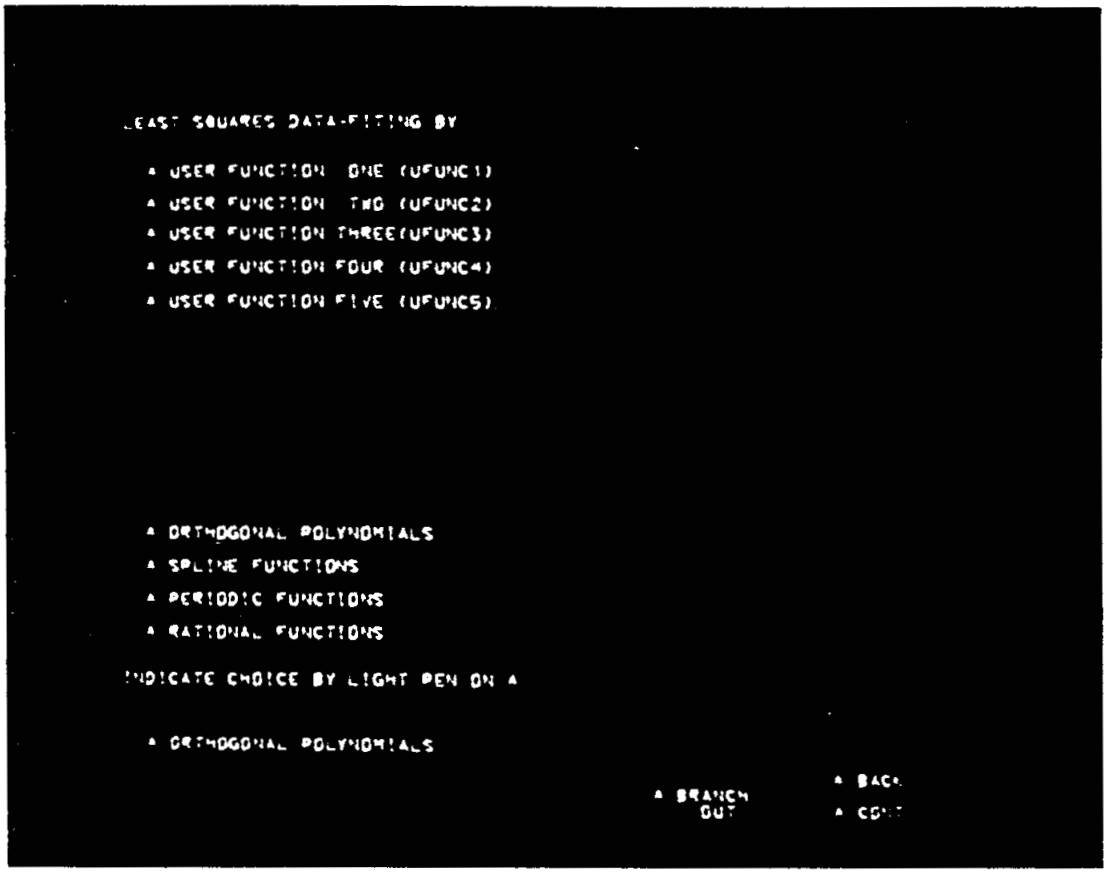

FIG. 3.3--Choosing the function.

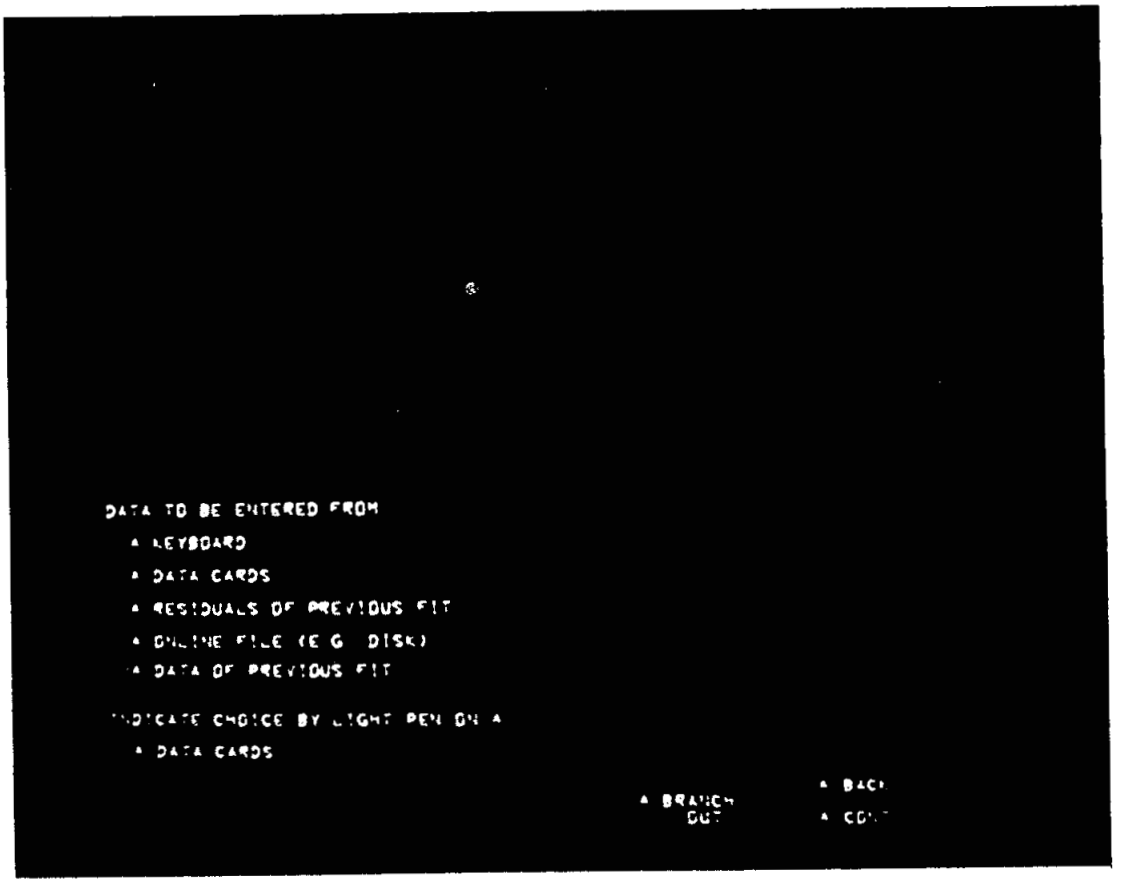

FIG. 3.4--Choosing the data mode.

Keyboard entry of data is accomplished through the display shown in Fig. 3.5. First one must specify whether or not there are weights associated with the data points. The weights, if given, are assumed to be the inverse of the error associated with the dependent variable, Y, at each point. After choosing "yes" or "no" on the question of entering weights, the data is entered from the keyboard, point by point.

Data card entry is accomplished by punching the data on cards in a specified format and submitting the data cards with the card deck required to activate the data-fitting program. Several sets of data may be put sequentially in the input deck. Each data set includes space for a title on the first card which is reproduced on the CRT at run time for identification purposes. The input from cards is required to be as follows:

1^{st} card ... has, in Fortran format 2I5, the number of points and a 0 or 1 in column 10 to indicate absence or presence of weights. Starting in column 21 there are 40 columns available for a title.

2^{nd} card ... and following have the data points, one per card, abscissa first, ordinate second and weight (if so desired) third. The Fortran format is 3E10.5 so data punched with either an F format or an E10.5 format will be read.

. (the above may be repeated many times for many data sets)

last card ... should have FINISH punched starting in column 1.

Selecting the residuals of previous fit allows a combination fit to be computed on a single set of data. In other words, the original data could be approximated by one function, say an orthogonal polynomial of degree 5, and the residuals of

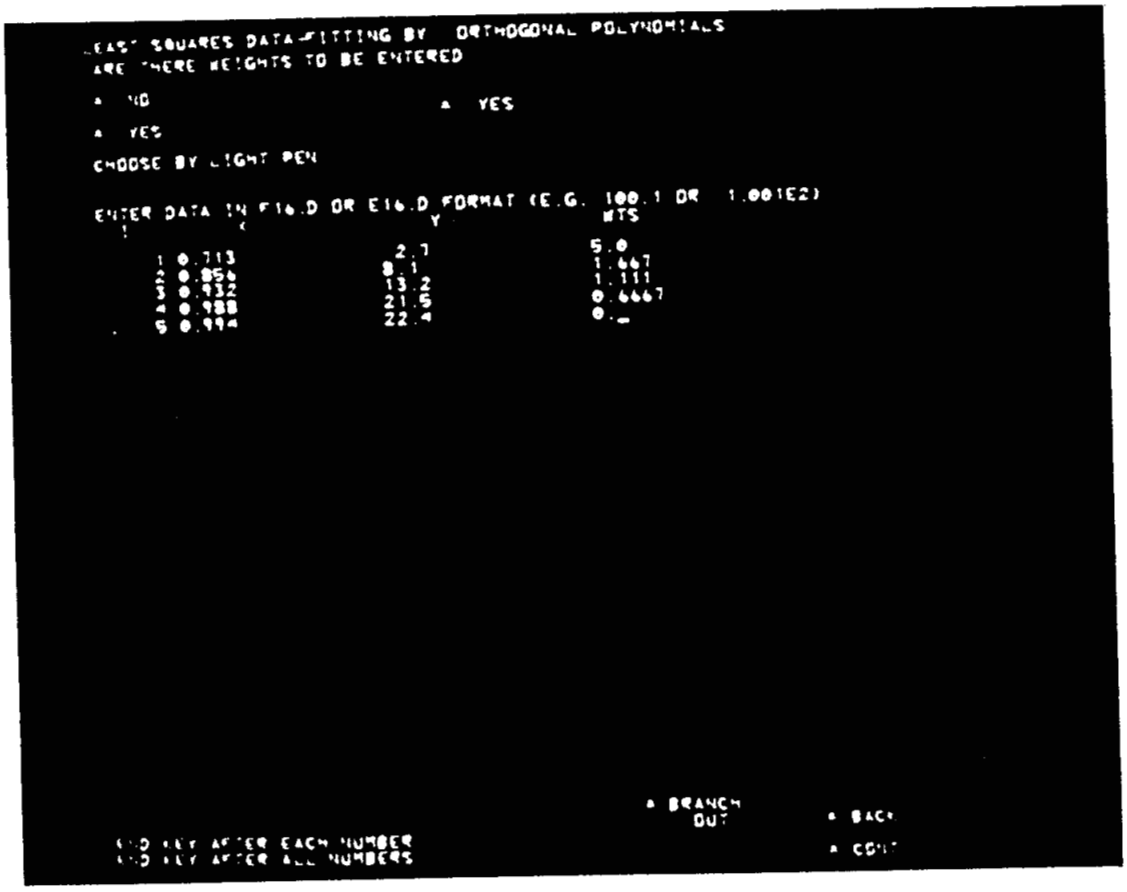

FIG. 3.5--Keyboard entry of data.

that polynomial fit could then be fit by a Fourier approximation. The resulting combined fit would give a closer (in the least squares sense) approximation to the data than either the fifth degree polynomial or the Fourier approximation alone. This capability would also be useful to an investigator who wishes to test if the residuals are time dependent.

Data may also be selected from an on-line file (e.g., a disk or tape). The user would enter from the keyboard information to allow the program to read data from a particular device. (This option has not yet been implemented.)

The data of previous fit choice of data mode allows comparative least squares fits to be computed. That is, the same data can be approximated over and over by different functions. As described in paragraph III.A.12 this program has a built-in comparison mechanism for the three standard functions, orthogonal polynomials, spline functions, and Fourier approximations.

Following entry of the data by any of the aforementioned means, a display appears which presents a plot of the data with a title for the data as well as a title for the chosen fitting function. Data titles are obtained from columns 21 through 60 of the first data card for each data set. Function titles, for user defined functions, are obtained from the user function which, when called with zero for the number of parameters, is expected to store a title for display purposes. The mechanics of accomplishing this are discussed in Section III.B. Figure 3.6 shows an example of this display whose purpose is to identify data and user functions during a run on the machine.

The choice of origin and scale for the plot shown in Fig. 3.6 and in all other plots displayed by the PEG system is made so as to utilize the given area as fully as possible. The maximum and minimum values for each axis are chosen to correspond to the maximum and minimum values of the values to be plotted.

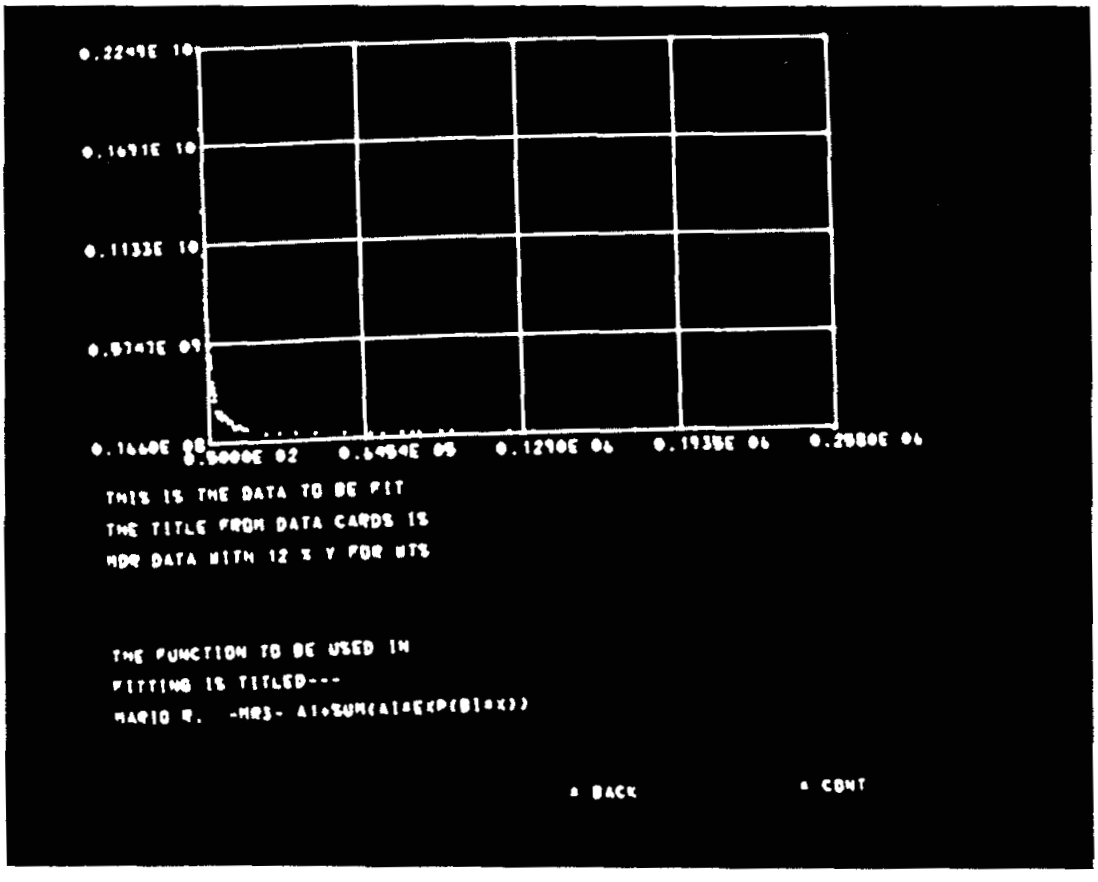

FIG. 3.6--Data and function titles display.

This causes the numbers displayed along each axis not to have "reasonable" scales. Reasonable scales can be obtained for plotting purposes by using an algorithm described by Dixon and Kronmal [1965]. Use of this algorithm will change the origin and scales so that the division points on the graph will be simple numbers (a simple number, s, is defined to have the form $s = p_i \, 10^k$, where p_i is taken from a set such as $\{1, 2, 5\}$).

4. Correction and/or Subset Selection

Once the data has been entered, either internally or from an external source, the display shown in Fig. 3.7 allows correction (alteration) of individual points or selection of a subset of the data.

Correction of a point is accomplished by entering the index of the point, the new (or unchanged) value of the independent variable, X, the new (or unchanged) value of the dependent variable, Y, and if weights are associated with the data, the new (or unchanged) value of the weight for that point. After entering all 3 (or 4 if weights) quantities defining the altered value of the point, a final "end" key replaces the point in the computer and on the CRT plotted graph of the data with the new value.

As shown in Fig. 3.7 this display shows a table of the data points as held in the computer memory. The limited size of the display area limits the table to 15 entries. To circumvent this problem the "FORWARD DATA" and "REVERSE DATA" lightpen options are presented. FORWARD DATA will cause the next (higher index) 15 points to be displayed.

Choosing "SELECT A SUBSET" with the lightpen allows the choice of a subset of the data which is concurrently graphed and listed tabularly. As many as 5 pairs of indices may be given to select a subset. Each pair (n_i, n_j), with $1 \leq n_i \leq n_j \leq N$ where N is the total number of points, specifies that data points

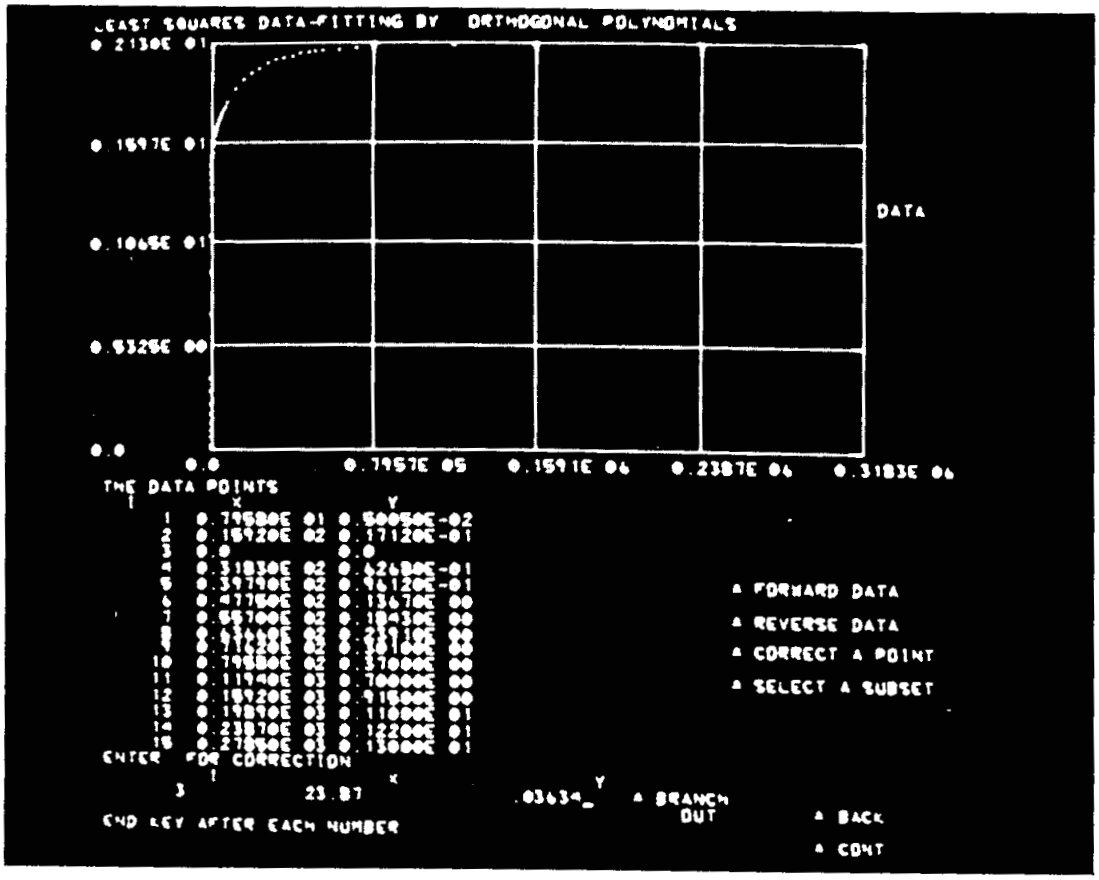

FIG. 3.7--Display for point correction and/or subset selection.

$\{(x_i, y_i)\}_{i=n_i}^{n_j}$ shall be included in the data set to be fit. If several (≤ 5) such pairs are given, their intersection must be empty. A final "end" key will cause the new data set as specified to be graphed and tabulated as the then current data set.

5. Data Transformation

It is sometimes desirable to transform the original data in some manner. For example, a semi-log or log-log plot of a curve can be more meaningful and/or easier to handle computationally than the original data. An example of such a case is shown in Fig. 3.8 which graphs data representing the magnetization curve of annealed ingot iron.

Several transformations are available for user selection by lightpen. The independent and dependent variables are acted upon separately. After choosing a pair of transformations the transformed data can be displayed for examination before proceeding with the fitting process. If a user wants to change his mind after seeing the results of his first choice of transformations, he may back up to the original data and try another transformation. The various options are shown in Fig. 3.8 and Fig. 3.9. It is easy to add additional transformations to the program in case a desired capability is not included. Figure 3.9 shows the results of transforming the data shown in Fig. 3.8 by choosing Y replaced by Y and X replaced by $LOG_{10}(X)$. The transformed curve is easier to handle computationally and could be approximated by orthogonal polynomials, for example. The resulting least squares fit could then be made to apply to the original data by a change of variable.

6. Data Point Deletion

Following the display which allows transformation of the data, another display appears which allows individual point deletion. The lightpen is used to

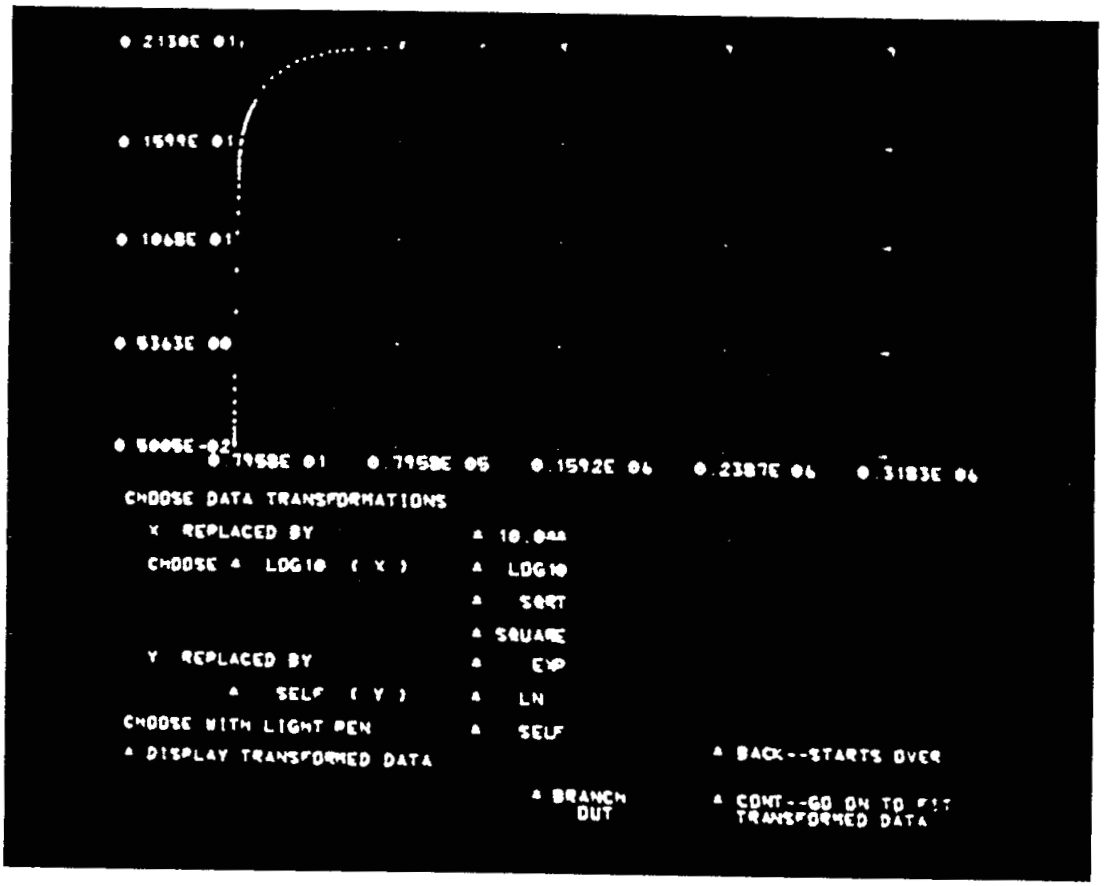

FIG. 3.8--A magnetization curve.

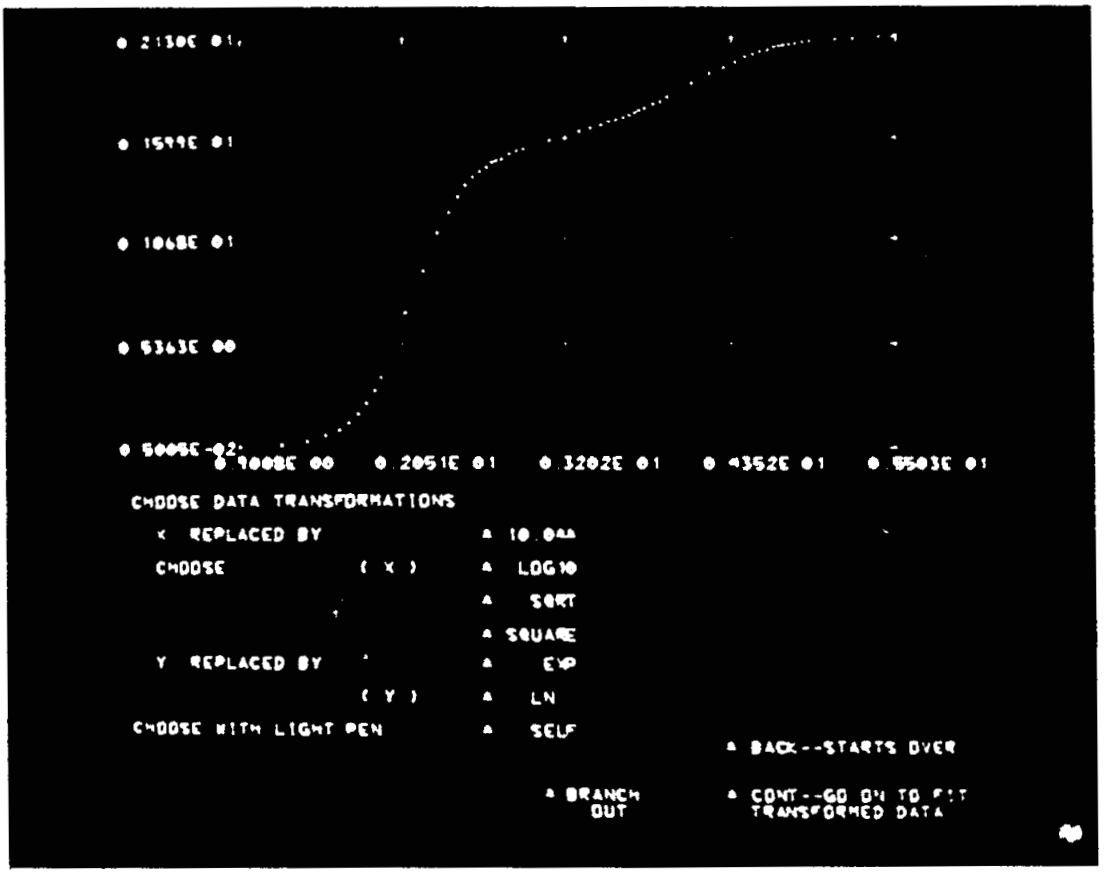

FIG. 3.9--Transformed magnetization curve.

select points for deletion. Pointing the lightpen at one of the tabulated points causes an arrow to mark the selected point in the table and a small rectangle to enclose the spot representing that point on the accompanying plot. Once the desired point is selected in this manner, the option "DELETE SELECTED POINT" is selected by lightpen and the point is eliminated from the set of data points under consideration. The other options available by lightpen selection are shown in Fig. 3.10.

7. Entering Parameters

After the data has been entered, and then modified as desired, the next step in the fitting process is to choose the degree and in the case of a non-linear function, the initial values for the parameters. The display which allows these choices to be made takes three different forms depending on the type of function being used.

Figure 3.11 shows the form of the display for orthogonal polynomials and for Fourier approximations. In these cases the only parameter to be specified is the degree. For polynomials, the degree entered is the highest degree polynomial to be used in the fit. For the Fourier approximation the degree is the number of sine (and cosine) terms to be used in the approximating function. That is, if the degree entered is d, the approximating function, f(x), is given by

$$f(x) = \frac{a_0}{2} + \sum_{i=1}^{d} \left\{ a_i \, \mathrm{COS}(ix) + b_j \, \mathrm{SIN}(ix) \right\} .$$

All three forms of this display for entering parameters include the graph of the data points and a table showing the numerical values of the data. The table has the FORWARD and REVERSE lightpen options as discussed previously.

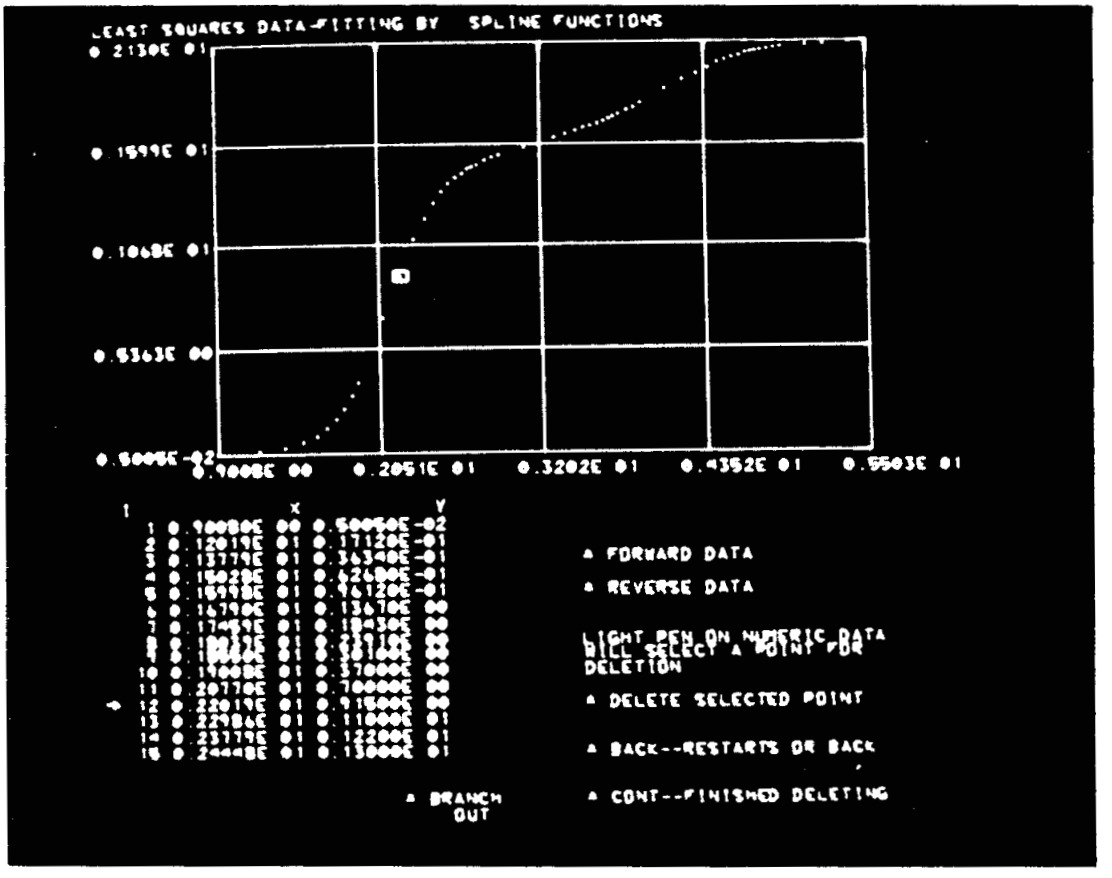

FIG. 3.10--Point deletion.

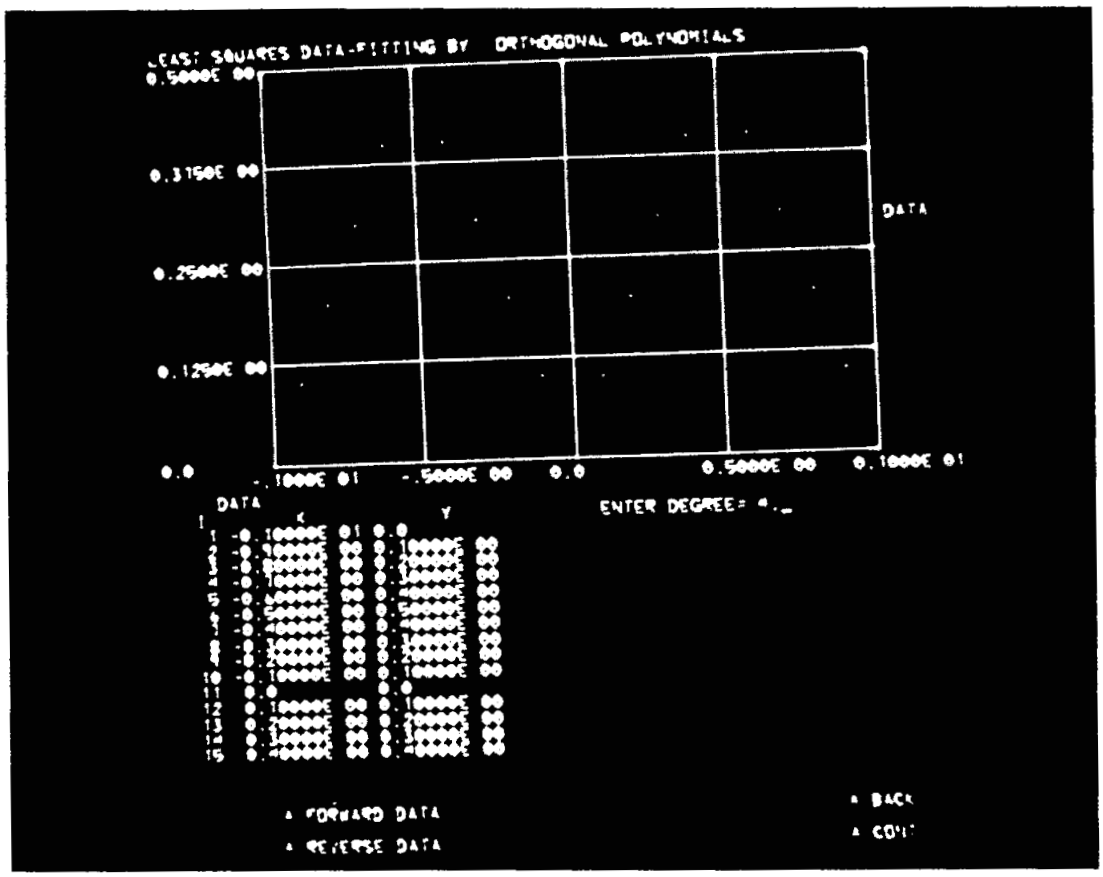

FIG. 3.11--Display for degree entry.

Figure 3.12 illustrates the form taken by the display in the case of spline functions. For splines the needed information includes the degree, the number of joints, and values for the joints. Also to be made is the decision as to whether the joint values entered are to be held fixed while computing the least squares fit or whether the joints are to be varied by the program in determining the least squares fit. If the joints are held fixed, a chance to enter new values through the keyboard is given as described in paragraph III.A.8.

When a user function has been selected as the least squares approximating function, the third form of this display, Fig. 3.13, comes into play. For these functions the items to be entered are the number of parameters, whether or not automatic parameter adjustment is desired, and initial guesses for the parameters. If automatic parameter adjustment is selected the parameters will be varied so as to minimize the sum of squares of the residuals. If not, the fit will be computed for the initial guess and then that fit will be displayed along with an option to enter another guess for the parameter values. Again, this option is discussed in more detail in paragraph III.A.8.

After parameter entry for a user function is completed, a display appears which allows selection of a method for minimization of the sum of squares. The available choices are the Direct Search method and the interactive minimizer (see paragraph III.A.14). More methods could be programmed and added to this list. Figure 3.14 shows how this selection is made.

8. Calculating the Fit

Depending on which type of function has been selected the calculation of the least squares approximation takes different forms. These may be classified in three groups.

a. non-iterative (for linear problems)

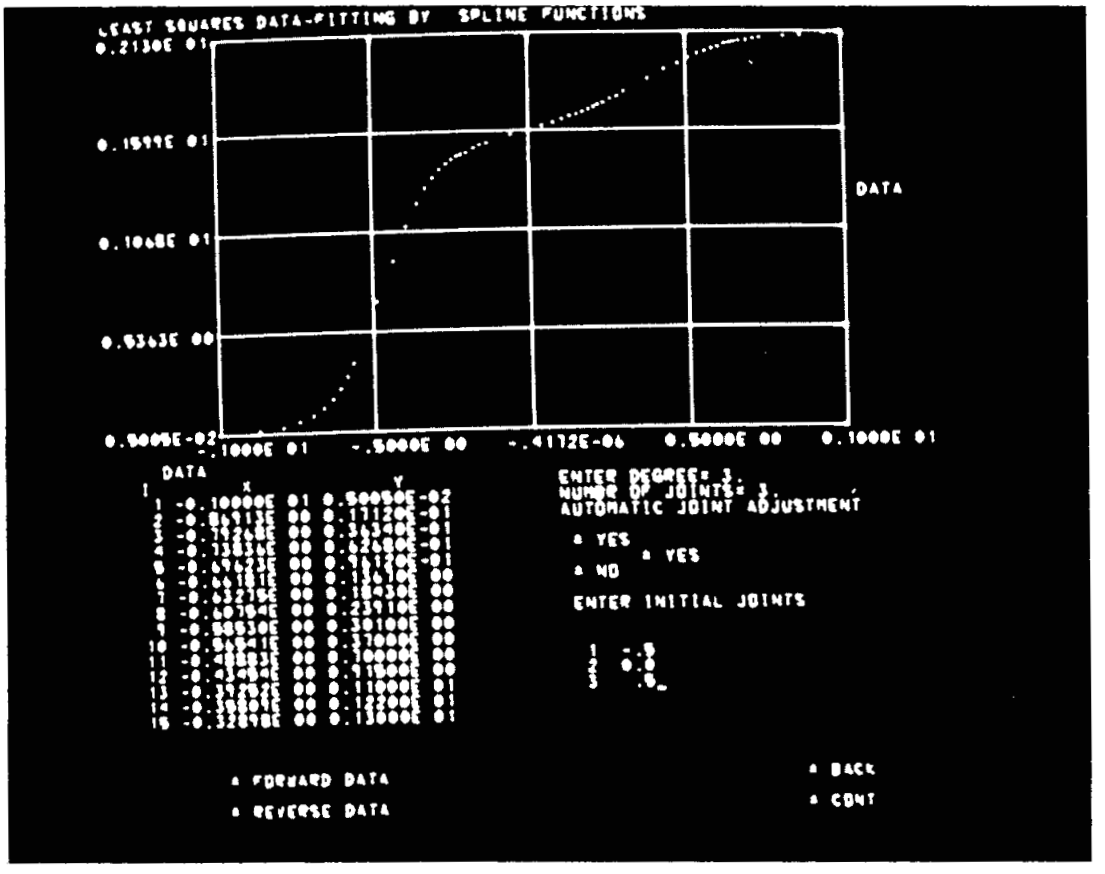

FIG. 3.12--Entering spline parameters.

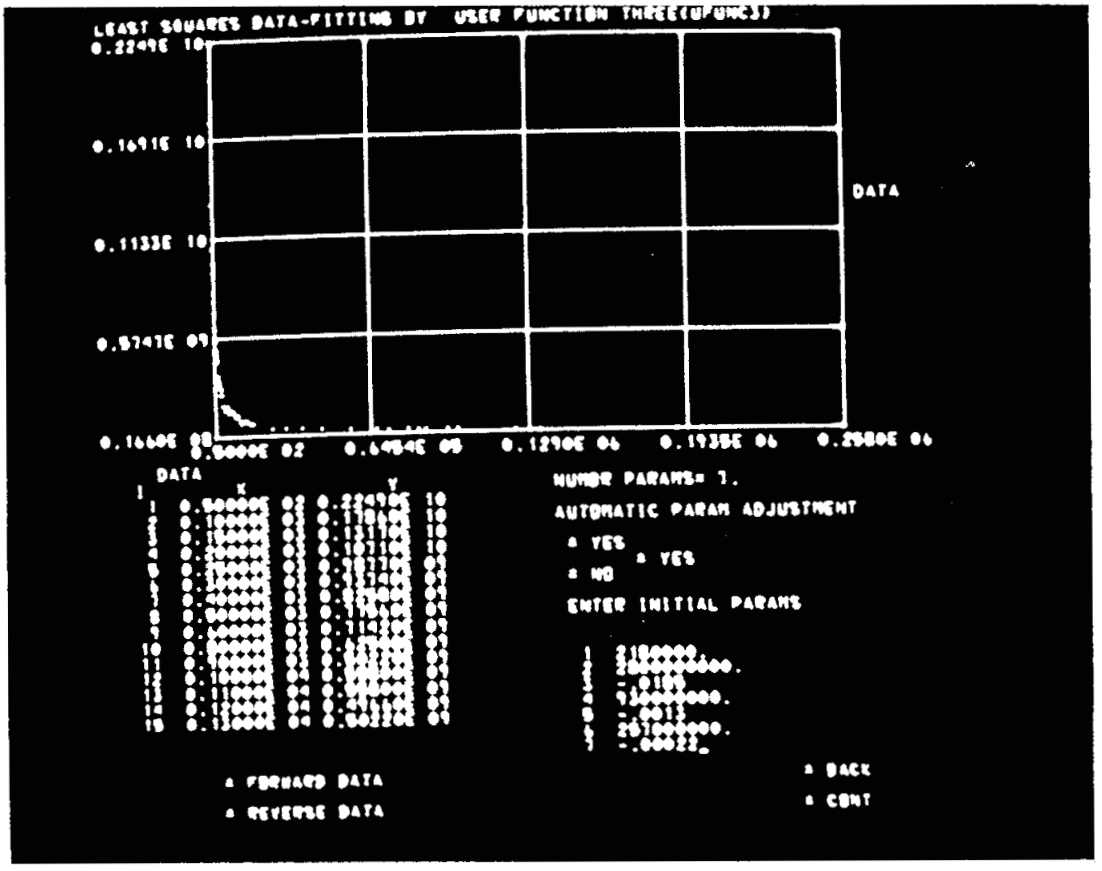

FIG. 3.13--Entering user function parameters.

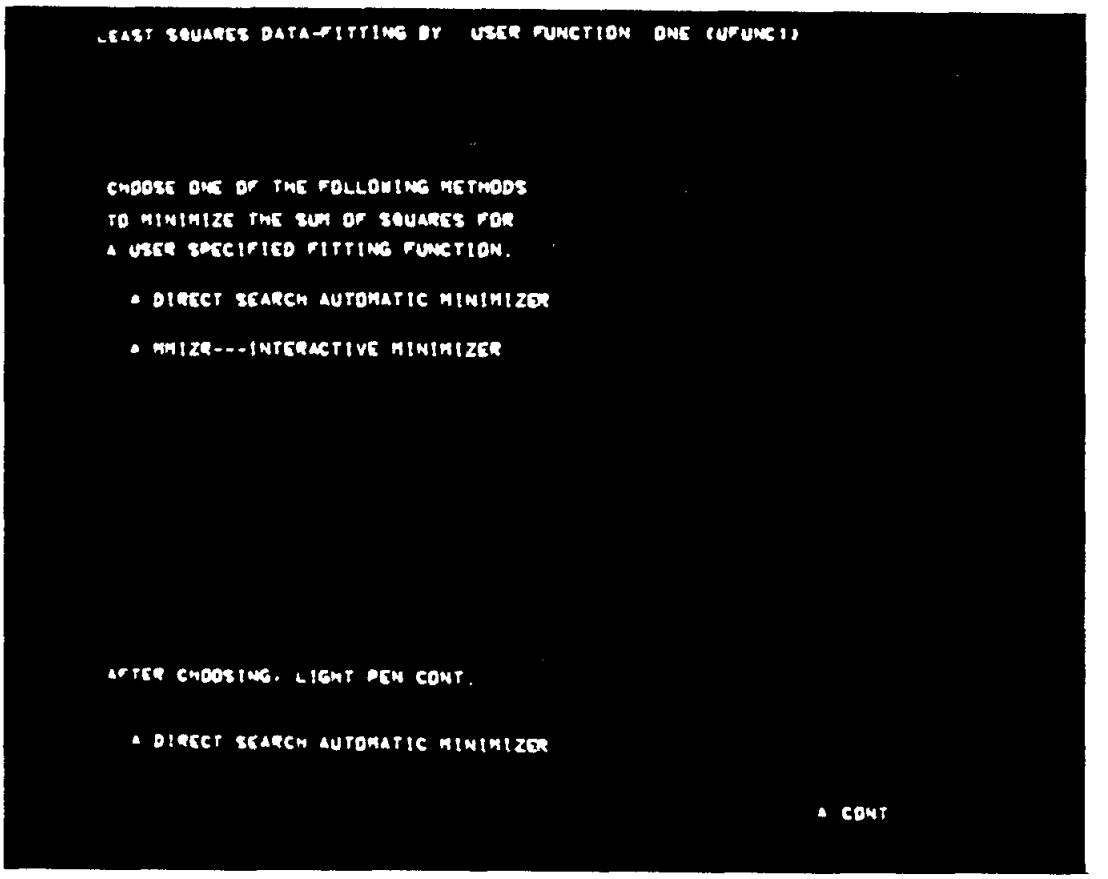

FIG. 3.14--Choosing a minimization method.

b. by Direct Search

c. by interactive minimizer

For orthogonal polynomials and Fourier approximations the fit is computed by a non-iterative algorithm. During the calculation a message, "FIT IS BEING COMPUTED," is displayed on the CRT. Upon completion of the least squares computation the next display, which allows the selection of a display mode (see paragraph III.A.9), appears on the screen.

Spline function fitting with fixed joints is accomplished directly (non-iteratively) while the "FIT IS BEING COMPUTED" message is displayed on the CRT. Following the calculation, a display showing the results of that fit and allowing entry of new guesses for the joints appears on the screen. This process can be repeated indefinitely allowing a user to locate values for the joints which give a "good" fit. The computer program keeps track of which of all previously tried sets of joints gives the least sum of squares and displays this information to the user as an aid to the selection of a new set of joints. Figure 3.15 shows this display. The positions of the joints are indicated by vertical lines of varying lengths. The shortest lines indicate previous values for the joints, the next longer lines show the current joints, and if new values are keyed in they will be indicated by the longest lines.

Another feature of this display is the capability to choose by lightpen whether to plot the current fit or the best of all previous fits on the displayed graph.

The direct search minimization routine has several parameters which control its convergence decisions and its stepping behavior. There are preset values for these parameters which are felt to be generally applicable; however, the option to change these parameters is presented to a user just before the program begins use of the direct search routine. The display which allows

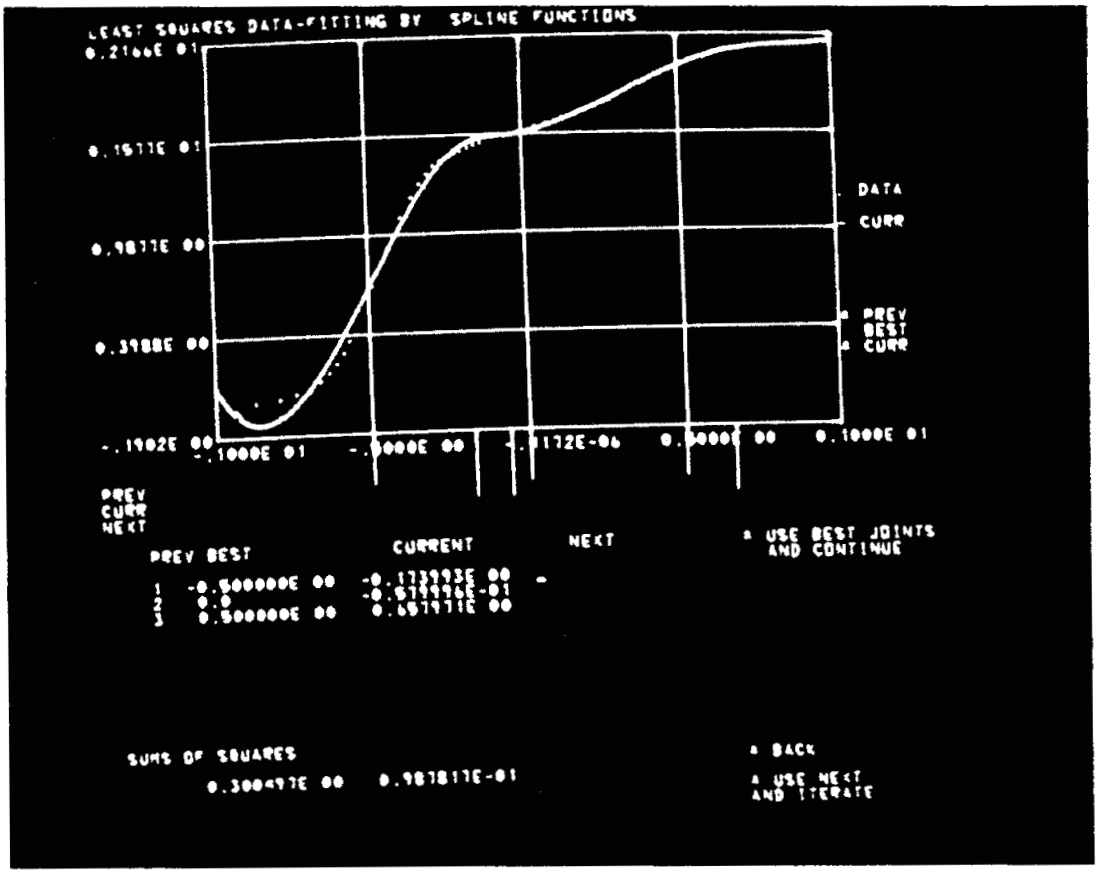

FIG. 3.15--Display of spline fit and selection of new joints.

user specification of the parameters is shown in Fig. 3.16. As shown, the display includes sufficient explanatory text so that someone only slightly familiar with the direct search routine can make reasonable guesses as to whether or not he wants to change any of the parameters.

Spline function fitting with variable joints uses the direct search minimization routine to adjust the joints. The sum of squares of the residuals is treated as a function of the joints and is minimized with respect to the joints. During each iteraction of the minimization process a plot of the data with the current fit superimposed is displayed on the CRT. The program stops after each iteration to allow user inspection of the fit at that point and waits for an "END" key signal or lightpen signal before continuing with the next iteration. A typical display during this iterative process for spline fitting is shown in Fig. 3.17. A lightpen option allows early termination of the iterative process.

When the Direct Search minimization is completed the program displays a picture exemplified by Fig. 3.15. In this case "previous" corresponds to the starting values, otherwise everything corresponds to the discussion of Fig. 3.15.

The use of the Direct Search method for a user function fit is similar to the spline fit with variable joints. During each iteration a display corresponding to that shown in Fig. 3.17 is put on the CRT and requires an "END" key or lightpen signal to continue with the next iteration. An option allows early termination of the iteration. When the Direct Search iterative process terminates, a display similar to the one given by Fig. 3.15 appears on the CRT. At this point we may optionally enter new values of the parameters as a new guess and repeat the fit calculation cycle.

Differing from the spline however, when the step corresponding to Fig. 3.15 is finished, the interactive minimizer is automatically brought into play for user

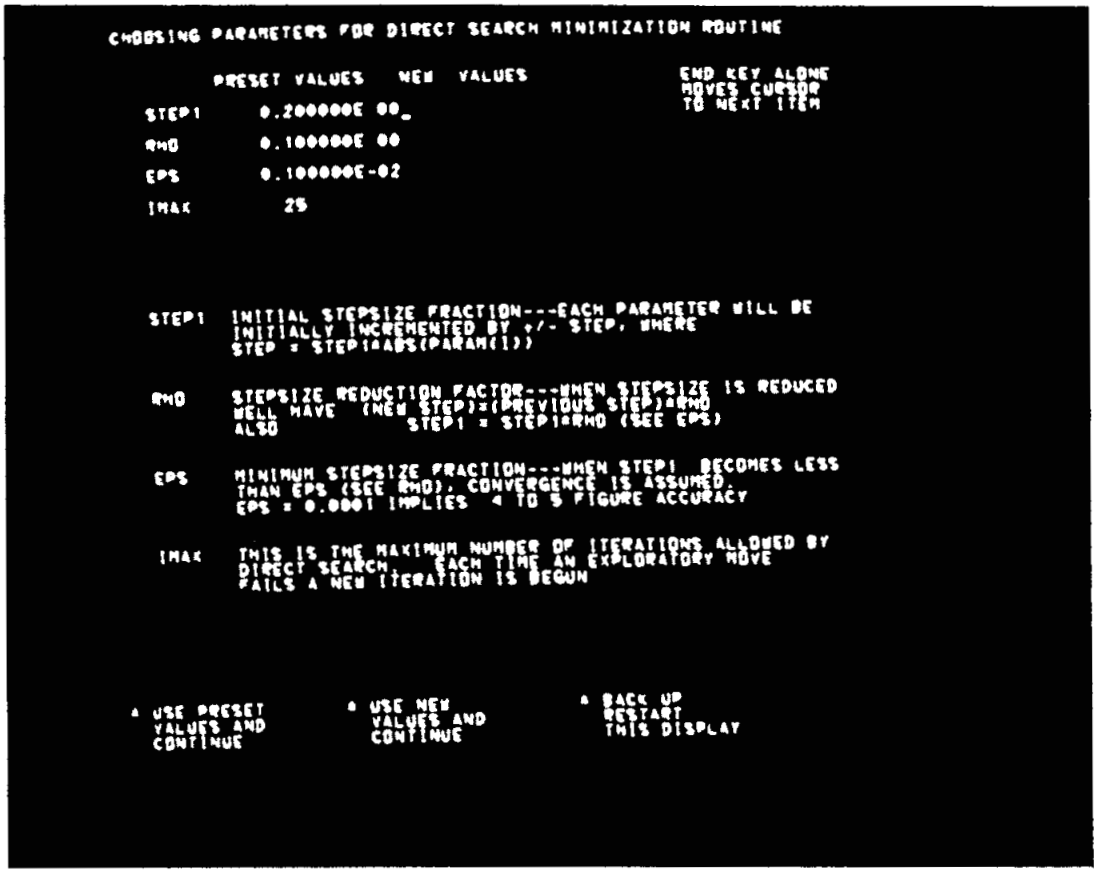

FIG. 3.16--Display allowing change of direct search parameters.

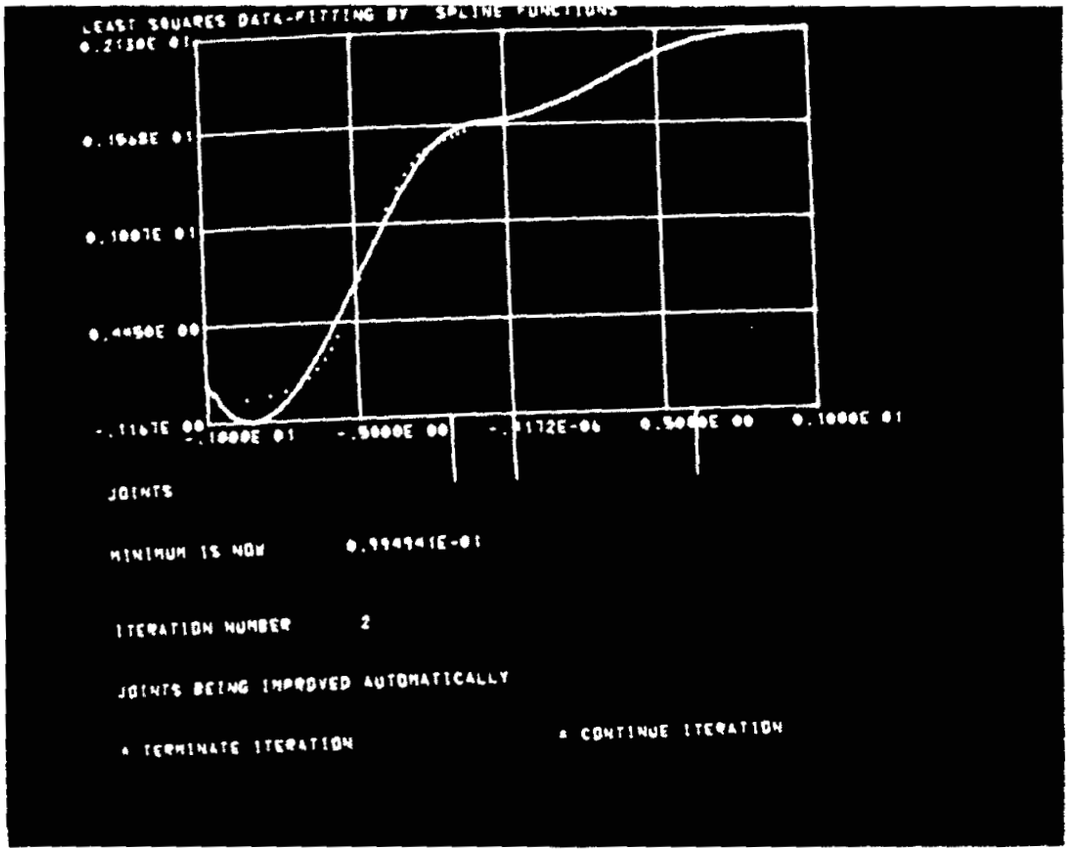

FIG. 3.17--Direct search spline fitting.

function fits. The interactive minimizer is described in detail in paragraph III.A.14. It will suffice to say here that the interactive minimizer can be easily bypassed or it can be used to check and/or improve the results of the Direct Search minimization.

The interactive minimizer may be selected for use by itself as shown in Fig. 3.14. In this case the routine for interactive minimization is brought in directly rather than after the Direct Search has been terminated.

9. Choosing Display Mode

After the least squares fit has been computed a display allowing user selection of a display mode is put on the CRT. There are seven different display modes, each showing a different facet of the computed fit. Figure 3.18 shows this display as it is presented to a user for his selection. In addition to selecting one of the display modes, if the fit has been computed using orthogonal polynomials or the Fourier approximation, a user may also specify the degree of fit he would like to see displayed. This option is presented, since for these functions all fits for degrees less than or equal to the degree specified in paragraph III.A.7 are available without the necessity of further computations.

The seven modes of displaying the calculated least squares fit are:

a. A plot of the data and fit with a tabular display of the fit, each on half of the CRT screen.

b. A plot of the data and fit with a display of the coefficients of the fit, each on half of the CRT screen.

c. A full CRT screen plot of the data and the fit.

d. A plot of the residuals with a tabular display of the fit, each on half of the CRT screen.

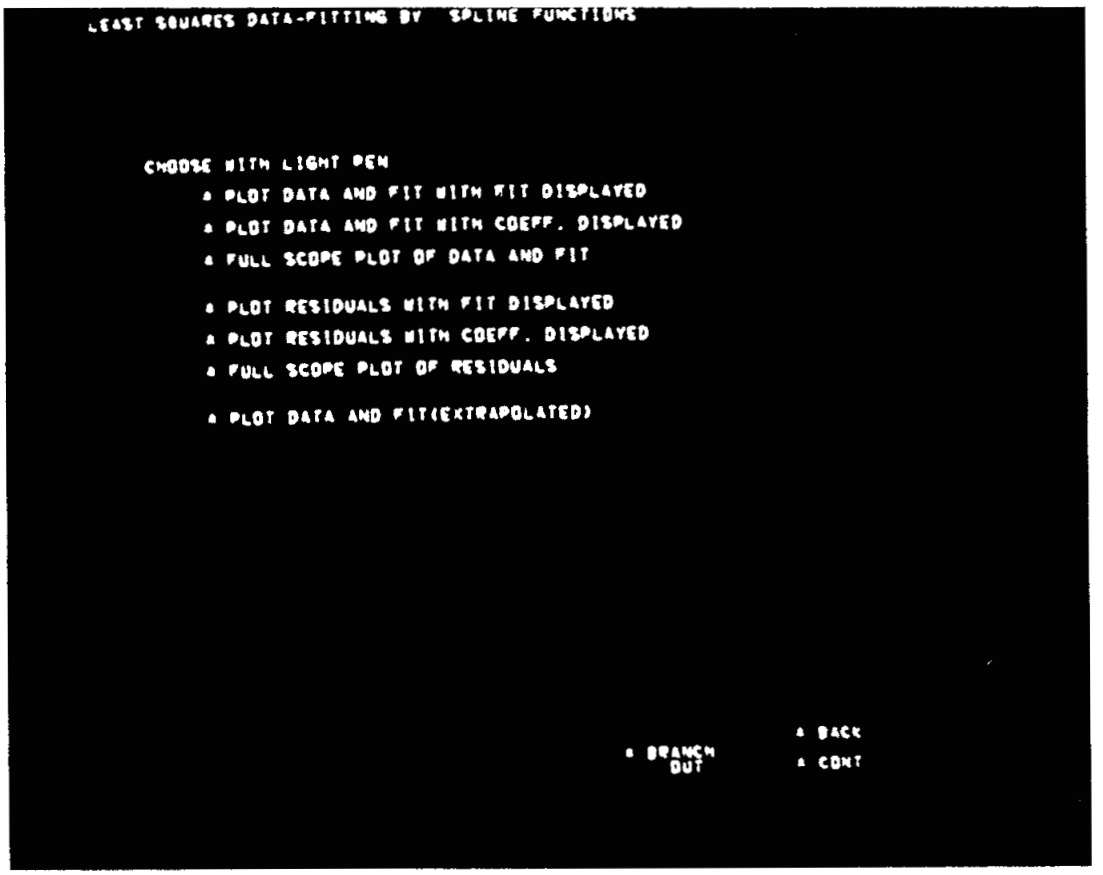

FIG. 3.18--Choosing display mode.

e. A plot of the residuals with a display of the coefficients of the fit, each on half of the CRT screen.

f. A full CRT screen plot of the residuals (connected by straight lines to make a continuous curve).

g. A full screen plot of the data and fit with the fitting function extrapolated in both directions.

10. Displaying the Fit

After choosing one of the seven display modes discussed above, the chosen display appears on the CRT screen. Figure 3.19 shows an example of the first mode. Notice that the tabular values showing the fit have been acted on by the lightpen "FORWARD" command to display points 16 through 21 (there were a total of 21 points in this case).

The second option of the display modes takes different forms depending on the fitting function involved. The upper half of the screen always shows a plot of the data points with the fitting function superimposed. The lower half of the screen displays the coefficients of the fit which are different for each function. Along with the coefficients is displayed a brief mathematical description of how they are used to formulate the given function. In the case of spline functions a "FORWARD" and "REVERSE" lightpen option is given to allow scrutiny of the polynomial segments which make up the spline function. Figures 3.20, 3.21 and 3.22 show the form of this display for orthogonal polynomials, Fourier approximations and spline functions, respectively.

An example of the third mode of display is shown in Fig. 3.23. In this case the plot of the data and fit is expanded to fill the whole screen of the CRT thereby allowing a closer examination of features of the plot.

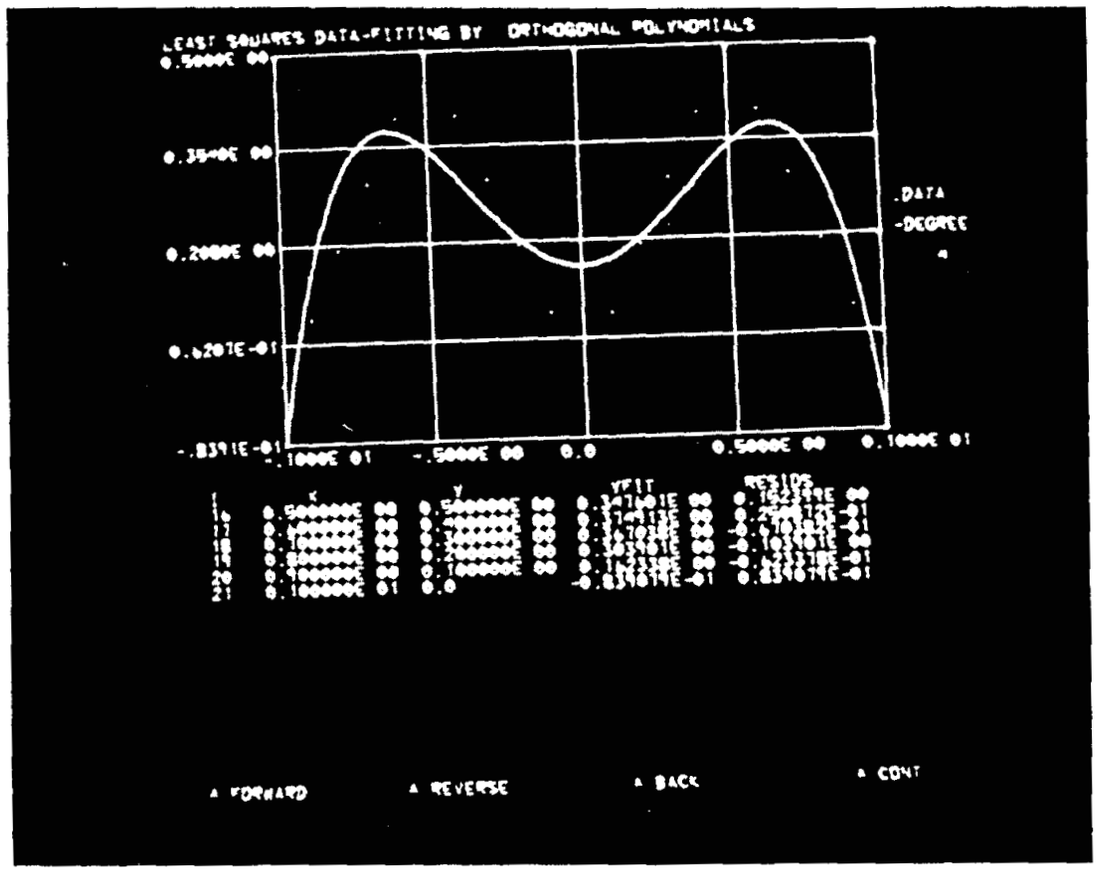

FIG. 3.19--Data and fit with fit displayed.

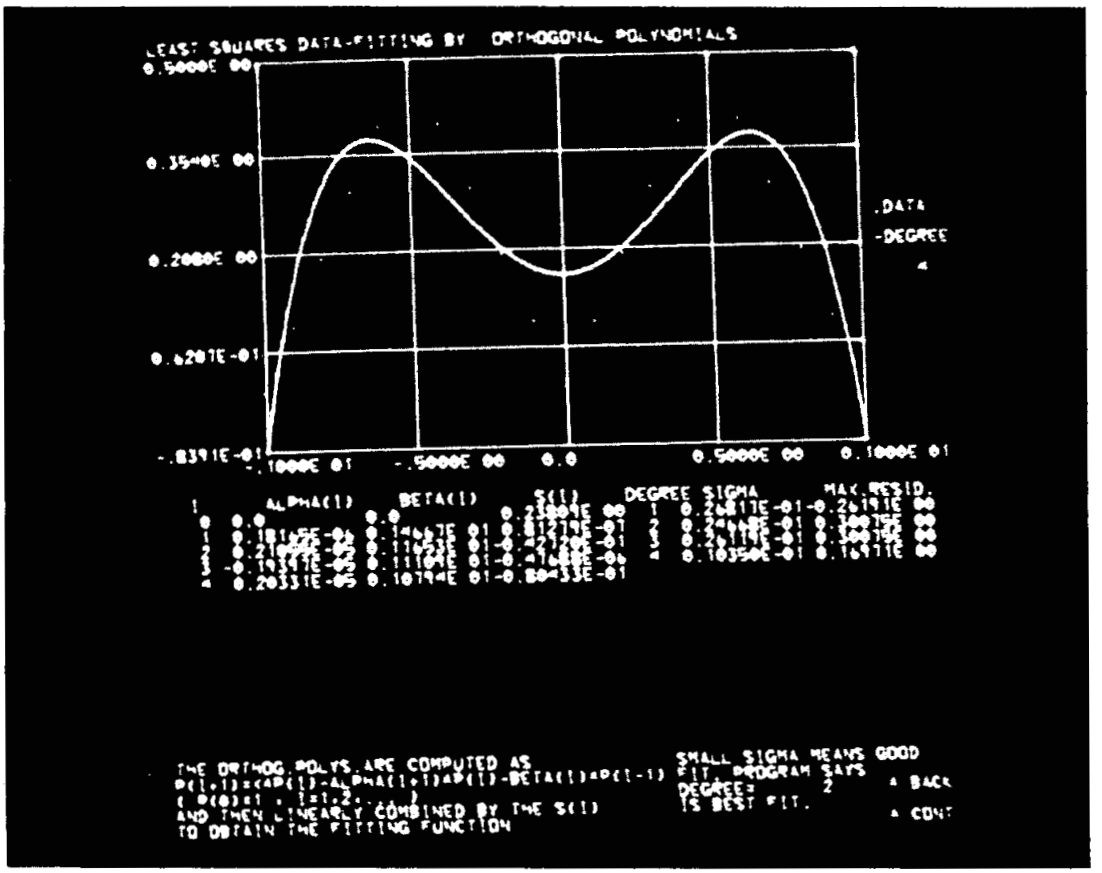

FIG. 3.20--Data and fit with orthogonal polynomial coefficients.

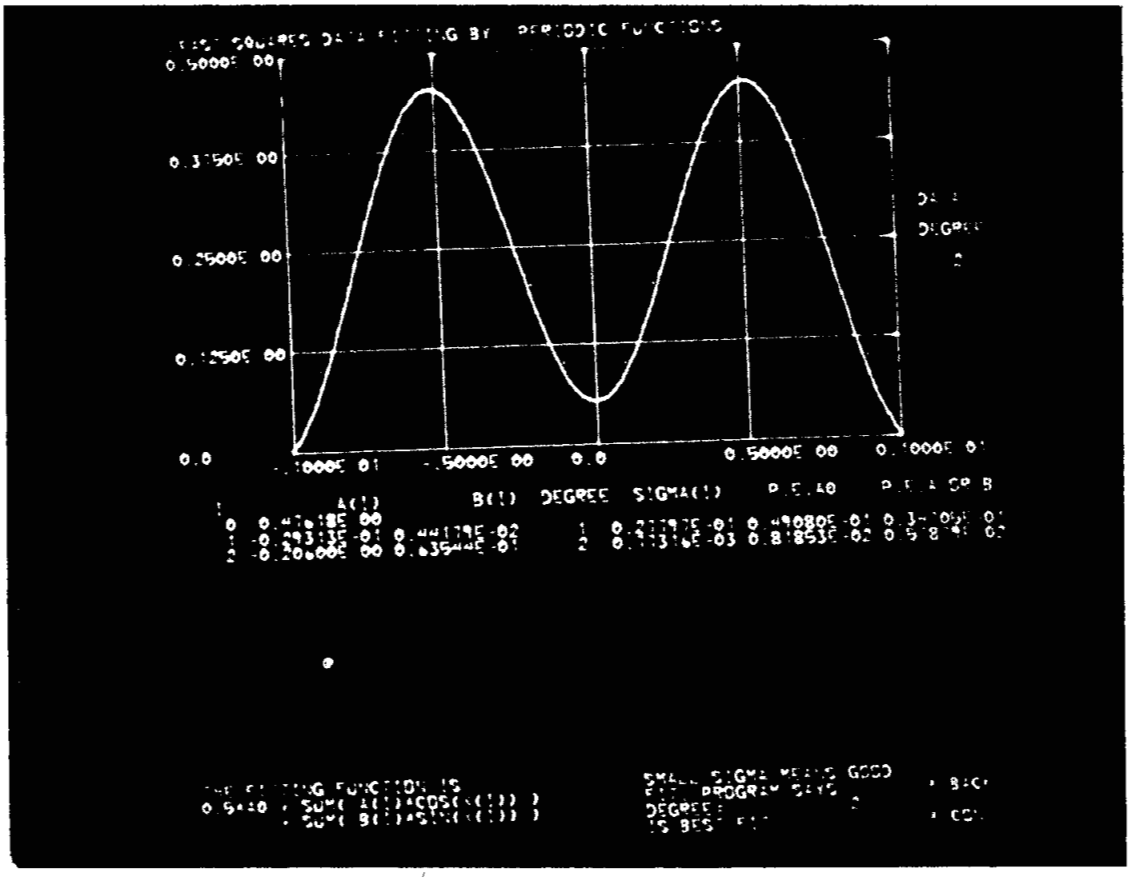

FIG. 3.21--Data and fit with Fourier approximation coefficients.

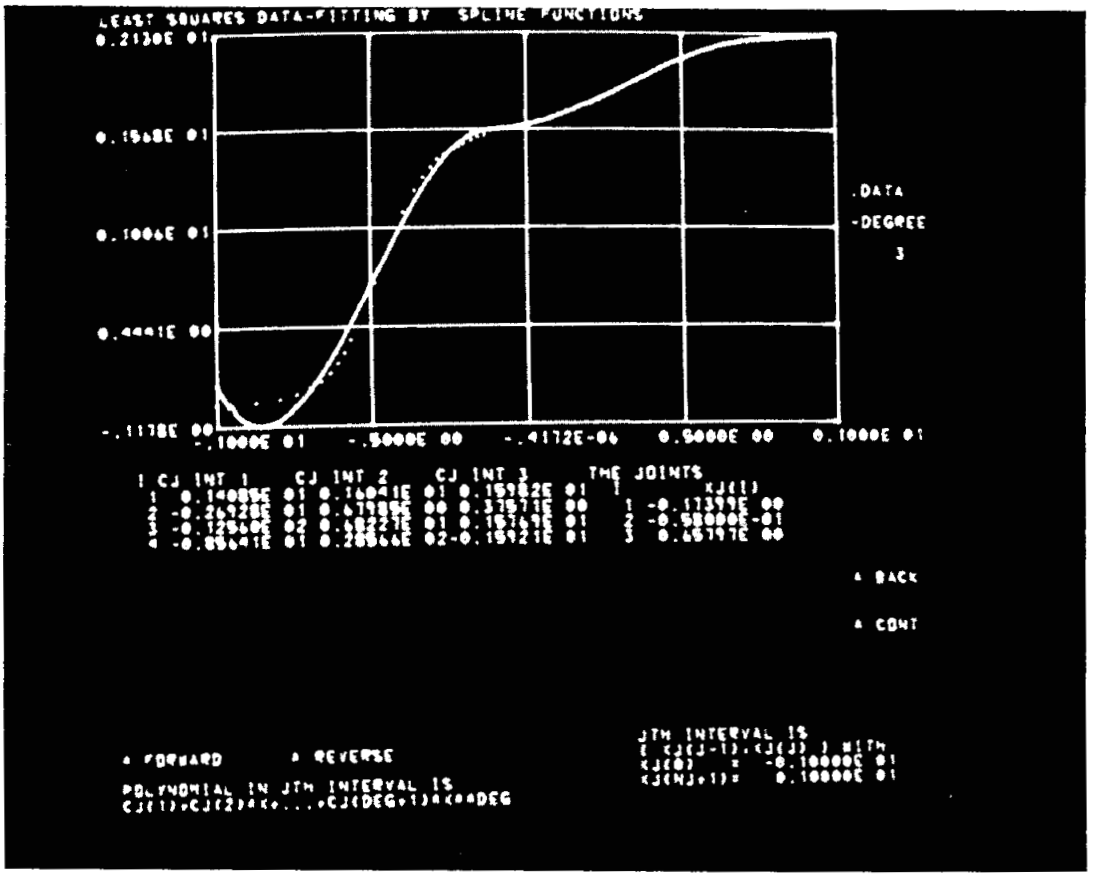

FIG. 3.22--Data and fit with spline function coefficients and joints.

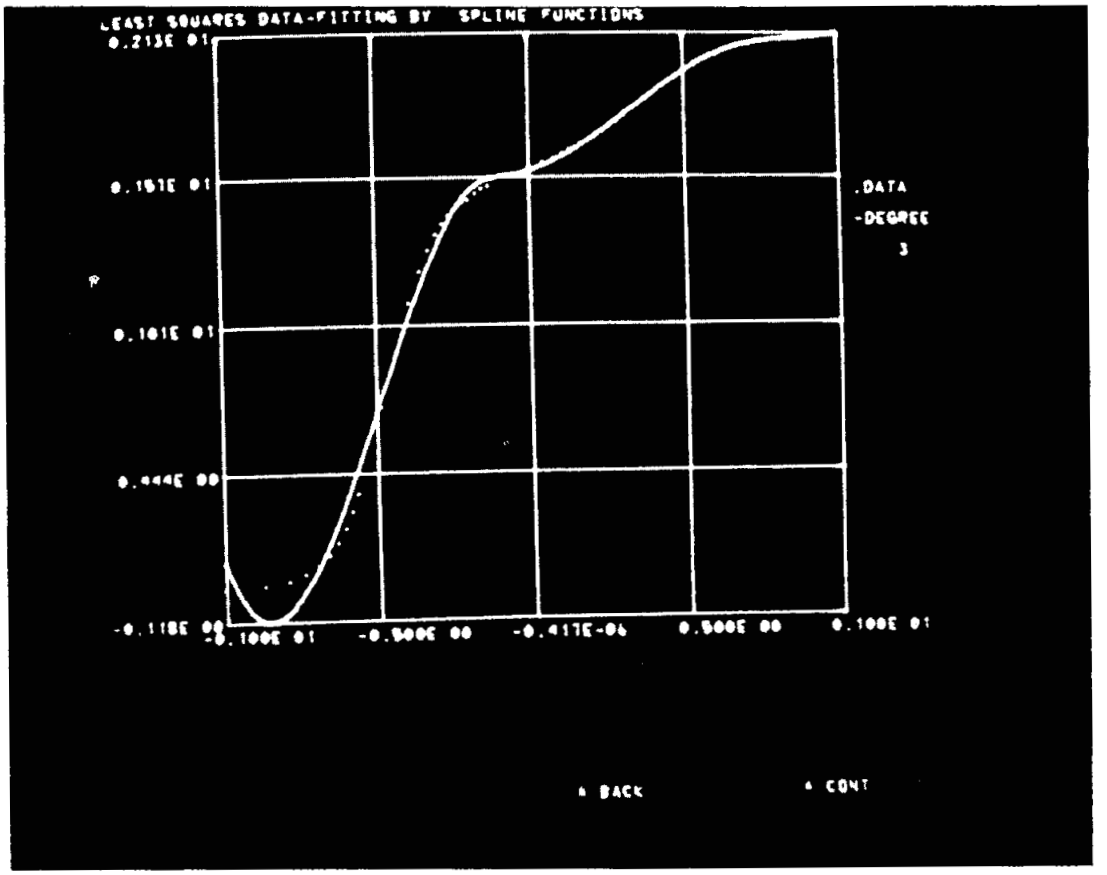

FIG. 3.23--Full scope plot of data and fit.

The fourth mode of display is as shown in Fig. 3.24. Here the residuals are plotted and the points are connected by straight line segments. On the lower half of the screen the data and computed fit with residuals are presented in a tabular fashion.

The residuals are plotted along with a display of the coefficients in the fifth mode of display. Again, as described for the second of these display modes, the coefficient displays vary according to the function being used. Figure 3.25 illustrates this display mode.

The sixth display mode is a full scope plot of the residuals. As before the points are connected by straight lines and the expanded plot allows any fine details to be examined fully. An example of the use of this mode of display is shown in Fig. 3.26.

The seventh and final display mode involves an extrapolation of the computed approximating function for both higher and lower values of the independent variable. Given that the independent variable, x, lies in the interval $a \leq x \leq b$ for the original data, the extrapolated function will be plotted for $\left(a - \frac{b-a}{6}\right) \leq x \leq \left(b + \frac{b-a}{6}\right)$. This choice of extrapolation limits is arbitrarily chosen and easily altered. Figure 3.27 shows an example of the extrapolation display.

11. The Branching Step

Following the display(s) showing the computed fit, the next step in the natural order of the data-fitting process is the "branching" step. From this point, program control is transferred to one of the previous steps to allow initiation of a new problem or a new attack on the problem at hand. The points to which control may be transferred are:

a. Starting over from the beginning

b. Choosing the function

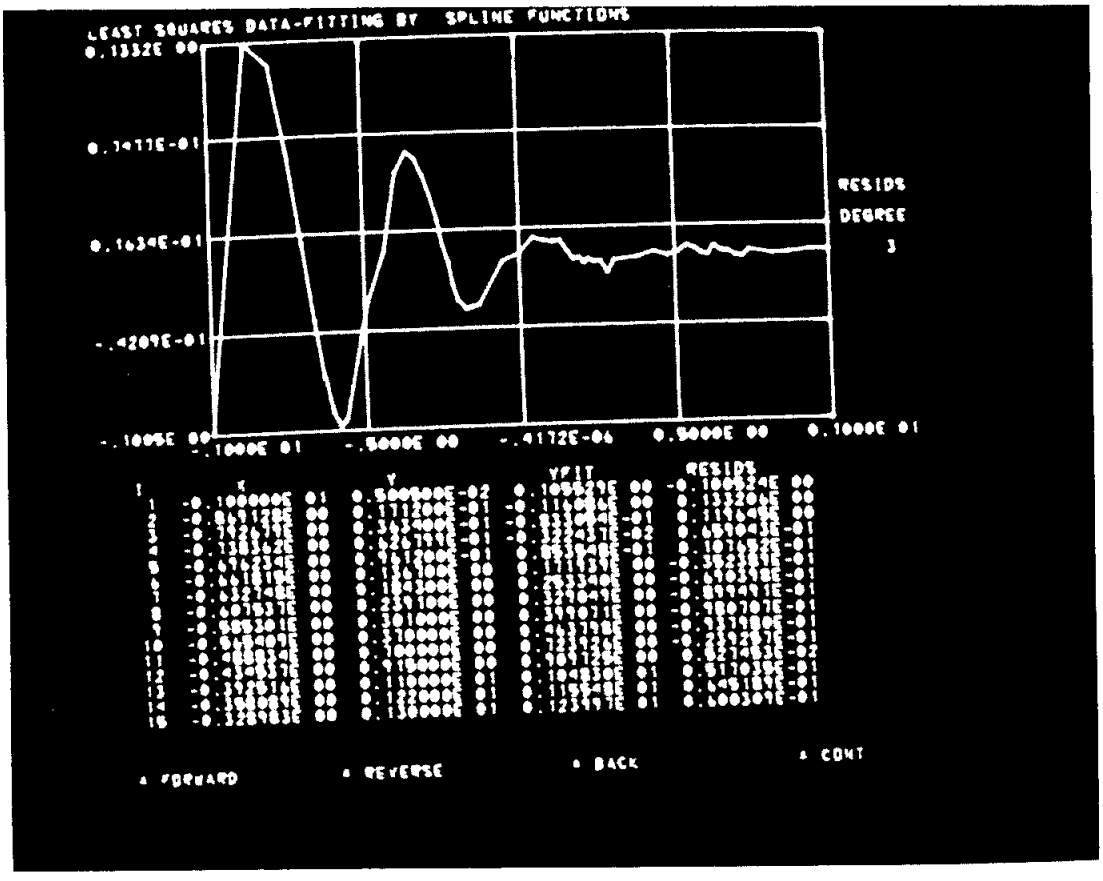

FIG. 3.24--Plot of residuals with fit displayed.

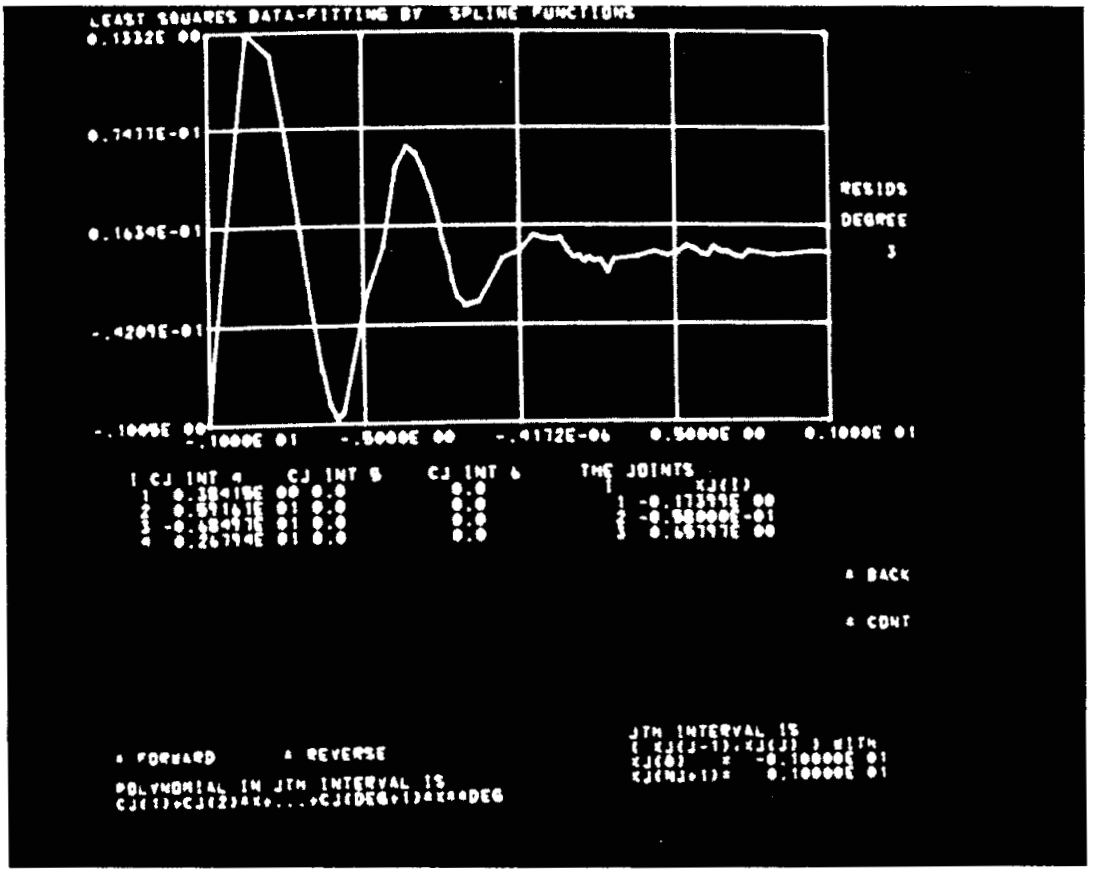

FIG. 3.25--Plot of residuals with coefficients displayed.

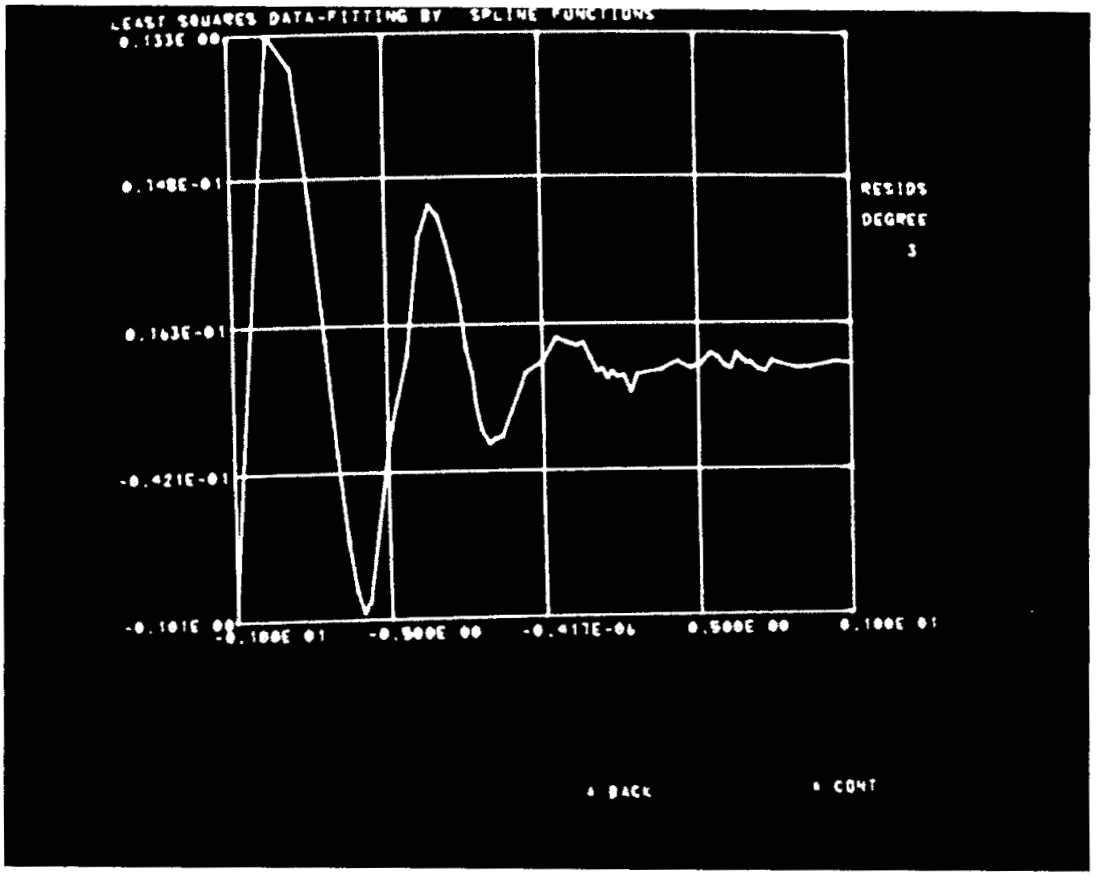

FIG. 3.26--Full scope plot of residuals.

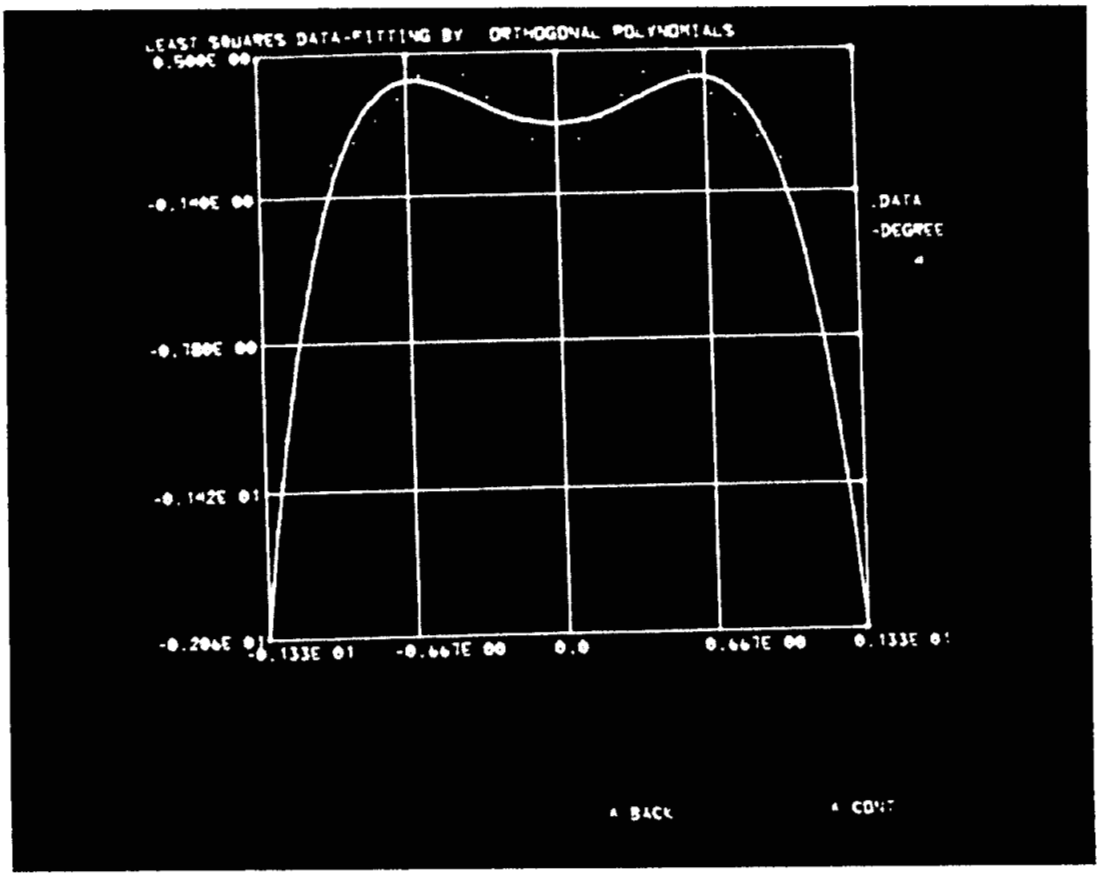

FIG. 3.27--Plot of data and fit extrapolated.

c. Choosing the data mode

d. Correction and/or subset selection

e. Data transformation

f. Point deletion

g. Choosing degree — entering initial parameters (joints for splines)

h. Choosing display mode

i. Examine fit comparison (to be described in paragraph III.A.12)

j. Return to where came from

k. TERMINATE COMPUTER PROCESSING.

In addition to the above there are two options which allow a user to look at further displays which are for information only and have no computational purpose. These are:

l. Display data and function titles (see Fig. 3.6 in paragraph III.A.3)

m. Tutorial displays .

Lightpen selection of item m brings out a display which lists various tutorial displays. Selection of an item from that list causes display of the corresponding tutorial material.

This branching step display is shown in Fig. 3.28. This display is activated in two ways. It is either reached in the natural order of events, following one of the various fit displays discussed in paragraph III.A.10, or it is reached by means of the "*BRANCH OUT" lightpen command which is mentioned in paragraph III.A.1 and is available on eight different displays. Item j above, "Return to where came from," is useful in connection with the "*BRANCH OUT" command as it allows return to the proper point in the data-fitting process without requiring the user to remember where he was when he selected the "*BRANCH OUT" option.

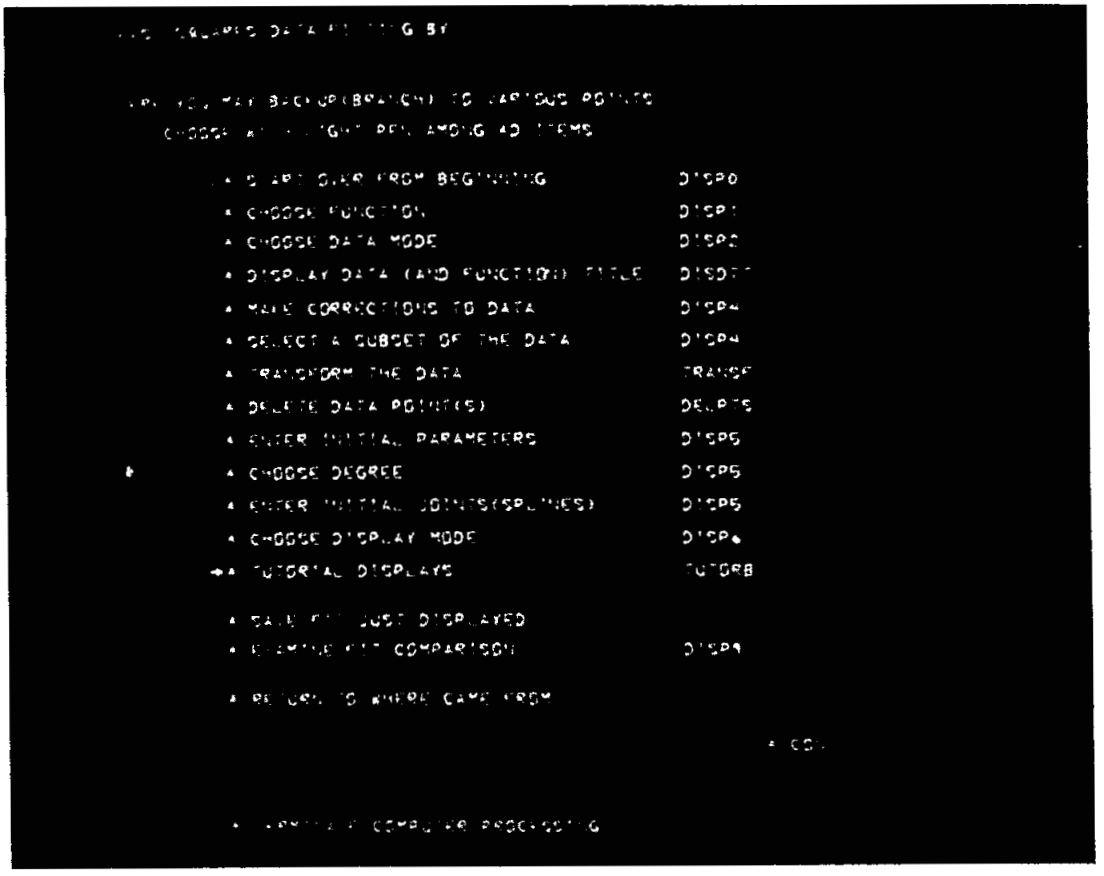

FIG. 3.28--Branching step display.

The program does not allow transfer of control to points in the process that are not yet meaningful. For example, if a fit has not yet been computed, a transfer to item h, "choose display mode," is meaningless. In such a case the program behaves as if "Return to where came from" had been selected.

In addition to the above choices there is a lightpen command to "Save the fit just displayed" for comparison purposes. This is necessary in order to specify which fits are to be compared when the display to be described in paragraph III.A.12 is requested.

Figure 3.28 shows the display which allows user selection of the many branching options.

12. A Fit Comparison

This display shows a comparison of the approximations obtained by orthogonal polynomials, spline functions, and the Fourier approximation. For each function the degree (and number of joints for splines) is given which relates to the number of parameters in that fitting function. Then the maximum residuals, and the sum of squares of the residuals for each fit are listed for comparison. In the example shown in Fig. 3.29 all three functions involved 5 parameters and so were equally good candidates in the sense of number of parameters; however, as shown by the maximum residuals and the sums of squares of residuals the spline function was by far the best fit to the particular data involved. Figures 3.20 and 3.21 show two of the fits compared here.

This fit comparison is most useful for comparing approximations to the same set of data. For this, the option to use the data of the previous fit as described in paragraph III.A.4, is very useful.

13. Tutorial Displays

Many displays in this data fitting program have a lightpen option which will cause transfer of control to the branching display described in paragraph III.A.11.

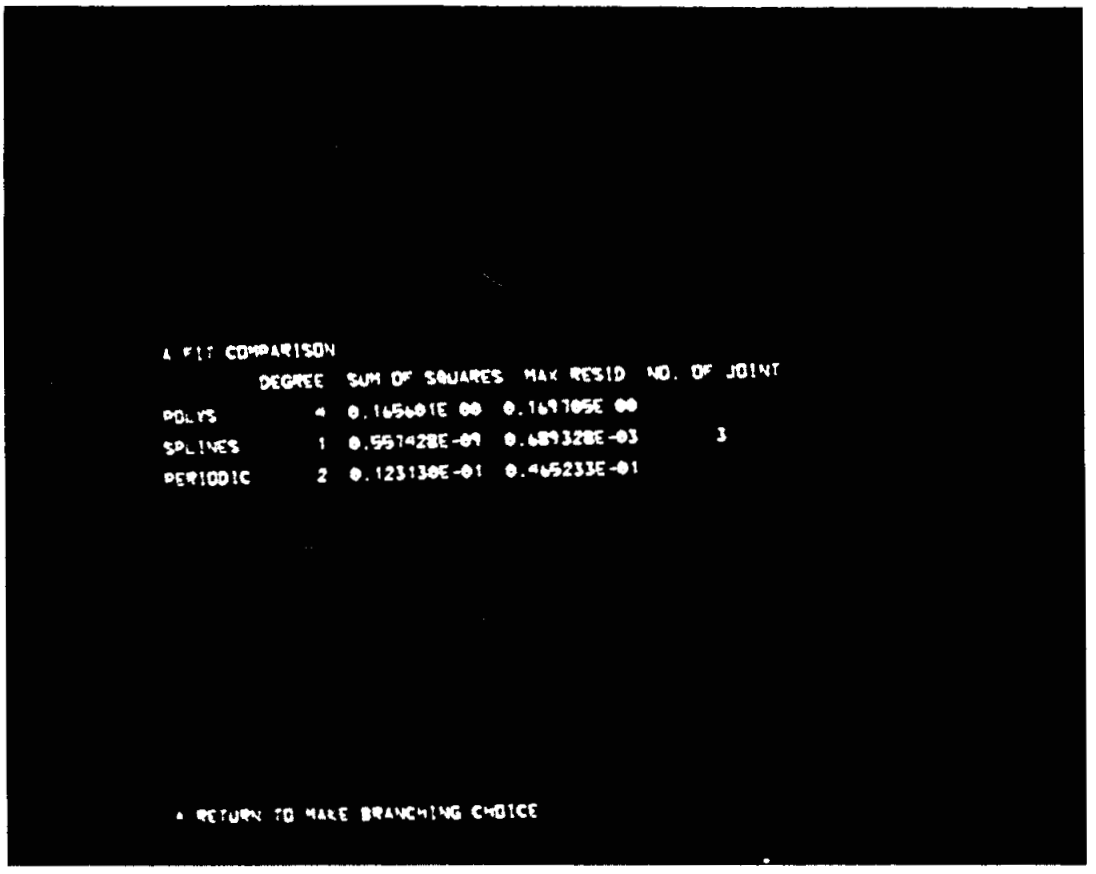

FIG. 3.29--A fit comparison.

From the branching display a user may then request tutorial information which may help him interact more profitably with the program. This essentially gives a user a "HELP" button which he may push at any time. To make the HELP facility applicable to many situations the tutorial displays cover a variety of subjects and situations that may occur while using the interactive data-fitting system.

Figure 3.30 shows the display that appears when the request for tutorial displays has been made.

Each of the individual tutorial displays has a lightpen command which will return control to the display shown in Fig. 3.30. From there control can be transferred back to the branching choice display where control can in turn be transferred to any one of the other steps in the data-fitting process.

Five of the tutorial displays are shown in Figs. 3.30a, 3.30b, 3.30c, 3.30d and 3.30e. Additional tutorial displays can be easily added to the system.

14. The Interactive Minimizer

Here we shall give only a brief description of the interactive minimizer. The minimization problem is to locate values of the n parameters which give a minimum value for the function (sum of squares in our problem). At such a minimum the function will be at a minimum with respect to each parameter considered by itself. Thus if we hold all parameters fixed except one, and plot the function as we vary that one parameter we will obtain a plot as shown in Fig. 3.31. The minimization problem is solved if we find values for all parameters which are at the minima of such curves drawn for each parameter. The interactive minimizer allows user modification of all parameter values until the curves for all parameters are simultaneously minimized at the chosen values.

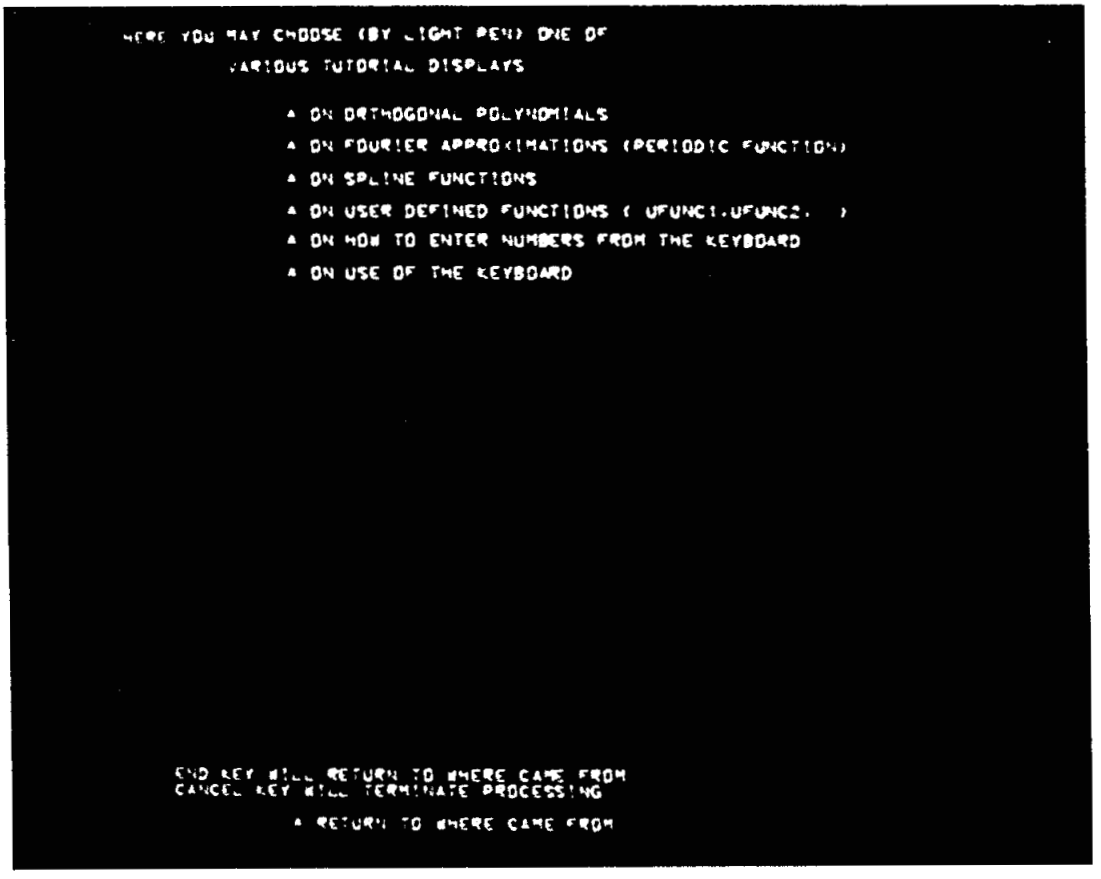

FIG. 3.30--Display allowing choice of tutorial display.

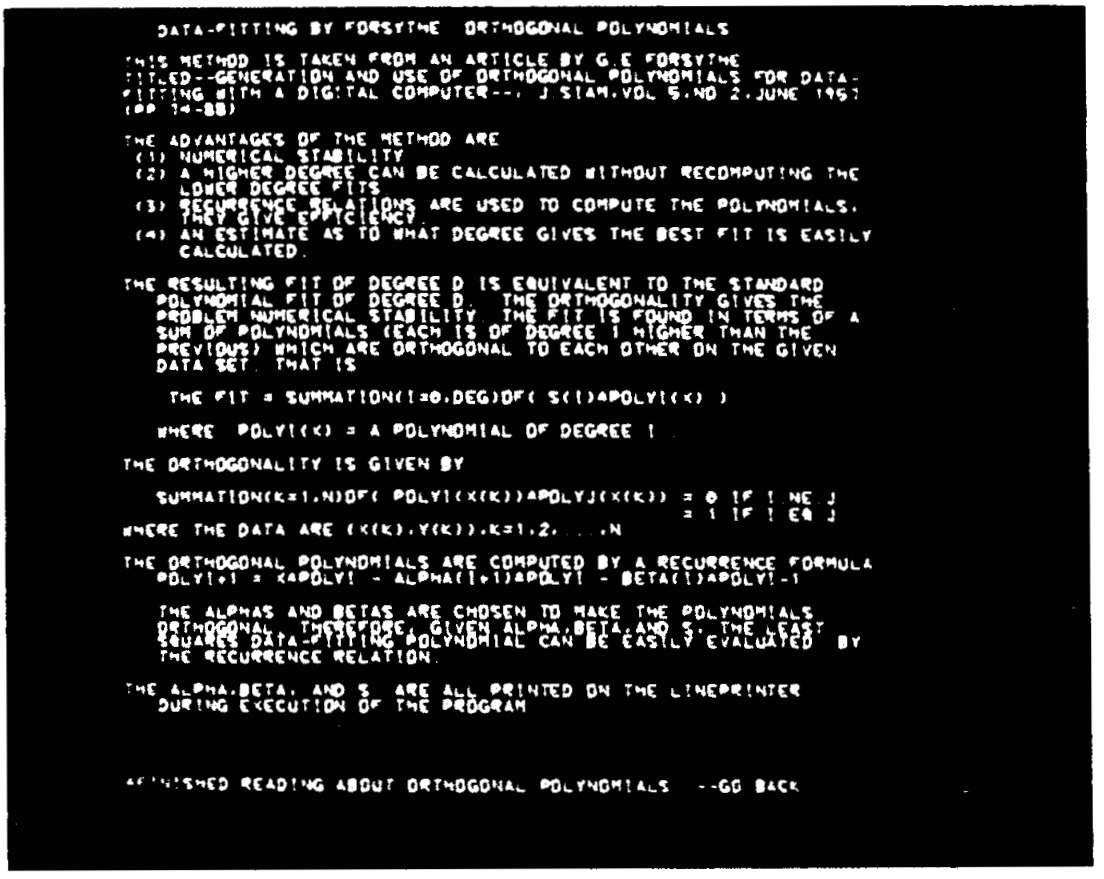

FIG. 3.30a--Tutorial display for orthogonal polynomials.

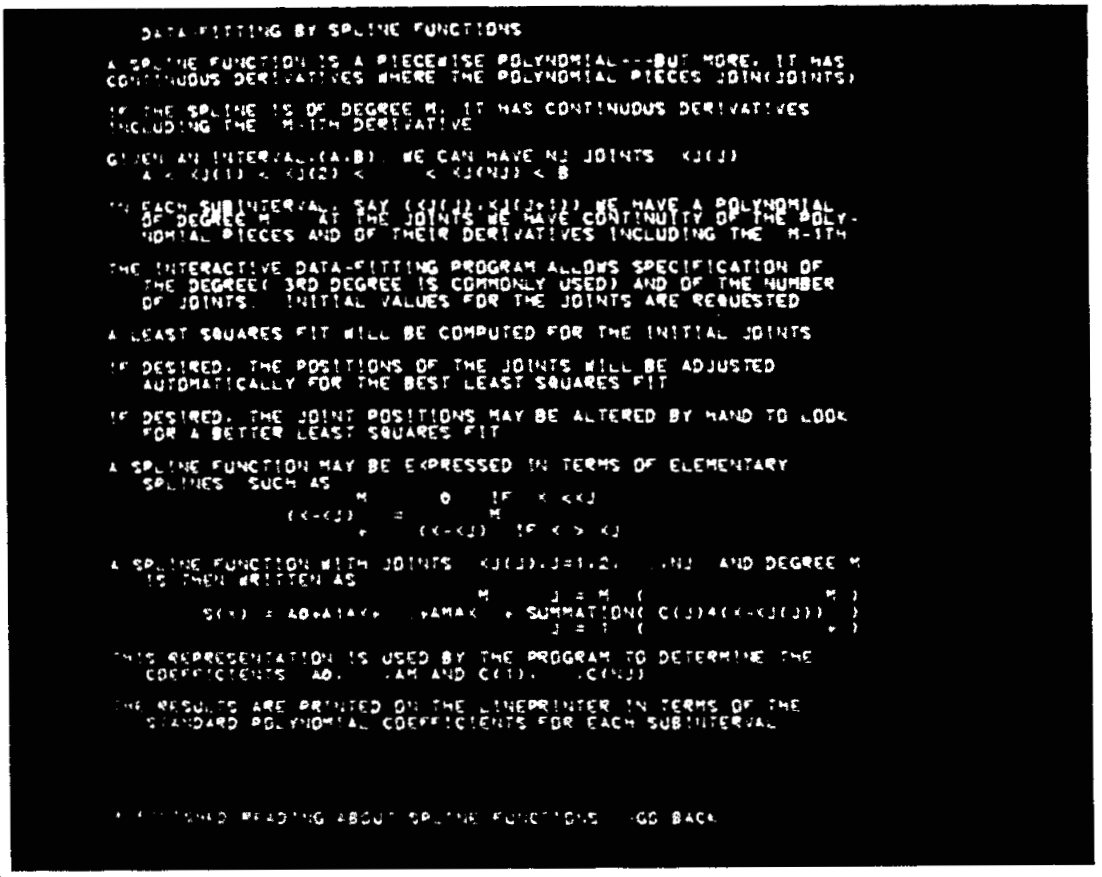

FIG. 3.30b--Tutorial display for spline functions.

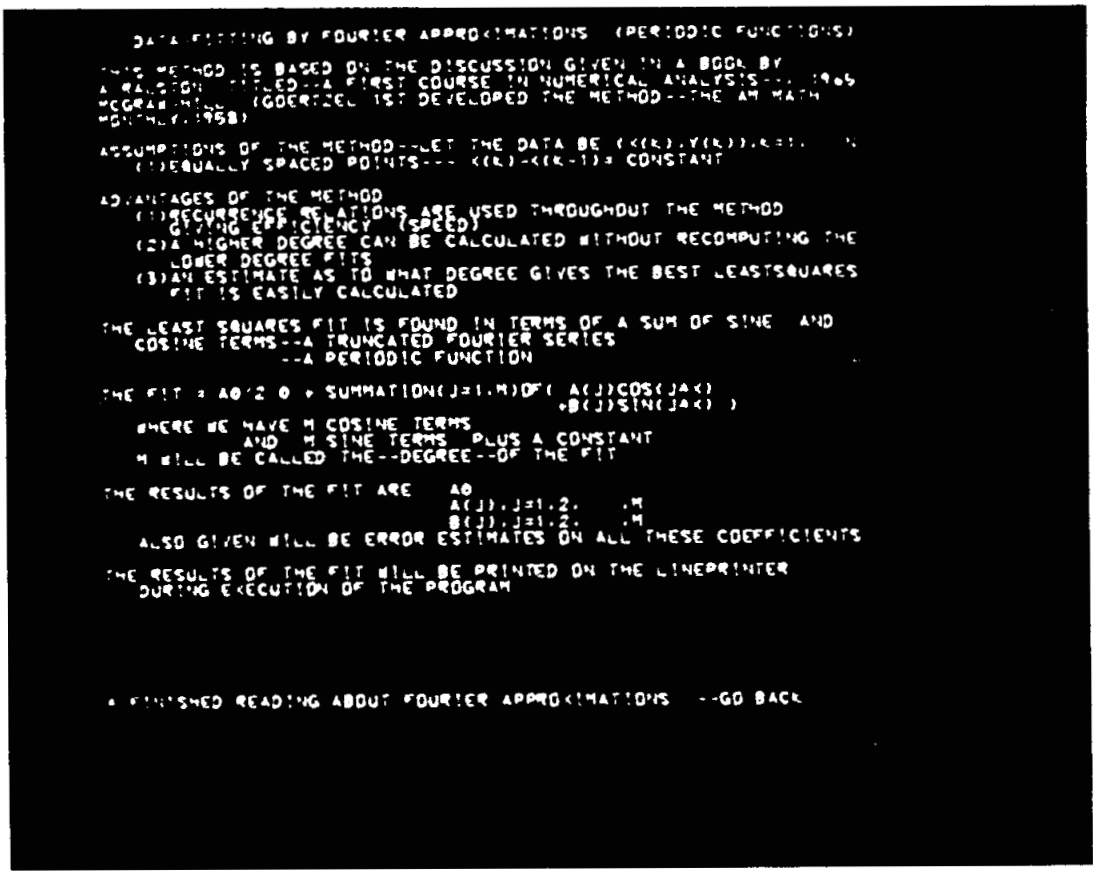

FIG. 3.30c--Tutorial display for Fourier approximations.

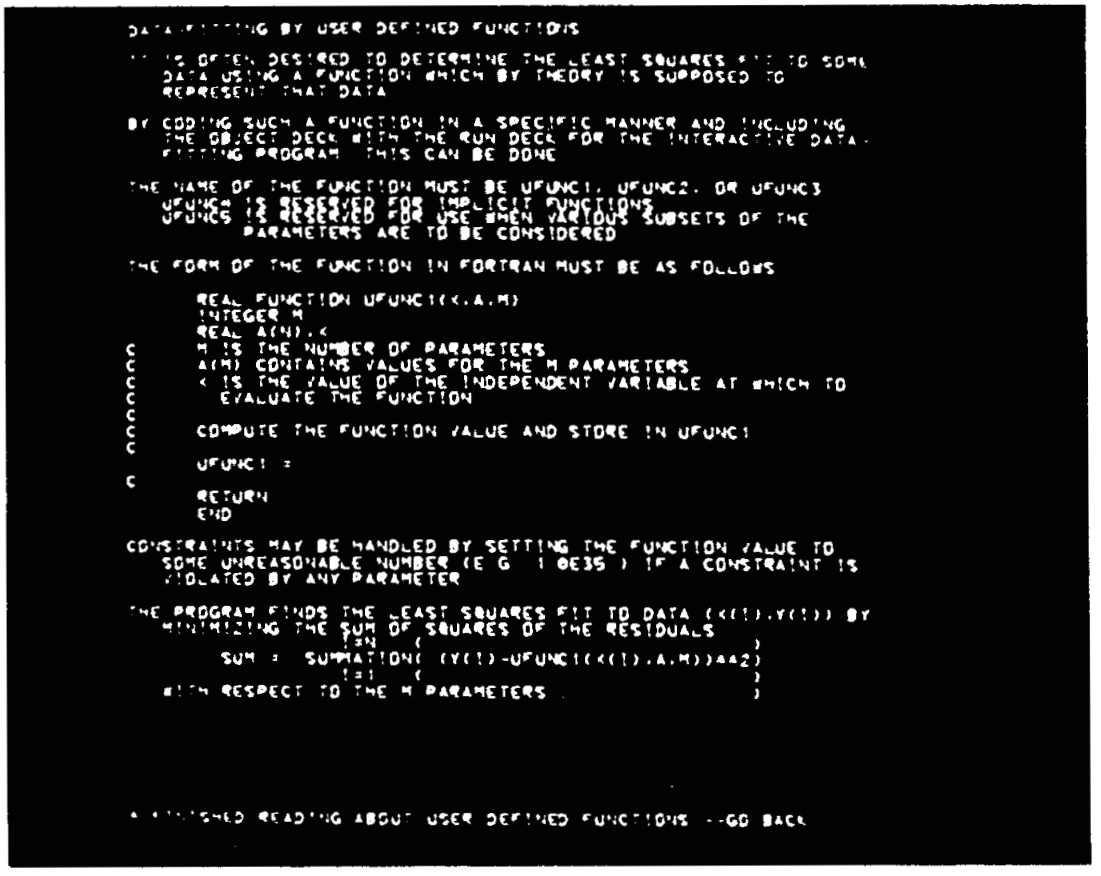

FIG. 3.30d--Tutorial display for user defined functions.

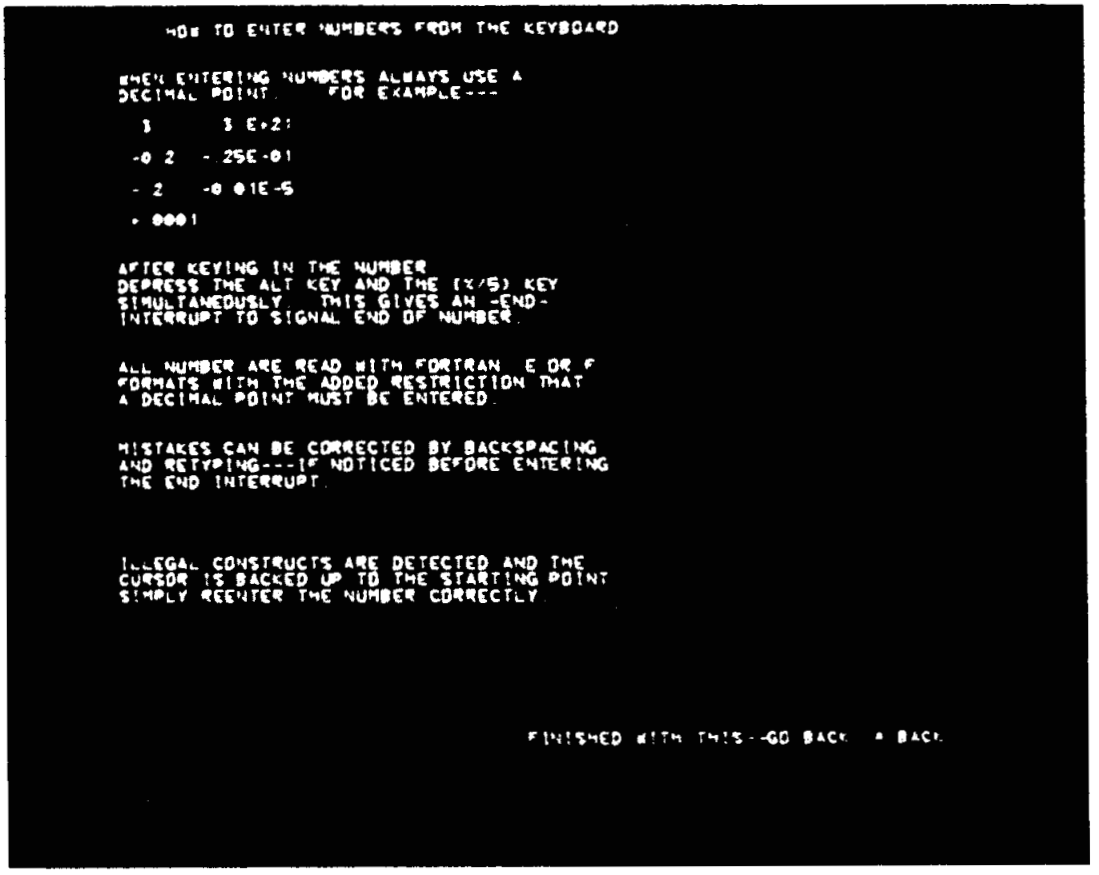

FIG. 3.30e--Tutorial display on how to enter numbers from the keyboard.

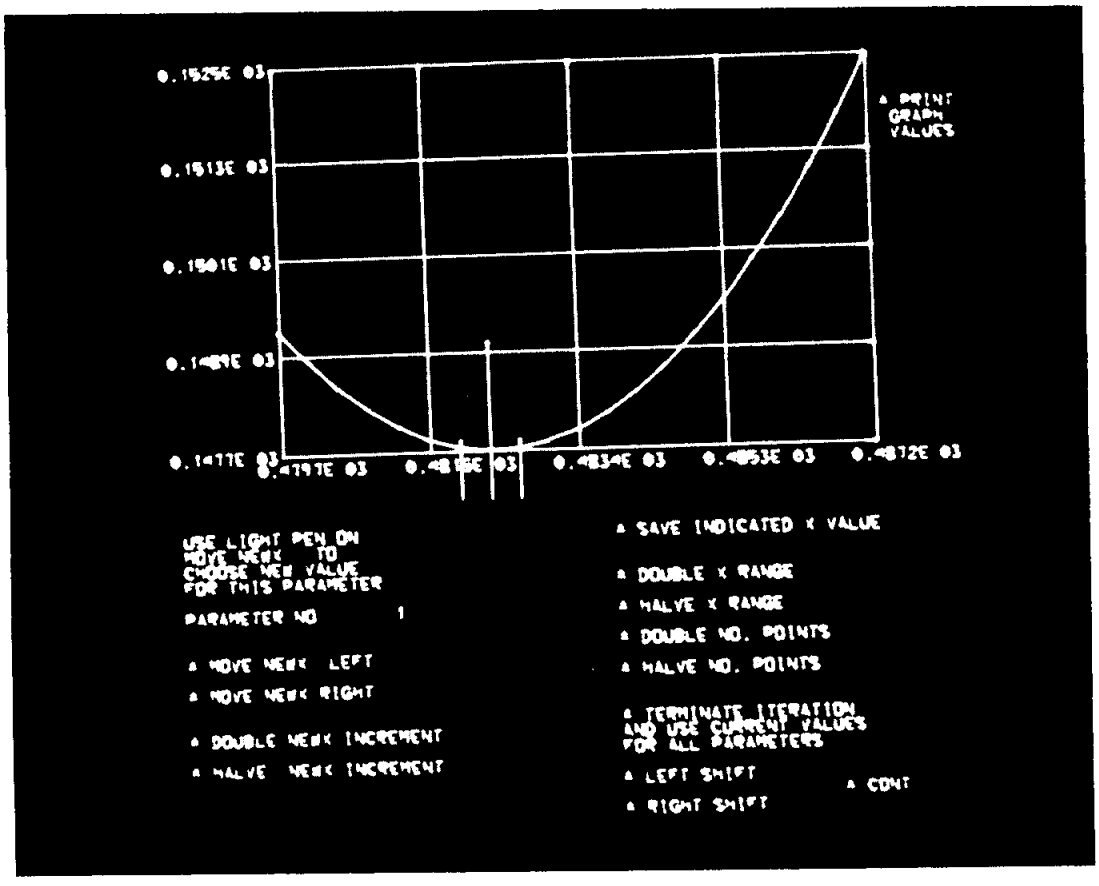

FIG. 3.31--Typical display during interactive minimization.

Figure 3.31 shows the lightpen commands that are available for the purpose of carrying out the minimization. After all parameters have been adjusted, a summary display is presented which shows current values for parameters and function and allows either another iteration (cycle through the parameters) or termination of the minimization by this method.

When the interactive minimization routine is entered at the termination of the minimizing efforts of the direct search routine it can be useful for checking purposes. One can simply cycle through the parameters once and check to be sure that for each parameter the value found by direct search is indeed at the minimum of the corresponding curve. If obvious improvements can be made then, of course, it is easy to make them at this point. In general the direct search results are good. However, if the convergence criteria in direct search are somewhat lenient, improvements can be made in the interactive minimizer.

B. User Defined Functions

In order to introduce flexibility into the interactive data-fitting program the ability to employ a user defined function as the least squares approximating function has been implemented. The only restrictions on these user defined functions are that they must have no more than fifteen parameters to be adjusted and that the parameter list conform to the requirements of the data-fitting program. A final restriction is that they be able to handle the case of zero parameters and preferably use this case to store a function title for later on-line display.

1. The Form (How to Code a User Function)

The form of these user defined functions must be as shown in Fig. 3.32.

The available names for user functions are UFUNC1, UFUNC2, UFUNC3, UFUNC4, and UFUNC5. The current implementation handles UFUNC4 and

```
            EXAMPLE OF A USER DEFINED FUNCTION


C
      REAL FUNCTION UFUNC1(X,A,N)
      INTEGER N
      REAL A(N),X
C
C     N     IS THE NUMBER OF PARAMETERS, .LE. 15 .
C     A(N)  IS AN ARRAY CONTAINING THE CURRENT VALUES
C           OF THE PARAMETERS.
C     X     IS THE VALUE OF THE INDEPENDENT VARIABLE
C           AT WHICH THE FUNCTION IS TO BE EVALUATED.
C
C     TO ILLUSTRATE----ASSUME THE FUNCTION TO BE FIT
C     IS  A CONSTANT PLUS AN EXPONENTIAL TERM.
C
C         A1 + A2*EXP(A3*X)
C
C
C     IF N EQUALS 0 WE PUT A TITLE IN
C
      COMMON/TITL/DTITL(10),FTITL(5,15)
      INTEGER DTITL,FTITL
      INTEGER TLE(15)/'PUT FUNCTION TITLE  E.G. A1+A2*EXP(A3*X)  HERE '/
      INTEGER TII
C
      REAL T
C
      IF(N .GT. 0) GO TO 7072
      DO 7071 TII=1,15
          FTITL(1,TII) = TLE(TII)
 7071 CONTINUE
      UFUNC1 = 0.0
      RETURN
 7072 CONTINUE
C
      T = A(3)*X
C
      IF( ABS(T) .GT. 170.0 ) GO TO 200
C
      UFUNC1 = A(1) + A(2)*EXP( T )
C
  100 RETURN
C
  200 UFUNC1 = 1.0E70
      GO TO 100
C     END UFUNC1
      END
```

FIG. 3.32--Example of a user defined function.

UFUNC5 in special ways to be described later. The use of the first three functions by the data-fitting program will now be described.

2. How User Functions are Used by the Program

Assume the data points are given as $\{(x_i, y_i)\}_{i=1}^{N}$ and that the desired fitting function has been coded as UFUNC1 (see the example shown in Fig. 3.32). A routine to calculate the sum of squares of the residuals then calls UFUNC1 as shown in Fig. 3.33. The actual code also provides for weights to be utilized in computing the sum of squares if weights are given for each data point. The weights enter into the calculation as shown by the comment in Fig. 3.33.

3. The Use of Object Decks for User Functions

The current implementation of the data fitting program on the IBM 360/91 allows object decks for user defined functions to be included with the deck that activates the program. The object deck overrides any user function with the same name that may be stored on disk, thereby allowing user defined functions to be changed very simply.

The user written routines which may be included as object decks are:

UFUNC1	(see paragraph III. B. 1)
UFUNC2	(see paragraph III. B. 1)
UFUNC3	(see paragraph III. B. 1)
IMPFCN	(see paragraph III. B. 5)
SQUELL	(see paragraph III. B. 6)
CEPLT	(see paragraph III. B. 7)
UFUNC5	(see paragraph III. B. 9)

4. Implicit Function Fitting (UFUNC4)

UFUNC4 has been reserved for least squares fitting of data to an implicit function of two variables, $f(x, y) = 0$. Thus the coding necessary for a user to

COMPUTING THE SUM OF SQUARES

```
C
      REAL FUNCTION SUMSQ(X,Y,N,A,NP)
      INTEGER N,NP
      REAL X(N),Y(N),A(NP)
C
C     X() AND Y() CONTAIN THE DATA POINTS.
C     N   IS THE NUMBER OF DATA POINTS.
C     A() CONTAINS THE CURRENT VALUES OF THE PARAMETERS.
C     NP  IS THE NUMBER OF PARAMETERS.
C
C
      INTEGER I
      REAL TEMP,TEMP1
C
      TEMP = 0.0
      DO 100 I=1,N
          TEMP1 = UFUNC1(X(I),A,NP)
C
          TEMP = TEMP + ( Y(I) - TEMP1 )**2
C     IF WEIGHTED  TEMP=TEMP+( WEIGHT(I)*(Y(I)-TEMP1) )**2
  100 CONTINUE
      SUMSQ = TEMP
      RETURN
C     END SUMSQ
      END
```

FIG. 3.33--Computing the sum of squares.

accomplish this kind of fitting is changed from that required for UFUNC1, UFUNC2, and UFUNC3.

In the case of an implicit function, two special routines must be coded by the user and the object decks included with the run deck. As currently implemented, UFUNC4 will fit data to an ellipse where the equation of the ellipse is given as

$$\frac{R^2 \cos^2(\theta - \theta_0)}{a^2} + \frac{R^2 \sin^2(\theta - \theta_0)}{b^2} - 1.0 = 0 \qquad \text{(III. 1)}$$

where

$$R = \sqrt{(x - x_0)^2 + (y - y_0)^2}\ , \qquad \theta = \tan^{-1}\left(\frac{y - y_0}{x - x_0}\right),$$

and a and b are the semi-major and semi-minor axes of the ellipse. The quantities x_0 and y_0 represent the coordinates of the center of the ellipse and θ_0 is the angle by which the major axis is tilted from the x-axis.

Given an implicit function, f(x, y) = 0, such as (III. 1), the three special routines to be coded by a user are

a. a routine to evaluate the function f(x, y),
b. a routine to compute $\sum_{i=1}^{N} \left[f(x_i, y_i)\right]^2$, and
c. a routine to compute points for plotting the computed function for display purposes.

5. Coding an Implicit Function

The implicit function must be evaluated by a routine like that shown in Fig. 3.34. The name and parameter list must correspond to that shown.

6. Coding the Sum of Squares for an Implicit Function

For implicit functions the user may encode the routine which will calculate the sum of squares which is to be minimized to determine the least squares fit. The reason for requiring this is to allow specification of constraints. If no

EXAMPLE CODE TO EVALUATE AN IMPLICIT FUNCTION

```
C
      REAL FUNCTION IMPFCN(X,Y,A,NP)
C
      INTEGER NP
      REAL X,Y,A(NP)
C
C     NP  IS THE NUMBER OF PARAMETERS.
C     A(NP) CONTAINS THE CURRENT VALUES OF THE PARAMETERS.
C     X,Y REPRESENT THE POINT AT WHICH TO EVALUATE THE
C         INPLICIT FUNCTION
C
C     FOR EXAMPLE---CONSIDER THE EQUATION FOR AN ELLIPSE,
C
C         X**2 / A**2  + Y**2 / B**2  = 1.0
C
C     THIS CAN BE REWRITTEN AS AN IMPLICIT FUNCTION
C
C         F(X,Y) = 0 = X**2 /A**2 + Y**2/B**2 - 1.0
C
C     AN ELLIPSE THAT'S ROTATED AND TRANSLATED WILL
C     HAVE FIVE PARAMETERS---
C         A(1) = X COORDINATE OF CENTER
C         A(2) = Y COORDINATE OF CENTER
C         A(3) = ANGLE OF TILT OF MAJOR AXIS
C         A(4) = SEMIMAJOR AXIS
C         A(5) = SEMIMINOR AXIS
C
C
      REAL T1,T2,T3
C
      T1 = X - A(1)
      T2 = Y - A(2)
      T3 = SQRT( T1*T1 + T2*T2 )
      T1 = ATAN2(T2,T1) - A(3)
C
      IMPFCN = (T3*COS(T1) / A(4))**2 + (T3*SIN(T1)/A(5))**2 - 1.0
C
      RETURN
C     END IMPFCN
      END
```

FIG. 3.34--Example code to evaluate an implicit function.

constraints are appropriate, the built-in code will compute the sum of squares and no user written code need be included.

The code for the implicit function sum of squares calculation must correspond to that shown in Fig. 3.35 with respect to the name and parameter list. Constraints may be handled as in the example where parameter values not within the allowable region cause the sum of squares to be very large. This essentially puts up a high wall around the allowable region in parameter space.

7. Coding the Plot of an Implicit Function

Since an implicit function may be a multiple valued function when one of the variables is considered as an independent variable, the plotting of the function for display purposes must be handled differently. To accomplish this a separate routine must be coded to compute points in the x, y plane which when connected sequentially will produce a faithful representation of the function at hand.

An example is the most appropriate method to describe the code needed for this routine since different implicit functions will necessarily be plotted differently. Some may best be plotted by polar coordinate representations and some by rectangular coordinates. Figure 3.36 shows the FORTRAN code needed to generate a set of points for plotting an ellipse.

As before, the name and parameter list of any user defined routine must correspond to that shown in Fig. 3.36.

8. Vary an Arbitrary Number of Parameters (UFUNC5)

Sometimes a problem arises which involves a function of many parameters which is to be used to approximate some data in the least squares sense. With many parameters however, one may desire to hold some of the parameters fixed and allow the remainder to vary. Thus a least squares fit involving a subset of the parameters can be obtained.

EXAMPLE SUM-OF-SQUARES FOR AN IMPLICIT FUNCTION

```
C
      REAL FUNCTION SQUELL(A,NP)
      INTEGER NP
      REAL A(NP)
C
      COMMON/INTGRS/I1,I2,I3,I4,I5,I6,N,I8,I9
      INTEGER       I1,I2,I3,I4,I5,I6,N,I8,I9
      COMMON/DATA/X(100),Y(100),WTS(100)
      REAL X,Y,WTS
      REAL IMPFCN
C
C     X() AND Y() CONTAIN THE DATA POINTS
C     N    IS THE NUMBER OF DATA POINTS
C     A() CONTAINS THE CURRENT VALUES OF THE PARAMETERS
C
      REAL TEMP,SQ
      INTEGER I
C
C     CONSTRAINTS MAY BE HANDLED HERE.
C     FOR EXAMPLE, THE LENGTHS OF ELLIPSE AXES
C         MUST BE POSITIVE. (SEE EXAMPLE IMPFCN ROUTINE)
C
      IF(A(4) .LE. 0.0) GO TO 90
      IF(A(5) .LE. 0.0) GO TO 90
      GO TO 95
C
   90 TEMP = 1.0E70
      GO TO 105
   95 TEMP = 0.0
      DO 100 I=1,N
          SQ = IMPFCN(X(I),Y(I),A,NP)
          TEMP = TEMP + SQ*SQ
  100 CONTINUE
  105 SQUELL = TEMP
      RETURN
      END
```

FIG. 3.35--Example of sum of squares for an implicit function.

EXAMPLE CODE TO COMPUTE AN IMPLICIT FUNCTION PLOT

```
C
      SUBROUTINE CEPLT(A,NFIT,XF,YF)
C
      INTEGER NFIT
      REAL XF(NFIT),YF(NFIT),A(5)
C                                                  (A(1),A(2))=(CENTER)
C     THIS ROUTINE COMPUTES POINTS FOR              A(3) = TILT
C     PLOTTING AN ELLIPSE USING THE                 A(4) = A---SEMI MAJOR AXIS
C     PARAMETRIC FORM OF AN ELLIPSE                 A(5) = B---SEMI MINOR AXIS
C
      INTEGER I
      REAL XN,CDP,SDP,CNDP,SNDP,X1,Y1,DP,CT,ST,TMP,TWOPI/6.28318531/
C
C
  100 XN = NFIT - 1
      DP = TWOPI / XN
      CT = COS(A(3))
      ST = SIN(A(3))
      CNDP = 1.0
      SNDP = 0.0
      CDP = COS(DP)
      SDP = SIN(DP)
C
  110 DO 120 I=1,NFIT
          X1 = A(4)*CNDP
          Y1 = A(5)*SNDP
C
          XF(I) = A(1) + X1*CT - Y1*ST
          YF(I) = A(2) + X1*ST + Y1*CT
C
          TMP =  CNDP*CDP - SNDP*SDP
          SNDP = SNDP*CDP + CNDP*SDP
          CNDP = TMP
  120 CONTINUE
C
      RETURN
C     END CEPLT
      END
```

FIG. 3.36--Example code to compute an implicit function plot.

User defined function, UFUNC5, has been implemented to allow this choice of parameters for least squares consideration. The choice of parameters is made at execution time through an interactive display. The display allowing the user to select which parameters are to be varied is shown in Fig. 3.37. For each of the original parameter set, a number is entered from the 2250 keyboard. Entry of zero causes the corresponding parameter to be held fixed at its current value. An entry of one causes the program to treat the corresponding parameter as one of those to be used in the minimization.

9. Coding UFUNC5 for a Variable Number of Parameters

Figure 3.38 illustrates the coding necessary for a user definition of UFUNC5. The function name and parameter list correspond to those for other user defined functions (UFUNC1, UFUNC2, and UFUNC3), but there are two arrays in COMMON storage which hold the information necessary to allow use of a variable number of parameters. One array, GLOBP, contains the original values of all parameters, and the other array, WHCP, signals which parameters are being varied. If WHCP(I)=0, the Ith parameter is held fixed and its value is given by GLOBP(I). On the other hand, if WHCP(I)=1, the Ith parameter is being varied and its value is found in the array A which has been passed as a parameter. If WHCP(I)=1, and there are J values of K such that WHCP(K)=1 with $1 \leq K \leq I$, then the current value of the Ith parameter is found in A(J+1).

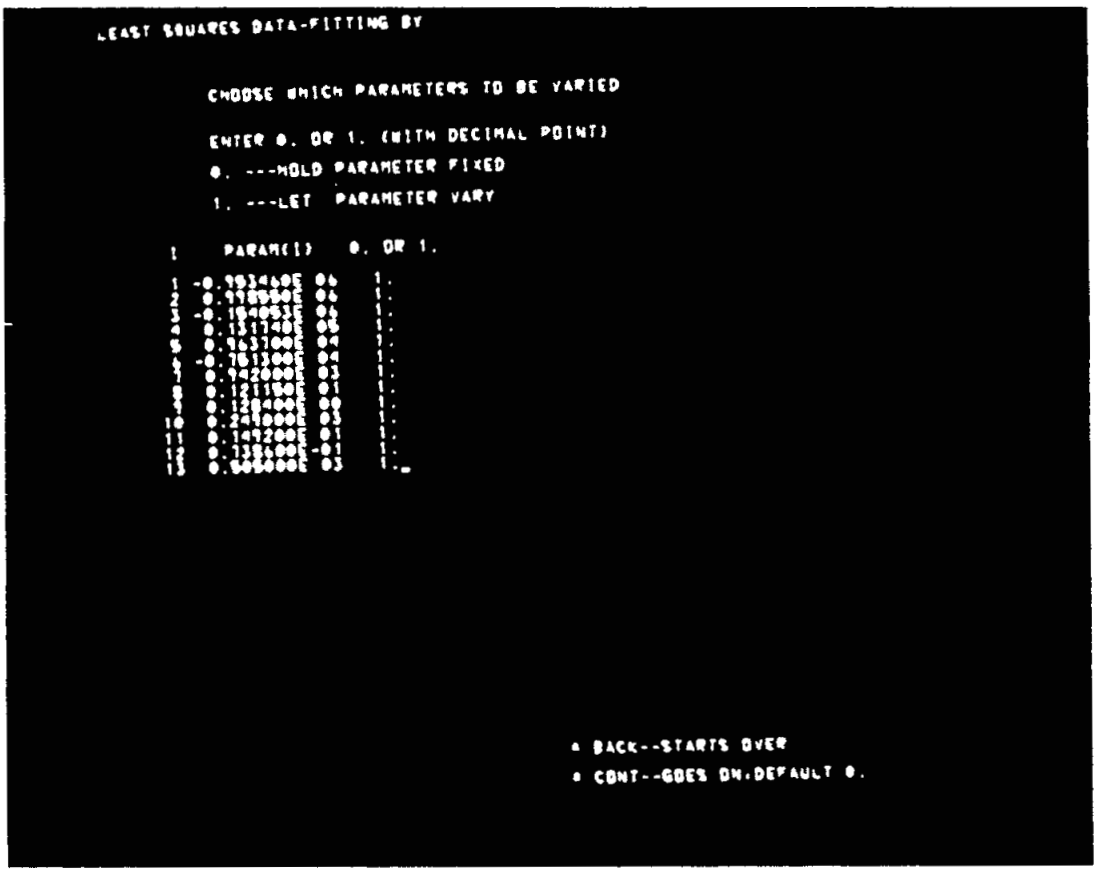

FIG. 3.37--Choosing which parameters to vary.

HOW TO CODE UFUNC5

```
      REAL FUNCTION UFUNC5(X,A,N)
      INTEGER N
      REAL X,A(N)
C
      COMMON/MURPH/GLOBP(15),WHCP(15)
      REAL GLOBP
      INTEGER WHCP
      REAL CH(15),F
      INTEGER I,J
C     PUT THE  DECLARATINS NECESSARY TO A PARTICULAR
C     FUNCTION HERE.
C     THIS STORES ALL PARAMETER VALUES IN CH()
C         SOME VALUES COME FROM GLOBP()---THE FIXED ONES
C         SOME VALUES COME FROM   A() ----THE VARIED ONES
C
C     TO HANDLE  N=0  (PUT FUNCTION TITLE IN COMMON)
      COMMON/TITL/DTITL(10),FTITL(5,15)
      INTEGER DTITL,FTITL
      INTEGER TII
      INTEGER TLE(15)/'THIS EXAMPLE IS A THIRD DEGREE POLYNOMIAL      '/
      IF(N.GT.0)  GO TO 7072
      DO 7071 TII=1,15
         FTITL(5,TII) = TLE(TII)
 7071 CONTINUE
      UFUNC5 = 0.0
      RETURN
 7072 CONTINUE
      J = 1
      DO 105 I=1,15
          IF(WHCP(I) .EQ. 0) GO TO 103
          CH(I) = A(J)
          J = J + 1
          GO TO 105
  103     CH(I) = GLOBP(I)
  105 CONTINUE
C
C     NOW INSERT HERE THE CODE TO
C     EVALUATE A PARTICULAR FUNCTION,F(X,CH),
C     USING PARAMETER VALUES IN CH().
C     FINALLY STORE THE FUNCTION VALUE
C     IN UFUNC5.
C     UFUNC5 =(FUNCTION VALUE)
C     POLYNOMIAL EXAMPLE (DEGREE 3 )
      J = 3
      F = CH(4)
      DO 110 I=1,3
          F= F*X + CH(J)
          J= J-1
  110 CONTINUE
C     VALUE OF POLYNOMIAL IS IN F
      UFUNC5 = F
      RETURN
      END
```

FIG. 3.38--How to code UFUNC5.

IV. DECK SETUP TO USE THE INTERACTIVE DATA-FITTING PROGRAM

There are two methods of using the interactive data-fitting program. If no newly defined user defined function is to be included, an existing load module can be used. However, if an object deck is included for a user function, the deck setup which includes all the linkedit and overlay instructions must be used. These two deck setups will now be described in more detail. Following that, we will give a description of how data cards must be prepared for inclusion with the deck.

A. Using the Load Module

When no object deck for a user defined function is to be included, an existing load module can be called in. Since the load module has already been link edited and prepared for overlay, considerable savings in machine time can be made by using the load module instead of using the deck setup which requires a linkedit and overlay preparation.

The deck setup for using the existing load module is shown in Fig. 4.1.

B. Using Object Deck(s) for User Defined Function(s)

When an object deck for a user defined function (see Section III.B) is to be included, the existing load module may not be used. In this case it is necessary to linkedit and prepare for overlay each time a new user defined function is to be used.

The deck setup for running with an object deck for a user defined function is outlined in Fig. 4.2.

Col. 72

```
//        Brown Card

      (DATA CARDS)
//SYSIN              DD *
//SCOPE2             DD UNIT=0D1
//FT06F001           DD DDNAME=SYSIN
//FT05F001           DD SYSOUT=A
//            EXEC   PGM=NOW
//                   DISP=(SHR,PASS)
//JOBLIB             DD DSNAME=PUB.LBS.PGG,VOLUME=SER=PUB001,UNIT=2314,    +
//                   PRTY=10
//LBSRUN     JOB     'LBS$CG,30M,10000L,(L B SMITH BIN 105)',105,REGION=150K,    +
```

12|4A2

FIG. 4.1--Deck setup to use load module (no user function object decks).

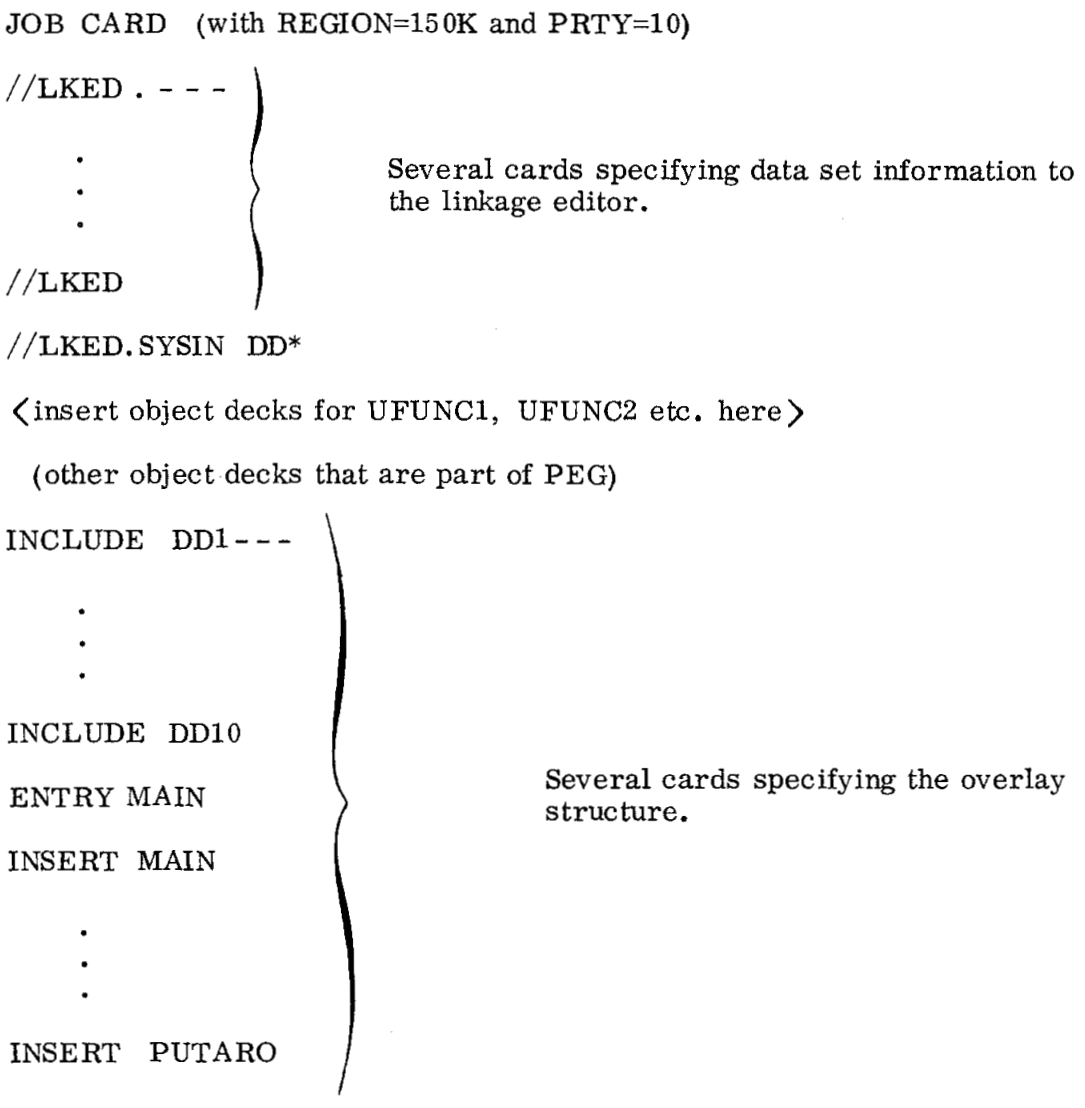

FIG. 4.2--Deck setup to use an object deck for a user defined function.

C. Preparation of Data Cards

A data set for one data-fitting problem consists of N+1 cards. The first card contains the following information:

N --- the number of data points.

WEIGHT --- 0 or 1. 0 if data includes no weights.
1 if data includes weights.

DTITLE --- 40 characters used as a title for the data set which will be displayed on-line on request.

The remaining N cards contain the N data points, one point per card. Each card contains

X(I), Y(I), and if WEIGHT=1, W(I).

X(I) is the abscissa value of the Ith point.

Y(I) is the ordinate value of the Ith point.

W(I) is the weight (or inverse of the error) associated with the Ith point.

The format for preparation of the data card is as follows:

Card 1 --- FORMAT (2I5, 10X, 10A4)

Thus N is in columns 1-5 (right justified)

WEIGHT is in columns 6-10 (0 or 1 in column 10)

DTITLE is in columns 21-60

Card 2 through N+1 --- FORMAT (3F10.5)

X(I) is in columns 1-10

Y(I) is in columns 11-20

W(I) is in columns 21-30 if WEIGHT=1.

Numbers punched in E10.d ($d \leq 4$) or any 10 column F format with an explicit decimal point will be entered correctly.

As many data sets as desired may be put one after the other in the input decks as shown in Fig. 4.1 and Fig. 4.2.

A card with FINISH punched beginning in column 1 should follow the last data set. The purpose of this is to signal to the program that no more data is there to be read. If no FINISH card is found, the program will halt with a read error.

An example of a data set is shown in Fig. 4.3.

```
 30     1              HUGHES--1  BACKGROUND PLUS GAUSSIANS
  1.        29.          .13
  2.        32.          .18
  3.        16.          .25
  4.        29.          .19
  5.        35.          .17
  6.        50.          .14
  7.        57.          .13
  8.        72.          .12
  9.        46.          .15
 10.       105.          .10
 11.        99.          .1
 12.       179.          .075
 13.       312.          .055
 14.       604.          .04
 15.       733.          .037
 16.       823.          .034
 17.       508.          .045
 18.       287.          .059
 19.        95.          .1
 20.        39.          .15
 21.        13.          .3
 22.        35.          .17
 23.        41.          .16
 24.        26.          .18
 25.        32.          .15
 26.        44.          .17
 27.        21.          .22
 28.        16.          .25
 29.        22.          .21
 30.        33.          .17
```

FIG. 4.3--Example of a data set.

REFERENCES

1. K. F. Gauss, Theoria motus corporum coelestium in sectimibus conicis solem aurbientium, (Perthes and Besser, Hamburg, 1809).
2. A. A. Markov, Wahrscheinlichkeitsrechumung, (Teubner, Leipzig), (German translation of Russian, second edition, 1908, appeared in 1921; original Russian edition appeared in 1900.)
3. S. S. Wilks, Mathematical Statistics, (John Wiley and Sons, Inc., New York, 1962).
4. R. Hooke and T. A. Jeeves, Journal of the ACM 8, 212-229 (1961).
5. G. E. Forsythe, SIAM Journal 5, 74-88 (1957).
6. A. Ralston, A First Course in Numerical Analysis, (McGraw-Hill Book Co., New York, 1965).
7. W. J. Dixon and R. A. Kronmel, Journal of the ACM 12, 259-261 (1965).