SLAC-96
UC-32
(MISC)

THE USE OF MAN-MACHINE INTERACTION IN DATA-FITTING PROBLEMS

LYLE B. SMITH

STANFORD LINEAR ACCELERATOR CENTER

STANFORD UNIVERSITY

Stanford, California

PREPARED FOR THE U.S. ATOMIC ENERGY

COMMISSION UNDER CONTRACT NO. AT(04-3)-515

March 1969

ABSTRACT

The aim of this work is to explore the possibility of providing useful tools for numerically solving scientific problems in an on-line environment. The development of such tools is then exemplified by interactive systems written for an IBM 2250 display console (CRT). Least squares data-fitting and the related problem of numerical functional minimization is chosen as an example of a problem area which can benefit from on-line graphical interaction.

Interactive systems can be designed for various levels of user sophistication. A system that can be useful to persons with any level of programming ability or training has to be designed to lead a user through the problem solving process. This can be accomplished by allowing the user to select from various options at each stage in the problem solution rather than requiring him to specify steps to the computer. The design of PEG, an on-line least squares data-fitting program, embodies this easy-to-use design philosophy. Several other aspects of the design of interactive systems are discussed and exemplified by the design of PEG.

PEG is related to other existing and proposed systems for on-line solution of mathematical problems. A brief survey of other systems is included.

The PEG system includes orthogonal polynomials, Fourier approximations, spline functions (with fixed or variable joints), and user defined functions as fitting functions. The user defined functions and spline functions with variable joints are non-linear problems whereas the other functions involve linear least squares problems. Fortran was used to do all the implementation except for a few 2250 routines coded in assembly language. The use of a high level procedural language such as Fortran, Algol, or PL/1 eases the implementation of such

systems. Also, by coding in Fortran we achieve some degree of machine independence because of its widespread use.

Results are given for problems on which PEG has been used successfully. One problem involves using polynomial or spline functions to obtain a fit to empirical data. Two other problems involve user defined functions: a sum of Breit-Wigner terms and a sum of Gaussian curves. Special consideration is given to the problem of fitting an ellipse to empirical data. The ellipse problem is treated both as an implicit function fitting problem and as a linear least squares problem.

The implementation of an interactive minimizer using a Gauss-Seidel type method is described. This illustrates how on-line graphics can show round-off error and aid a user in locating subsidiary minima. The interactive minimizer is used by the PEG system as an alternative or supplement to the direct search minimization used on non-linear problems.

Finally, an interactive data-fitting program which employs the maximum likelihood method is given. This program uses the interactive minimizer to perform the maximization and produces on-line contour plots for examination of results. A problem involving empirical data has been solved using this program.

ACKNOWLEDGMENTS

I would like to express my thanks to my thesis advisor, Professor Gene Golub, for his advice and counsel during the preparation of this thesis. I am also grateful to Professor George Forsythe and to Dr. Dennis Moore for their constructive readings of the thesis and suggestions for its improvement. Special thanks are due Professor William F. Miller, who not only made it possible for me to do this research at SLAC and was an active member of my reading committee but also provided much sound advice during the last eighteen months of this work.

I would like to thank the members of the SLAC Computation Group, especially Maryanne Fisherkeller and my fellow student, James George, for their help during the implementation and debugging of the PEG system and Charles Zahn who volunteered to perform the test chronicled in Appendix C. The members of the SLAC staff who have used PEG on their problems also deserve my gratitude.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A. Raison d'être

To ease the task of the programmer, many computer installations maintain a library of frequently used subroutines or procedures that eliminate much programming effort in some problems. Often such a library will also include self-contained programs that require a user only to prepare data in some specified form. These "canned" programs eliminate all programming for certain well defined problems. However, in any case there is a delay, called the turn-around time, of "about an hour" to several days (depending on the particular installation) before the results of a computer run are available from a batch processing system.

In the past most installations have employed a batch processing monitor system to process programs on their computer(s). In this batch mode programs are gathered into groups called batches. The number of programs in a batch is usually determined by the estimated total processing time of the collection. Each batch is then presented to the computer and the individual programs are processed sequentially. The time taken to accumulate a batch of programs, process it on the computer and return the output to the individual users is called the turnaround time. From the programmer's point of view, turnaround time is the elapsed time between submission of a program to be processed, and return of the output. Each time a program is submitted to the computer, one must wait for the turnaround time often only to discover that a data or control card was mispunched.

In the last few years several installations have successfully implemented time-sharing systems on their computers. Time-sharing systems have several

consoles or terminals such as typewriters or teletypes attached to the main computer. Each user has a console through which he communicates with the central computer. The purpose of a time-sharing system is to make it appear to each user that the computer is processing his program alone. Thus several users communicate through their consoles with the same computer in a manner that appears to require simultaneous use of the central computer. This is accomplished by dividing a unit of time into small fractions and using each fraction to process a different user's requests. When the unit of time is small compared to human reaction time each user appears to be receiving the sole attention of the central computer. Such time-sharing systems provide what is known as on-line computing. One does not have to submit a program and wait for a turnaround time before getting back results as in a batch system. With time-sharing each user is on-line to the computer. His console, which need not be physically near the central computer, is essentially the operator's console for his own private computer.

With the availability of on-line computing the question arises as to how one should use the new power to best advantage. The first step, of course, is the elimination of the waiting of batch processing. This is eliminated by the use of a time-sharing system. However, rather than use a time-shared, on-line environment to simply provide very fast turnaround for the same kinds of programs used on a batch system, it is desirable to have programs designed to take advantage of the close relationship between man and computer that can be engendered by an on-line environment.

The work reported here was begun with the aim of exploring the possibility of providing a useful tool for numerically solving scientific problems in an on-line environment with graphical capability. Furthermore, it was of interest to strive for as much machine independence as possible. This goal has not been

included in the design of existing systems. The degree to which these aims are accomplished is the subject of this dissertation. Some related problem areas are also recognized, but are not yet solved. These related problem areas are discussed further in Chapter VIII.

Several decisions must be made in order to start such a project. First, an area in numerical analysis most likely to benefit from graphical display and interaction should be chosen. Second, the usefulness of such a system should be determined and the characteristics of the users should be recognized. And third, a definition of what is meant by a "machine-independent" program should be given. A degree of machine independence has been achieved in this work by the choice of programming language.

1. The Problem Area

Data-fitting and the associated problems of numerical functional minimization is an area in which a batch processing environment is especially frustrating, because intermediate results and graphs of results are often needed in order to proceed intelligently. For example, in solving a non-linear least squares problem many different initial values for the parameters may be tried before a satisfactory solution is found. If the various starting points are tried sequentially the results of previous calculations are available to aid in choosing a next starting point. This is a slow tedious process in a batch processing environment. The alternative is to input a spectrum of starting points to be tried on a single, long computer run and hope that the desired result will be somewhere in the range covered. This, of course, is wasteful of machine time and cumbersome, but it does save several turnaround times. Because of the need for intermediate results and graphs in the solution of data-fitting problems, these problems are particularly well suited for interactive treatment. Benefits

should be considerable in the areas of elapsed man-hours and "customer satisfaction." But, beyond the obvious advantages, more insight into the problem may be gained through use of a well designed interactive program. Perhaps the graphical display of certain intermediate quantities will provide insight not otherwise available and thereby aid in the solution of some previously unsolved problem. This is the hope of interactive programming — to provide a new dimension to problem solving capabilities. In a sense, interactive computing is similar to the use of a hand computer in that intermediate results are available for inspection. The relaxation methods that used to be used for the iterative solution of a system of equations were certainly interactive.

2. The Users

Computer user communities can be divided in several ways. Two such ways are by problem area and by expertise in (or desire to engage in) programming. The above mentioned problem area reduces the user group of interest to those having data-fitting problems. A further reduction is accomplished by providing for the non-programming users. This does not mean that expert programmers will be bored with the system. It means the interaction will be designed so that no knowledge of programming is necessary in order to use the system. The scientists in a physics laboratory such as the Stanford Linear Accelerator Center (SLAC) are represented by this division. Many of them are competent programmers but even so when using an existing program they will find it easier and they will make fewer errors if the interaction is designed to be "user-proof." Thus, the question to be answered, is, "Given a prospective user whose problem area is data-fitting and who does not know how to program a computer, what kind of interactive computer program can serve him (her) best?"

PEG, the on-line data-fitting program, described in this report, is an answer to this question. PEG utilizes simple interaction and graphical displays with user selection of options, rather than user specification, to accomplish

least squares data-fitting in an on-line environment. PEG also provides tutorial information to aid a user and essentially leads the user "by the hand" through a data-fitting problem.

3. The Language

A high level language was chosen for coding the interactive programs in order to show that interactive programming can be accomplished effectively at that level of coding. If programs such as PEG can be coded in a "popular" language (e.g., Fortran) without the need for extensive use of assembly language or the use of a special purpose language, then the implementation of such programs can be accomplished on a wider scale than if restrictions or extensions to a language are needed. This is not meant to say that a special purpose language might not be more effective and efficient if it is available, but if the special purpose language is not available, a general purpose algebraic language can be used effectively.

At the time the development of PEG was begun there were other interactive systems for scientific problem solving already in existence. Several such systems are mentioned in the next section. These systems are strongly oriented toward particular machine configurations, peripheral devices, and operating systems. Such systems are often "one-shop" systems and cannot be implemented at another installation very easily. In order to provide some degree of machine independence, the decision was made to implement PEG in a high level language which would be machine-independent to some degree. There are three major candidates for a high level scientific problem solving language: Fortran, Algol, and PL/1. Since Fortran IV is the most widely used and is available on many manufacturers machines, PEG was coded almost entirely in Fortran IV. The only exceptions are certain low level routines which communicate with the display hardware.

A third reason for choosing to code interactive data-fitting routines in a high level language was to ease the implementation at perhaps some cost in run-time efficiency. The PEG system as detailed in this report was implemented and checked out in less than one man-year of designing and programming effort. PEG involves 63 overlay segments, about 70 Fortran subroutines, and well over 200,000 bytes of program. The amount of overlay necessary was dictated by the hardware and operating system environment. The original implementation was done on an IBM 360/75 and an IBM 2250 Model II display unit. A 100K byte partition of memory was made available by the operating system for the high priority execution of "scope" jobs. An IBM 360/91 has since replaced the 360/75. PEG is running on the new computer with no programming changes.

PEG was partially inspired by the DATAN system (Simonsen and Anketell [1966]) which is an attempt to mechanize the curve fitting process in a batch processing environment. Other references of interest are Conn and vonHoldt [1965] which describes an on-line display for the study of approximating functions and deMaine [1965] which discusses the "self-judgement" method of curve fitting. A paper by Pyle [1965] describes a system allowing data input by question and answer which is similar in concept to the method employed by PEG to obtain information from the user.

B. Other Systems and Their Relation to PEG

Here we list some interactive graphical systems for mathematics. This list is compiled from a search of the literature and thus may omit some very interesting systems that are in use or under development but have not been described in the computing literature. The systems are grouped according to their applicability. That is, a system that provides the general mathematical capabilities to solve a variety of problems will be classed as a "general purpose"

system. On the other hand, a system oriented toward a specific problem area, such as data-fitting by least squares, will be classed as a "special purpose" system. A paper by Ruyle et al. [1967] contains a comparison of AMTRAN, Culler-Fried, the Lincoln Reckoner and the MAP system. In Smith [1969] a more complete survey and detailed discussion of the following systems is given. Here we will give one reference for each system. Additional references are given in Smith [1969].

1. General Purpose Systems — Operation Oriented

   a. Culler-Fried (see Culler and Fried [1963])

      This system employs pushbutton programming of mathematical problems through a 96-key console with a storage CRT for display of results.

   b. TOC (see Ruyle [1967])

      The TOC system is based on the Culler-Fried system with extensions and modification.

   c. AMTRAN (see Reinfelds et al. [1966])

      AMTRAN is also based on the Culler-Fried system. It employs 224 function buttons and two 5-inch CRTs for alphanumeric and graphical display.

   d. The Lincoln Reckoner (see Stowe et al. [1966])

      The aim of this system is to provide a system whereby a scientist can reduce his laboratory data without programming experience.

   e. OPS-3 (see Greenberger et al. [1965])

      This system includes many standard operators to ease the user's job. A user can also write and use

his own compound operators. Simulation is also a primary use of OPS-3. The system has no graphic capability.

2. General Purpose Systems — Language Oriented

   a. MAP (see Kaplow et al. [1966])

      MAP employs a language for man-machine communication which includes high level functions such as integration and linear least squares analysis.

   b. NAPSS (see Rice and Rosen [1966])

      NAPSS is being developed to "make the computer behave as if it had some of the knowledge, ability and insight of a professional numerical analyst." Polyalgorithms (routines which include decision making code as well as one or more algorithms to perform a certain calculation) are included in the NAPSS language.

   c. POSE (see Schlesinger and Sashkin [1967])

      POSE is similar to NAPSS except instead of poly-algorithms it employs single general-purpose algorithms.

   d. JOSS (see Shaw [1964])

      The JOSS language permits editing, computing and typing of small mathematical problems. No graphics are included in the JOSS system.

3. Special Purpose Systems

a. STATPAC (see Goodenough [1965])

STATPAC is oriented toward statistical operations. A "menu" of the vocabulary is displayed on a CRT for user composition of commands.

b. Marchuk and Yershov (see Marchuk and Yershov [1965])

In this reference a system for on-line solution of differential equations is proposed.

c. Gear's System (see Gear [1966])

This system is for solving ordinary differential equations on-line. Graphs of the results are generated on the teletype consoles.

d. Dixon's System (see Dixon [1967])

This system uses an on-line CRT for interactive use of packaged statistical programs.

e. PEG (see Chapter III of this report)

PEG is an on-line data-fitting system using the method of least squares. The design encourages use by non-programmers as well as programmers.

4. Other Systems of Interest

Several other systems of interest are described in Smith [1969]. Here we will list those systems with references.

a. DIALØG (Cameron et al. [1967])

b. MATHLAB (Engelman [1965])

c. MAGIC PAPER 1 (Clapp and Kain [1963])

d. DISPLAY (Bowman and Lickhalter [1968])

e. Klerer and May (Klerer and May [1964])

f. University of Western Australia (Moore and Erickson [1966])

g. PROMENADE (Ball and Hall [1967]).

Two IBM publications which contain various articles of interest are:

IBM Systems Journal

Volume Seven | Numbers Three and Four | 1968

"Interactive Graphics in Data Processing"

Proceedings

IBM Scientific Computing Symposium

Man-Machine Communication

(held at Thomas J. Watson Research Center In Yorktown Heights, New York on May 3, 4, and 5, 1965)

5. A Language Comparison

Just as we have various levels of programming languages for coding in a batch processing environment we have various levels of communication in interactive systems. Each level has its usefulness for particular classes of problems and programmers. These levels are summarized in Table 1 where for each level in the batch processing environment we give examples of on-line systems with a similar level of communication.

C. Preview of Later Chapters

Chapter II begins with a discussion of the value of interactive graphics. This is followed by a presentation of design criteria which if utilized bring the advantages to an implementation of an on-line system. These design ideas have been incorporated in the PEG system wherever appropriate.

TABLE 1

Programming Language Level Comparison

Batch Processing	On-Line Interactive Processing
Assembly Language	(1) Culler-Fried System (the function buttons provide very basic capabilities) (2) AMTRAN
Procedural Language (e.g. Fortran, Algol, PL/1)	(1) JOSS (provides on-line algebraic computation)
Procedural Language with a library of subroutines (procedures)	(1) MAP (has operators backed by comprehensive algorithms) (2) NAPSS
Using "canned" programs (which only require data)	(1) Dixon's (on-line statistical analysis programs) (2) PEG (given the data, off-line or on-line, PEG then leads a user through the data-fitting process)

In Chapter III is a general description of the on-line data-fitting system, PEG, and its capabilities. A flowchart shows the modular structure of PEG and photographs of the 2250 screen during operation illustrate its appearance to a user. A discussion of how to code and use user defined functions concludes Chapter III.

The numerical methods implemented in PEG are discussed in Chapter IV. Where appropriate, the weaknesses of the methods are pointed out and alternative methods are suggested. A discussion of the program organization and how alternative or additional methods can be added to PEG is given in Appendix C.

Chapter V contains a discussion of two minimization methods. An interactive minimizer that uses a method of Gauss-Seidel type and an implementation of the "direct search" method are both available in PEG for use on non-linear least squares problems. Other methods could be added or substituted as discussed in Appendix C.

Several examples of the use of PEG are given in Chapter VI. Two examples illustrate the usefulness of the user defined function option. A third example shows the use of the built-in functions. Finally, a discussion of the least squares fitting of an ellipse is given. The ellipse problem illustrates the use of PEG with functions expressed as implicit functions, $f(x,y) = 0$.

In Chapter VII a description of an interactive program to determine parameter values by the maximum likelihood method is given. This is a data-fitting program, separate from PEG, which employs the maximum likelihood method instead of least squares. The program includes the capability of interactive examination of the results by means of contour plots.

Chapter VIII contains a summary of the work presented in this dissertation. Some conclusions, drawn of the basis of this work are also given. Finally, several unsolved problems are mentioned and possible extensions to the work in on-line numerical problem solving are discussed.

CHAPTER II

DESIGNING AN INTERACTIVE SYSTEM

The first section of this chapter contains a discussion of some of the advantages accrued by working in an on-line interactive environment with graphical capability. The remaining sections present design criteria which bring the advantages to an implementation of an on-line system.

A. The Value of Interactive Graphics

The following list gives some of the advantages of an on-line or interactive computing system with the additional capability of graphical display, such as that provided by a cathode ray tube display unit with lightpen and keyboard control. These advantages are amplified and exemplified in what follows.

1. speed (instant turnaround)
2. immediate graphical display
3. simple control of calculations
4. various attacks on a problem
5. for production problems — choice of method

1. Speed (Instant Turnaround)

One of the most obvious advantages of interaction to a computer user who has used a batch processing environment, with its often long turnaround times, is simply that he can obtain results quickly. In fact a system with instant (a few seconds to a few minutes) turnaround is essentially an interactive system.

Consider an installation with an on-line system. Assume that installation has, as part of the computer's on-line library, interactive programs to solve certain standard problems. Take the case of a physicist who comes to the computer center wanting to find the roots of a fourth degree polynomial for

which he knows the coefficients. The fastest way to obtain those roots in the batch processing shop was to "pull from the shelf" a program which reads degree and coefficients for a polynomial and computes the roots, punch the coefficients in the proper format, submit the job (with appropriate control cards) to the computer and wait for the results. In the shop with an interactive polynomial rootfinder, however, the roots can be obtained much faster. The procedure would consist of logging on the system, calling the rootfinder routine, and entering the degree and coefficients as requested by the computer program. Then after only a second or so the roots would be printed. Many of the steps involved in the two methods are equivalent (see Table 2), but the on-line method puts the computer at the physicist's fingertips where he can obtain results now. In addition, a well-designed interactive system can make a user aware of mistakes (while keying in numbers or instructions) before it is too late, whereas a mispunched control card in a batch processing environment costs one turnaround time.

Table 2 points out several interesting comparisons between batch processing and on-line computing. Three steps in the batch environment do not exist in the on-line environment. In the on-line case all steps can be carried out while sitting at the terminal; probably within five minutes. This allows continuation of work on a problem which required an intermediate computer solution without the loss of a train of thought. Using conventional batch methods requires going from place to place to assemble the deck for submission to the computer, and waiting an hour or so (depending on turnaround parameters) for results. This can be quite discouraging to progress on a theoretical calculation which requires a computer solution at an intermediate stage.

TABLE 2

A Comparison of Problem Solution Steps

(Numbered by order of execution)

In a Batch Processing Environment	In an On-Line (Interactive) Environment
1. Look in program library to find appropriate program.	1. Look in program library to find appropriate program.
2. Obtain deck for program (or card(s) which invoke the program from an on-line storage device, e.g. disk)	3. Type in instructions to execute desired program
3. Obtain writeup for program	Not necessary.
4. Prepare data cards according to program writeup (prone to key-punching errors).	4. Type in data as instructed by computer. (Computer will detect errors to some extent.)
5. Prepare system control cards (prone to keypunching errors).	2. Log on computer.
6. Submit job to dispatch.	Not necessary.
7. WAIT	Not necessary.
8. Retrieve deck and listing of output.	5. Log off computer and take listing of results.

The number of mechanical details that need to be looked at in a writeup or memorized is minimized by a well-written user program in an on-line environment. The only information needed is how to log-on (and off) and how to call the particular program. The program can then inform the user what parameters and data should be entered when and in what format. This type of interactive program is well suited to the casual user who wants quick solutions to well defined and well conditioned problems.

2. Immediate Graphical Display

"One picture is worth a thousand words." The graphical display used in this work involved an IBM 2250 display unit with five thousand bytes of memory assigned as a buffer area. With four bytes comprising one word, we have that one picture is worth 1250 words; a 25 percent increase due to modern technology.

The IBM 2250 was the basis for the interactive graphics to be discussed in this paper; however, there are other hardware units that can be used for interaction. The basic concepts do not change with the hardware, only the details of implementation. A teletype could be used as an interactive-graphical terminal with a slow rate of textual output and typewriter plots used for graphical output.

One example of the value of graphical display of results is the following. Consider the data shown in Fig. 2.1 with points closely spaced along the horizontal axis except for the last few points whose abscissas are more widely separated. Figure 2.2 shows a least squares polynomial fit of degree six. The residuals, displayed in Fig. 2.2, show the maximum residual to be a reasonably small number. One might assume on that basis that the sixth degree fit was satisfactory for interpolation. However, Fig. 2.2 convinces one that this sixth degree fit is unsatisfactory especially between the last two points. If we had used spline functions to approximate the same data we could have obtained the third degree (with two joints) fit shown in Fig. 2.3.

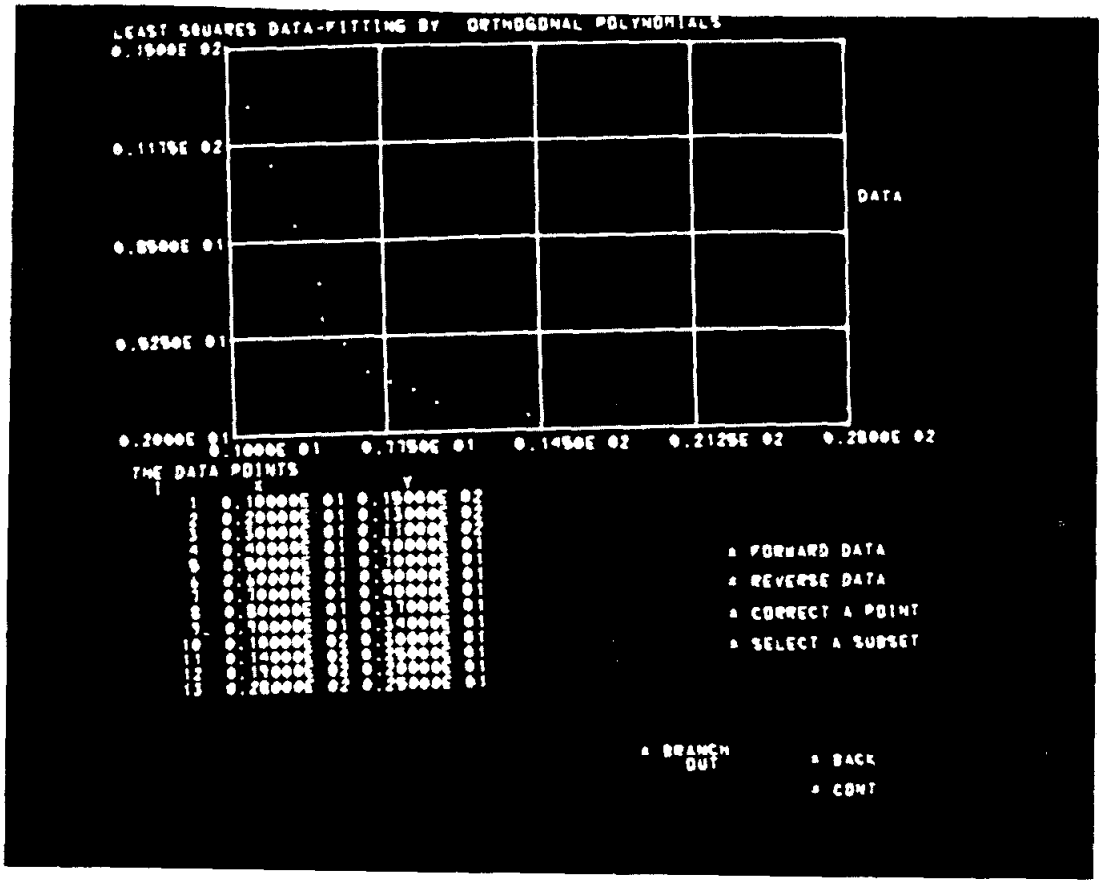

FIG. 2.1--Data with unequally spaced abscissa values.

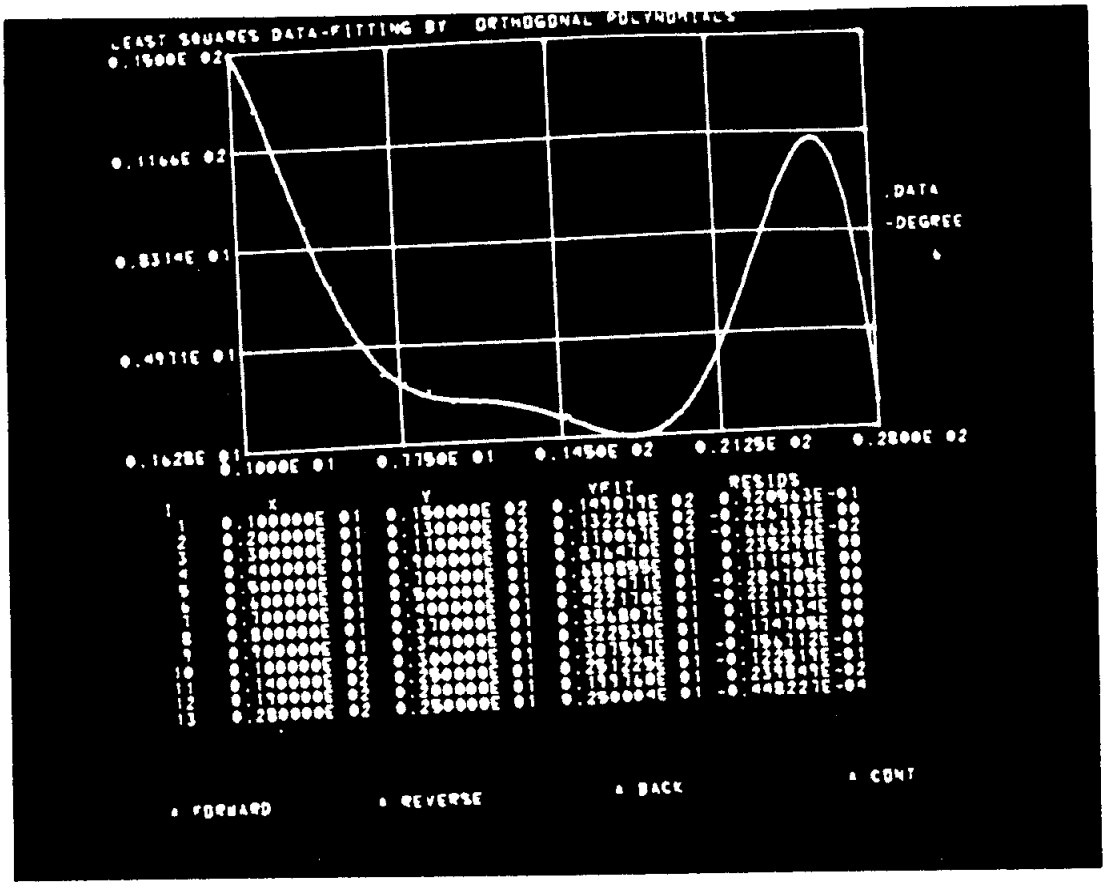

FIG. 2.2--Least squares polynomial fit — degree 6.

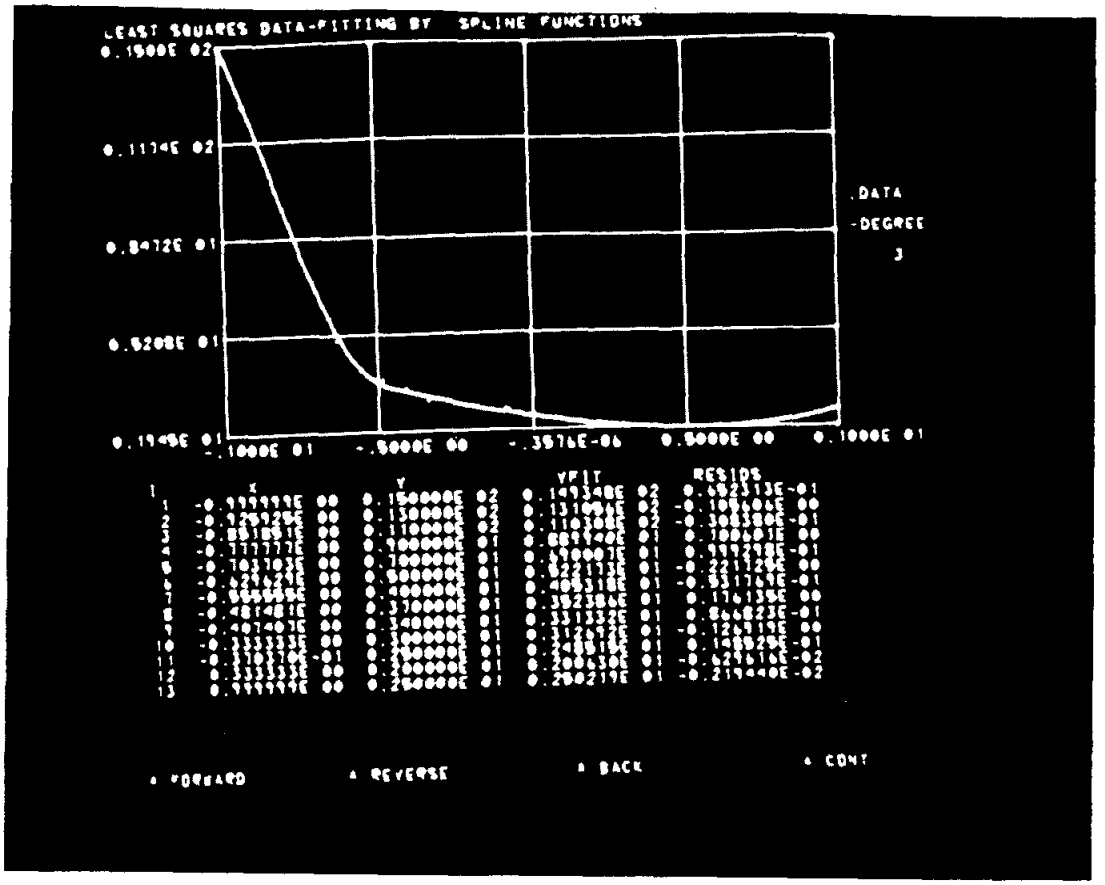

FIG. 2.3--Least squares spline fit — degree 3, 2 joints.

Other examples of the value of graphical display could be given. In data-fitting problems there are often characteristic maxima that are expected to appear in the fitting function. A graph of the computed fit can reveal the presence or absence of the desired character. In general, a graphical representation of a function is more revealing of its properties than a tabular representation of the same function. The old saw could be reworded to say that "one picture is worth a thousand (or 1250) numbers."

3. Simple Control of Calculations

An interactive system using a display terminal with a lightpen or similar device can be programmed to require extremely simple actions by a user. The degree of simplicity depends on the design goals of the system. A system designed to accept textual or FORTRAN-like input statements requires more typing by a user than a system designed to accept only lightpen or special key interrupts and numbers entered through a keyboard. PEG, the data-fitting program to be described herein is of the simple type. All choices such as which function, which numerical method, what mode of display, and what step to execute next are made by lightpen selection of an option displayed on the CRT. In addition to lightpen actions, the only other required user action is the keyboard entry of numbers for specifying degree of fit, and in some cases initial values for parameters.

A lightpen with graphical input capability (not available with an IBM 2250) can make the task of input easier than it is with a passive lightpen. For example, the guesses for joint position when performing a spline fit could be made by pointing the lightpen at the graph rather than by entering numbers through the keyboard as is done in the PEG system.

4. Various Attacks on a Problem

An interactive system with a variety of available options allows a user to try various methods to solve a particular problem during a single session with the computer. Again the data-fitting program of this report exemplifies this aspect of on-line interactive use of a computer.

If one is given a set of raw data for which an approximating function is desired for use in a later calculation, and there is no requirement as to what kind of function should be chosen to represent the data, an interactive data-fitting program with several built-in options for the approximating function could be very effectively employed to obtain the desired result. One would simply approximate the data with various functions of various degree and then make a final choice. The choice would probably depend on two things.

a. Knowledge of the functions and their efficiency of calculation in later work.

b. Information generated by the program for each case (function and degree), such as maximum residual, sum of squares of residuals, and average error.

Other problem areas are also well suited to take advantage of this aspect of an on-line system. Any problem for which there are several known methods of solution could be attacked by using a computer program which allows a choice of method. The problem can then be solved in a variety of ways and the results compared.

5. For Production Problems — Choice of Method

Often a computing problem arises which is repeated many times on different but similar sets of data. This is what is meant by a production problem. From a programmer's viewpoint it means developing a program that handles a typical

data set and then using the program as a "black box" which will process all similar data sets in the desired manner. Again the data-fitting problem provides an example of how an interactive system can be effective in this situation.

Assume, for example, that a physicist plans an experiment that will produce a thousand sets of data, each of which involves an approximation problem. He would like to take a typical data set, determine the function and numerical method which best (best in some sense that he defines) approximate that set of data. He would then use that function and method to process automatically (no human intervention) the remaining nine hundred and ninety-nine sets of data. Using an on-line system such as the system described in this paper, with many varied options, he could interact with the computer as he works with a typical data set until a function and method are found which provide a satisfactory approximation. The chosen method could then be incorporated in a production program to process the remaining data.

B. What Makes an Interactive System Useful?

1. Effectiveness Versus Simplicity

There are two properties of an interactive system which are essential to the usefulness of the system but which compete with each other to some extent. These properties can be labeled:

a. effectiveness and

b. simplicity of use.

By effectiveness is meant the capability to solve "significant" problems in a "reasonable" time. To provide this power there must be generality and flexibility. An example of an effective system for on-line mathematics is the Culler-Fried system (Culler and Fried [1965]) where the intent of the designers was to provide a system with the capability of performing classical mathematical analysis on-line with graphic

output. This system can be used to solve a large variety of problems, namely any that can be stated in terms of the wide range of mathematical capabilities of the system. The Culler-Fried and other similar on-line systems are certainly effective by the above definition. However, this type of system does sacrifice simplicity. For example, to use the Culler-Fried system on a complex problem, one must generate a program by means of a function keyboard which can be utilized on many different "levels" (the meaning of a key depends on the current "level" of operation). This is not meant to say that the Culler-Fried system should not be used. On the contrary, it is very effective in the role for which it was designed, namely, as a tool for fairly sophisticated users.

The simplicity of an interactive system is inversely dependent on the quantity and complexity of actions required by the user. It is easy to see how a requirement of simplicity competes with a desire for effectiveness, since as more generality and flexibility are put in a system, more choices (actions) must be made and more complex information must be given by the user to the system to tailor its power to the particular problem at hand. The problem of resolving the conflict between the desire for generality and flexibility and the desire for simplicity is one of the most significant problems facing the designer of an interactive system.

PEG, the data-fitting system described in Chapter III, resolves to some extent the effectiveness versus simplicity dilemma. The generality of a complete mathematical system is avoided by restricting the scope of problems to a single well-defined area, that of least squares data-fitting. Flexibility within that problem area is maintained by providing the capability of performing least squares data-fitting by a variety of functions. Simplicity is achieved by designing the interaction to require only very simple actions by the user. This is

possible because the problem area has been restricted to include a class of problems among which (hopefully) all possible desired actions by a user can be anticipated. Thus, a list of actions can be presented from which the desired action can be chosen by lightpen.

2. The Man-Machine Interface

When designing an interactive system, a very important consideration is the appearance of the system to the user. If the system is to be used by a variety of people with varied backgrounds and experience it should be designed so that any user can understand what is happening and what is expected of him at each step.

The design of an effective but simple man-machine interface is sometimes referred to as the "human engineering" aspect of the system. Even a special purpose interactive system developed for a select group of users should be designed to have an easily understood and easily handled man-machine interface.

Even seemingly trivial considerations such as the position of a set of "light buttons" on a CRT used for interaction can affect the useability of the system. A right-handed user would be more comfortable selecting items by lightpen from the lower right-hand corner of a CRT than from the upper left-hand corner. In using the upper left corner he would not only have to reach further but he would be blocking his own view of the rest of the CRT as he did so. Another example of something to be considered in an interactive environment is the choice of words to use in a message to the user. Words should be chosen so as to convey the same meaning to all varieties of users and yet the message should be concise enough to keep the users from becoming bored or disgruntled by long messages. These and other considerations which may seem insignificant are very important to the design of a "user-oriented" interactive system.

C. Design Criteria for User-Oriented Interaction

There are several general design principles that should be embodied in a user-oriented interactive system. These principles pertain to CRT based interaction as well as to systems employing teletypes (typewriters) as user consoles. Some of the design principles are:

a. Self explanatory

b. Self helping (helps user)

c. Simple interface with user

d. Interaction by anticipation

e. Optional verbosity

In what follows, these ideas are discussed in some detail. The PEG system has been designed to adhere to these principles to the extent that they apply to a CRT based system.

1. Self-Explanatory Systems

A self-explanatory system is one which displays to the user sufficient explanatory information about the process being carried out to enable him to carry on without reference to some external source of explanation (a manual or textbook, for example). This can be accomplished in most cases by always displaying certain basic items and allowing the user to ask for optional additional tutorial material. For example, the data-fitting program, PEG, allows the choice of spline functions as an approximating function. A user may optionally choose (by lightpen) to have a mathematical description of spline functions displayed (a tutorial display). On the other hand, when the coefficients of a spline fit are displayed they are always accompanied by a short mathematical description of how those coefficients are used to evaluate the computed spline approximation.

2. Self-Helping Systems

A self-helping system provides checking of user inputs to the system and provides reminders or instructional displays on user request. Input items are checked for validity or reasonableness and if appropriate, a diagnostic message is sent to the user and the system awaits his further instructions. For example, PEG checks numbers keyed in by a user to insure that the string of characters entered is a valid number. If not, an audible signal is sent to the user, the input is rejected, and the computer awaits further action by the user. Reminders, or instructional displays are usually made available through the action of some type of "help button" capability. If a person is interacting with an on-line system, and has forgotten the proper procedure for executing some phase of the problem solution, he can (by lightpen, for example) ask for help in the form of a display detailing the actions needed to execute the desired step(s). This can be exemplified by PEG. In PEG, there is a "BRANCH OUT" option on several displays which allows a user to select a tutorial display describing some aspect of the system and then return to where he was.

The proper choice of diagnostic messages can sometimes be difficult. One may want to keep the message short but have it contain enough information so even unsophisticated users can understand it clearly. Some existing compilers and operating systems circumvent this dilemma by giving a number as a message and providing a manual which contains the messages corresponding to each number. This method can be awkward to use, especially in an on-line environment.

The teaching concepts used in the design of programmed learning texts could also be employed by an interactive system. The choice of error message could be made to depend on the type or seriousness of the error committed. Various levels of instructional displays could also be made available to user requests.

PEG avoids the problem of diagnostic messages by employing interaction by anticipation (see paragraph II.C.4). Only those actions that are legitimate to the system are presented for user selection thus avoiding the entry of illegitimate requests requiring diagnostics. The STATPAC system (Goodenough [1965]) and the DIALØG system (Cameron et al. [1967]) also avoid diagnostics in a similar manner. These two systems employ CRT displays to display elements of their languages from which a user picks items to compose a command. However, only those items which will generate a syntactically correct command are displayed thus obviating the need for diagnostics.

3. Simple Interfacing

Simple interfacing with the user is accomplished by ensuring that the actions required of him are short and simple as discussed previously. Another aspect of the interface is the ordering of requests made by the computer. That is, as items are needed from the user one by one, their order should be made "natural" with respect to the problem at hand. To make a simple example, if a date is requested it should be requested as month, day, then year since that is the "natural" order (in European countries the natural order would be day, month, then year).

4. Interaction by Anticipation

The interaction implemented in the data-fitting system of this paper and a desirable feature of any on-line system is interaction by anticipation. By this we mean that all possible desires of a user are anticipated and choices are presented which include all those possibilities. This method allows the user to select a desired option rather than specify that option, thereby allowing, for example, a simple lightpen action rather than requiring entry of an alphabetic

command. With a teletype or typewriter terminal the options could be numbered sequentially as they are printed and the user would select by simply entering the number corresponding to his choice.

Again, interaction by anticipation adds to the simplicity desired in an interactive environment and can be exemplified by the data-fitting program. When the choice of approximating function is to be made a list of available functions is displayed on the CRT as follows. (The role of user functions will be explained in Chapter III.)

LEAST SQUARES DATA-FITTING BY

USER FUNCTION ONE

USER FUNCTION TWO

USER FUNCTION THREE

USER FUNCTION FOUR

USER FUNCTION FIVE

ORTHOGONAL POLYNOMIALS

SPLINE FUNCTIONS

PERIODIC FUNCTIONS

RATIONAL FUNCTIONS

INDICATE CHOICE BY LIGHTPEN

A user would then point the lightpen at the function of his choice and proceed. On the other hand, a system which required that the user specify a function for further use would expect the user to type in the name of an available function (and probably expect him to spell it correctly!).

5. Optional Verbosity

An interactive system featuring optional verbosity can be considered as a double system with two levels of detail in the interface with the user. For the

novice or first time user, the interface will contain detailed explanations and in general be verbose in order to insure that the user understands what he is expected to do and what the computer program is doing. On the other hand a user who is already familiar with the system might choose the mode of operation with few or no explanations and abbreviated communications (terse messages) both to and from the computer program. This optional verbosity is especially desirable for an on-line system which employs a teletype or typewriter as a terminal device. With the relatively slow printing rate of such devices lengthy measages are time consuming and therefore frustrating to an experienced user. With a CRT display, the verbosity option is not an essential requirement since the practically instantaneous display does not consume the time and patience of the user.

The use of optional verbosity is illustrated by MAP (Kaplow et al. [1966] and [1966a]) and OPS-3 (Greenberger et al. [1965]). Both of these systems provide short and long forms for calling compoundoperators such as the solution of simultaneous linear equations. In the short form the name of the operator is followed immediately by the appropriate parameters. When the long form is invoked the system takes control and asks the user for the parameters one by one.

D. Design Criteria for CRT Interaction

Several features which are desirable in an implementation of an interactive system with a CRT display with a lightpen (or equivalent device) are the following:

a. roll-by of tabular information

b. hard copy option

c. continual trace

d. lightpen echo

e. reversible versus non-reversible requests

f. lightpen on a string of characters rather than on a single character

g. display requests one-at-a-time unless order is optional.

Each of these features will be detailed and where the idea is appropriate to more than CRT displays this will be pointed out. All of these ideas have been implemented to some degree in PEG, the on-line data-fitting program described in Chapter III. A related discussion of how to improve man-machine graphical communication is given by Joyce and Cianciolo [1967].

1. Roll-by of Tabular Information

It is often desired to display on the CRT a table of numbers (a matrix) whose size exceeds the capacity of the CRT to display numbers. For example, there may be space to display 30 lines of alphameric information on the screen when a table with 100 rows is to be displayed. The solution is to create a 30 line "window" and roll the table past the window allowing the whole table to be seen 30 rows at a time. This is implemented by providing forward and reverse (or up and down) options to apply to the viewing area. Thus, for example, a lightpen (or key or button) activation of the forward option would change the display from rows n+1 through n+30 to rows n+31 through n+60. The reverse option would simply go back (or down) thirty rows. For display of matrices with more characters to be displayed per row than can be handled by the CRT a similar scheme can be used to move the viewing window to the right or left. The interactive routines described by Dixon [1967] utilize four keys on a function button keyboard to accomplish matrix display. The four buttons are chosen

such that with the viewing window over the matrix the top button moves the viewing window up, the bottom button moves the window down, the left button moves the window to the left and the right button moves the window to the right. Thus any portion of the matrix can be displayed by using the correct sequence of button pushes to position the viewing window.

2. Hard Copy Option

While using an interactive system to solve a problem there is often a graph or a large quantity of output for which the user would like to have hard copy. When using a CRT for interaction there is no associated hard copy. Consequently, a CRT system should have an option to provide hard copy of various quantities on demand. For example, in the data-fitting program, when a graph is displayed on the screen there could be an option which could be selected by lightpen to provide an off-line (line printer) plot of the same graph as that displayed on the CRT. This same principle can also be applied to tables of results and other quantities.

The hard copy option is also appropriate to a system utilizing a teletype or typewriter terminal. In this case the slow speed of the terminal prohibits any large quantity of output on-line so an option to have off-line output of certain quantities is very valuable.

3. Continual Trace

After a session with a CRT oriented interactive system one often would like to review the actions which were taken during the on-line operation. This review is useful in interpreting results as well as for debugging purposes. A trace, or printout indicating what actions were taken in what order and some indication of the results of those actions can be recorded off-line during the on-line interaction with the CRT. It would also be useful to make such a trace available for on-line inspection during the session.

A system with a typewriter-like terminal device provides a trace in the hard copy produced by the typewriter itself. However, even in this environment an off-line trace might prove useful since it would provide a trace of the interaction with any additional hard copy output inserted in its proper chronological position.

4. Lightpen Echo

An important feature to implement when using a lightpen, or similar device, to select items on a CRT is the "echo" of any lightpen selection. The "echo" is created by visibly marking the selected item in some manner. This confirms the acceptance of the lightpen command as well as marking the choice for later reference. Echoed requests can then be executed by a second lightpen action. In paragraph II.D.5 we discuss when the echo feature should be used.

Several methods have been used to accomplish echoing. A special character can be displayed next to the chosen item to mark it, the chosen item can be made brighter or made to blink on or off, or a copy of the selected display can be copied somewhere else on the CRT to confirm the choice. These methods can be summarized as follows:

Methods of lightpen echo

a. display special character adjacent to choice

b. increase intensity of chosen display item

c. make chosen item blink (continually vary intensity)

d. display elsewhere a copy of chosen item

Echoing is not so important for a typewriter-like terminal; however it could prove useful if abbreviations or codes are used as input to the system. In this case the system could type back the non-abbreviated (decoded) choice for user verification.

5. Reversible Versus Non-Reversible Requests

With some types of lightpens, for example, those activated by a foot pedal, it is possible to obtain a lightpen interrupt that was not intended. This is easy to accomplish if the foot-hand-eye coordination is somewhat lacking. For this reason and also to allow mind-changing to some extent, it is important to distinguish between reversible and non-reversible requests to be made by lightpen action. A reversible request is one which can easily (without significant computer utilization) be reversed such as the forward and reverse options on a tabular display (see II. D. 1). A non-reversible request is a request for action which can not be reversed easily such as a request to terminate or to perform a calculation which uses a significant amount of computer time. Non-reversible requests should be acted upon only after echoing (see II. D. 4). On the other hand, a reversible request should be carried out immediately since requiring an echo would only be frustrating to the user.

6. Lightpen on a String of Characters — Not Single Characters

One consideration in the design and implementation of an interactive system with a CRT and lightpen is to make the lightpen sensitive area for each item as large as possible and at the same time ensure the separation of the sensitive areas. This allows a non-precise human operator to execute lightpen commands without the painstaking care needed to place the lightpen on a small target area such as a single point or even a single character. Sometimes the nature of the problem dictates that a single character must be selected by the lightpen; however, if this is not the case, a string of characters makes the interaction simpler and less frustrating from a human engineering standpoint.

As mentioned above, in addition to creating large sensitive areas for lightpen action the separation of these areas one from another is also important to

avoid inadvertent selection of the wrong item. For example, if several choices are presented to a user as lines of text describing them, not only should each line be lightpen sensitive along its whole length, but in addition, the lines should be separated by one or more blank lines to insure easy selection by the user.

7. Display Requests One-at-a-Time

If during execution of an interactive program a sequence of input items or choices by the user is needed by the program and the order of these items is fixed by the nature of the problem, the requests for these items should be displayed singly and in order. This avoids the necessity of displaying ordering information to the user and avoids any chance of entry of items in an incorrect order. If the order of items is not important then requests may be displayed simultaneously and user inputs should be allowed in any order. This question of ordering does not arise in the implementation of a typewriter-like terminal system since the nature of the device dictates that requests to the user are made singly.

E. Human Testing of an Interactive System

The test of an interactive system is whether a new user can use it successfully. To test PEG a person with programming and some mathematical background who had not previously seen PEG in operation volunteered to try the system while someone sat behind him and took notes. These notes are listed in Appendix D. This test pointed to displays which could have more explanations shown on the CRT as well as other ways in which the interaction could be improved. Along with the notes in Appendix D are comments as to how PEG could be changed and/or improved to eliminate the puzzlement of future new users and to make PEG more flexible and more useful.

CHAPTER 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.

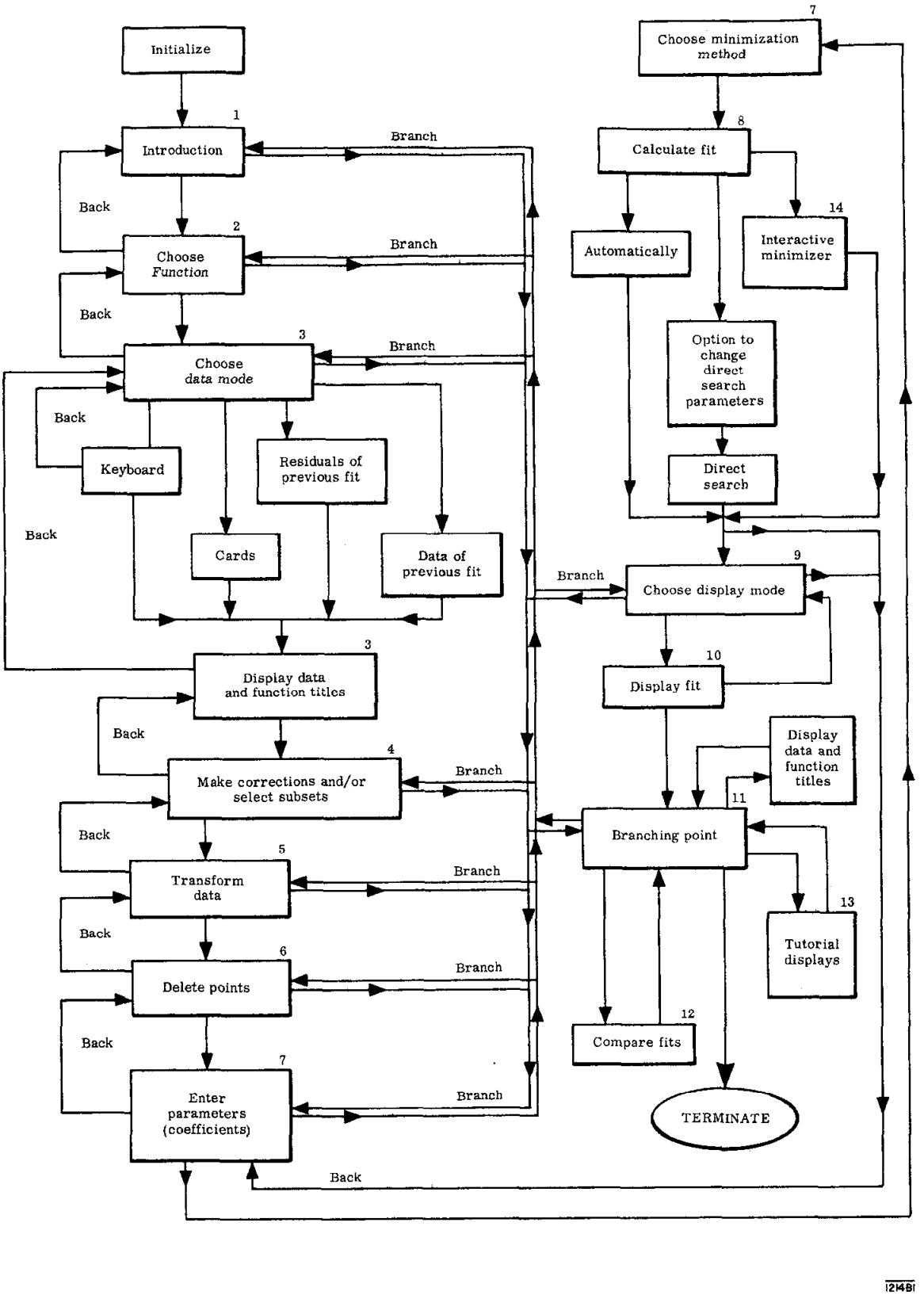

FIG. 3.1--Flowchart — Interactive data-fitting program.

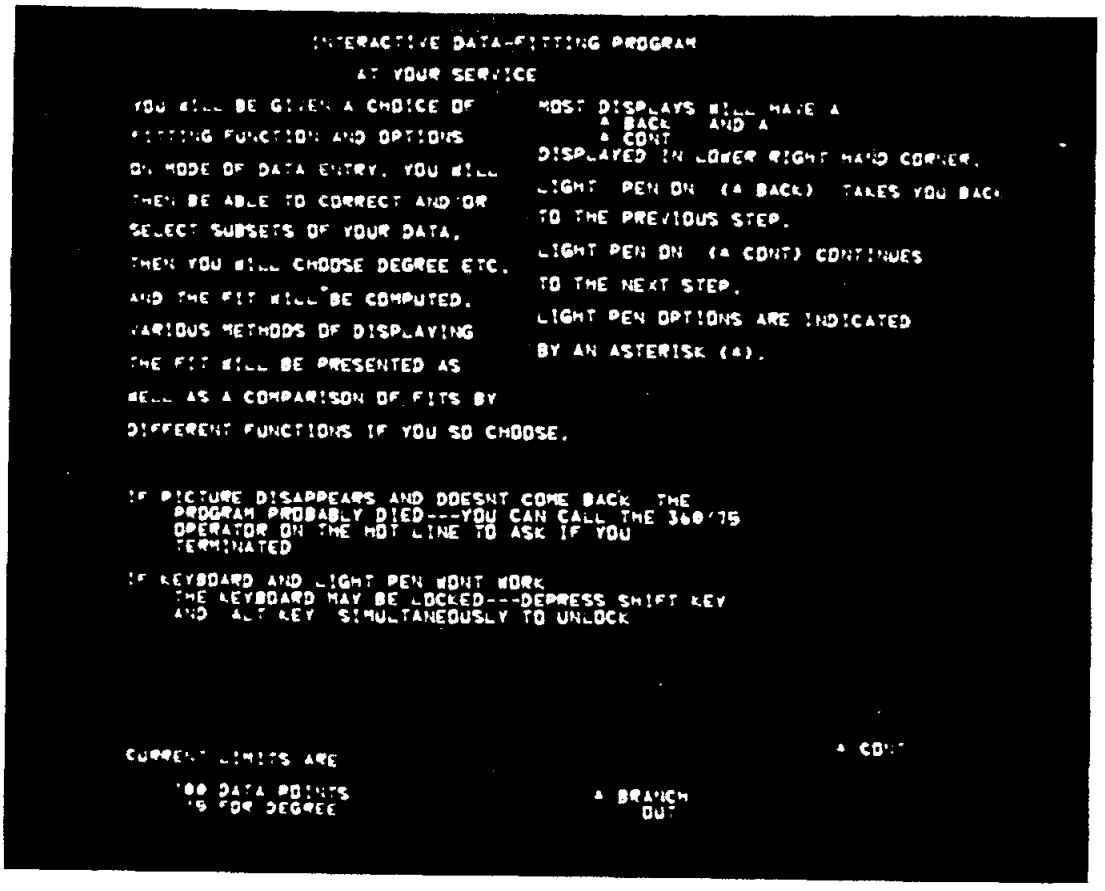

FIG. 3.2--Introductory display.

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.

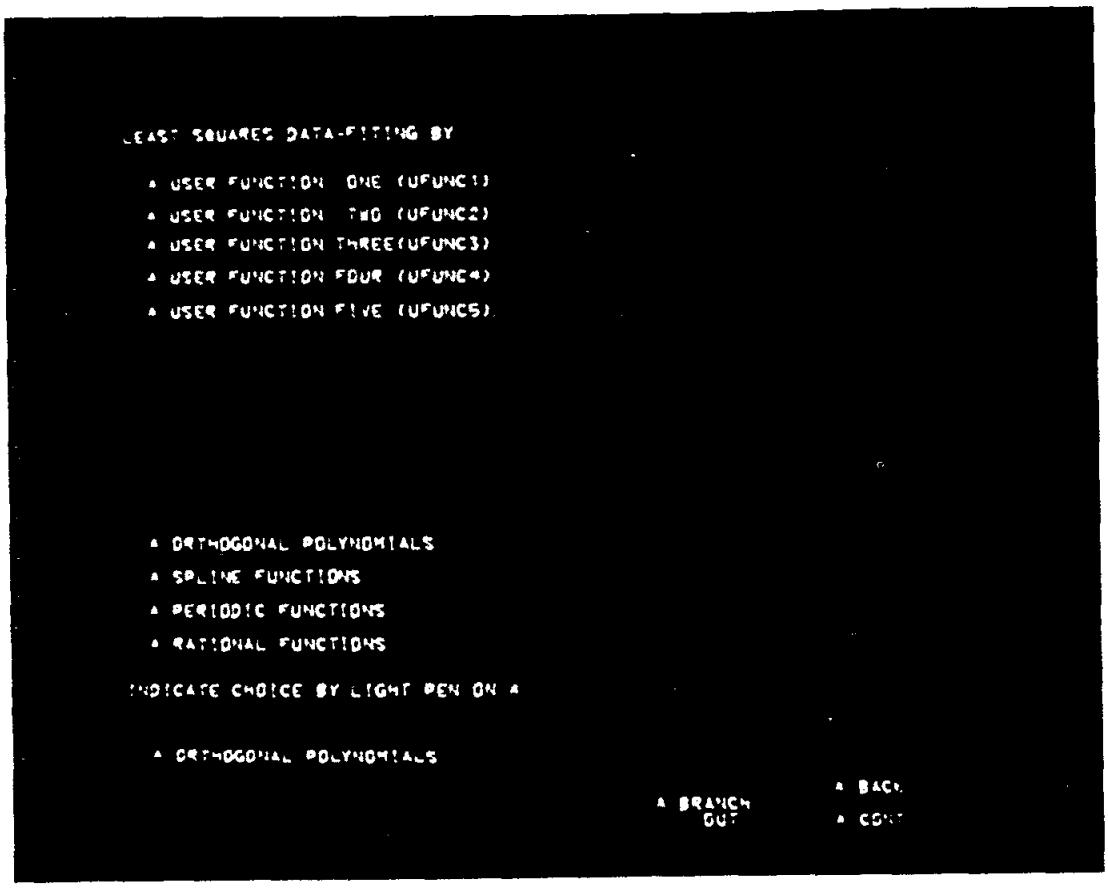

FIG. 3.3--Choosing the function.

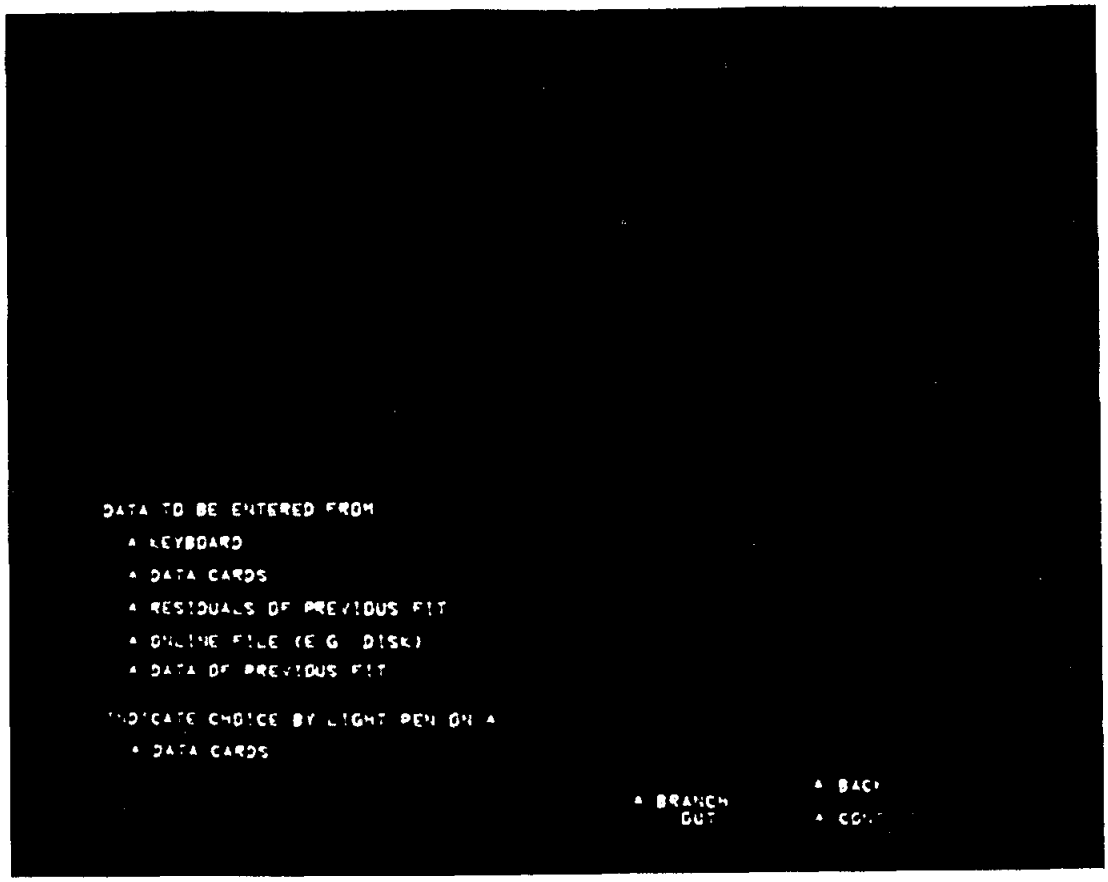

FIG. 3.4--Choosing the data mode.

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

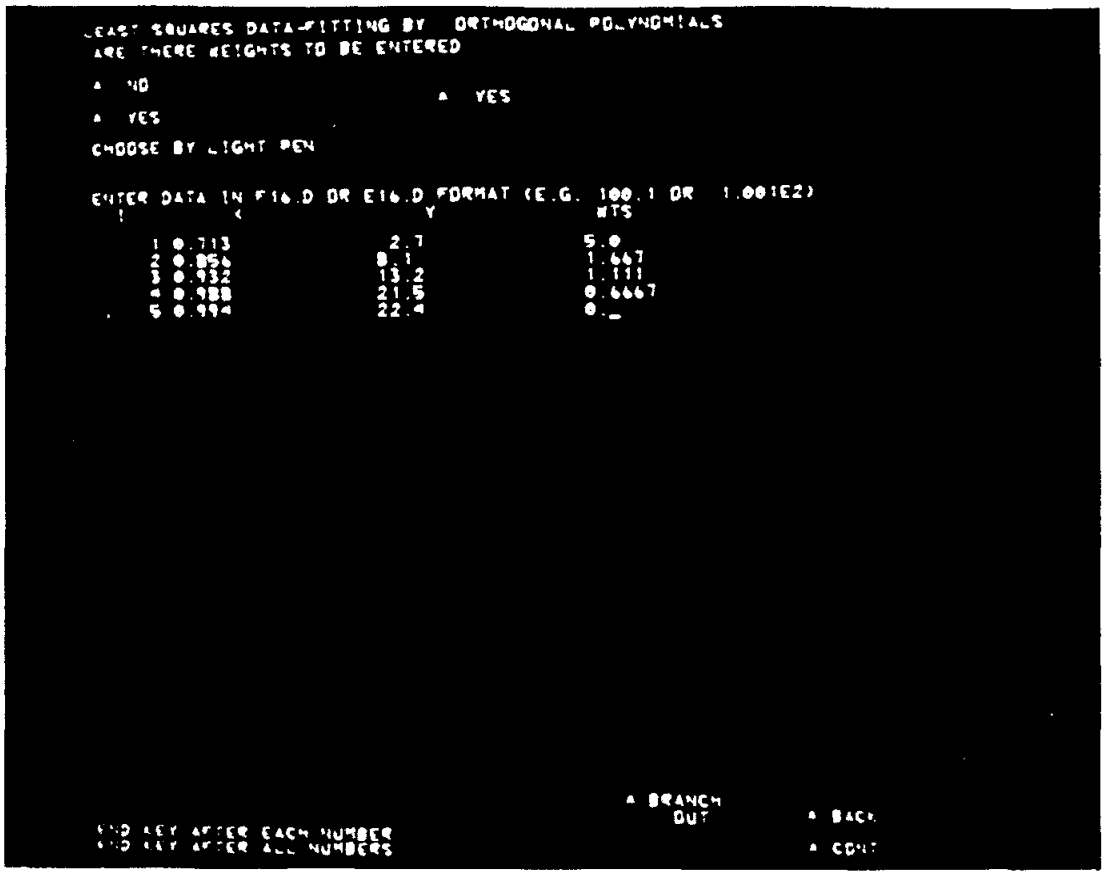

FIG. 3.5--Keyboard entry of data.

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.

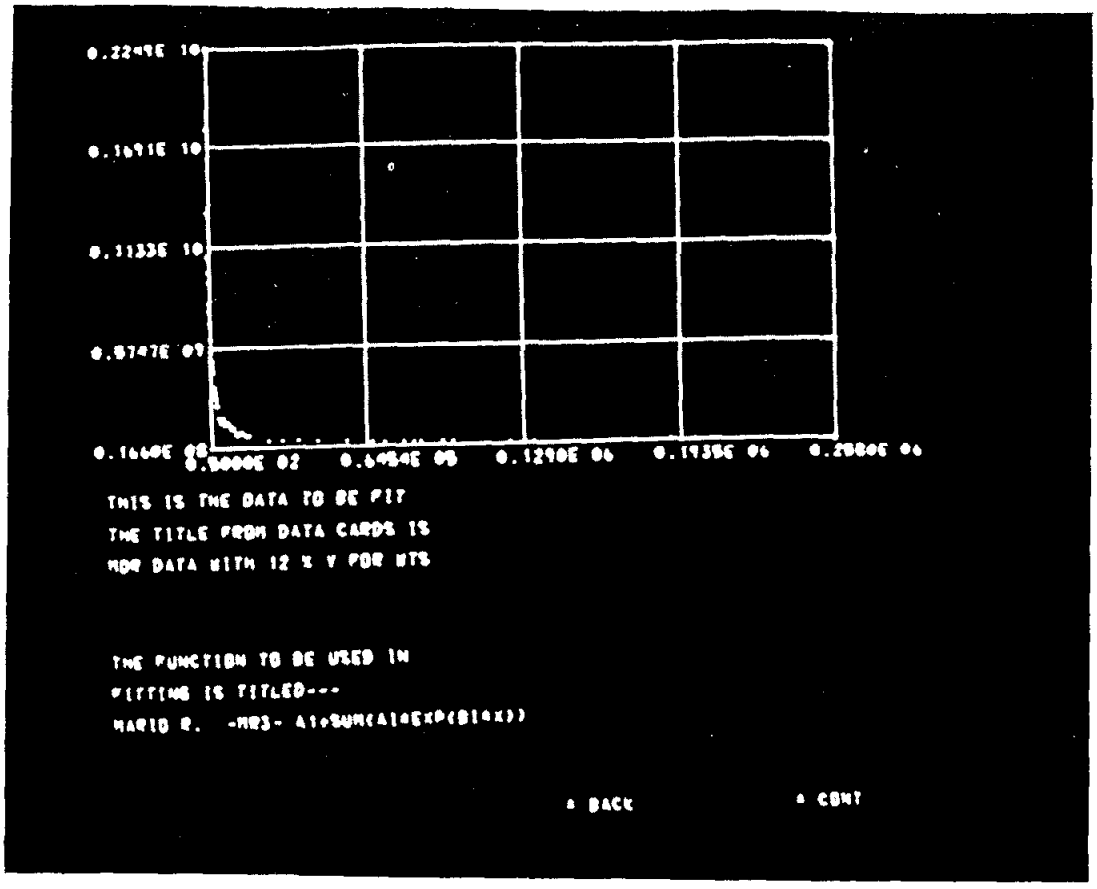

FIG. 3.6--Data and function titles display.

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

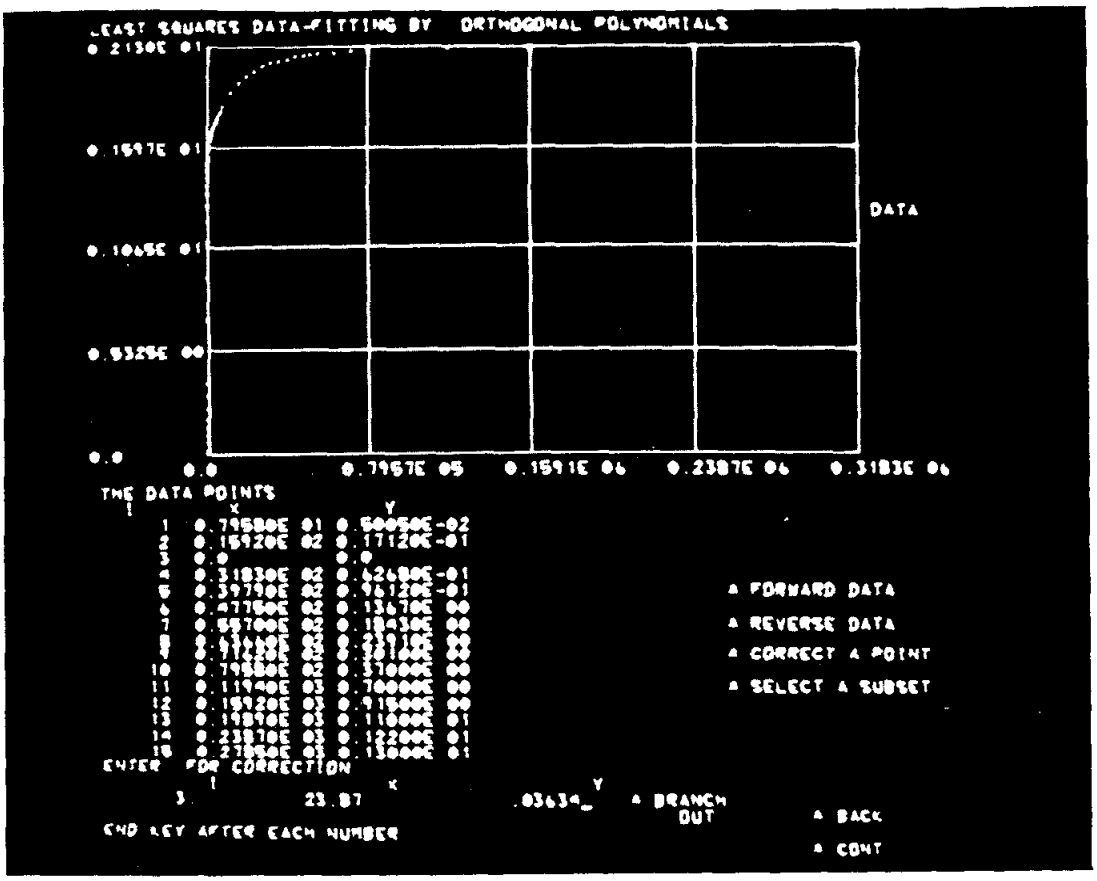

FIG. 3.7--Display for point correction and/or subset selection.

$\{(x_i, y_i)\}_{i=n_i}^{n_j}$ 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

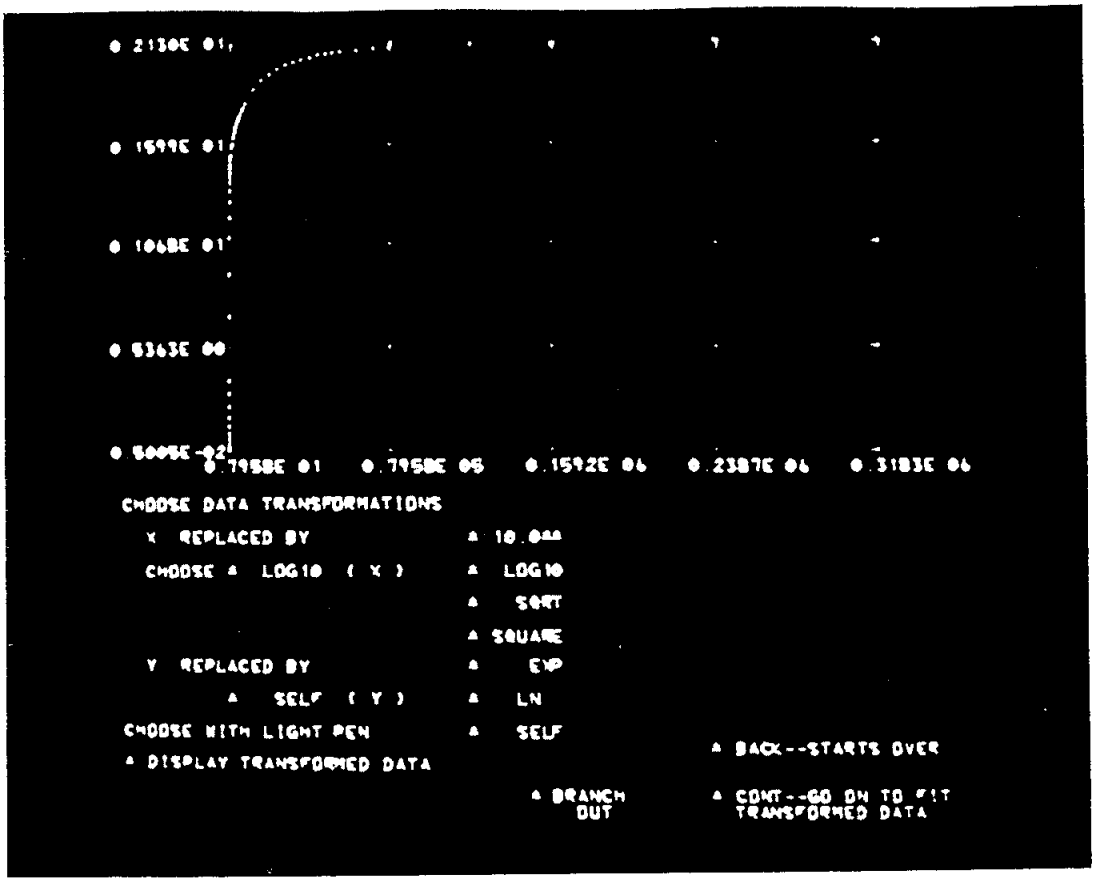

FIG. 3.8--A magnetization curve.

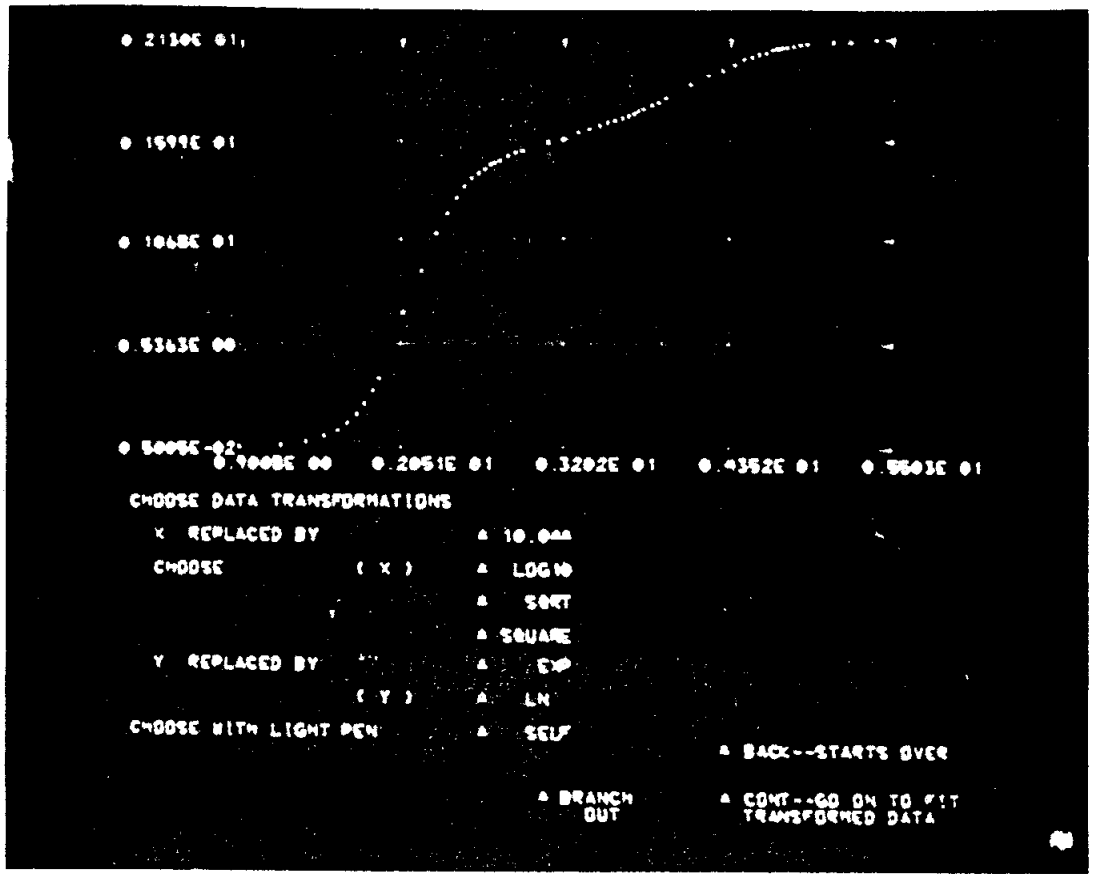

FIG. 3.9--Transformed magnetization curve.

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.

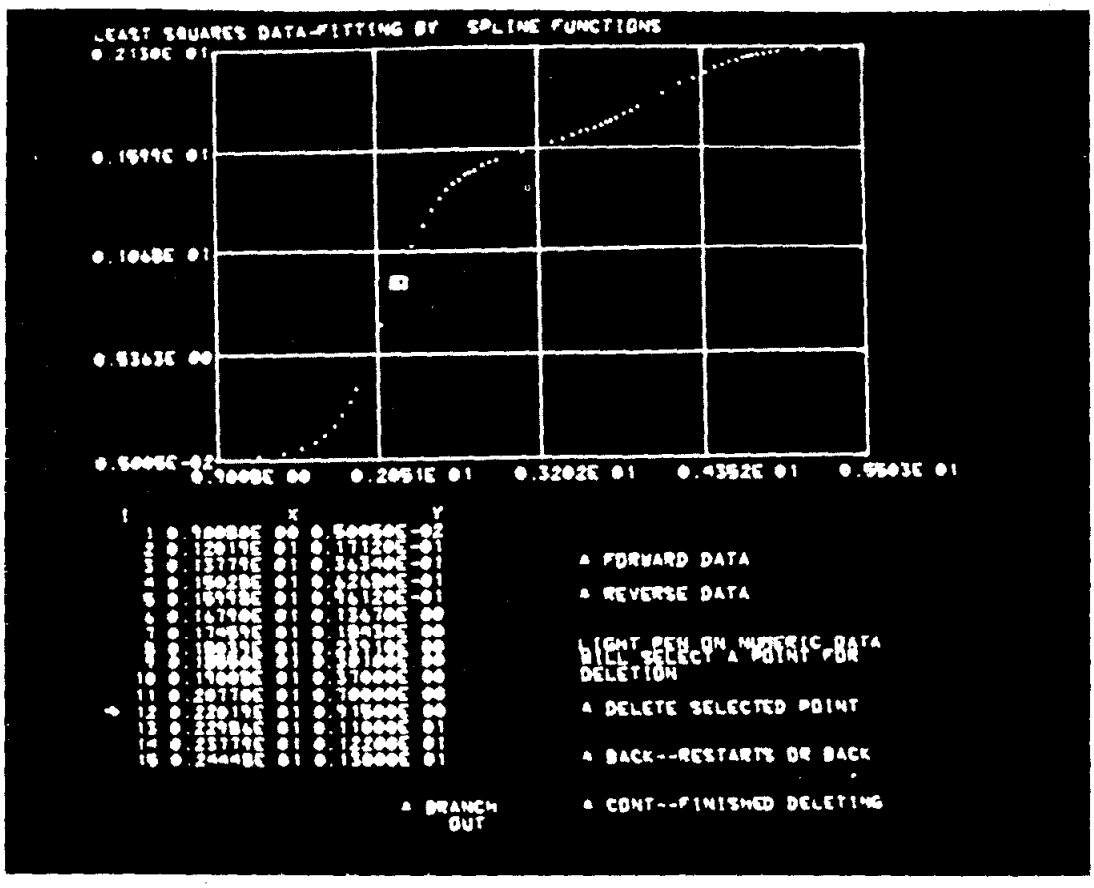

FIG. 3.10--Point deletion.

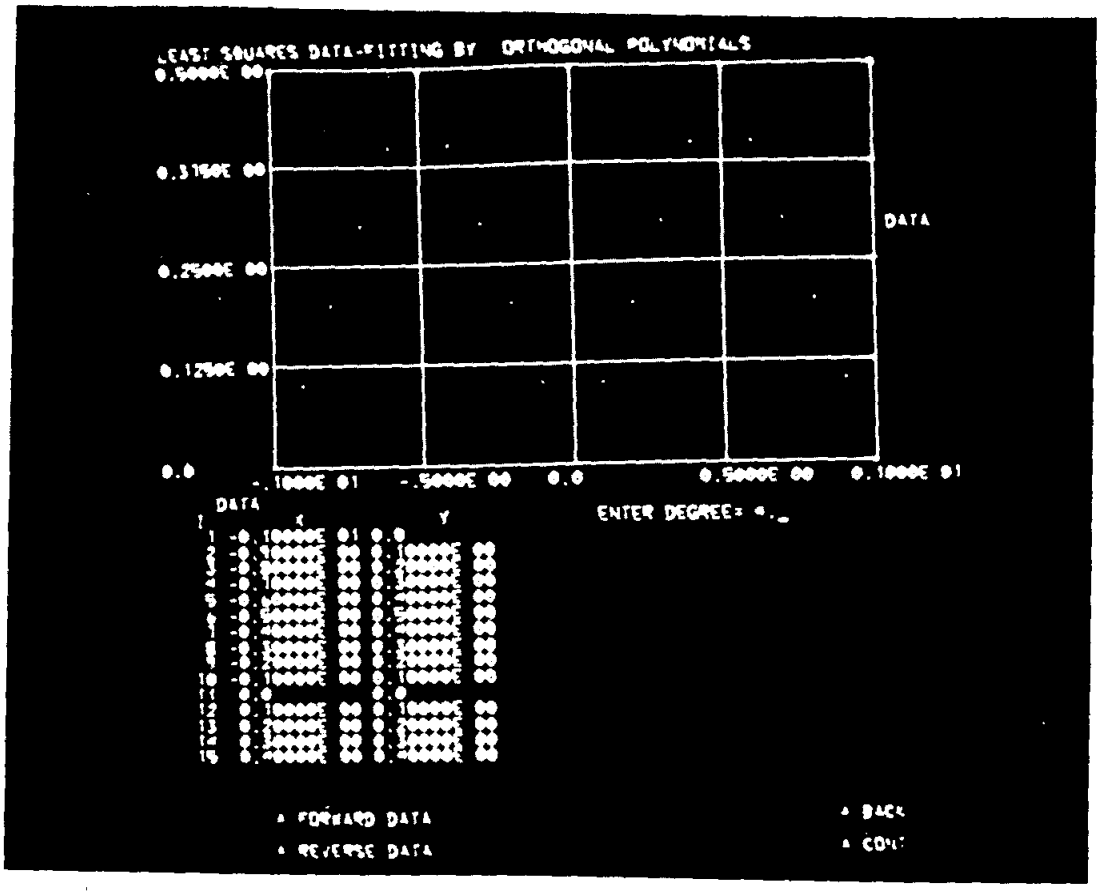

FIG. 3.11--Display for degree entry.

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)

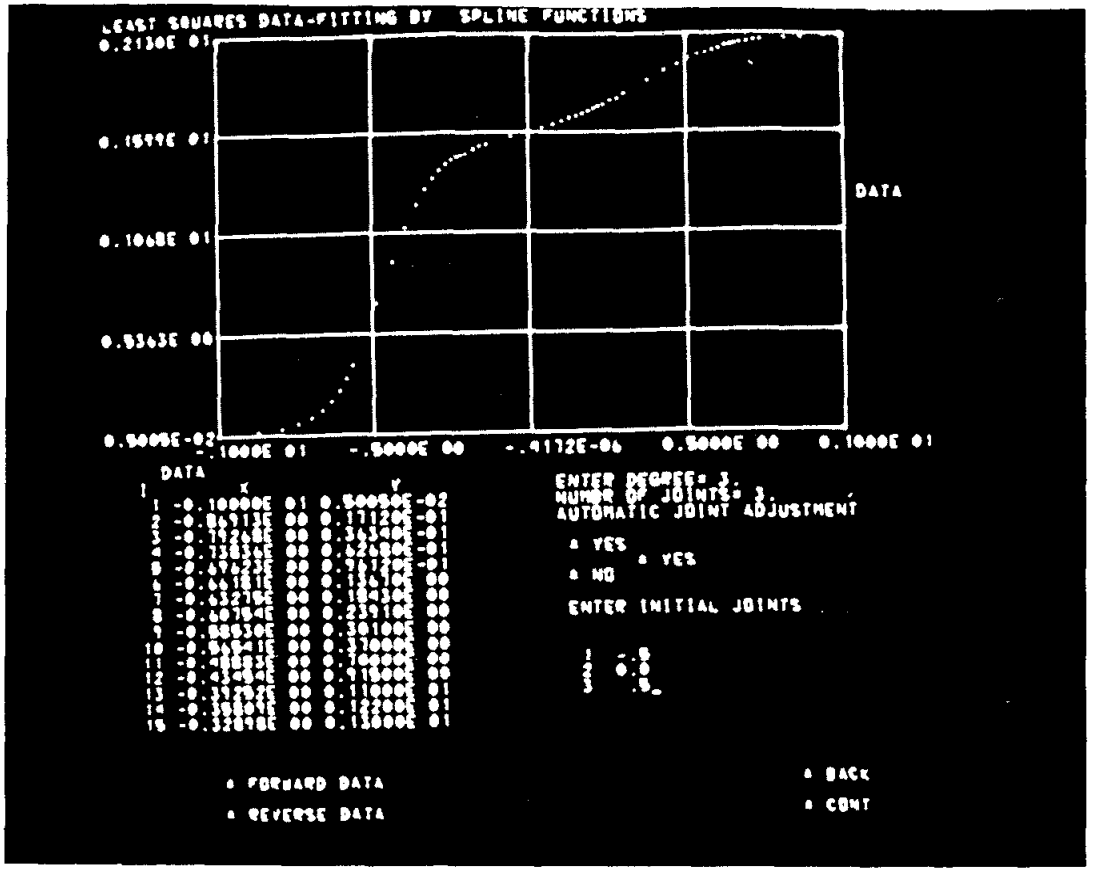

FIG. 3.12--Entering spline parameters.

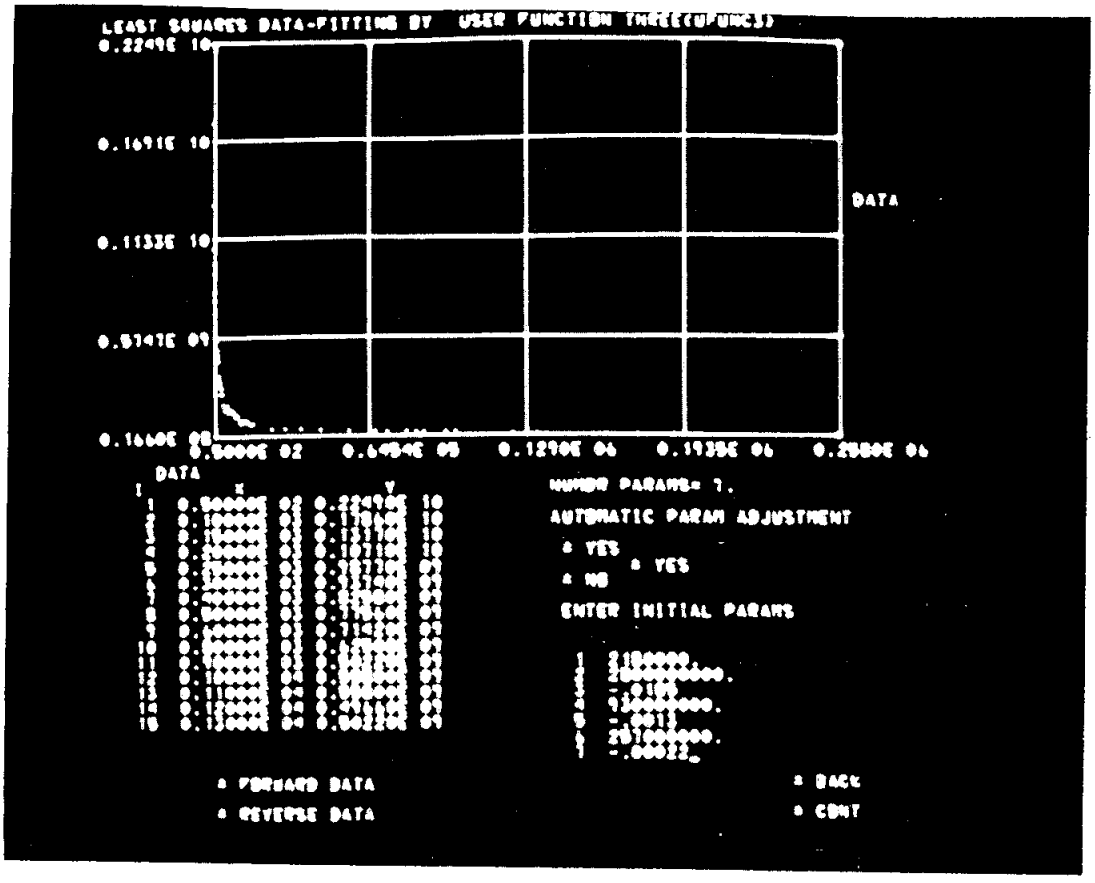

FIG. 3.13--Entering user function parameters.

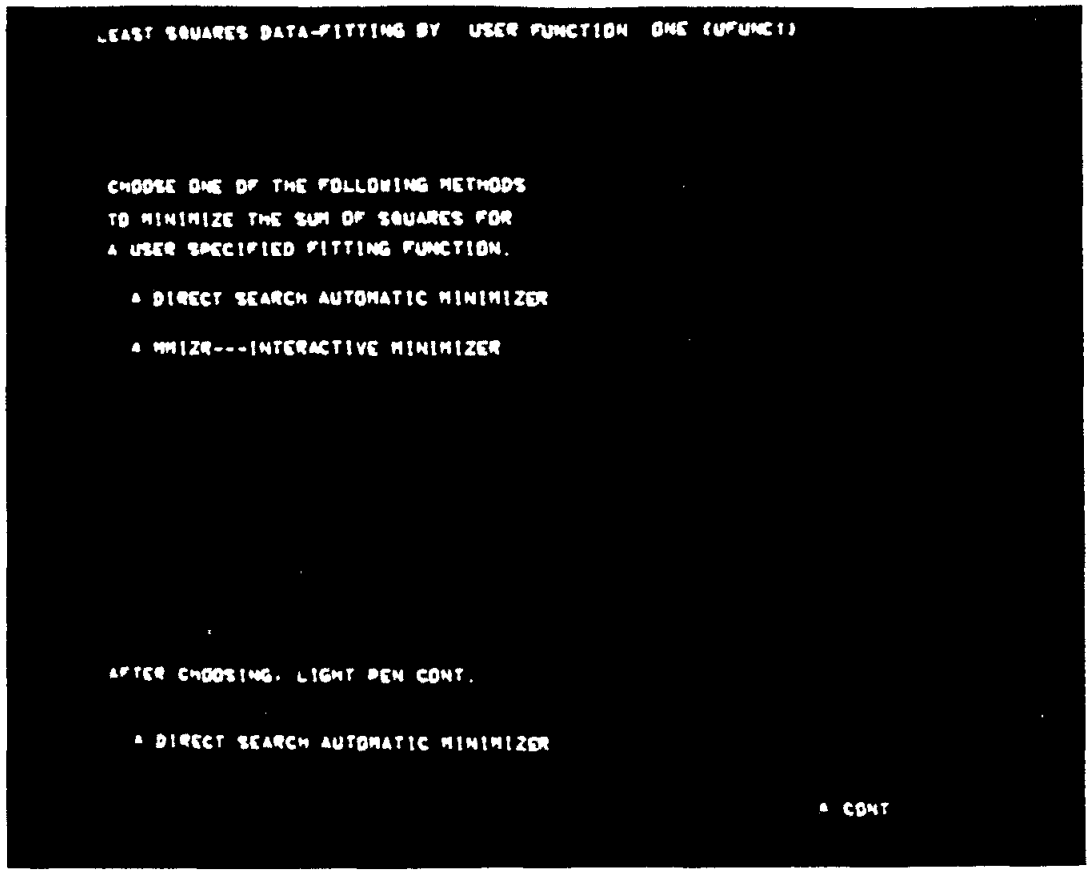

FIG. 3.14--Choosing a minimization method.

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

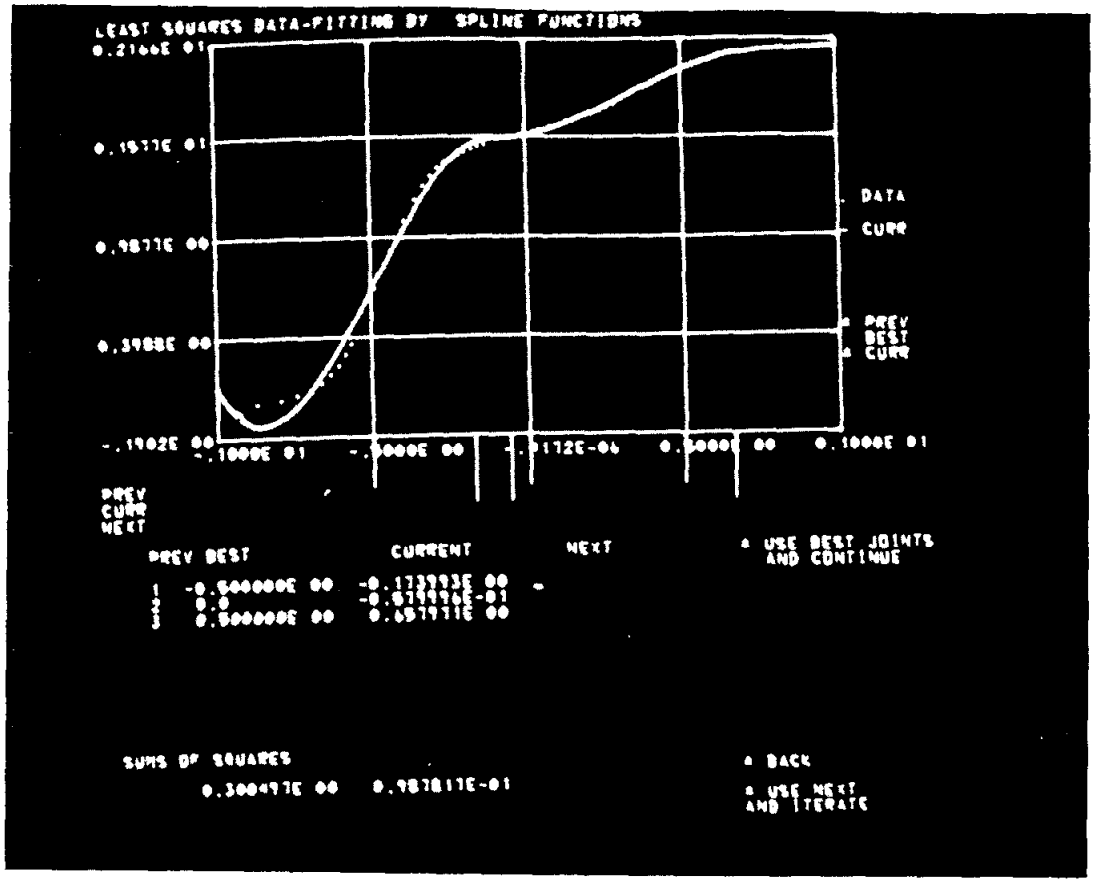

FIG. 3.15--Display of spline fit and selection of new joints.

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

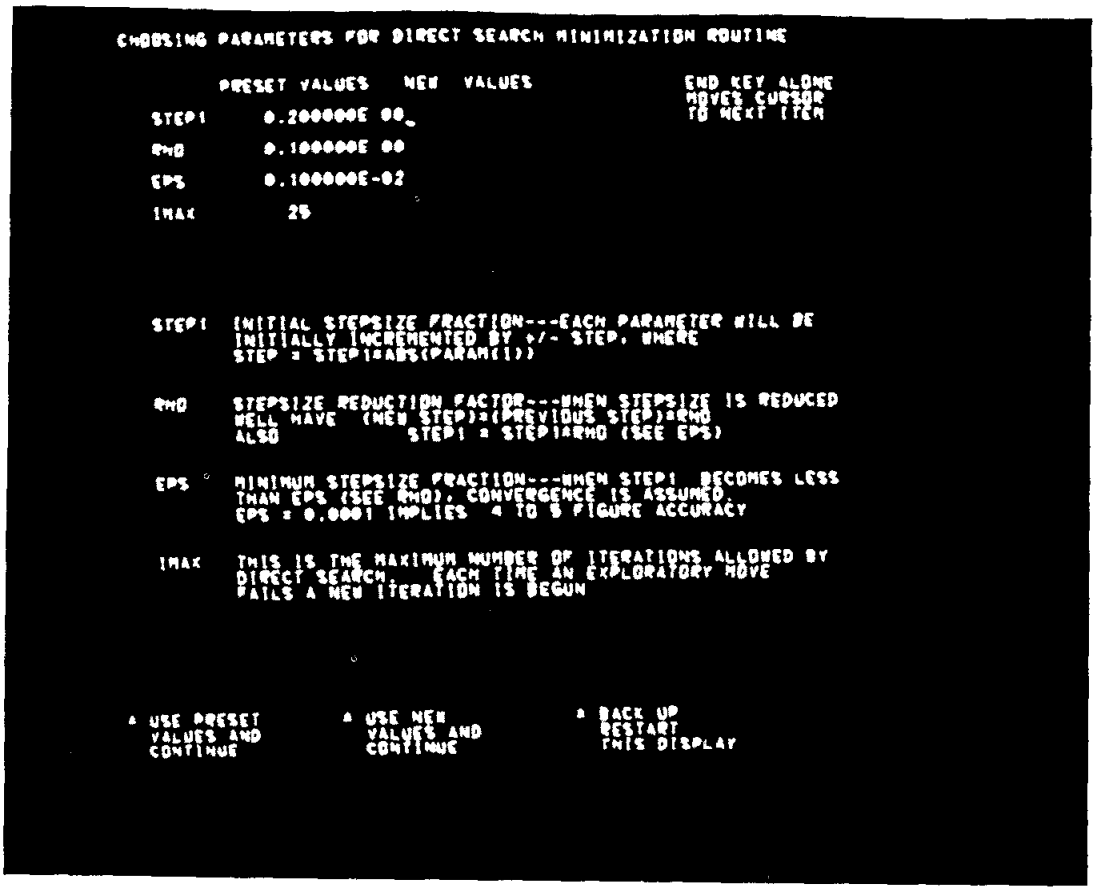

FIG. 3.16--Display allowing change of direct search parameters.

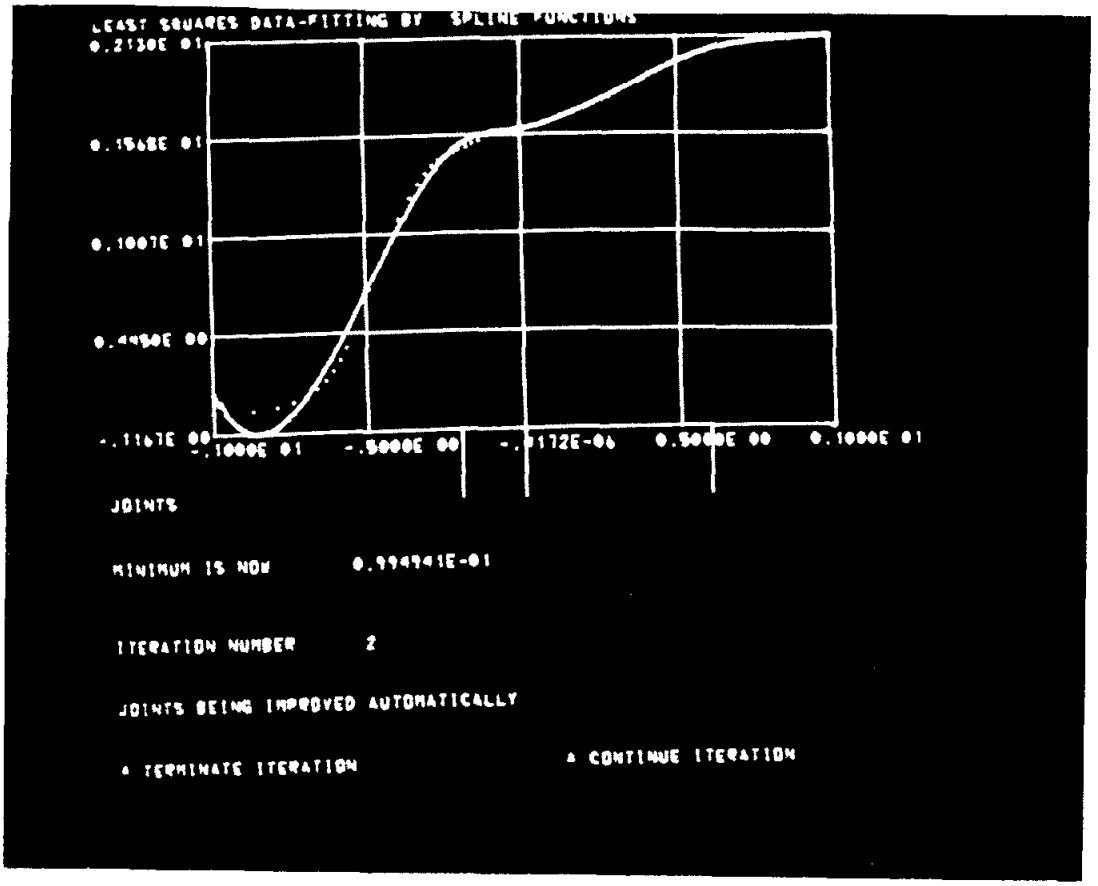

FIG. 3.17--Direct search spline fitting.

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.

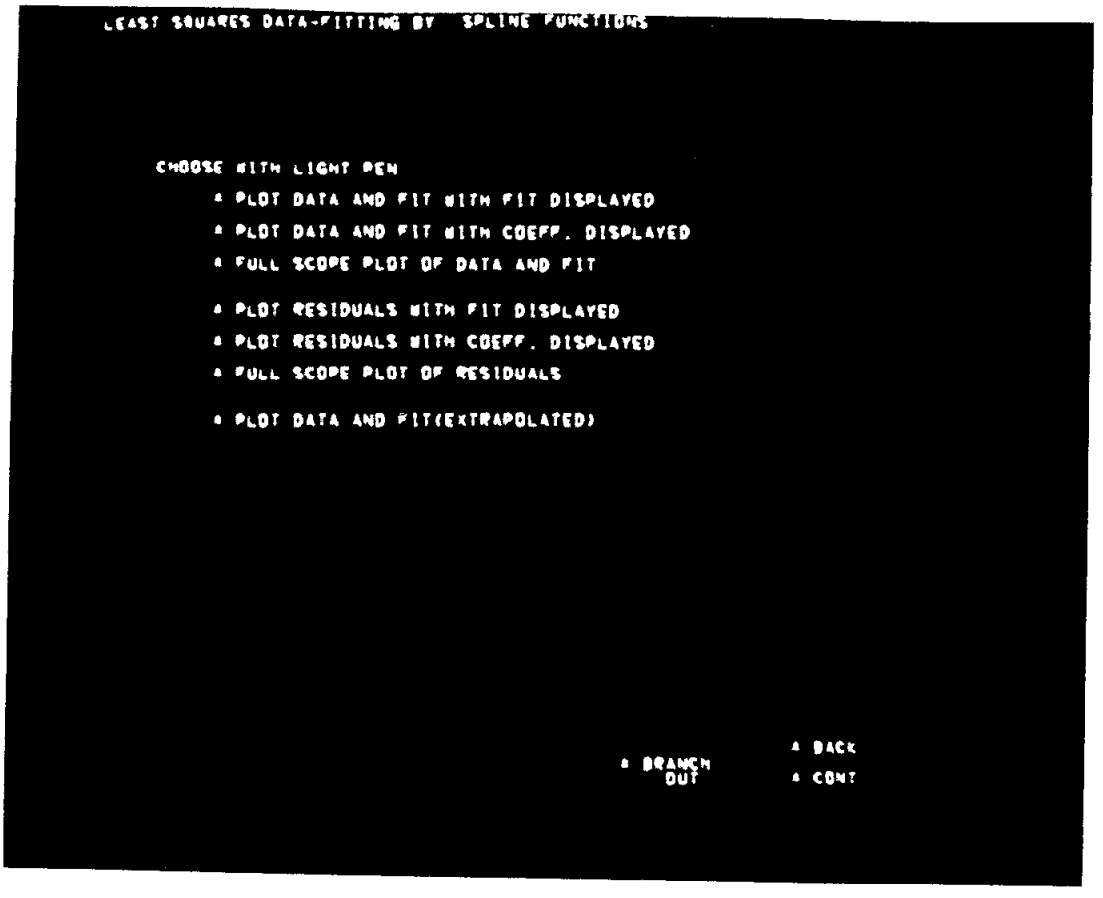

FIG. 3.18--Choosing display mode.

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.

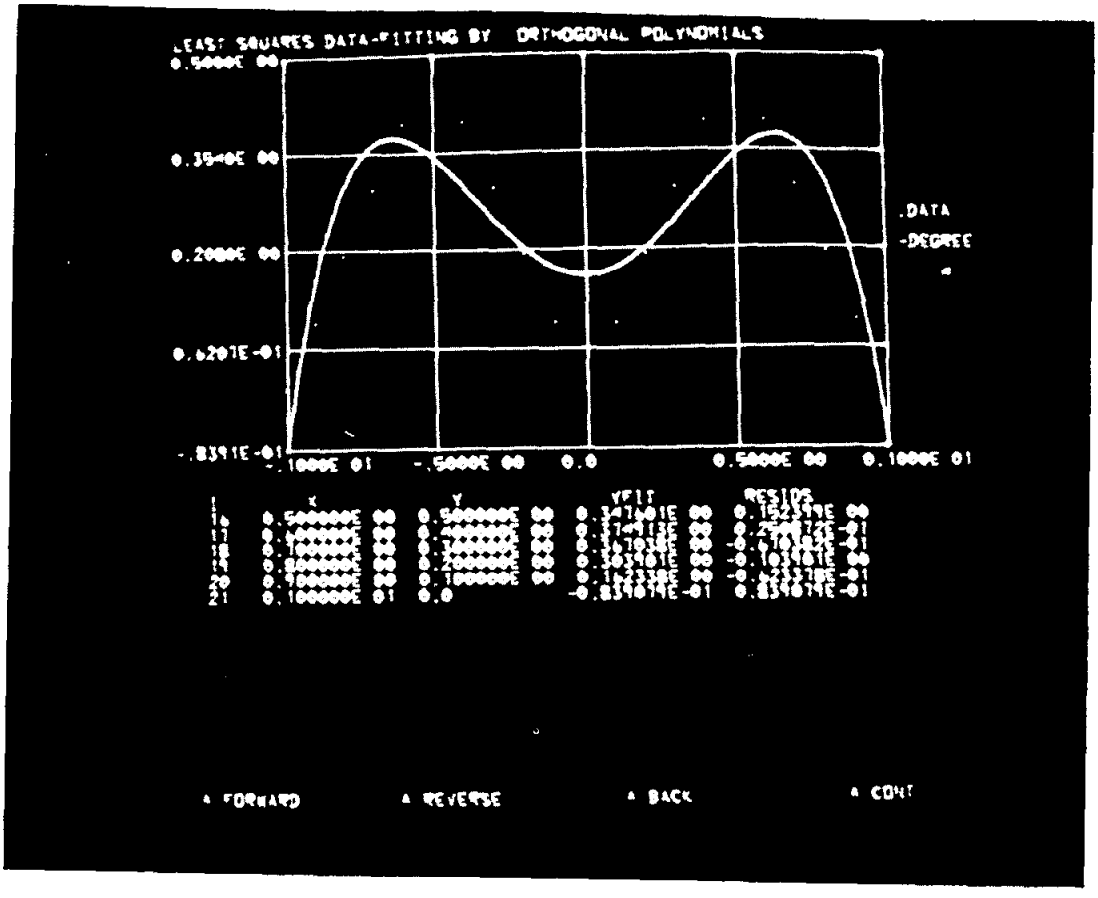

FIG. 3.19--Data and fit with fit displayed.

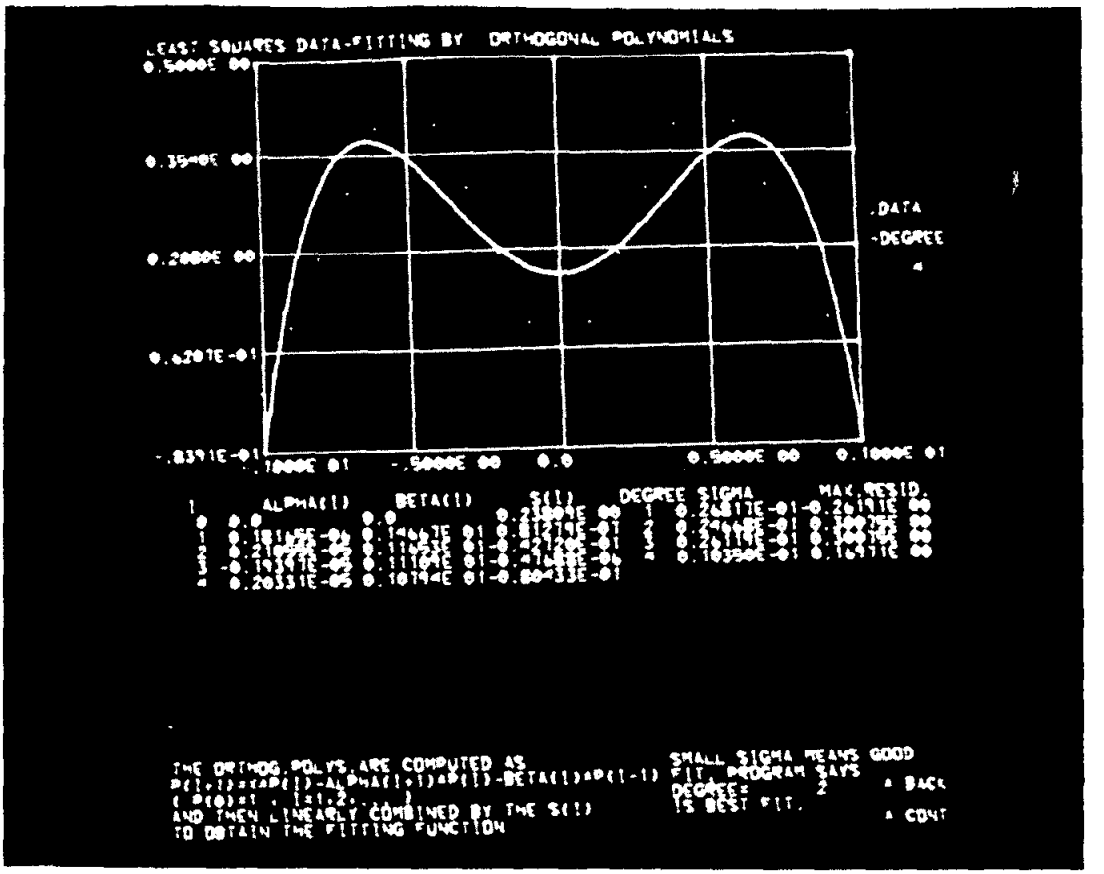

FIG. 3.20--Data and fit with orthogonal polynomial coefficients.

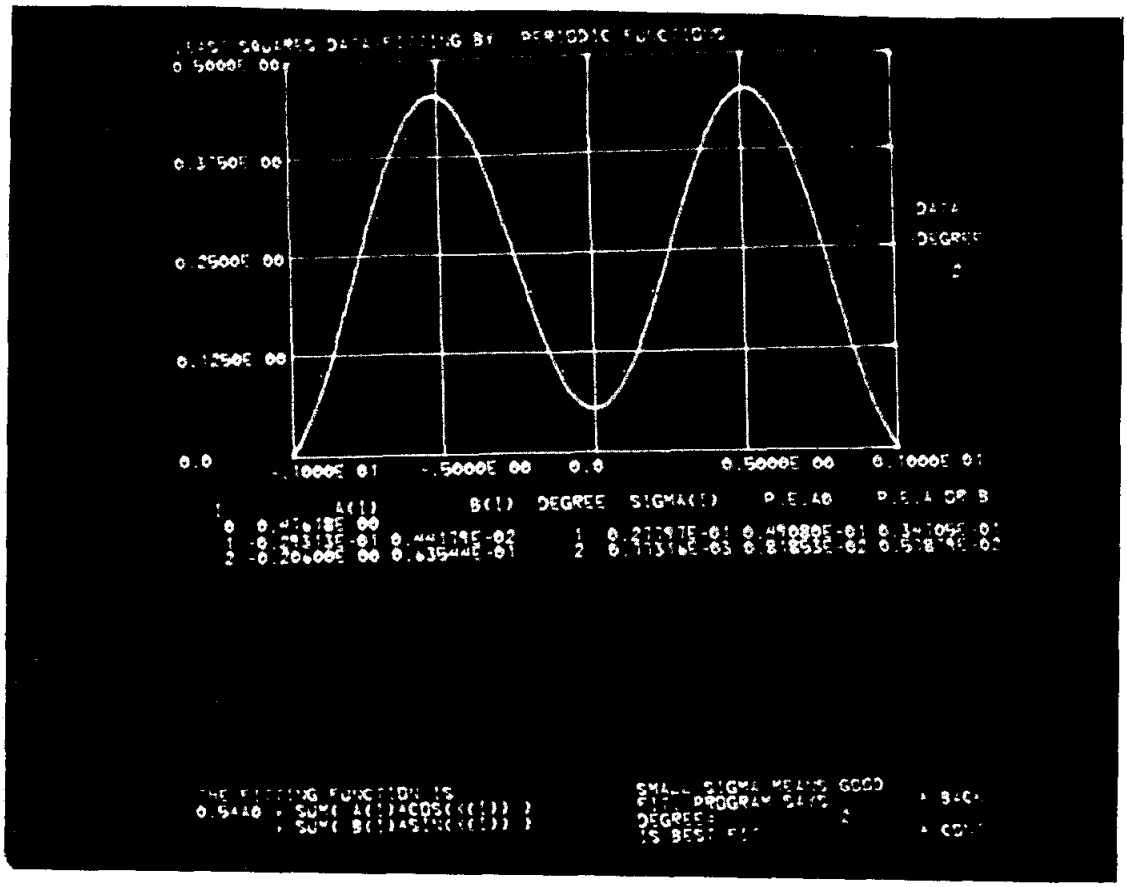

FIG. 3.21--Data and fit with Fourier approximation coefficients.

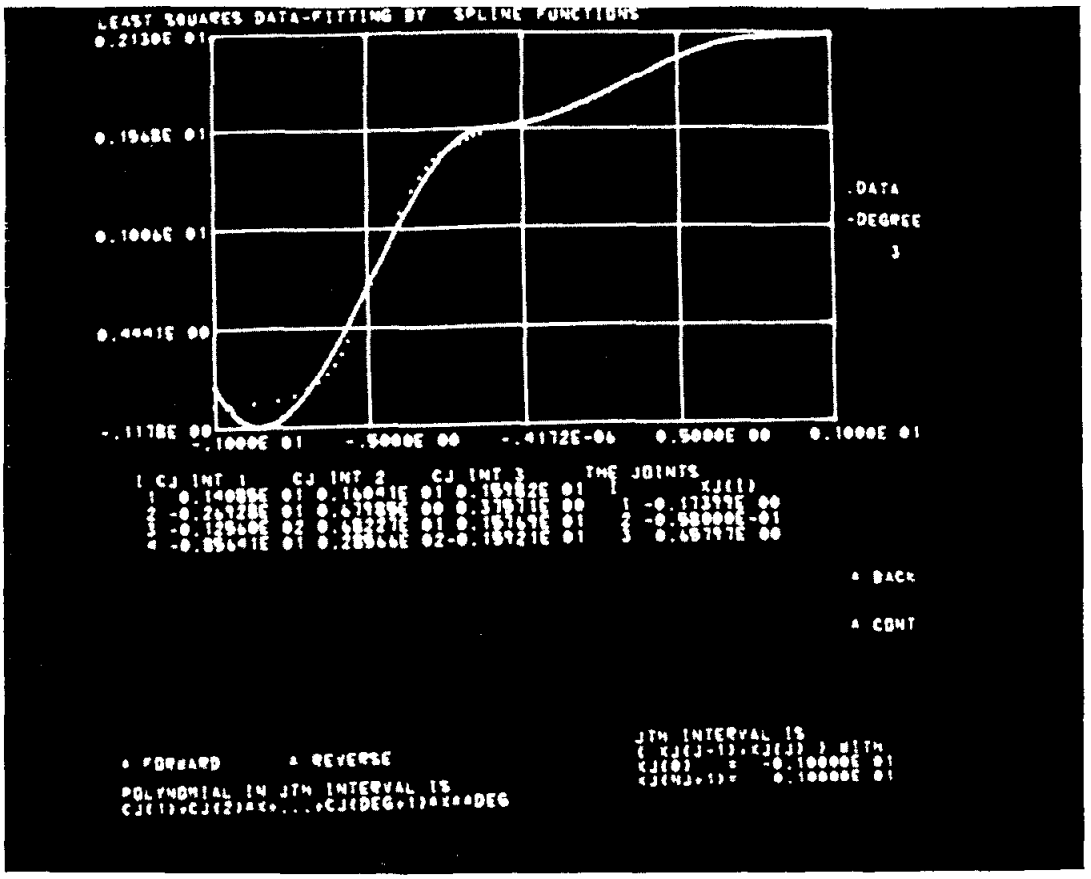

FIG. 3.22--Data and fit with spline function coefficients and joints.

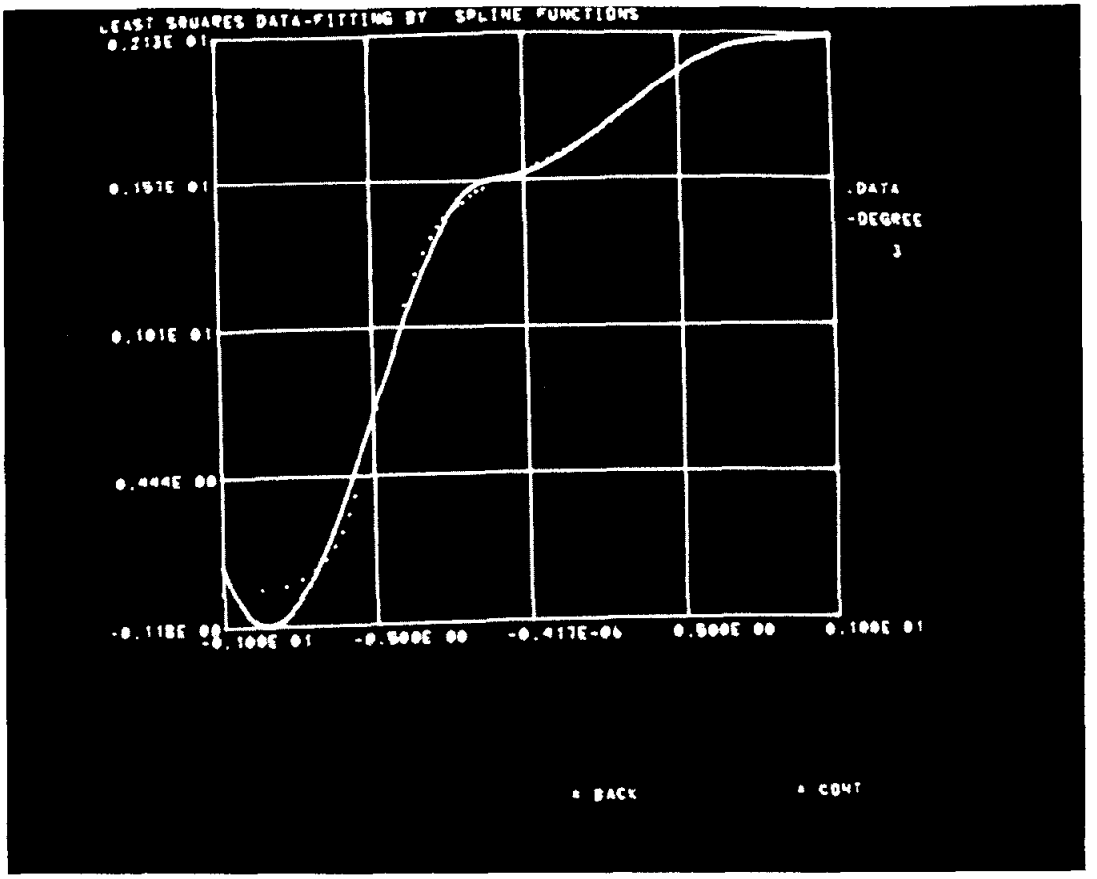

FIG. 3.23--Full scope plot of data and fit.

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

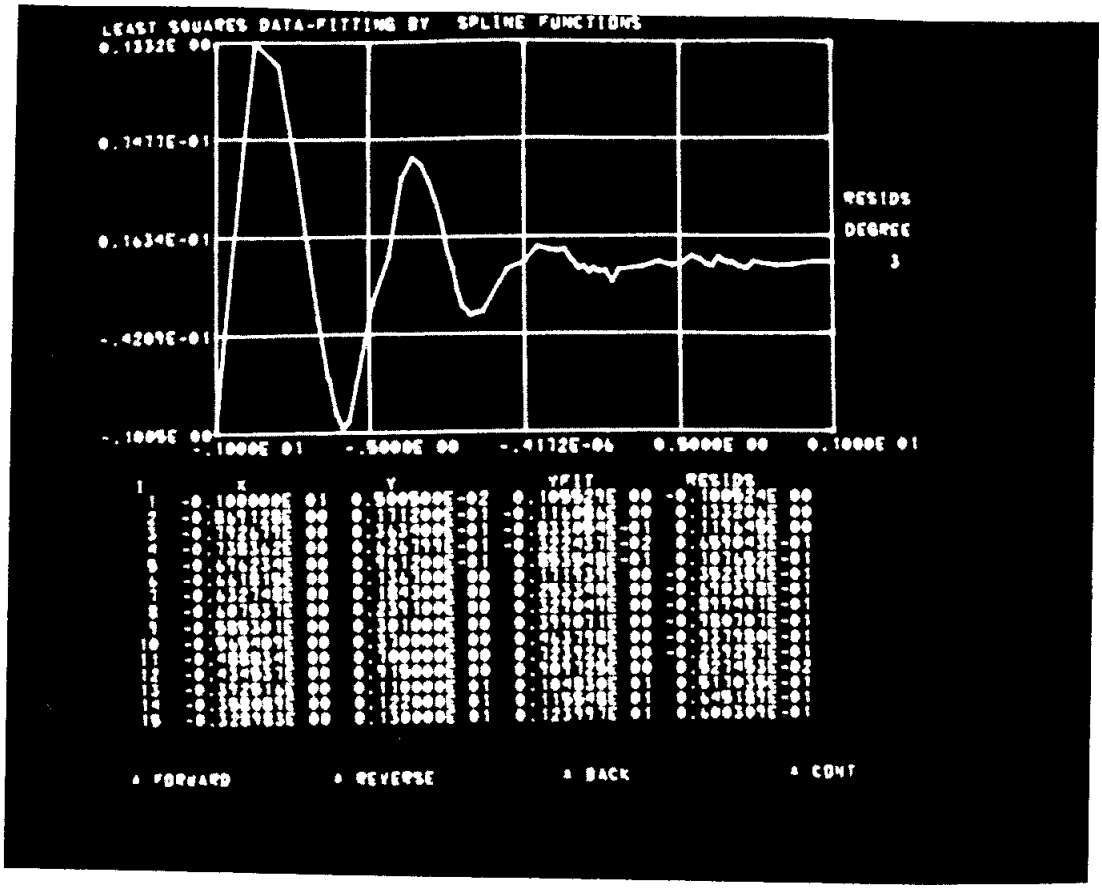

FIG. 3.24--Plot of residuals with fit displayed.

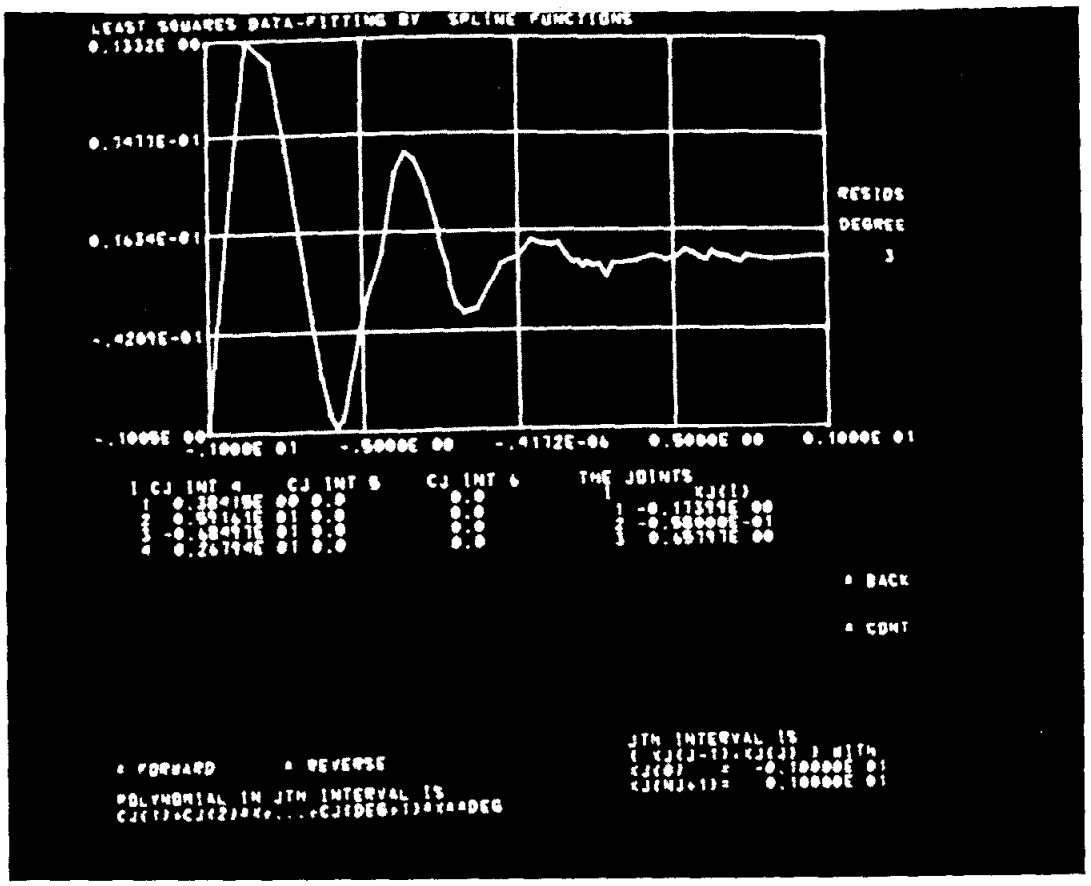

FIG. 3.25--Plot of residuals with coefficients displayed.

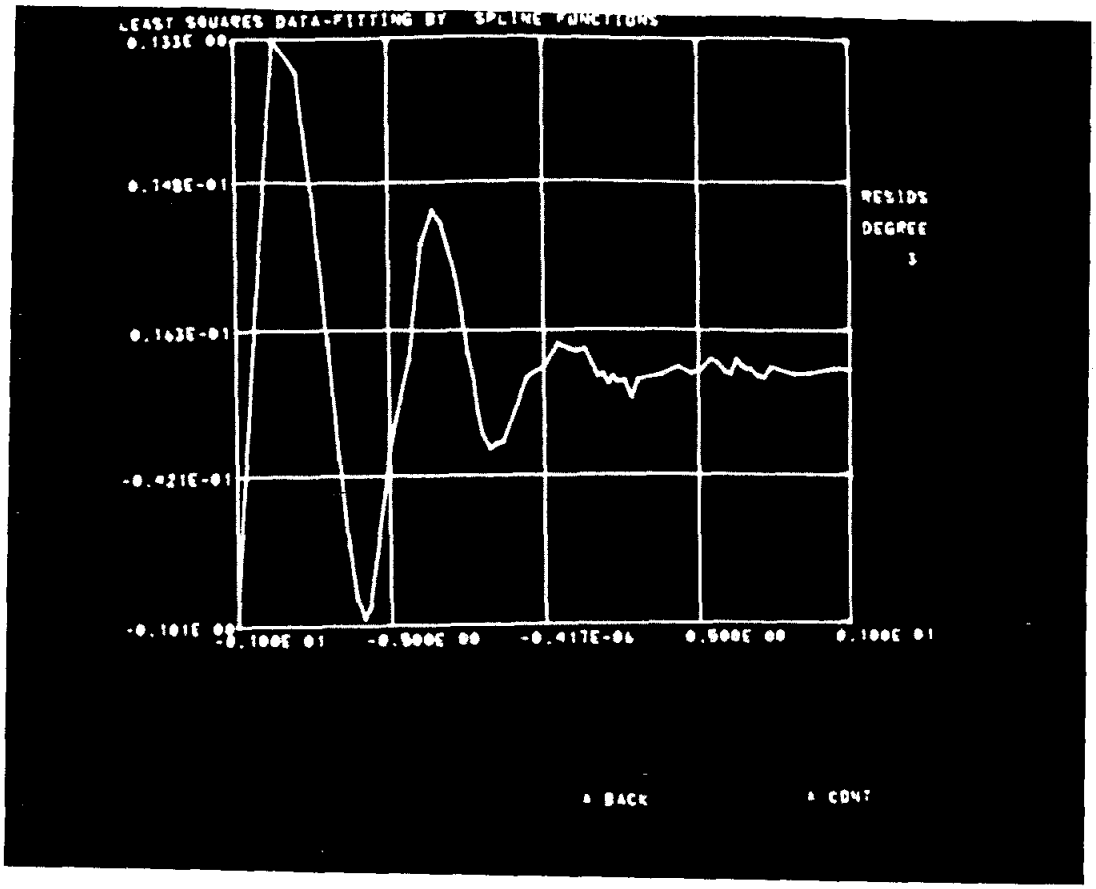

FIG. 3.26--Full scope plot of residuals.

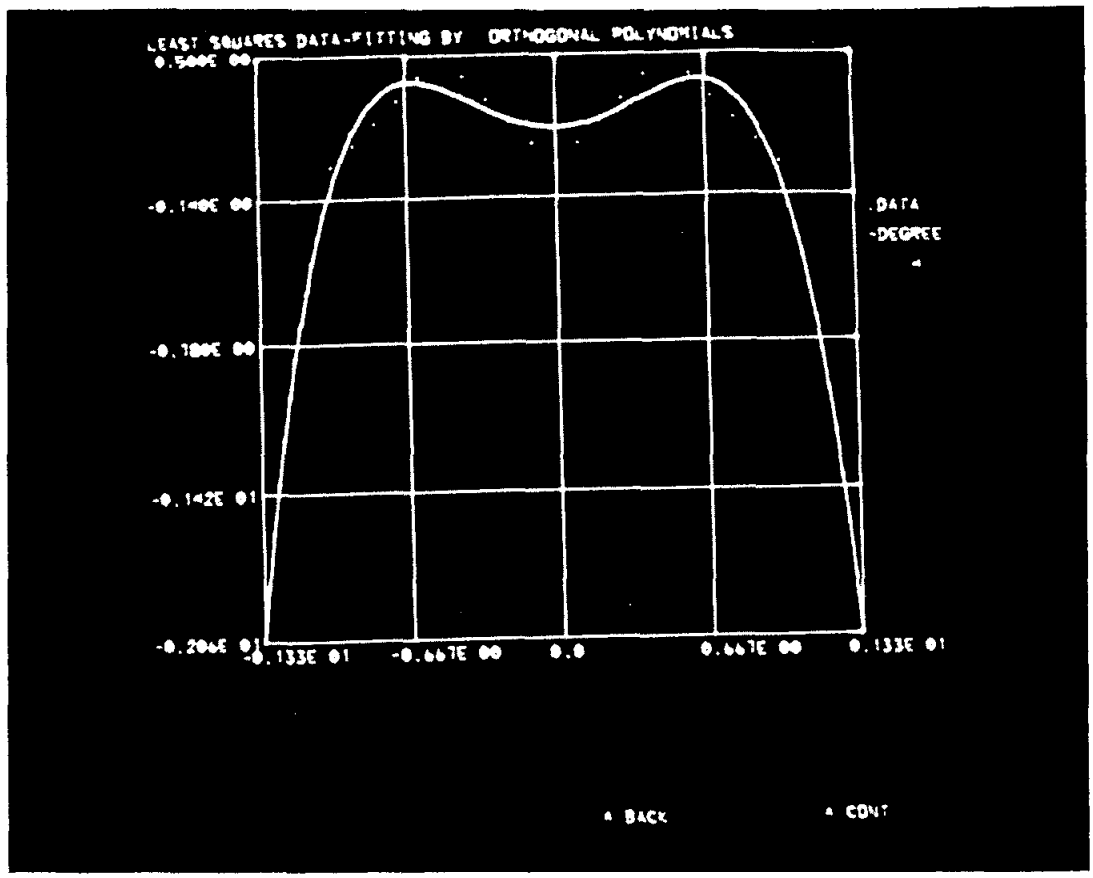

FIG. 3.27--Plot of data and fit extrapolated.

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.

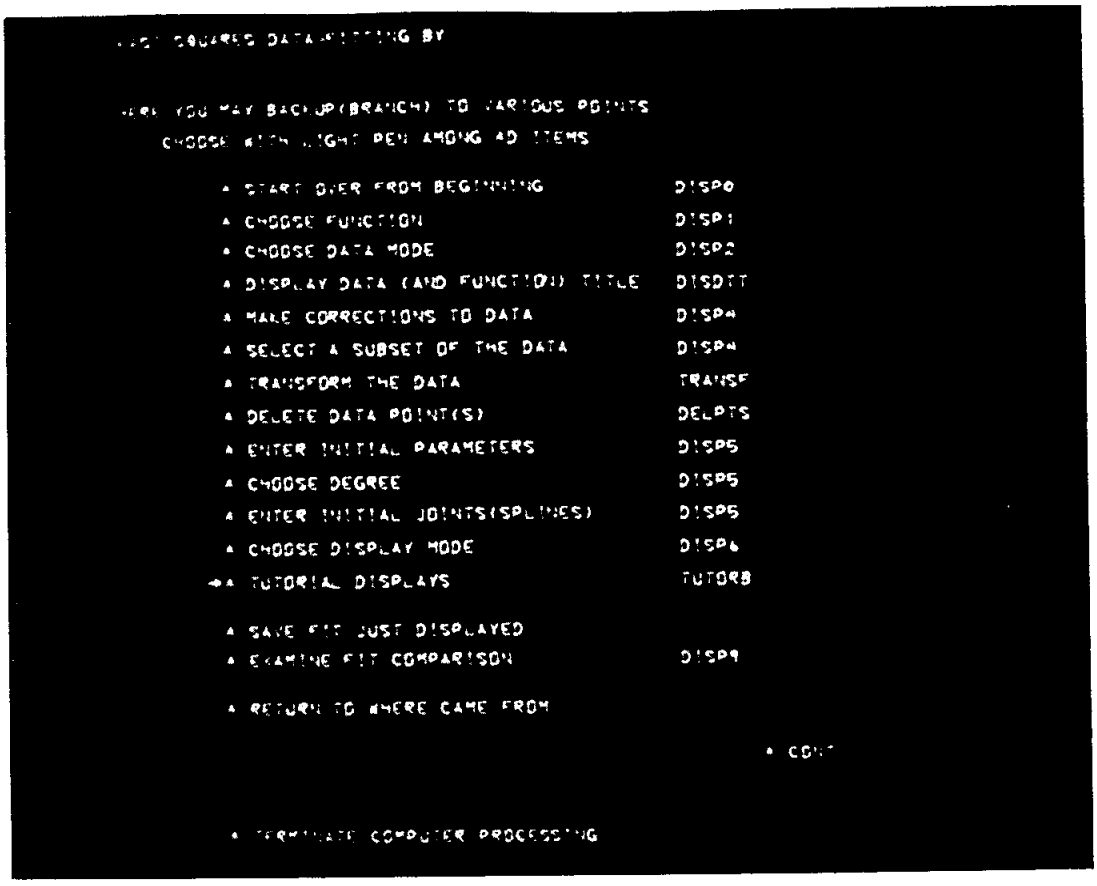

FIG. 3.28--Branching step display.

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.

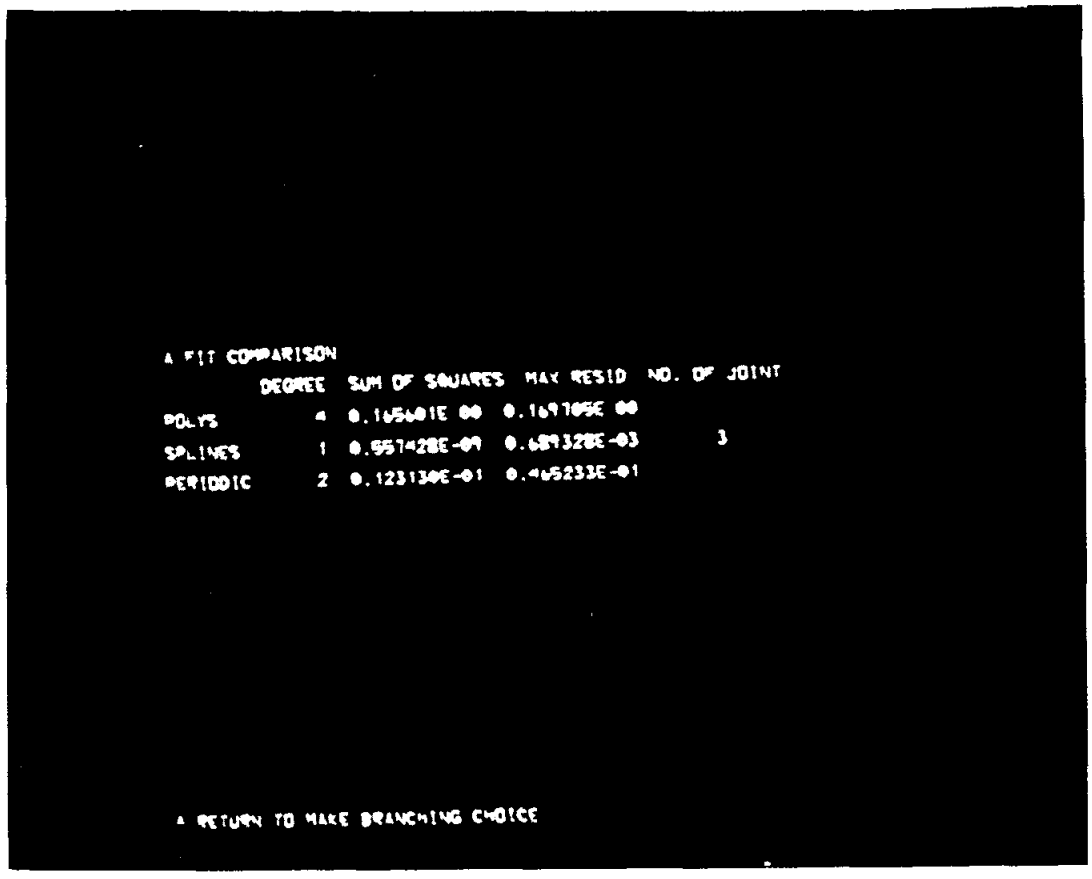

FIG. 3.29--A fit comparison.

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.

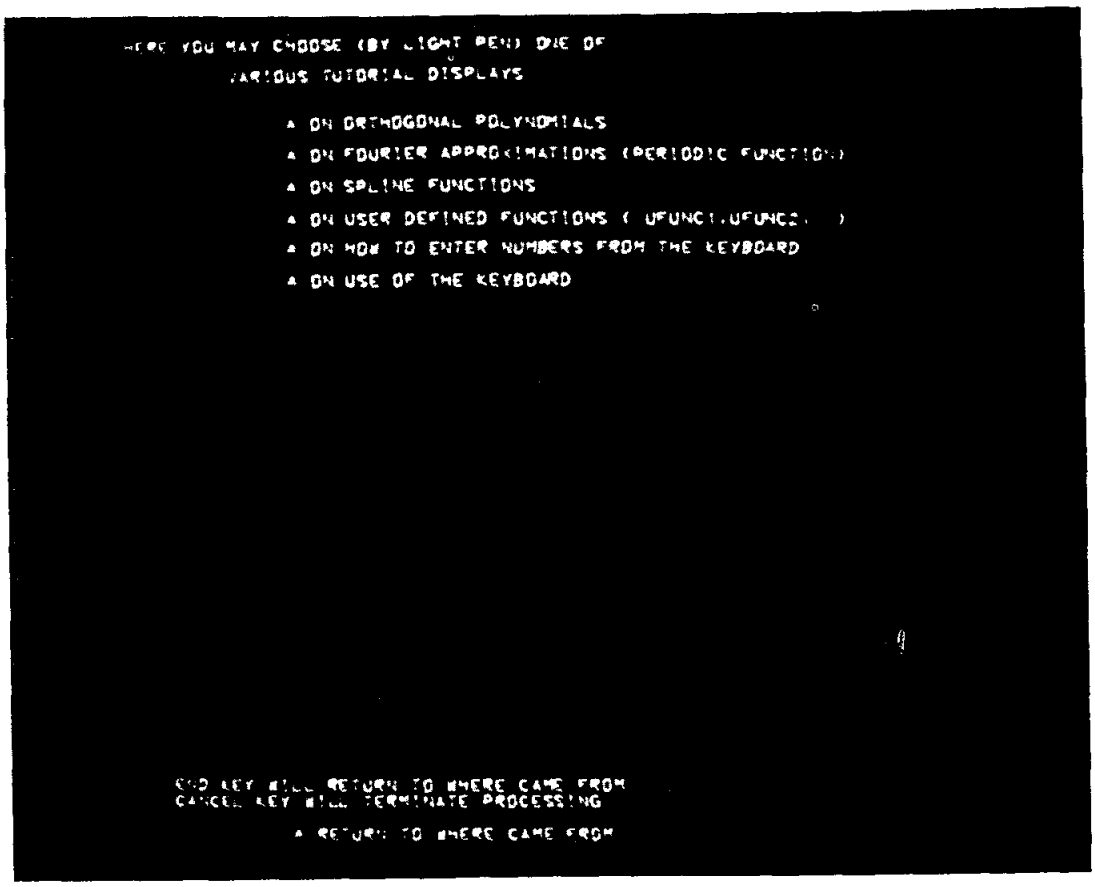

FIG. 3.30--Display allowing choice of tutorial display.

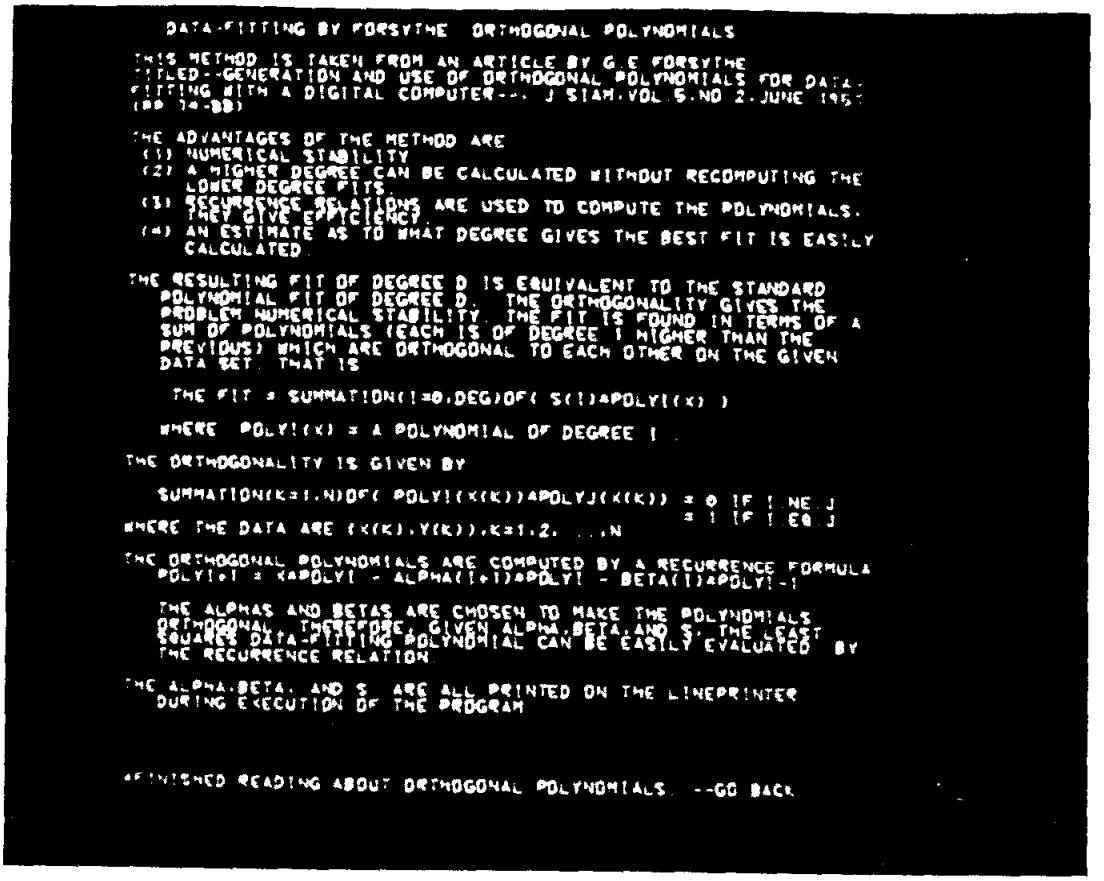

FIG. 3.30a--Tutorial display for orthogonal polynomials.

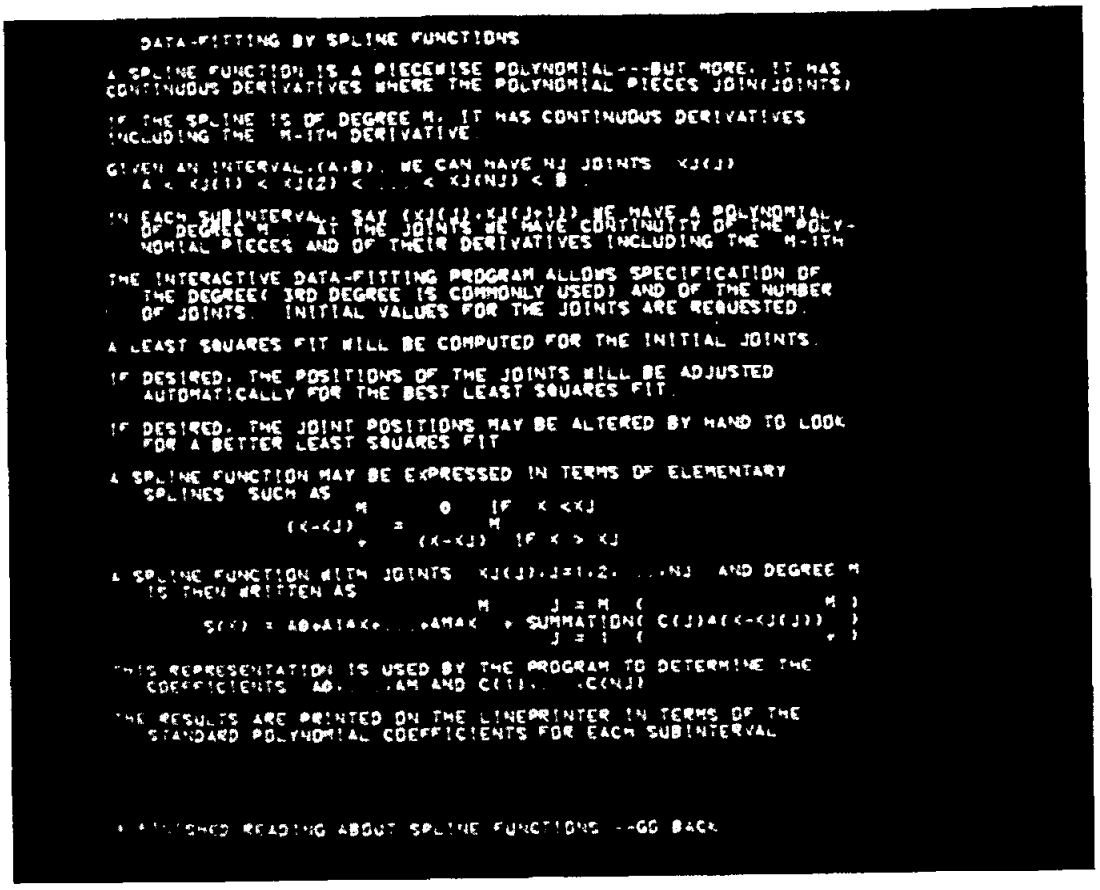

FIG. 3.30b--Tutorial display for spline functions.

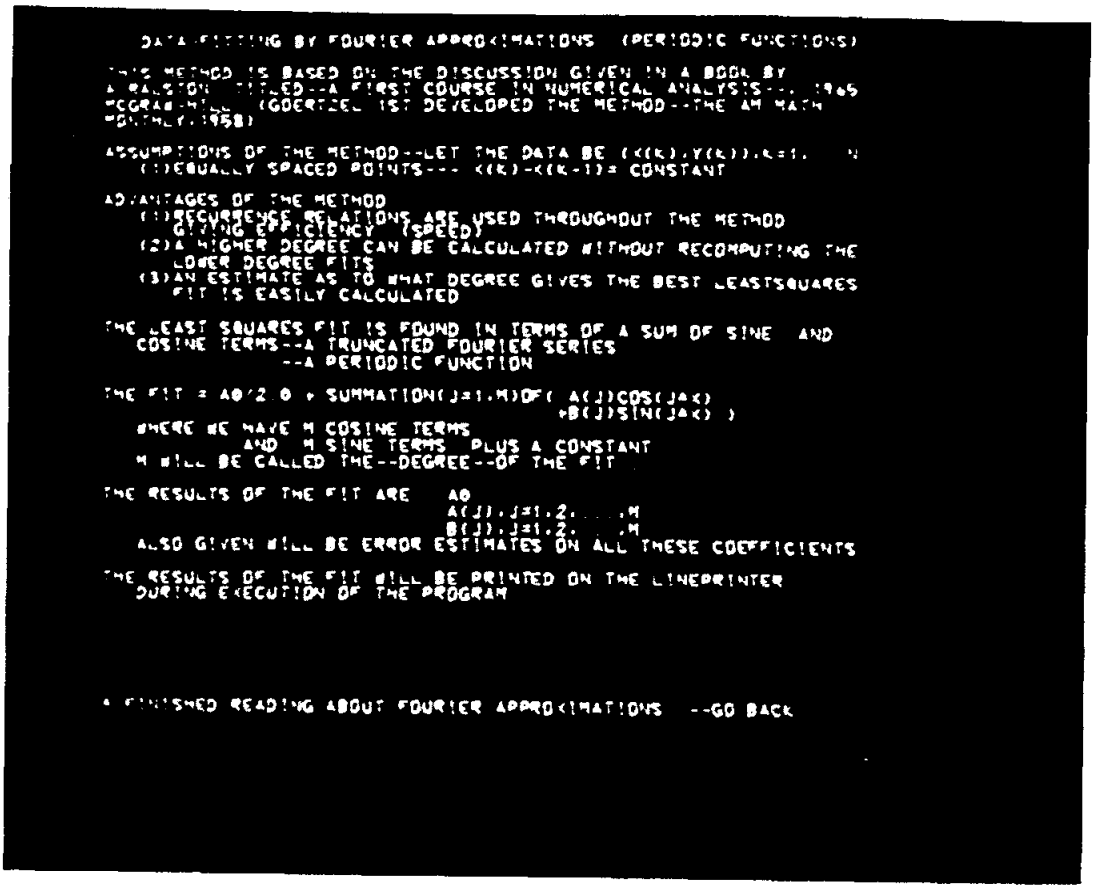

FIG. 3.30c--Tutorial display for Fourier approximations.

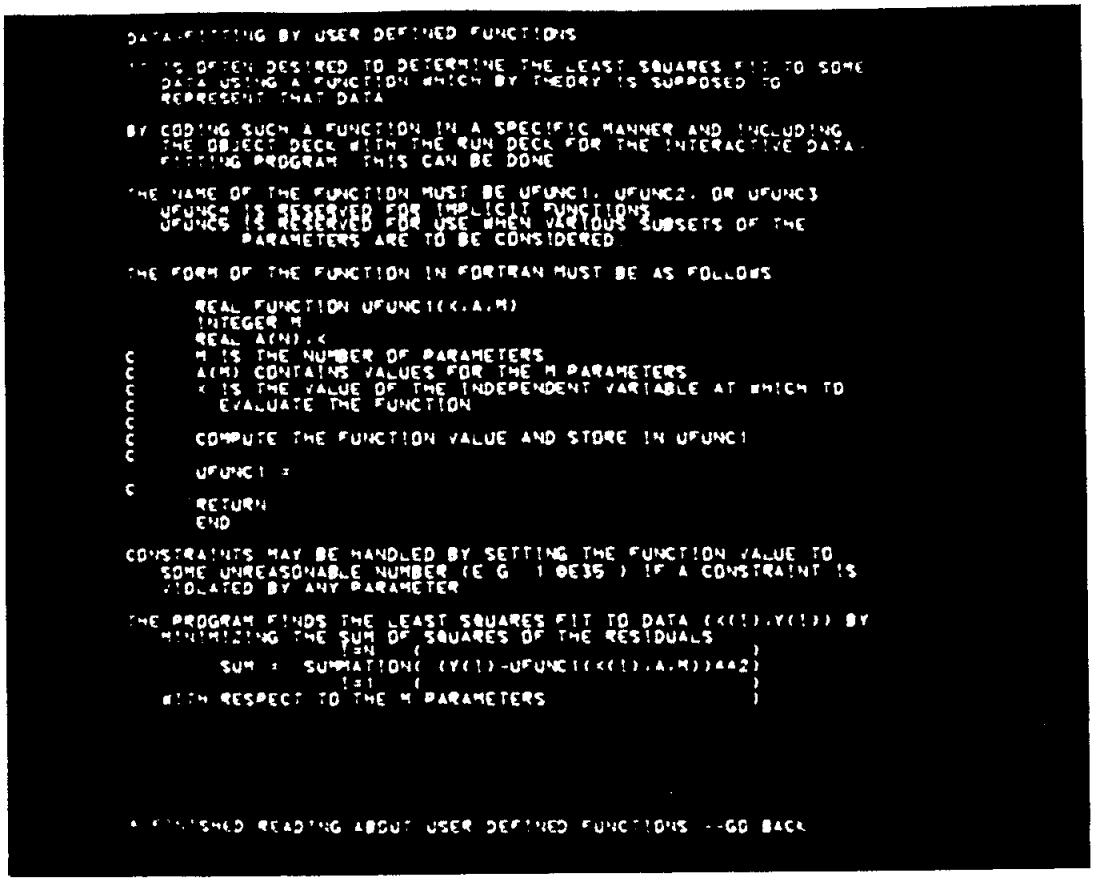

FIG. 3.30d--Tutorial display for user defined functions.

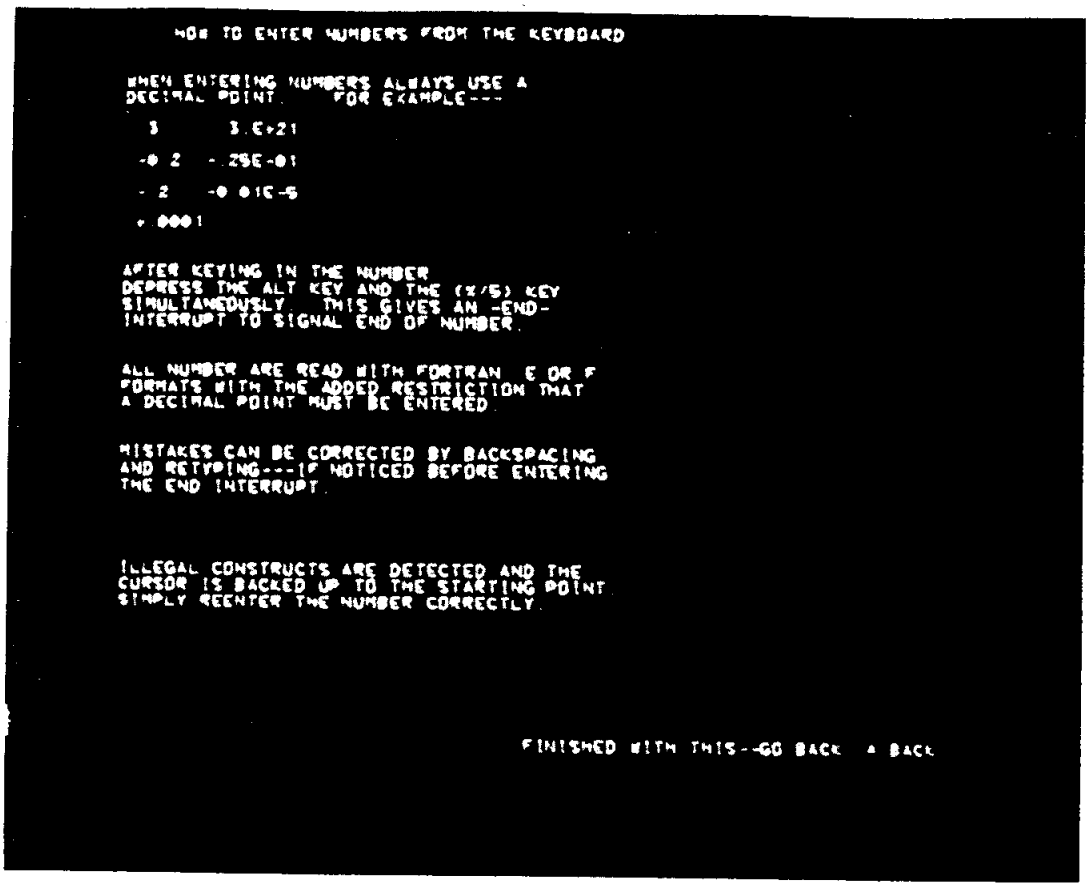

FIG. 3.30e--Tutorial display on how to enter numbers from the keyboard.

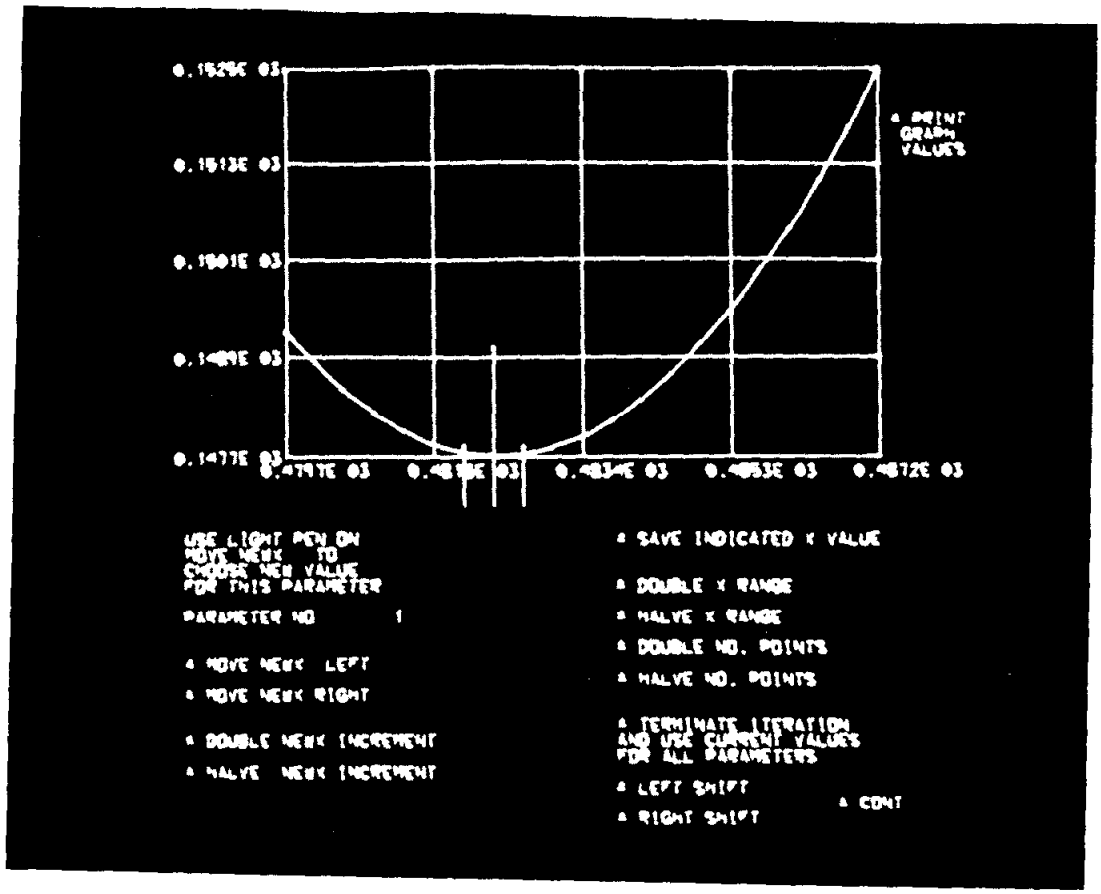

FIG. 3.31--Typical display during interactive minimization.

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).

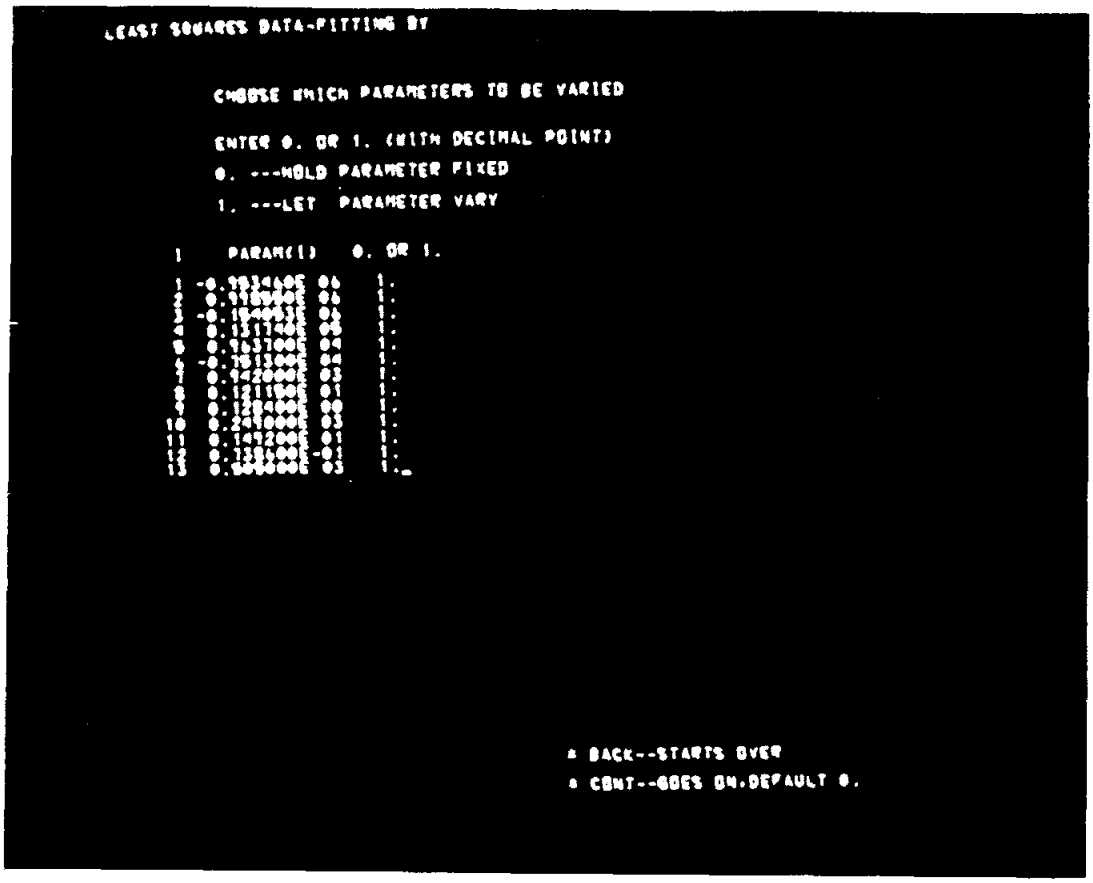

FIG. 3.37--Choosing which parameters to vary.

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.

CHAPTER IV

PEG — NUMERICAL METHODS

This chapter contains descriptions of the numerical algorithms used in the implementation of PEG. There are currently three "built-in" functions as well as the facility to employ a user defined function in the data-fitting process. The built-in functions are orthogonal polynomials, Fourier approximations, and spline functions. The methods used for each of these will be discussed and where appropriate their weaknesses will be pointed out and alternative methods suggested. In Appendix C we will point out the ease with which alternative methods can be incorporated in PEG.

A. Orthogonal Polynomials

Given a set of points for which an approximating curve is desired, if there is no known function which is supposed to approximate the data on theoretical grounds, a polynomial is often used as the approximating function. Polynomial approximation has a sound basis. It is well known that any real valued continuous function defined on a closed interval can be uniformly approximated by a polynomial (Weierstrass' approximation theorem, see Apostol [1957], p. 481, for example). Thus it seems intuitively reasonable to use a polynomial to approximate an unknown continuous function. Polynomials also have the advantage of being fairly east to handle computationally.

1. Why Not Standard Polynomials?

Let us define the "standard" representation of a polynomial, p(x), of degree k to be a sum of monomials,

$$p(x) = \sum_{i=0}^{k} a_i x^i \quad .$$

An orthogonal polynomial representation of p(x) is expressed as

$$p(x) = \sum_{i=0}^{k} b_i \, p_i(x) \qquad \text{(IV.0)}$$

where each $p_i(x)$ is a polynomial of proper degree i and the set of polynomials $\{p_i(x)\}_{i=0}^{k}$ are orthogonal in some sense to be discussed later.

The main disadvantage to the use of standard polynomials in least squares data-fitting, as stated by Forsythe [1957], is "when $k \geq 7$ or 8 (where k is the degree of polynomial) ... one begins to hear strange grumblings of discontent in the computing laboratory." The reason is that the system of equations to be solved for the least squares polynomial coefficients becomes quite ill-conditioned as the degree of polynomial increases. In fact, as shown by Forsythe, the coefficient matrix approximates the principal minor of the infinite Hilbert matrix,

$$H = \begin{bmatrix} 1 & 1/2 & 1/3 & \cdots \\ 1/2 & 1/3 & 1/4 & \cdots \\ 1/3 & 1/4 & 1/5 & \cdots \\ \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \end{bmatrix}$$

Systems of equations involving minors of H are well known to be very difficult to solve numerically.

Another disadvantage to the use of standard polynomials for least squares data-fitting is that once an approximating polynomial of degree m has been found, and a higher degree, say m + 1, is desired, the entire calculation must be redone. Thus in a statistical evaluation of the fit where the approximating polynomials for degree m = 1, 2, 3, ... are all desired, each polynomial requires the solution of a system of m equations in m unknowns.

2. The Generation and Use of Orthogonal Polynomials

The use of orthogonal polynomials essentially eliminates both the numeric computational difficulty and the excessive computation problem associated with the use of standard polynomials. Polynomials which are orthogonal with respect to a weight function are exemplified by the Chebyshev polynomials $T_i(x)$ on the interval (-1, 1) which satisfy

$$\int_{-1}^{1} T_i(x)\, T_j(x)\, (1 - x^2)^{-1/2}\, dx = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases}, \qquad \text{(IV. 1)}$$

where the weight function is $(1 - x^2)^{-1/2}$. Forsythe [1957] gives an algorithm for generating polynomials orthogonal over a set of points, $\{x_i\}_{i=1}^{n}$. These polynomials are extremely useful in least squares data-fitting.

Let the given data be $\{(x_i, y_i)\}_{i=1}^{n}$ and let the polynomials $\{p_i(x)\}_{i=1}^{m}$ be orthogonal over the point set $\{x_i\}_{i=1}^{n}$. These polynomials, with $p_i(x)$ a polynomial in x of proper degree i then satisfy

$$\sum_{\mu=1}^{n} p_i(x_\mu)\, p_j(x_\mu) = \begin{cases} 0 \text{ if } i \neq j \\ \neq 0 \text{ if } i = j \end{cases}. \qquad \text{(IV. 2)}$$

Using these orthogonal polynomials gives us a coefficient matrix for the system of equations to be solved for the least squares polynomial that is diagonal. This removes the difficulties of solving the normal equations (each equation is solved by one division). Also, given the least squares fit of degree m, the use of these orthogonal polynomials allows the determination of the m + 1 degree fit by solving a single equation.

It is well known that a three term recurrence is used to generate the orthogonal polynomials.

$$p_0(x) = 1;$$

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

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

$$\vdots$$

$$p_{i+1}(x) = xp_i(x) - \alpha_{i+1} p_i(x) - \beta_i p_{i-1}(x) \quad (i = 1, 2, \ldots) \quad . \tag{IV.3}$$

The α_i and β_i are chosen to make these polynomials satisfy (IV.2), the orthogonality relation. As shown by Forsythe [1957], they can be expressed as follows:

$$\alpha_{i+1} = \sum_{\mu=1}^{n} x_\mu \left[p_i(x_\mu)\right]^2 \Big/ \sum_{\mu=1}^{n} \left[p_i(x_\mu)\right]^2 \quad ; \tag{IV.4}$$

$$\beta_i = \sum_{\mu=1}^{n} x_\mu p_i(x_\mu)\, p_{i-1}(x_\mu) \Big/ \sum_{\mu=1}^{n} \left[p_{i-1}(x_\mu)\right]^2 \quad . \tag{IV.5}$$

If we define

$$\omega_{ij} = \sum_{\mu=1}^{n} p_i(x_\mu)\, p_j(x_\mu) \; , \qquad \omega_{ij} = 0 \text{ for } i \neq j \quad ,$$

an alternative formulation for the α_i and β_i's is:

$$\alpha_{i+1} = \sum_{\mu=1}^{n} x_\mu \left[p_i(x_\mu)\right]^2 / \omega_{ii} \quad ; \tag{IV.6}$$

$$\beta_i = \omega_{ii} / \omega_{i-1,i-1} \quad . \tag{IV.7}$$

Using (IV.6) and (IV.7) the orthogonal polynomials can be generated recursively.

The normal equations whose solution determines the least-squares polynomial represented as in (IV.0), can now be solved for the coefficients, $\{b_i\}_{i=0}^{k}$, as follows:

$$b_i = \left[\sum_{\mu=1}^{n} y_\mu p_i(x_\mu)\right] \Big/ \left[\sum_{\mu=1}^{n} \left[p_i(x_\mu)\right]^2\right], \quad i=0,\ 1,\ \ldots,\ k\ . \tag{IV.8}$$

The coefficients $\{\alpha_i\}_{i=1}^{k}$ and $\{\beta_i\}_{i=0}^{k-1}$, which determine the orthogonal polynomials, and the coefficients $\{b_i\}_{i=0}^{k}$ determine the desired least squares polynomial which minimizes S given by

$$S = \sum_{\mu=1}^{n} \left[y_\mu - \sum_{i=0}^{k} b_i p_i(x_\mu)\right]^2 \ . \tag{IV.9}$$

3. Weighted Least Squares

Often errors or positive weights (usually the inverse of any error) are associated with each data point. The weights may be denoted by $\{w_\mu\}_{\mu=1}^{n}$ where $w_\mu > 0$ for $\mu = 1, 2, \ldots, n$. The least squares problem is then to minimize S_w, where

$$S_w = \sum_{\mu=1}^{n} w_\mu^2 \left\{y_\mu - \sum_{i=0}^{k} b_i p_i(x_\mu)\right\}^2 \ . \tag{IV.10}$$

In the weighted case the orthogonal polynomials are generated so as to satisfy

$$\sum_{\mu=1}^{n} w_\mu^2 p_i(x_\mu)\ p_j(x_\mu) = \begin{cases} 0 \text{ if } i \neq j \\ \neq 0 \text{ if } i = j \end{cases} \ . \tag{IV.11}$$

This changes the formulae for α_{i+1} and β_i to be (compare (IV.6) and (IV.7))

$$\alpha_{i+1} = \frac{\sum_{x_\mu} w_\mu^2 \{p_i(x_\mu)\}^2}{\omega_{ii}} \tag{IV.12}$$

and

$$\beta_i = \omega_{ii}/\omega_{i-1,\,i-1} \quad , \tag{IV.13}$$

where the definition of ω_{ij} now includes the weighting as follows:

$$\omega_{ij} = \sum_{\mu=1}^{n} w_\mu^2 p_i(x_\mu)\, p_j(x_\mu) \quad , \qquad \omega_{ij} = 0 \text{ for } i \neq j \quad . \tag{IV.14}$$

These equations with weighting reduce to the non-weighted equations in case all the w_μ, $\mu = 1, 2, \ldots, n$ are equal to unity. The normal equations for the weighted least squares problem are:

$$b_i \sum_{\mu=1}^{n} w_\mu^2 \left\{p_i(x_\mu)\right\}^2 = \sum_{\mu=1}^{n} w_\mu^2 y_\mu p_i(x_\mu), \qquad i = 0, \ldots, k \tag{IV.15}$$

Using (IV.3), (IV.12) and (IV.13) we can solve (IV.15) for the $\{b_i\}_{i=0}^{k}$ which solve the weighted least squares problem (IV.10).

4. Choosing the Best Degree

A lucid discussion of a regression theory interpretation of polynomial least squares is given by Forsythe [1957]. The question to be answered is "which value of k should be used?", where k is the highest degree polynomial used in the fitting process.

If we assume that the $\{y_\mu\}_{\mu=1}^{n}$ are independently normally distributed about some polynomial trend $p_{h+1}(x) = \sum_{i=0}^{h+1} r_i x^i$ with a variance σ^2 independent of μ, we can make the null hypothesis that $r_{h+1} = 0$, no matter what values r_0, r_1, $\ldots$, r_h and σ^2 may have. The statistical test function for testing the hypothesis that $r_{h+1} = 0$ is defined below.

Let us consider the weighted case where we wish to minimize S_w in (IV.10) and let $p_k^*(x)$ denote the polynomial of degree k which accomplishes this

minimization,

$$p_k^*(x) = \sum_{i=0}^{k} b_i p_i(x) \quad . \tag{IV.16}$$

Define

$$\delta_k^2 = \sum_{\mu=1}^{n} w_\mu^2 \left\{ y_\mu - p_k^*(x_\mu) \right\}^2 \quad , \tag{IV.17}$$

and let

$$\sigma_k^2 = \delta_k^2/(n - k - 1). \tag{IV.18}$$

Under the null hypothesis it follows that the expected value of the statistic σ_k^2 is independent of k, for k=h+1, h+2, ..., n-2. For the case of unit weights (no weighting) we simply assume $w_\mu = 1$, $\mu = 1, 2, \ldots, n$ in (IV.17).

To determine which value of k does indeed best represent the data, we monitor σ_k^2 as k increases. When there is no further significant decrease in σ_k^2 for $k > k_1$, we may assume that k_1 gives the best representation of the data. In practice we usually require calculation of at least 2 higher values of k, say $k_1 + 1$ and $k_1 + 2$ to show no significant improvement before accepting k_1 as the best least squares fit, since it often happens in practice that $\sigma_{k_1+1}^2$ can even increase over $\sigma_{k_1}^2$ but then show a decrease well below $\sigma_{k_1}^2$ at $\sigma_{k_1+2}^2$.

5. Estimating the Errors in the Coefficients

An estimate of the errors in the computed least squares coefficients can be computed during the fitting process. A discussion of how these error estimates are derived is given by Ralston [1965], (p. 247). If the computed least squares fit is given by (IV.16) an estimate of the variance of the errors Δb_i, in the computed b_i, $i = 0, 1, \ldots, k$ is given by

$$\sigma_{\Delta b_i}^2 = \sigma^2/\gamma_i \tag{IV.19}$$

where the errors in the data, $\{y_\mu\}_{\mu=1}^n$, are assumed to have variance σ^2/w_i^2 at each point, and γ_i is given by

$$\gamma_i = \sum_{\mu=1}^{n} \left\{ w_\mu^2 \, p_i(x_\mu) \right\}^2 \quad . \tag{IV.20}$$

An estimate for the quantity σ^2 is given by (IV.18) so that in practice we use

$$\sigma^2_{\Delta b_i} = \sigma_i^2/\gamma_i, \; i = 0, \; 1, \; \ldots, \; k \tag{IV.21}$$

to estimate the variance of the error in the coefficients of (IV.16). $\sigma_{\Delta b_i}$ then is an estimate of the standard deviation of the error in b_i.

6. Fortran Code for Least Squares by Forsythe Orthogonal Polynomials

A detailed description of the algorithm is given in Forsythe [1957]. In Appendix A, we present the algorithm for calculation of the Forsythe orthogonal polynomial least squares fit in the form of a Fortran subroutine as used in PEG. Forsythe [1957] suggests scaling the interval containing the $\{x_\mu\}_{\mu=1}^n$ to the interval [-2, 2] for reasons of computational stability. In using the code given in Appendix A, it is suggested that the data be so scaled, however, the code in no way depends on such scaling.

7. Other Methods

Berztiss [1964] gives a comparison of several polynomial fitting methods. He compares machine time, storage space, accuracy, and applicability to various problems of five methods which are listed below.

a. Use of monomials x^k.

b. Use of Legendre polynomials $p_k(x)$.

c. Use of Chebyshev polynomials $T_k(x)$.

d. Forsythe's method.

e. Clenshaw's method.

Method d should not be used for interpolation since the expansion arrived at is dependent on the original data but for other general purpose usage, method d is probably as good or better than the other methods mentioned.

Clenshaw and Hayes [1965] also discuss the same five methods compared by Berztiss. They point out applications in curve and surface fitting where method e is the most desirable.

B. Truncated Fourier Series

Sometimes physical knowledge of the data indicates that a periodic function should be used in a least squares approximation. Visual inspection of the data can also lead to an attempt to find a periodic approximation as opposed to say a polynomial approximation. An efficient algorithm for calculation of the Fourier approximation for equally spaced data points is discussed by Ralston [1965]. The computational algorithm is originally due to Goertzel [1958].

1. Goertzel's Algorithm

Given data points $(x_i, y_i)_{i=0}^{n}$ we wish to approximate y_i by

$$f_m(x) = (1/2)\, a_0 + \sum_{j=1}^{m} (a_j \cos jx + b_j \sin jx) \quad . \tag{IV.22}$$

If the data points are equally spaced the set of approximating functions, 1, cos x, sin x, ... cos mx, sin mx, satisfy certain orthogonality relations over the data points. As pointed out by Ralston [1965] these orthogonality relations can be used to simplify the normal equations obtained from a direct approach to the least squares problem. Even so the resulting equations for calculating

$a_0, a_1, \ldots, a_m, b_1, \ldots, b_m$ are cumbersome compared to Goertzel's method. Goertzel's method employs a recurrence relation to obtain the desired coefficients. The method is detailed in Ralston [1965] for the case of an odd number of equally spaced data points. If the number of equally spaced data points is even, the method still holds with only minor modifications.

An estimate of the errors in the computed coefficients can be easily calculated for the case of evenly spaced points. These estimates are given in Ralston [1965] (p. 258, problem 30).

Let us denote by "degree" the number of sine terms and cosine terms in the approximating function (m in (IV.22)). It is possible to determine what degree gives the best representation of the data in a manner similar to that used for orthogonal polynomials. Again, Ralston [1965] presents a discussion of how this can be accomplished.

It has been pointed out by Dr. J. F. Traub of the Bell Telephone Laboratories at Murray Hill, New Jersey, that Goertzel's algorithm is numerically unstable for high degree computations (Traub [1968]). However, Dr. M. Gentlemen also of the Bell Telephone Laboratories will soon be publishing a modification to Goertzel's algorithm which will correct the instability.

2. Other Methods

In case the points are not equally spaced or there are weights given, the least squares solution must take a different form from that given by Goertzel's algorithm. The method employed by PEG is a straightforward solution of the normal equations derived from (IV.22). The matrix of coefficients of the normal equations is inverted and the right-hand side is multiplied by the inverse matrix. This method is less efficient than, for example, using Gaussian elimination and backsolving, but the diagonal elements of the inverse are used to give estimates of the errors in the computed coefficients.

A more accurate solution to this problem could be obtained by using an algorithm given by Björck and Golub [1968]. This method employs Householder transformations and iterative refinement to obtain accurate solutions to linear least squares problems with or without constraints.

3. Fortran Codes for Fourier Approximation

In Appendix B we give Fortran subroutines for the calculation of least squares Fourier approximation to equally spaced data points using Goertzel's method. For unequal spacing of the data points or if weights are associated with the points, PEG employs a direct attack on the normal equations by inversion of the matrix of coefficients. As mentioned previously, the inverse matrix gives estimates on the errors in the least squares coefficients. Given a subroutine for matrix inversion, this method is quite straightforward so we will not present Fortran code for this problem.

C. Spline Functions

First we will present a direct approach to the problem of least squares approximation by spline functions as implemented in PEG. Following that we will discuss various other methods. The straightforward approach can lead to numerical difficulties due to a somewhat ill-conditioned matrix, consequently as the sum of the degree of the spline used and the number of joints gets larger the alternative methods become more valuable.

1. A Direct Approach

Any spline function can be expressed uniquely as the sum of a polynomial and a linear combination of elementary spline functions (see Schoenberg and Whitney [1953]). We recall that a spline function $S_m(x)$ is a piecewise polynomial of degree m with the additional restriction that all derivatives up to and including the m-1st must be continuous at the joints (where the polynomial

pieces join together). An elementary spline function, $(x-\hat{x}_\alpha)_+^m$, is defined as

$$(x-\hat{x}_\alpha)_+^m = \begin{cases} 0 \text{ if } x \le \hat{x}_\alpha \\ (x-\hat{x}_\alpha)^m \text{ if } x > \hat{x}_\alpha \end{cases} \tag{IV.23}$$

A spline function $S_m(x)$ of degree m with J joints $\hat{x}_1 < \hat{x}_2 < \ldots < \hat{x}_J$ can be represented as

$$S_m(x) = a_0 + a_1 x + \ldots + a_m x^m + \sum_{i=1}^{J} c_i (x-\hat{x}_i)_+^m \,. \tag{IV.24}$$

Thus any spline function with distinct joints can be expressed as a polynomial plus a sum of elementary splines.

Given a set of joints, $\hat{x}_1 < \hat{x}_2 < \ldots < \hat{x}_J$, the least squares approximation problem is to determine the values of the coefficients $\{a_i\}_{i=0}^m$ and $\{c_i\}_{i=1}^J$ in (IV.24) which minimize the sum of squares of the residuals. If we have data points $\{(x_\mu, y_\mu)\}_{\mu=1}^n$ the quantity to be minimized is D, where

$$D = \sum_{\mu=1}^{n} \left[y_\mu - S_m(x_\mu) \right]^2 \,. \tag{IV.25}$$

Since the coefficients in (IV.24) occur linearly, this is a linear least squares problem if we assume the joints, $\{x_i\}_{i=1}^J$, are fixed. One method of solving this problem is simply to form the partial derivatives of D with respect to the m + 1 + J parameters, equate these derivatives to zero and solve the resultant set of equations, the normal equations. The partial derivatives of D are

$$\begin{aligned} \frac{\partial D}{\partial a_i} &= 2 \sum_{\mu=1}^{n} \left[y_\mu - S_m(x_\mu) \right] \cdot \left[x_\mu^i \right], \quad i = 0, 1, \ldots, m \\ \frac{\partial D}{\partial c_i} &= 2 \sum_{\mu=1}^{n} \left[y_\mu - S_m(x_\mu) \right] \cdot \left[(x_\mu - \hat{x}_i)_+^m \right], \quad i = 1, 2, \ldots, J \,. \end{aligned} \tag{IV.26}$$

Equating these m + 1 + J equations to zero we obtain the normal equations

$$Au = b \quad , \tag{IV.27}$$

where, if we denote $\sum_{\mu=1}^{n}$ by Σ ,

$$A = \begin{bmatrix} n & \dots \Sigma x_\mu^m & \Sigma (x_\mu - \hat{x}_1)_+^m & \dots \Sigma (x_\mu - \hat{x}_J)_+^m \\ \Sigma x_\mu & \dots \Sigma x_\mu^{m+1} & \Sigma x_\mu (x - \hat{x}_1)_+^m & \dots \Sigma x_\mu (x_\mu - \hat{x}_J)_+^m \\ \vdots & & & \\ \Sigma x_\mu^m & \dots \Sigma x_\mu^{2m} & \Sigma x_\mu^m (x_\mu - \hat{x}_1)_+^m & \dots \Sigma x_\mu^m (x_\mu - \hat{x}_J)_+^m \\ \Sigma (x_\mu - \hat{x}_1)_+^m & \dots \Sigma x_\mu^m (x_\mu - \hat{x}_1)_+^m & \Sigma (x_\mu - \hat{x}_1)_+^{2m} & \dots \Sigma (x_\mu - \hat{x}_1)_+^m (x_\mu - \hat{x}_J)_+^m \\ \vdots & & & \\ \Sigma (x_\mu - \hat{x}_J)_+^m & \dots \Sigma x_\mu^m (x_\mu - \hat{x}_J)_+^m & \Sigma (x_\mu - \hat{x}_1)_+^m (x_\mu - \hat{x}_J)_+^m & \dots \Sigma (x_\mu - \hat{x}_J)_+^{2m} \end{bmatrix} \tag{IV.28}$$

$$b = \begin{bmatrix} \Sigma y_\mu \\ \Sigma y_\mu x_\mu \\ \vdots \\ \Sigma y_\mu x_\mu^m \\ \Sigma y_\mu (x_\mu - \hat{x}_1)_+^m \\ \vdots \\ \Sigma y_\mu (x_\mu - \hat{x}_J)_+^m \end{bmatrix} \tag{IV.29}$$

and

$$u = \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_m \\ c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_J \end{bmatrix} . \qquad \text{(IV.30)}$$

Equation (IV.27) can be solved for the coefficients which minimize D by any standard method for solving a linear system of equations such as Gaussian elimination.

Comments on the normal equations. In computing the elements of A and b as shown in (IV.28) and (IV.29) note that any element containing $(x_\mu - \hat{x}_j)_+^m$ as a factor need only be summed from $\mu = \mu_j$ to $\mu = n$ (rather than from $\mu = 1$ to $\mu = n$) where μ_j satisfies

$$\begin{cases} x_\mu \leq \hat{x}_j & \text{for } \mu < \mu_j \\ x_\mu > \hat{x}_j & \text{for } \mu \geq \mu_j \end{cases} \qquad \text{(IV.31)}$$

If a least squares spline is desired which differs from a previous calculation in that one or more joints have been moved, total recalculation of the matrix of coefficients, A, is unnecessary. Only the rows and columns of A which correspond to the altered joints need to be recomputed. This applies to the elements of the right-hand side, b, as well. Thus considerable effort in recomputing the normal

equations can be eliminated when changing only a few of the joints. In any case, the $(m + 1) \times (m + 1)$ principal minor is never changed as the joints change and the first $m + 1$ elements of the right-hand side are also independent of the joint positions. The importance of this will be seen later as we discuss the least squares problem with variable joints.

If one joint is added to the system, the normal matrix is simply augmented by one row and one column while the right-hand side has one element added to it. Thus, calculation of the new system of equations is easily accomplished. In making changes to the normal matrix remember that it is symmetric.

Restrictions on the joints. It is easy to see from (IV.28) that if any joint $\hat{x}_j$ is such that $\hat{x}_j > x_\mu$ for $\mu = 1, 2, \ldots, n$, the matrix A will be singular, since $(x_\mu - \hat{x}_j)_+^m = 0$ for all μ and the row and column corresponding to the j^{th} joint will be all zeros. Therefore, we must require all joints to satisfy

$$\hat{x}_j < \max_\mu (x_\mu) \quad . \tag{IV.32}$$

In addition, if any two joints are equal, $\hat{x}_j = \hat{x}_k$, we will have two identical rows and columns in (IV.28) which again gives singularity. This leads to the requirement that all joints be distinct,

$$\hat{x}_j \neq \hat{x}_k \quad \text{for } j \neq k \quad . \tag{IV.33}$$

Evaluating the spline function. Evaluation of the spline function in the form (IV.24) requires considerably more calculation than that required to evaluate a polynomial of degree m which is what the spline represents between each pair of points. Thus we need to reduce (or expand) $S_m(x)$ so it can be expressed as a simple polynomial of degree m at any point. One method of accomplishing this is to evaluate $S_m(x)$, given by (IV.24), at $m + 1$ points between each pair of joints and compute the Lagrange interpolating polynomial on those points. This will

produce polynomials of degree m in each subinterval between joints which are identical to $S_m(x)$ on those subintervals. The calculation of these interpolating polynomials can be simplified by choosing equally spaced points on which to interpolate. This representation will have $(m + 1) \times (J + 1)$ coefficients, c_{ij}, $i = 0, 1, \ldots, m$ and $j = 0, 1, \ldots, J$, which represent the spline function as follows:

$$\begin{cases} S_m(x) = \sum_{i=0}^{m} c_{i0} x^i, & x \le \hat{x}_1 ; \\ S_m(x) = \sum_{i=0}^{m} c_{ij} x^i, & \hat{x}_j \le x \le \hat{x}_{j+1}, \quad 1 \le j \le J-1 ; \\ S_m(x) = \sum_{i=0}^{m} c_{iJ} x^i, & \hat{x}_J \le x \quad . \end{cases} \tag{IV.34}$$

deBoor and Rice [1968a] discuss five representations of the spline function (IV.24) with varying numbers of parameters. They point out that the representation (IV.34) is the computationally most desirable in spite of the fact that the requirement of $(m + 1) \times (J + 1)$ parameters makes it highly redundant. The use of (IV.34) to evaluate the spline function at a particular value of the independent variable, say x', involves testing to see which subinterval contains x' and evaluating the corresponding polynomial.

Weighted spline function approximation. If there are positive weights, $w_\mu > 0$, associated with the data points so that the data is $\{(x_\mu, y_\mu, w_\mu)\}_{\mu=1}^{n}$, the least squares spline approximation just discussed can be modified to give a weighted least squares approximation. For each data point we have the weighted equation

$$w_\mu y_\mu - w_\mu S_m(x_\mu) = w_\mu r_\mu \quad , \tag{IV.35}$$

where r_μ is the residual at that point. We wish to minimize D_w, where

$$D_w = \sum_{\mu=1}^{n} \left[w_\mu r_\mu\right]^2 \qquad \text{(IV.36)}$$

As in paragraph IV.C.1, this problem can be expressed in the form of normal equations

$$A_w u_w = b_w \ , \qquad \text{(IV.37)}$$

where A_w and b_w are given by (IV.28) and (IV.29) with the addition of the square of the weights as a multiplicative factor under each summation. The vector u_w is the same as u in (IV.30). The subscript, w, simply indicates that these coefficients are for the weighted least squares problem.

As before, Eq. (IV.37) can now be solved by any suitable method for solving a linear system of equations. The previous comments on the normal equations apply to the weighted case also.

Estimating errors in the coefficients. When the spline function approximation is computed by solving the normal equations, weighted or non-weighted, as discussed above, a simple method is available for obtaining estimates of the errors in the coefficients.

In paragraph IV.B.2 we discussed the use of the inverse of the normal matrix as a source of error estimates. The same remarks apply to the present case. If we solve the equations (IV.27) and (IV.37) by inverting A and A_w respectively and computing $A^{-1}b$ and $A_w^{-1}b_w$ to obtain the coefficients of the spline functions, the elements of A^{-1} and A_w^{-1} can be used in estimates of the variance of each coefficient. For example, in the non-weighted case, if we denote the variance

of a_i by Δa_i, and that of c_i by Δc_i, we have

$$\begin{cases} \Delta a_i = \left(A^{-1}_{\ell\ell}\right)^{1/2}, & i = 0,\ 1,\ \ldots\ m\ (\ell = i + 1) \\ \Delta c_i = \left(A^{-1}_{\ell\ell}\right)^{1/2}, & i = 1,\ 2,\ \ldots,\ J\ (\ell = m + 1 + i)\ . \end{cases} \tag{IV.38}$$

2. Other Methods

As mentioned previously, the matrix (IV.28) can be ill-conditioned if the sum of the degree and number of joints is large. In practice (on an IBM 360 using single precision) if m + J is around 7 or 8 we begin to see deterioration of the method using (IV.28).

A method which has not been implemented but which would provide a more accurate solution to the least squares problem is the use of Householder transformations as discussed by Golub [1965] and by Bussinger and Golub [1965]. The algorithm given by Bjorck and Golub [1968] which employs Householder transformations and iterative refinement could also be used to improve the handling of cases where m + J is larger than 7 or 8.

In deBoor and Rice [1968a] and [1968b] the use of an orthonormal basis to represent a cubic spline function (m = 3) for the least squares approximation problem is discussed. Their approach to least squares is somewhat different from that used in PEG in that they consider

$$\int_{x_1}^{x_n} \left[w(x)\ (S(x) - y(x))\right]^2\ dx = \min. \tag{IV.39}$$

The integral is approximated in practice by the trapezoidal rule. S(x) is a cubic spline and w(x) represents a weight function (w(x) = 1 if no weights). By choosing an orthonormal basis for the space of all cubic splines they are able to obtain a more accurate solution to (IV.39) than if some other representation of the spline were used. This method could be extended to splines of other than third degree.

In Carasso and Laurent [1968] a basis is used to represent spline functions for the interpolation problem. They consider only splines of odd degree and are concerned with interpolation and not the least squares problem but the basis they suggest could also be used in the least squares problem. The basis they suggest is defined as follows.

Consider the $(J + 2q)$-dimensional space of spline functions of degree $2q - 1$ with J joints defined on the interval $[a, b]$. Let δ_j^q be the divided difference functional (of order q with respect to the abscissa $\hat{x}_j, \ldots, \hat{x}_{j+q}$). Then a basis for the space of spline functions is given by

$$s_i(x) = \delta_{i-q}^{2q} (x - \hat{x}_i)_+^{2q-1}; \quad i = -q+1, \ldots, J+q,$$

where we use the supplementary "joints" $\hat{x}_{-2q+1}, \ldots, \hat{x}_{-1}, \hat{x}_0 = a$ and $\hat{x}_{J+1} = b$, $\hat{x}_{J+2}, \ldots, \hat{x}_{J+2q}$ $(\hat{x}_i < \hat{x}_{i+1})$. We have $s_i(x) = 0$ if $x \notin [\hat{x}_{i-q}, \hat{x}_{i+q}]$. No criteria for choosing the supplementary joints are given in Carasso and Laurent [1968], only the requirement that $\hat{x}_i < \hat{x}_{i+1}$.

This basis could be used in the least squares problem with fixed joints. Let S_m be a spline function of degree m (odd) with J joints and represent it by the above basis as

$$S_m(x) = \sum_{i=1}^{m+1+J} c_i s_i(x) \quad .$$

Given data points $\{(x_\mu, y_\mu)\}_{\mu=1}^n$ the quantity to be minimized is D, where

$$D = \sum_{\mu=1}^{n} \left[y_\mu - S_m(x_\mu)\right]^2 \quad .$$

Since the coefficients $\{c_i\}_{i=1}^{m+1+J}$ occur linearly, it is a straightforward process to develop the normal equations whose solution gives the least squares spline

function in terms of the above basis. Carasso and Laurent [1968] compare the use of this basis to other methods for solving the interpolation problem and point out that it is not only as accurate as other methods but more efficient.

Anselone and Laurent [1968] discuss a general method for constructing interpolating or smoothing spline functions. The method is based on the properties of orthogonality of the spline functions and has proven to be stable even for a large number of functionals. This paper includes a bibliography containing 58 references related to the study of spline functions.

D. User Defined Functions

Very often it is desired to obtain a least squares fit to some empirical data by a specific function which is neither a polynomial nor some other standard function. In physics, for example, a theory might indicate that the data should be approximated by a function of a certain form and the problem is then to determine values for the parameters of that function.

Assume we are given data $\{(x_\mu, y_\mu)\}_{\mu=1}^{n}$ or weighted data $\{(x_\mu, y_\mu, w_\mu)\}_{\mu=1}^{n}$ and the approximating function is $f(x, a_1, \ldots, a_m)$, where the $\{a_i\}_{i=1}^{m}$ are the parameters to be determined by the least squares analysis. It is assumed that the parameters occur non-linearly so that we have a non-linear least squares problem requiring an iterative method for solution.

The problem to be solved is the following. Minimize S_w with respect to the parameters $\{a_i\}_{i=1}^{m}$, where

$$S_w = \sum_{\mu=1}^{n} w_\mu^2 \left[y_\mu - f(x_\mu, a_1, \ldots, a_m)\right]^2 \quad . \tag{IV.40}$$

If no weights are given, assume $w_\mu = 1$ for $\mu = 1, 2, \ldots, n$. There are various methods available which could be used to accomplish this minimization. These include, gradient methods, steepest descent, and others. Some of these require

the first and some require the first and second partial derivatives of S_w to be evaluated (either explicitly or by some approximation). The methods available in the PEG system are chosen for their simplicity from the users viewpoint. The only item for which a user must provide the code is the function $f(x, a_1, \ldots, a_m)$.

The two methods currently implemented in the data-fitting program are the direct search method and the interactive minimizer, both of which are described in detail in Chapter V. Both methods consider the function S_w and search for its minimum by systematic variation of the parameters.

Other methods could be used to perform the required minimization. Appendix C contains a discussion of how additional or substitute methods could be employed by the PEG system.

CHAPTER V

INTERACTIVE MINIMIZER AND DIRECT SEARCH

In this chapter we shall describe two minimization methods employed by PEG. The first is an interactive minimizer with which a user can locate a minimum by looking at certain graphs displayed on the CRT and making appropriate lightpen commands. The second method, called Direct Search, is an algorithm for internal (non-interactive) use. Some of the parameters of Direct Search can be specified by a user under the implementation described in Chapter III.

Other minimization methods can be substituted or added to those currently implemented in the PEG system. Appendix C contains a discussion of how this could be accomplished. Some references appropriate to a study of other methods of minimization are Zangwill [1967], Witte and Holst [1964], Fletcher [1965], Fletcher [1968] and Box [1965].

A. Interactive Minimizer

Given a functional defined on E^n a frequently arising problem is to find a point in E^n which minimizes the functional. If we denote the functional by $F(a_1, a_2, \ldots, a_n)$, where $\{a_i\}_{i=1}^n$ are the parameters, the problem can be generally stated as follows:

Global Minimization Problem

Given $F(a)$ defined for $a \in E^n$

Find $a^* \in E^n$ such that

$$F(a^*) \leq F(a) \text{ for all } a \in E^n$$

These are often constraints on some of the parameters in which case we are restricted to some subset of E^n. There are various algorithms which will

compute a solution for such a problem but the minimum is most often a local minimum and not necessarily a global minimum. The program to be described here will converge to a local minimum with certain minimum restrictions.

1. The Interactive Method of Minimization

The method of minimization employed by the interactive minimizer is a Gauss-Seidel type method. It is based on a search for a point where the partial derivative of F with respect to each parameter is zero. At such a point F will be minimized or maximized and the interactive environment makes the distinction obvious.

Assume the functional to be minimized is $F(a)$ where $a = (a_1, a_2, \ldots, a_n) \in E^n$. The method involves choosing new values for a_i, $i = 1, 2, \ldots, n$ one at a time and then repeating this process until the relative changes in the $\{a_i\}$ are small enough to claim convergence. This is accomplished by solving the following set of simultaneous equations iteratively.

$$\frac{\partial F(a)}{\partial a_i} = 0, \; i = 1, 2, \ldots, n \; . \qquad (V.1)$$

Given initial values for $\{a_i\}_{i=1}^n$, the Gauss-Seidel method for solving such a set of equations is to solve the i^{th} equation for a_i, $i = 1, 2, \ldots, n$, while using the most recently calculated values for a_j, $j \neq i$. The n equations are then solved repeatedly until convergence is obtained.

To illustrate, if $n = 3$, and the initial values are $a = (a_1^1, a_2^1, a_3^1)$ we have the following steps:

a. Set $k = 1$.

b. Set $a_2 = a_2^k$, $a_3 = a_3^k$ and solve $\frac{\partial F(a)}{\partial a_1} = 0$ for a_1^{k+1} .

c. Set $a_1 = a_1^{k+1}$, $a_3 = a_3^k$ and solve $\frac{\partial F(a)}{\partial a_2} = 0$ for a_2^{k+1} .

d. Set $a_1 = a_1^{k+1}$, $a_2 = a_2^{k+1}$ and solve $\frac{F(a)}{\partial a_3} = 0$ for a_3^{k+1}.

e. If the changes $|a_i^k - a_i^{k+1}|$, i = 1, 2, 3, are less than some convergence criterion, say $|a_i^k| \cdot \epsilon$, where ϵ is a small number, claim convergence (if k > 1).

f. If $|a_i^k - a_i^{k+1}|$, i = 1, 2, 3, are not small (enough) set k = k + 1 and go back to step (b).

In a more general version of the Gauss-Seidel method the choice of which equation to solve next can be made according to some criterion such as the relative change in the corresponding parameter on the previous iteration. This iterative method can be used to solve both linear and non-linear sets of equations.

In the interactive environment the solution of each equation in the set (V.1) or at each step (b), (c), or (d) in the above example is accomplished by choosing a point by hand (and eye) which is the minimum point on a plotted (displayed) graph. An example of such a graph is shown in Fig. 5.1. Consider F(a) with $a = (a_1, a_2, \ldots, a_n)$ and hold all parameters fixed except a_j. Having a guess, say a_j, for the solution of Eq. (V.1) for i = j, then we allow a_j to take on a sequence of equally spaced values in the interval $\left[a_j - \delta a_j, a_j + \delta a_j\right]$. This will give us points on a curve which represents $F_j(a)$, where $F_j(a)$ denotes F(a) considered as a function of a_j alone with a_i, $i \neq j$ held fixed. A minimum point on this curve corresponds to the value of a_j which satisfies Eq. (V.1) for i = j.

2. Interactive Minimizer Applied to Least Squares

The functionals to be minimized in the data-fitting program of Chapter III are the sums of squares of the residuals. That is, we are given N data points, $\{(x_i, y_i)\}_{i=1}^N$, and an approximating function, $f(x, a_1, a_2, \ldots, a_n)$. For any given values of the parameters $\{a_i\}_{i=1}^n$ we compute the sum of squares of the

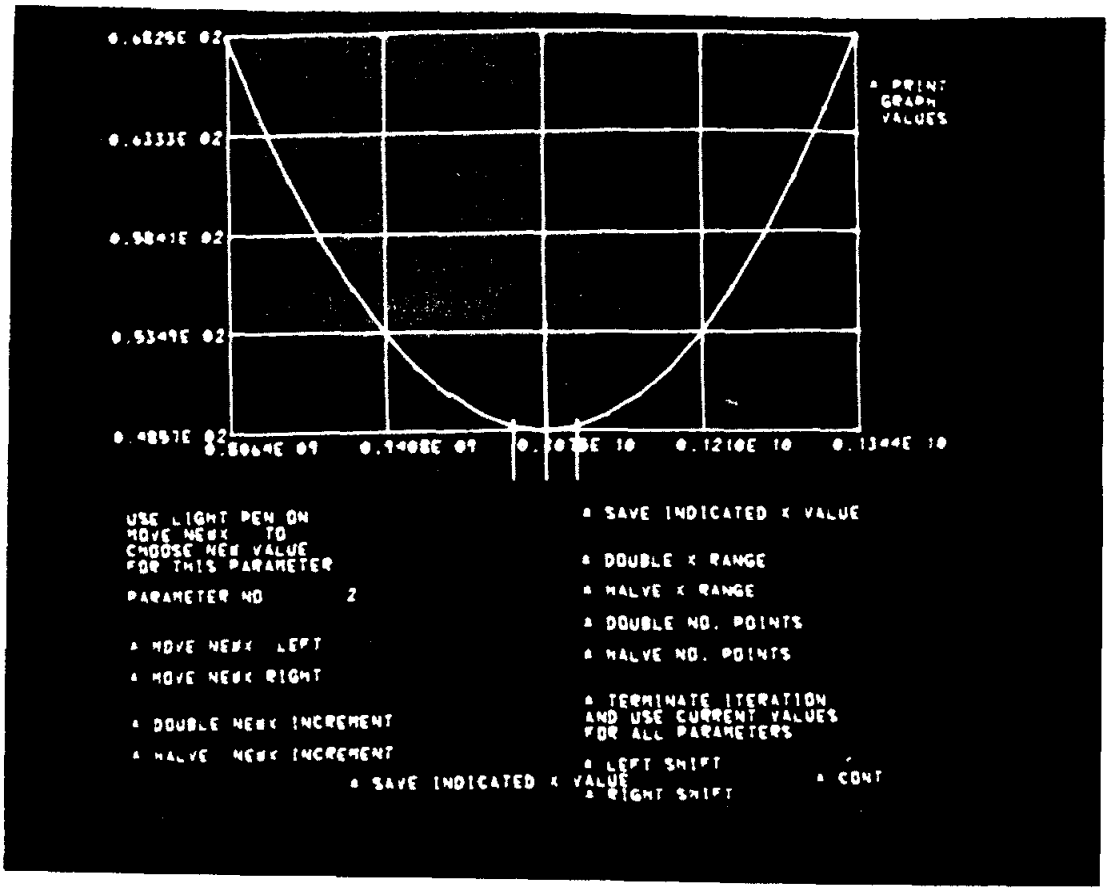

FIG. 5.1--A plot of $f_2(a)$ as a_2 varies.

residuals, denoted by F, as

$$F = \sum_{i=1}^{N} \left[y_i - f(x_i, a_1, a_2, \ldots, a_n) \right]^2 . \qquad (V.2)$$

F is a functional defined on the space E^n. The least squares problem is to determine a point $a^* = (a_1^*, a_2^*, \ldots, a_n^*) \in E^n$ such that

$$F(a^*) \leq F(a) \text{ for all } a \in E^n \ . \qquad (V.3)$$

Therefore this least squares problem is in the class of problems discussed in paragraph V.A.1 and is solvable by the Gauss-Seidel method used in the interactive minimizer.

Inequality constraints may be imposed on the parameter space very simply. This is accomplished by programming the sum of squares calculation to result in a very large number if any parameter violates its constraint. This will essentially put a "wall" around the allowable region in parameter space. For example, if a_i is constrained by $a_i > 0$, the routine to calculate the sum of squares could return 10^{70} if $a_i \leq 0$ and the true sum of squares if $a_i > 0$. Then the graph of $F_i(a)$ as a_i varies over an interval including zero would have a large jump discontinuity at zero. This would clearly indicate to the viewing user that negative values of a_i were quite undesirable for minimizing F. Other forms of constraints could be handled similarly.

3. Convergence of the Interactive Minimizer

Convergence properties and proofs for several iterative minimization schemes are discussed by Elkin in his thesis, see Elkin [1968]. Among those discussed is the Gauss-Seidel method used by the interactive minimizer.

Convergence for an arbitrary functional, F, is an open question. However, if certain assumptions on F are made convergence can be proven. First we establish some notation. Let a^k be a vector in E^n containing the values of the

parameters after the k^{th} iteration of the Gauss-Seidel method where each iteration involves the cyclical solution of the n equations

$$\frac{\partial F}{\partial a_i} = 0, \quad i = 1, 2, \ldots, n \quad . \tag{V.3}$$

Let $D \subset E^n$ be the space of vectors on which F is defined. Now we can state a relatively weak theorem (weak because we require $F'(a) = 0$ to have a unique solution) as given by Elkin [1968]. E^n denotes Euclidean n-space and R denotes the field of real numbers.

Theorem V.1. Assume $F'(a) = 0$ has a unique solution, that the sequence $\{a^k\}$ lies in a compact subset of D, and that $F:D \subset E^n \longrightarrow R$ is continuously differentiable, then $\{a^k\}$ converges to a^* and $F'(a^*) = 0$.

A stronger result also given by Elkin [1968] but due to Ostrowski [1964] is possible when $\|a^k - a^{k+1}\| \longrightarrow 0$. The norm is the Euclidean norm,

$$\|a^k - a^{k+1}\| = \left[\sum_{i=1}^{n} \left(a_i^k - a_i^{k+1}\right)^2\right]^{1/2} \quad .$$

Theorem V.2. Let $F:D \subset E^n \longrightarrow R$ have a continuous derivative on an open set D; suppose the sequence $\{a^k\}$ is contained in a compact set $D_0 \subset D$, and assume that $F'(a^k) \longrightarrow 0$, $\|a^k - a^{k+1}\| \longrightarrow 0$, and the set $\{a:F'(a) = 0\}$ consists only of isolated points. Then the sequence $\{a^k\}$ converges to a limit a^* and $F'(a^*) = 0$.

Proofs of both theorems are given in Elkin [1968].

With these theorems a given functional such as the sum of squares based on a user defined approximating function can be tested to see if it has the required properties to ensure convergence. The only a priori knowledge we have about the sequence $\{a^k\}$ is that a sequence does indeed exist with the property that $F(a^{k+1}) < F(a^k)$ and that such a sequence can be guaranteed by the interactive minimizer. This is based on the fact that as we look at displayed graphs such as shown in Fig. 5.1 we can be sure that we are always reducing the value of the functional (sum of squares) as we move to a lower point on the curve. Whether or not we can choose a sequence $\{a^k\}$ interactively such that $\|a^k - a^{k+1}\| \longrightarrow 0$ will undoubtedly depend on the functional involved.

4. Using the Interactive Minimizer

The use of the interactive minimizer is quite straightforward. For each parameter of the functional to be minimized a display such as that shown in Fig. 5.2 is placed on the CRT.

The purpose of the display and the lightpen commands is to choose as accurately as possible the minimum point of the plotted curve. The number or index of the parameter under consideration is displayed for identification. The purpose and function of each of the lightpen options is detailed below.

Three vertical lines are superimposed on the displayed graph. The longer center line (of the three) indicates the current value of the parameter under consideration. It is essentially a pointer used to select a new value for the parameter. The two shorter lines to the right and left

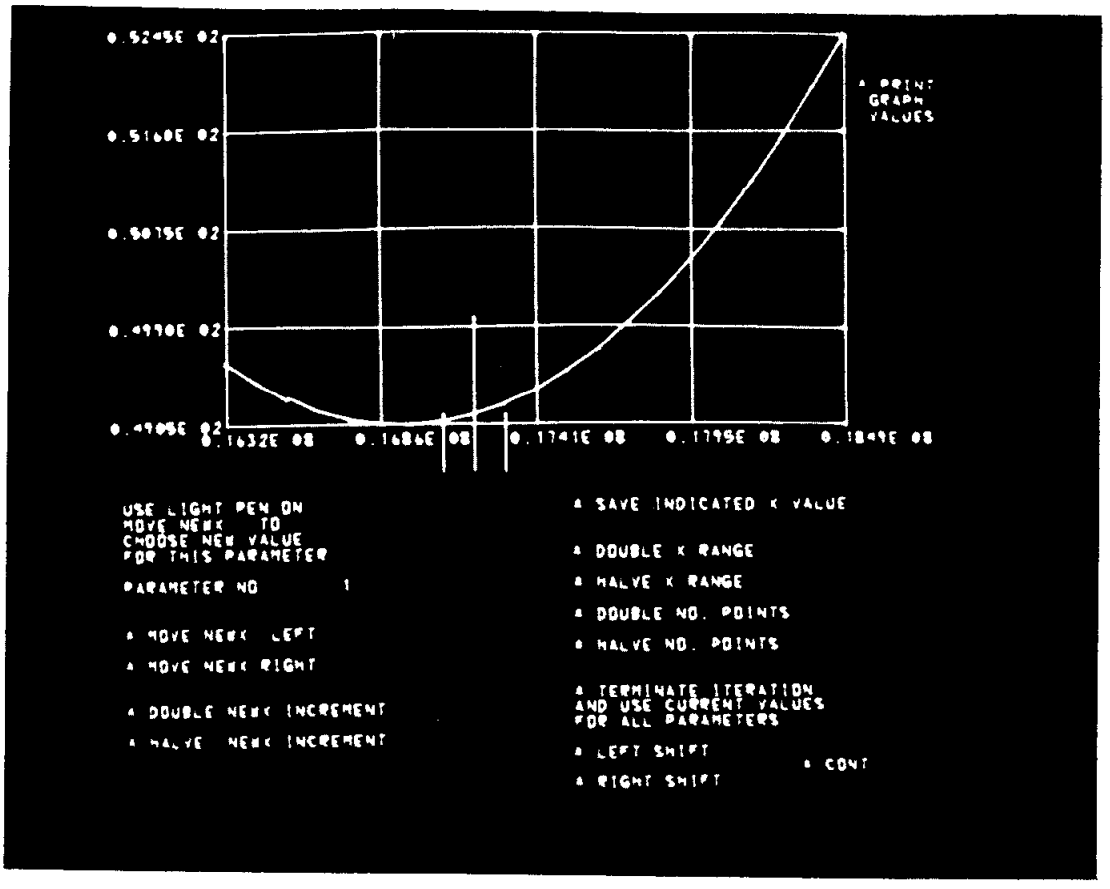

FIG. 5.2--Interactive minimizer at work (*MOVE NEWX LEFT has been selected twice).

of the pointer indicate where the pointer will move to if it is moved right or left by a lightpen command. The distance separating the pointer and its increment markers may be doubled or halved at will by lightpen commands, thereby allowing the positioning of the pointer to be as accurate as desired.

Now we will detail the purpose of each lightpen option shown in Fig. 5.2:

MOVE NEWX LEFT ---------- moves the pointer to the left — the distance indicated by the left-hand marker.

MOVE NEWX RIGHT ---------- moves the pointer to the right — the distance indicated by the right-hand marker.

DOUBLE NEWX INCREMENT ------ doubles the distances between the pointer and the right- and left-hand markers (i.e., moves the markers away from the pointer).

HALVE NEWX INCREMENT ------- halves the distances between the pointer and the right- and left-hand markers.

Given that the displayed values for the abscissa (the parameter at hand) range from $a_i - \delta a_i$ to $a_i + \delta a_i$ with a_i at the center, we have the following lightpen options.

DOUBLE X RANGE ---------- changes the display to plot abscissa values in the interval $\left[a_i - 2\delta a_i, a_i + 2\delta a_i\right]$ (wide angle lens).

HALVE X RANGE ---------- changes the display to plot abscissa values in the interval $\left[a_i - \frac{1}{2}\delta a_i,\ a_i + \frac{1}{2}\delta a_i\right]$ (telephoto lens).

LEFT SHIFT ---------- changes the display to plot abscissa values in the interval $\left[a_i - \frac{3}{2}\delta a_i,\ a_i + \frac{1}{2}\delta a_i\right]$. This essentially moves the viewing window to the left to examine the curve in that direction.

RIGHT SHIFT ---------- changes the display to plot abscissa values in the interval $\left[a_i - \frac{1}{2}\delta a_i,\ a_i + \frac{3}{2}\delta a_i\right]$. This essentially moves the viewing window to the right to examine the curve in that direction.

The remaining lightpen commands are:

DOUBLE NO. POINTS ---------- this doubles the number of points which are computed to represent

	the displayed curve. The effect is to "smooth" the curve.
HALVE NO. POINTS ----------	this halves the number of points which are computed to represent the displayed curve. Halving the number of points reduces the amount of calculation (machine CPU time) necessary to prepare a curve for display.
SAVE INDICATED X VALUE ----- (requires lightpen on CONT to execute)	this saves the value indicated by the pointer as a new value for the parameter under consideration. The display is then changed to plot the functional as the next parameter is varied.
TERMINATE ITERATION -------- AND USE CURRENT VALUES FOR ALL PARAMETERS (requires lightpen on CONT to execute)	this allows termination of the Gauss-Seidel iteration at this point. That is, a complete cycle through all parameters is not necessary. This option allows escape from the iteractive minimizer.

After the cyclical adjustment of all the parameters, each iteration invokes the display shown in Fig. 5.3. This summarizes the results of all iterations up to that point. The table on the left shows the functional value at the beginning of each iteration. On the right side of the display the results of the iteration just

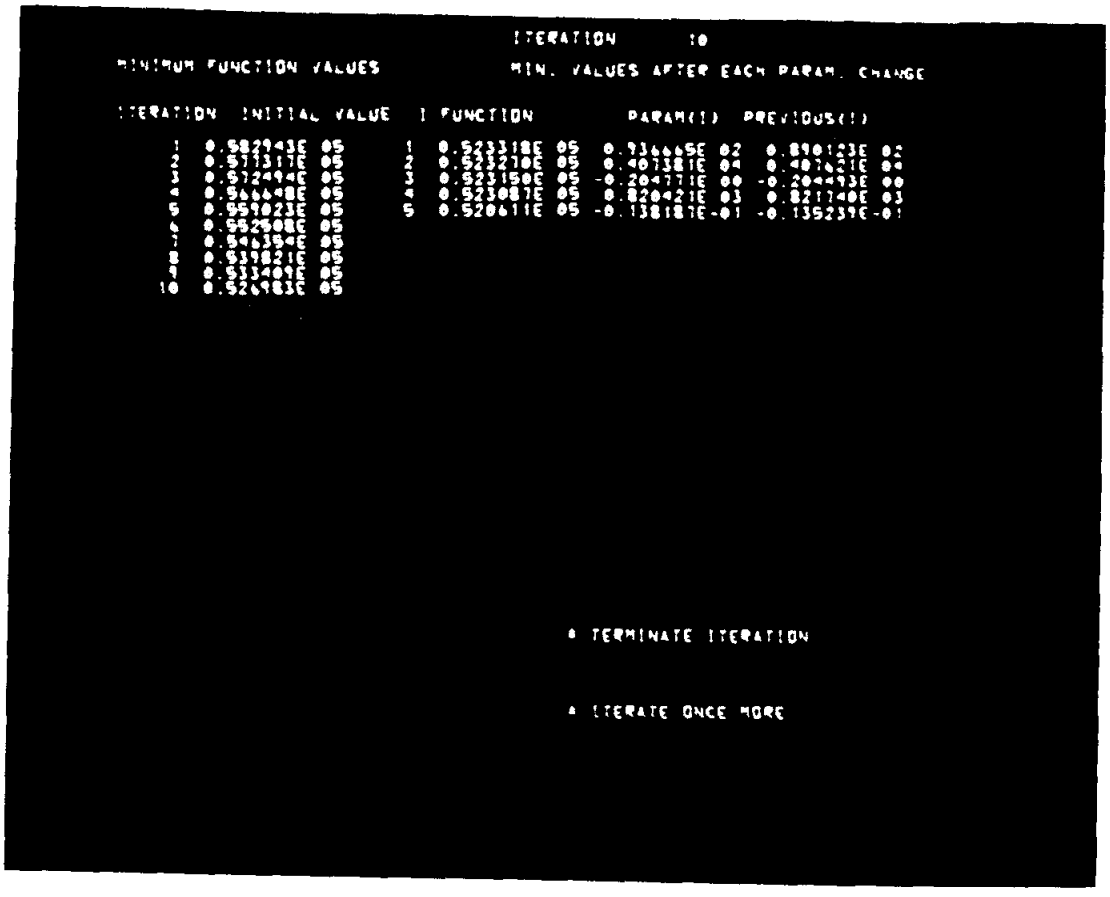

FIG. 5.3--Iteration summary display.

completed are displayed. For each parameter, the previous value and the new value just chosen are tabulated along with the functional value after the new value for that parameter was selected. This information allows the user to decide when to terminate iteration. His decision may be based on the change or lack of change in functional value or on the size of the changes in the parameters, or both.

5. Advantages and Disadvantages of Interactive Minimization

Some of the advantages accrued by minimizing with the interactive minimization routine are listed below. Not all of these benefits are peculiar to the problem of minimization but they do appear sometimes dramatically in this environment.

a. "Round-off" appears graphically and allows iteration to round-off limits.

b. A good "feel" for the problem can be obtained from the graphical display of the curves as each parameter is varied.

c. Weak dependence of certain parameters is shown graphically.

d. Other local minima appear graphically and can be easily examined.

e. Insight into the minimization problem can be gained. For example, when iterating by hand it is obvious when convergence is very slow.

Some of the above merit further discussion. Round-off, for example, is illustrated by Fig. 5.4. Such a display is obtained by having a very small interval spanned by the abscissa. In this case the functional is evaluated for values

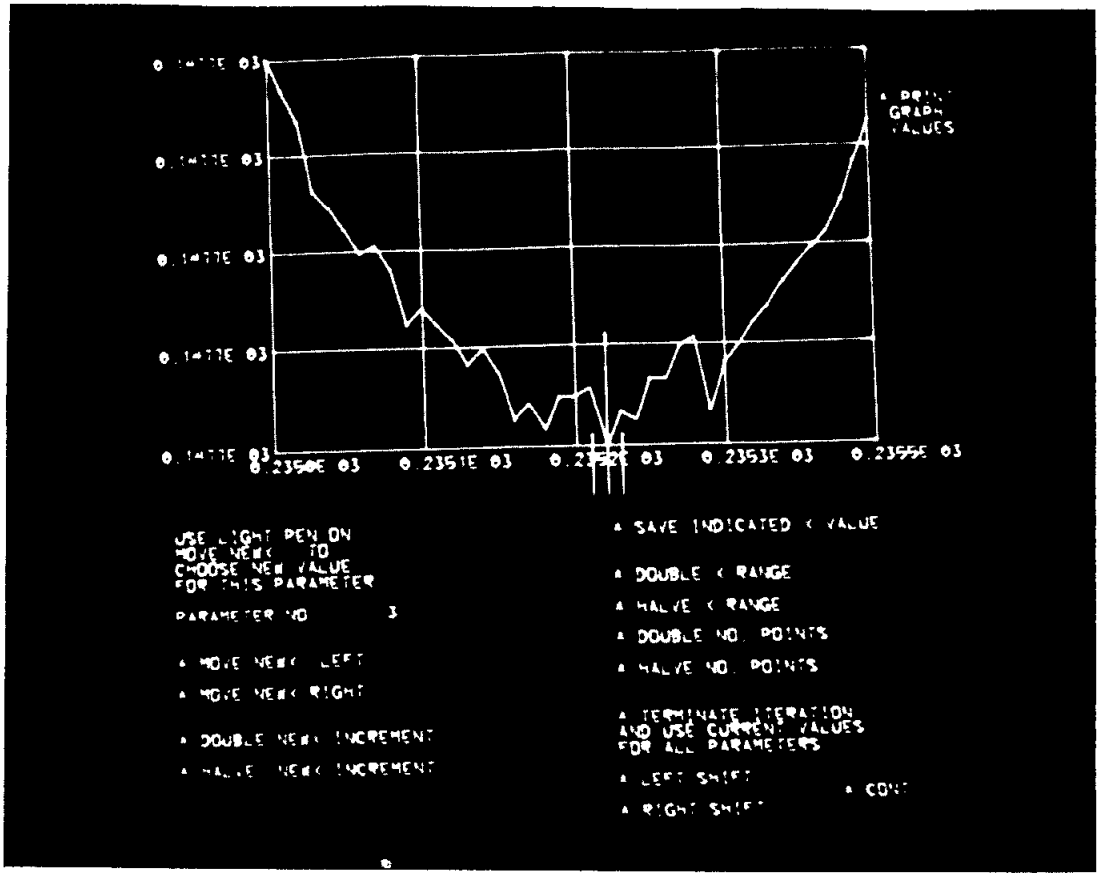

FIG. 5.4--Round-off limits.

of the parameter which are very close and round-off errors in the calculation of the ordinate values cause the "non-smooth" character of the curve. When the graph takes on such an appearance there is no further improvement possible in the value of the parameter associated with the display.

Weak dependence of a parameter is shown graphically by a plot that has no vertical variation. In other words, if the functional does not change as the parameter varies over the specified interval, the plot will show only a horizontal line for the functional values. For example if the changes in a certain parameter change the functional values in the eighth significant digit, but we are using a computer which only holds seven significant digits, this fact will be graphically demonstrated by showing no dependence on that parameter.

Figure 5.5 illustrates the appearance of subsidiary minima. In the example shown, the extra local minima are obviously unwanted as they represent higher functional values, but in other cases the location of a lower local minimum could be detected by this graphical display. This is obviously a very desirable feature in a minimization program.

Another example of subsidiary minima is shown in Fig. 5.6. In this case the parameter involved was an angle and the functional dependence was obviously modulo some fraction of π. Therefore the extra local minima are meaningless but again the graphical representation makes this fact obvious.

The most obvious disadvantage of this interactive minimizer is that convergence often requires several iterations and a considerable amount of lightpen activity by the user. Another criticism might be that after the values have been computed for graphing the function as each parameters varies it would be simple to have the computer perform the minimum search that is done by the user. However, if the minimization were done completely automatically the human

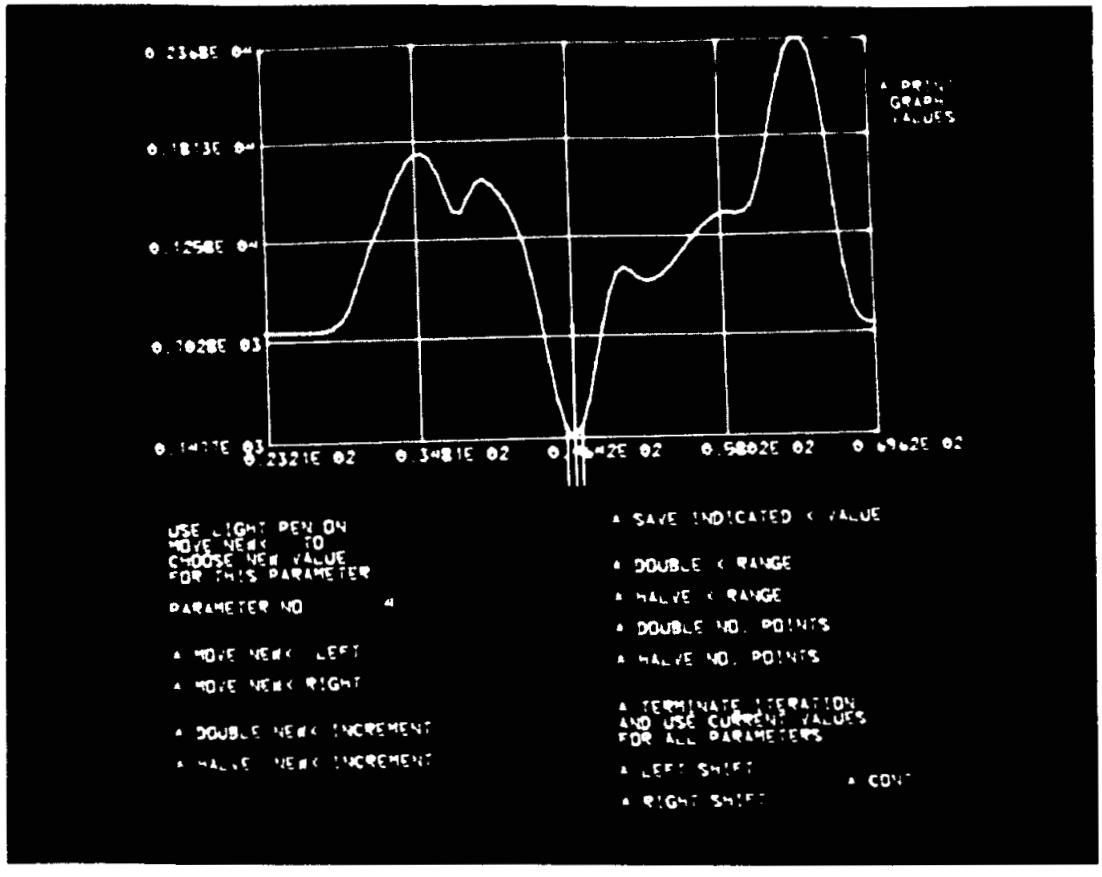

FIG. 5.5--Multiple local minima.

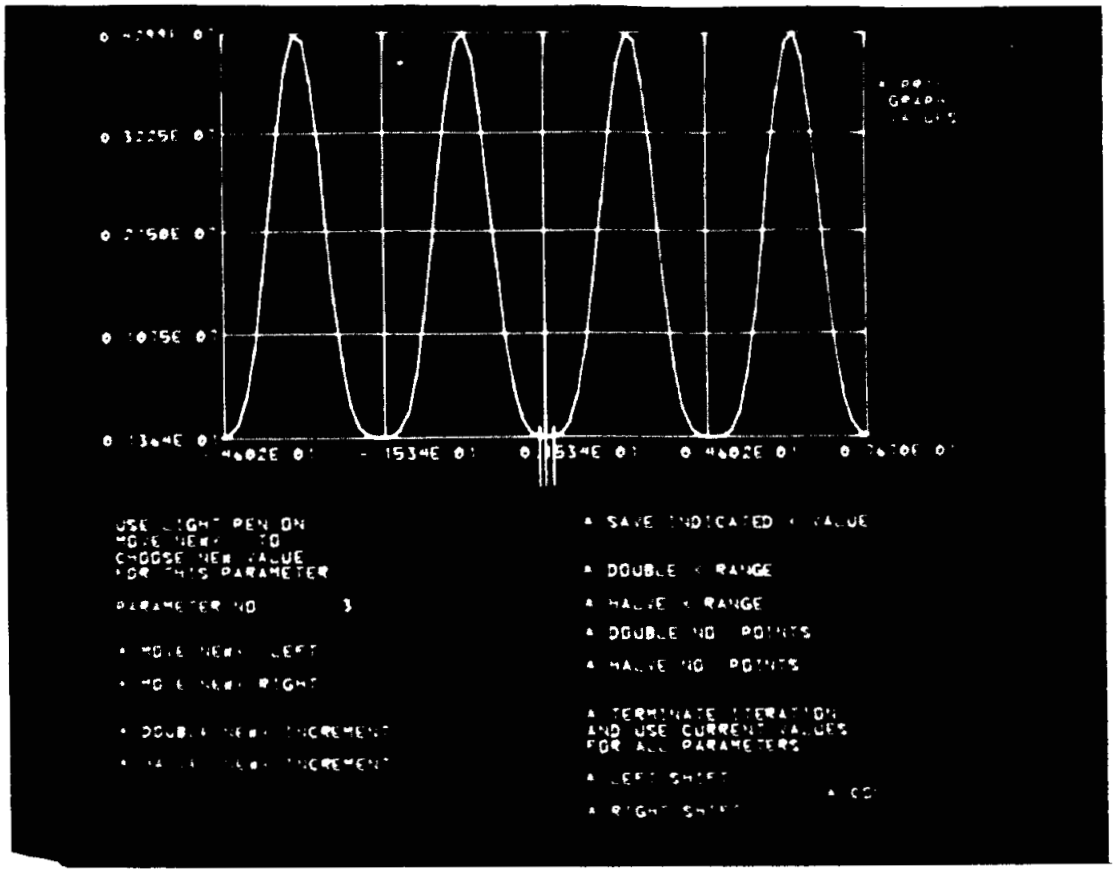

FIG. 5.6--More local minima.

insight that might detect a subsidiary minimum or some anomalous behavior is eliminated. A combination of interactive and automatic minimization would probably be best. A user could interactively locate a point that seemed close to a minimum and the computer could then automatically iterate until convergence was achieved.

B. Minimization by Direct Search

Minimization algorithms can be classed in three broad classes. The first class includes quadratic methods which involve the first and second derivatives of the functional whose minimum is sought. Secondly we have the class of linear methods which require knowledge of the first derivatives. This includes steepest descent, Newton's method, gradient and conjugate gradient methods. The third class includes directional methods requiring no derivative calculation. The Gauss-Seidel method used by the interactive minimizer and the direct search algorithm to be described here are in the latter class.

1. The Direct Search Method

Given a priori knowledge about a functional and its derivatives a quadratic or linear method could in many cases be chosen to minimize the functional efficiently and accurately. However, in the data-fitting environment where we choose to allow a user to define his own approximating function we do not have any a priori knowledge of the function. We also do not wish to restrict the data-fitting to functions of a certain class in order to ensure that first and second derivatives of the sum-of-squares functional exist and are computable. Requiring a user to provide code to compute derivatives is a restriction we want to avoid. Finally, the presence of constraints on certain parameters can cause unwanted complications if a quadratic or linear minimization algorithm is being used,

whereas constraints are relatively easy to handle with a directional method. For these reasons a directional method is embodied in the data-fitting program. It is used to minimize the sum of squares for user defined functions, as well as for the least squares adjustment of the joints in the case of spline approximation with variable joints.

There are many variations on directional methods for locating functional minima. One method is the Gauss-Seidel method as discussed in paragraph V.A.1 and used by the interactive minimizer. The method chosen for use in the interactive data-fitting program is the direct search method discussed by Hooke and Jeeves [1961]. This method varies the parameters by an initial step size until no further decrease in functional value can be obtained. The step size is then decreased and the search for a minimum is continued. This process is repeated until the step size is below some specified tolerance at which point convergence is claimed. An Algol implementation of the direct search method is given by Kaupe [1963], a corrected and modified version is given by Bell and Pike [1966] and a related remark is given by DeVogelaere [1968].

The details of the method as applied to a functional $F(x)$, where $x \in E^n$ are as follows:

Given: $F(x)$, a functional.

x^0, an initial guess for the solution vector.

δx, an initial step size for varying the parameters.

ρ, a multiplier, $\rho < 1$, used to decrease step size.

δ, a minimum tolerance on the step size.

Direct search method. The algorithm involves two methods of choosing a better value for the minimum point. First exploratory moves are made by cyclically varying each parameter by δx (add δx then subtracting δx) and testing

for decrease in the functional after each parameter change. Any point at which a decrease is seen is kept as a better value for the corresponding parameter. Following each exploratory move a "pattern" move is made. This consists of incrementing those parameters which were changed during the preceding exploratory move by $\pm \delta x$. The sign of the change corresponds to the sign of the change made in the exploratory move. Whenever an exploratory move yields no decrease in the functional, the step size is decreased by replacing δx by $\rho \cdot \delta x$, and the search is continued. Convergence is assumed when δx becomes less than δ, the given tolerance.

A two dimensional example of the use of direct search will illustrate the method.

In Fig. 5.7 we illustrate the direct search method by following the search path it generates as it seeks a minimum. The path shown is that generated by the version of direct search given by Bell and Pike [1966]. We begin at the point $x^0 = (2, 2)$ with $F(x^0) = 6$ and

$$\begin{aligned} \delta x &= 2 \text{ (initial step size)} \\ \rho &= 1/2 \text{ (step size reduction factor)} \\ \delta &= 1.5 \text{ (stopping step size criterion —} \\ &\quad \text{stop when } \delta x \leq \delta). \end{aligned}$$

Starting at point (2, 2) the routine first performs an exploratory move, then a pattern move, etc. The details are as follows:

1^{st} explore	tries (4, 2) gets $7 > 6 \Rightarrow$ fail
	tries (0, 2) gets $10 > 6 \Rightarrow$ fail
	tries (2, 4) gets $5 < 6 \Rightarrow$ success
1^{st} pattern	moves to (2, 6) gets 4

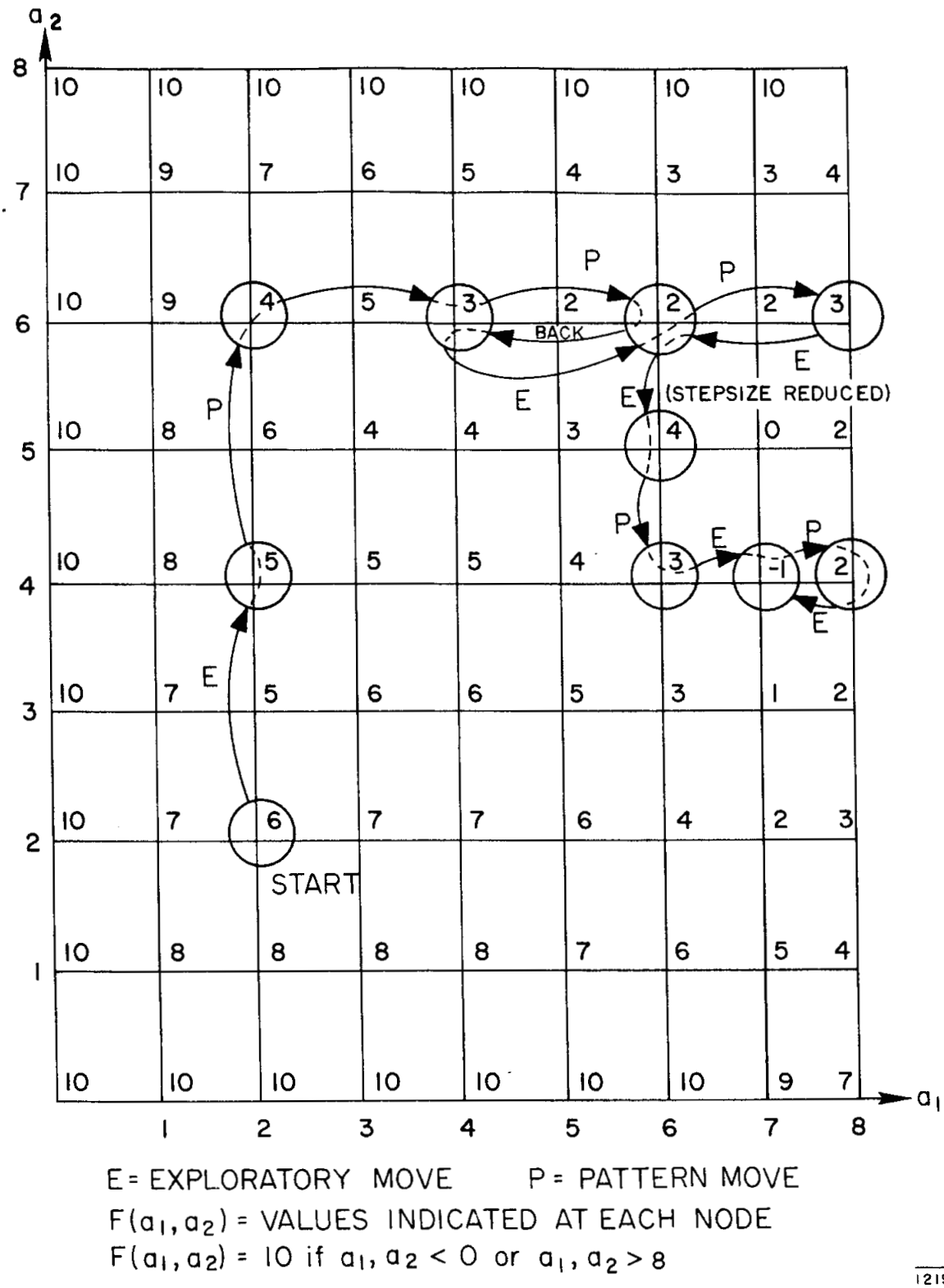

FIG. 5.7--A two-dimensional direct search path.

2^{nd} explore tries (0, 6) gets 10 > 4 ⇒ fail

tries (4, 6) gets 3 < 4 ⇒ success

tries (4, 8) gets 10 > 3 ⇒ fail

tries (4, 4) gets 5 > 3 ⇒ fail

2^{nd} pattern moves to (6, 6) gets 2

3^{rd} explore tries (8, 6) gets 3 > 2 ⇒ fail

tries (4, 6) gets 3 > 2 ⇒ fail

tries (6, 4) gets 3 > 2 ⇒ fail

tries (6, 8) gets 10 > 2 ⇒ fail

Since explore failed to go down we go back to (4, 6).

4^{th} explore tries (6, 6) gets 2 < 3 ⇒ success

tries (6, 4) gets 3 > 2 ⇒ fail

tries (6, 8) gets 10 > 2 ⇒ fail

3^{rd} pattern moves to (8, 6) gets 3

5^{th} explore tries (10, 6) gets 10 > 3 ⇒ fail

tries (6, 6) gets 2 < 3 ⇒ success

tries (6, 4) gets 3 > 2 ⇒ fail

tries (6, 8) gets 10 > 2 ⇒ fail

Since explore got no lower than before the last pattern move, we go back to (6, 6) the point found by the last explore.

6^{th} explore tries (8, 6) gets 3 > 2 ⇒ fail

tries (4, 6) gets 3 > 2 ⇒ fail

tries (6, 4) gets 3 > 2 ⇒ fail

tries (6, 8) gets 10 > 2 ⇒ fail

No reduction at this point causes the step size to be multiplied by ρ. Then another explore is performed.

7^{th} explore	tries (7, 6) gets 2 = 2 ⟹ fail
	tries (5, 6) gets 2 = 2 ⟹ fail
	tries (6, 5) gets 1 < 2 ⟹ success
4^{th} pattern	moves to (6, 4) gets 3
8^{th} explore	tries (5, 4) gets 4 > 3 ⟹ fail
	tries (7, 4) gets -1 < 3 ⟹ success
	tries (7, 3) gets 1 > -1 ⟹ fail
	tries (7, 5) gets 0 > -1 ⟹ fail
5^{th} pattern	moves to (8, 4) gets 2
9^{th} explore	tries (9, 4) gets 10 > 2 ⟹ fail
	tries (7, 4) gets - 1 < 2 ⟹ success
	tries (7, 3) gets 1 > -1 ⟹ fail
	tries (7, 5) gets 0 > -1 ⟹ fail

Since explore got no lower than before the last pattern move we go back to (7, 4), the point found by the last explore.

10^{th} explore	tries (8, 4) gets 2 > -1 ⟹ fail
	tries (6, 4) gets 3 > -1 ⟹ fail
	tries (7, 3) gets 1 > -1 ⟹ fail
	tries (7, 5) gets 0 > -1 ⟹ fail

Convergence is claimed at this point since no further reduction in the functional $F(a_1, a_2)$ can be achieved with the current step size and the step size is already less than the stopping criterion.

2. A Modification to the Direct Search Method

As presented in the literature the direct search method uses the same increment (step size) for all parameters. A modification to improve convergence in some cases has been introduced in the implementation of direct search used in the data-fitting program. The change provides for a different step size for each parameter. Thus if two parameters vary by several orders of magnitude their corresponding step sizes also vary by several orders of magnitude. This is accomplished by computing the initial step size for each parameter as a fraction, r, of its absolute value. The initial step size for the i^{th} parameter is then calculated as $\delta x_i = r \cdot |x_i^o|$, where x^o is the starting guess for the solution vector. If $x_i^o = 0$, we take $\delta x_i = r$. Subsequent reductions in step size are accomplished by replacing δx_i by $\rho \cdot \delta x_i$, where $\rho < 1$ is the specified fractional reduction in step size. At the same time r is replaced by $\rho \cdot r$ and when r becomes less than the specified tolerance, convergence is claimed. In this manner the step size for each parameter remains meaningful in terms of the relative size of that parameter. This modification is also discussed in a remark on direct search submitted for publication (see Smith [1968]).

An example of a user defined approximating function which requires these step size modifications is a sum of exponential terms

$$f(x) = \sum_{i=1}^{m} a_i e^{-b_i x} \quad . \tag{V.4}$$

For an actual problem, the parameters $\{a_i\}$ had values in the neighborhood of 10^8 whereas the parameters $\{b_i\}$ had values in the neighborhood of 10^{-2}. A fixed step size appropriate for the $\{a_i\}$ would have been meaningless if applied to the $\{b_i\}$.

Other variations of the direct search algorithm have been tried by various researchers. For example, the method has been modified to consider parameter variations in directions other than parallel to the parameter axes in parameter space. Randomly chosen angles in parameter space have been tried with some success as directions to explore rather than the simple scheme described here. Many other variations on the direction to explore and step size can be considered.

3. Direct Search Applied to Least Squares

The comments given in paragraph V.A.2 apply here as well. The imposition of constraints can be implemented as suggested there by introducing a large number for the sum of squares if any parameter violates its constraint. The effect will be that of putting a "wall" in parameter space that will repel any attempts by the direct search explorations to penetrate it.

4. Step Size and Tolerance Criteria for Direct Search

There are several externally controllable features in the direct search algorithm. These are:

a. The initial step size (fraction of parameter size).

b. The step size tolerance criterion.

c. The step size reduction factor.

d. The maximum number of iterations.

Values for these quantities are set automatically by the data-fitting program, but they may also be set by a user at run time. A display shown in Fig. 5.8 allows entry of new values for these items or use of the preset values. The display shows the preset values so a user can decide whether or not he would like to try something different. Each item may be either left alone or changed.

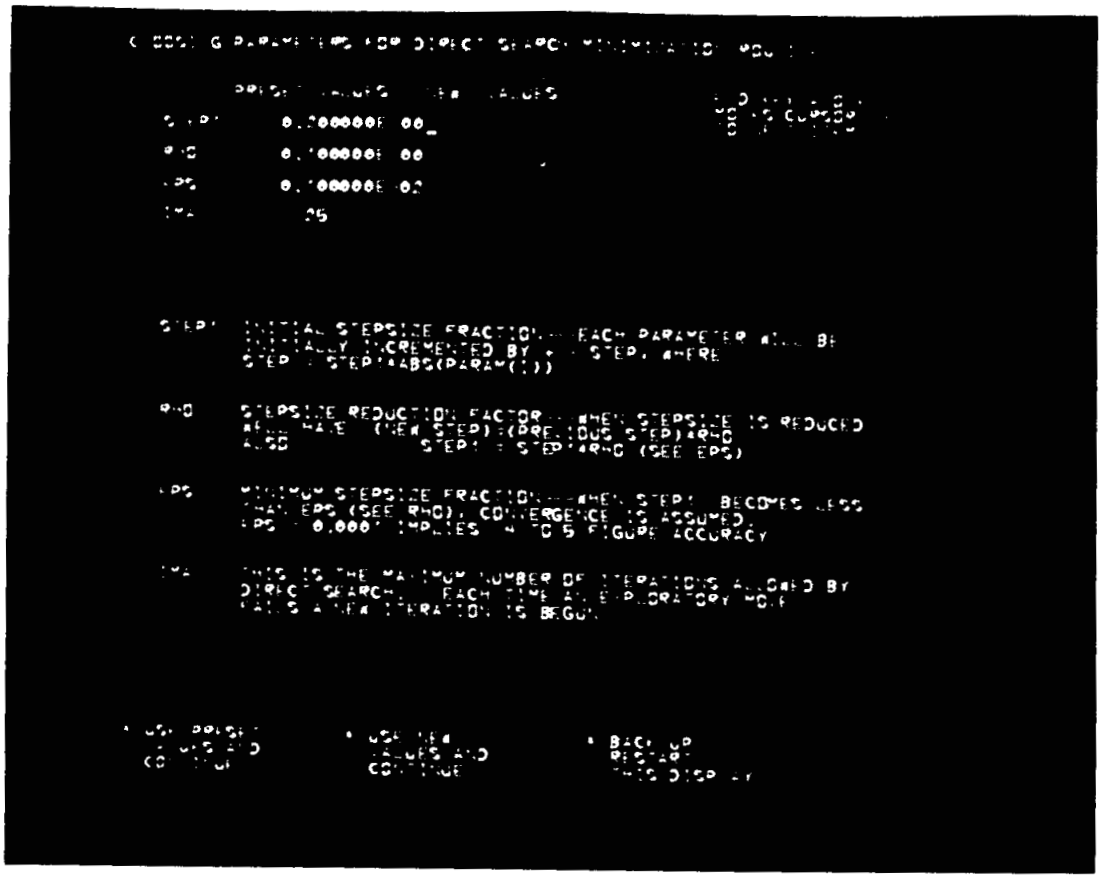

FIG. 5.8--Entry of direct search parameters.

The values for the above parameters to direct search as stored in the data-fitting program are:

initial step size fraction	$= r = .2$	initial step size on parameter a_i will be $.2\|a_i\|$
step size reduction factor	$= \rho = .1$	step size will be multiplied by .1 for each reduction
minimum step size fraction	$= \epsilon = .001$	each reduction will multiply the step size fraction by ρ. When r becomes $< \epsilon$ convergence is assumed.
maximum number of iteration	$= 25$	essentially eachtime an exploratory move fails a new iteration is begun

CHAPTER VI

THE USE OF INTERACTIVE DATA-FITTING

In this chapter we give four examples of data-fitting problems which have been solved using PEG, the interactive data-fitting program described in Chapter III. Two examples employ user defined functions and the third uses polynomials (or splines) to obtain a "good" fit for further calculations. The fourth example also employs a user defined function but it is a special case involving an implicit function (see discussion of UFUNC4, paragraph III.B.4) to determine a least squares approximation to data representing an ellipse.

A. Background Plus Gaussian Curves

The data for this problem came from experimental measurements (counts) using the accelerator on the Stanford campus operated by the physics department. The physicist working on the problem was Dr. Barnie Hughes (Hughes [1968]).

Figures 6.1 and 6.2 show typical data sets for this problem. The function supposed to approximate the data is given by (VI.1). Table 3 shows the data in tabular form.

The approximating function is

$$f(x) = a_0 + a_1 x + a_2 x^2 + \sum_{i=1}^{m} b_i \, e^{-\frac{(x-\hat{x}_i)^2}{s_i^2}} . \qquad \text{(VI.1)}$$

This represents a quadratic background plus a sum of Gaussian curves. The parameters to be determined are a_0, a_1, a_2 and $\{b_i, \hat{x}_i, s_i\}_{i=1}^{m}$. The $\{b_i\}$ are the amplitudes, the $\{\hat{x}_i\}$ the center positions, and the $\{s_i\}$ the standard deviations

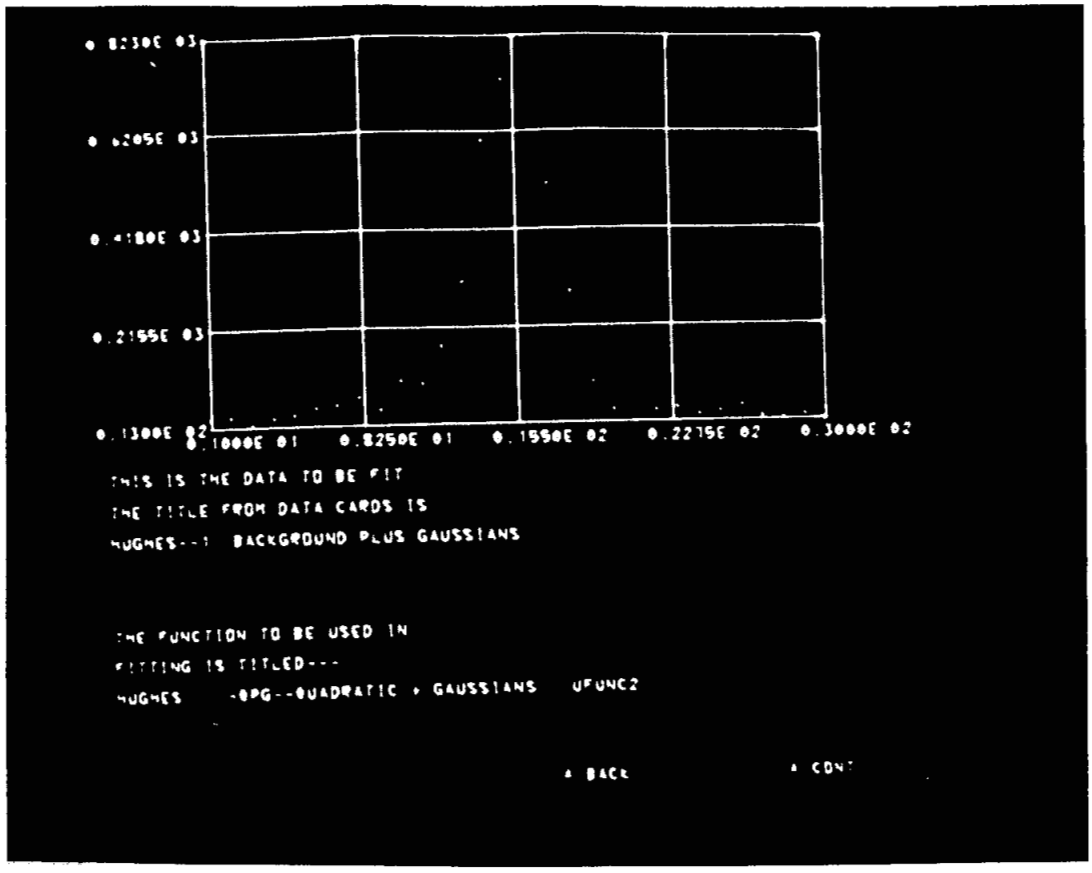

FIG. 6.1--Data to be approximated by (VI.1) (Sample 1).

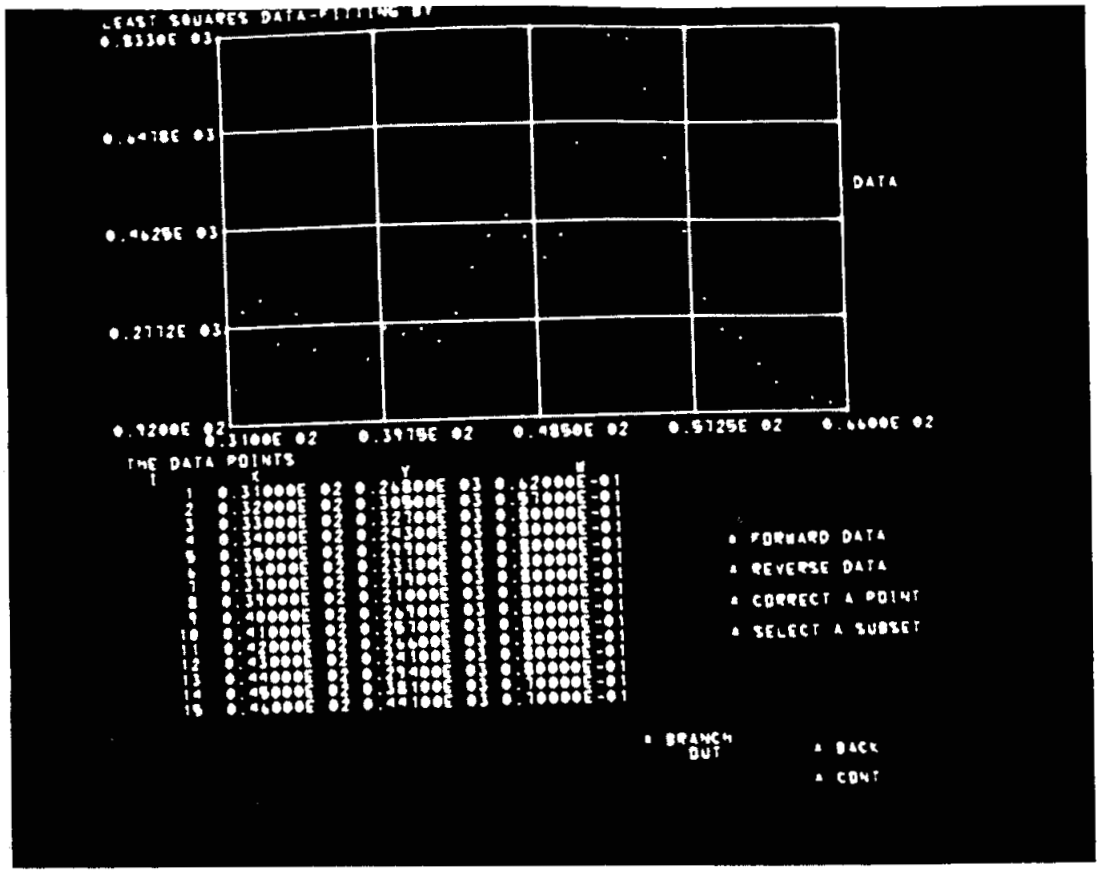

FIG. 6.2--Data to be approximated by (VI.1) (Sample 2).

TABLE 3

Data to be Approximated by Eq. (VI. 1)

	Sample 1			Sample 2		
i	X_i	Y_i	W_i	X_i	Y_i	W_i
1	1.0	29.0	.13	31.0	269.0	.062
2	2.0	32.0	.18	32.0	305.0	.057
3	3.0	16.0	.25	33.0	327.0	.09
4	4.0	29.0	.19	34.0	243.0	.09
5	5.0	35.0	.17	35.0	299.0	.09
6	6.0	50.0	.14	36.0	231.0	.09
7	7.0	57.0	.13	37.0	279.0	.09
8	8.0	72.0	.12	39.0	210.0	.08
9	9.0	46.0	.15	40.0	269.0	.08
10	10.0	105.0	.10	41.0	257.0	.08
11	11.0	99.0	.10	42.0	266.0	.08
12	12.0	179.0	.075	43.0	241.0	.08
13	13.0	312.0	.055	44.0	294.0	.09
14	14.0	604.0	.04	45.0	381.0	.07
15	15.0	733.0	.037	46.0	441.0	.07
16	16.0	823.0	.034	47.0	478.0	.06
17	17.0	508.0	.045	48.0	438.0	.06
18	18.0	287.0	.059	49.0	396.0	.06

Table 3 (continued)

	Sample 1			Sample 2		
i	X_i	Y_i	W_i	X_i	Y_i	W_i
19	19.0	95.0	.10	50.0	437.0	.06
20	20.0	39.0	.15	51.0	611.0	.05
21	21.0	13.0	.30	52.0	833.0	.05
22	22.0	35.0	.17	53.0	817.0	.05
23	23.0	41.0	.16	54.0	809.0	.05
24	24.0	26.0	.18	55.0	712.0	.05
25	25.0	32.0	.15	56.0	579.0	.06
26	26.0	44.0	.17	57.0	439.0	.06
27	27.0	21.0	.22	58.0	308.0	.07
28	28.0	16.0	.25	59.0	251.0	.08
29	29.0	22.0	.21	60.0	235.0	.08
30	30.0	33.0	.17	61.0	184.0	.09
31				62.0	148.0	.09
32				63.0	92.0	.11
33				64.0	115.0	.10
34				65.0	107.0	.11
35				66.0	111.0	.11

of the Gaussian curves. Some similar data sets were fit with $a_2 = 0$; thus using a linear background.

The approximation was accomplished by coding the function given in (VI. 1) as a user function and minimizing the sum of squares of residuals by the direct search method described in Chapter V. Figures 6.3 and 6.4 show the results of fitting the data by least squares. The data had weights associated with it which were derived from the square root of the number of counts taken at each data point. These weights are related to the confidence intervals (error bars) shown in Figs. 6.3 and 6.4.

The parameters shown in Figs. 6.3 and 6.4 correspond to those of (VI. 1) as follows:

TABLE 4

Parameter Correspondence for Gaussian Fit

Parameter		
1	a_0	the background polynomial coefficients
2	a_1	
3	a_2	
4	b_1	3 coefficients for each Gaussian curve
5	$\hat{x}_1$	
6	s_1	
7	b_2	
8	$\hat{x}_2$	
9	s_2	
.	.	
.	.	
.	.	

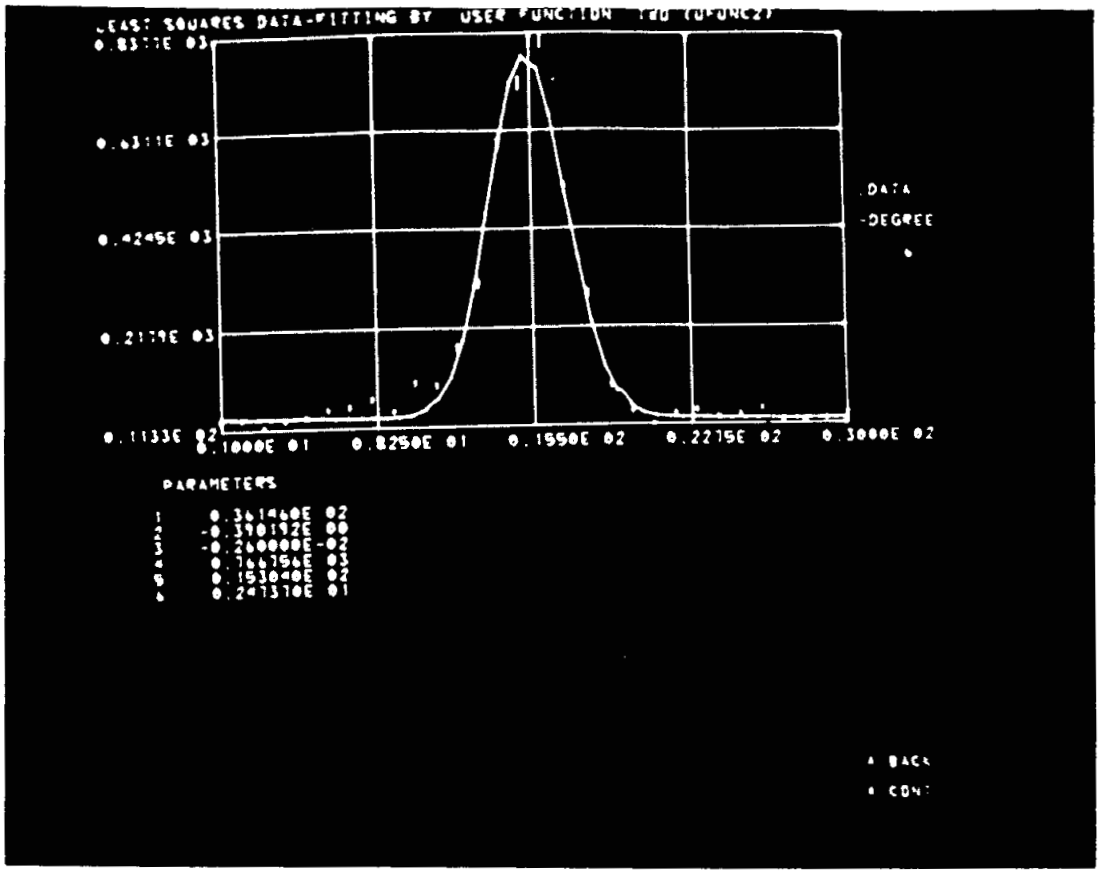

FIG. 6.3--The computed approximation to Sample 1 (with error bars).

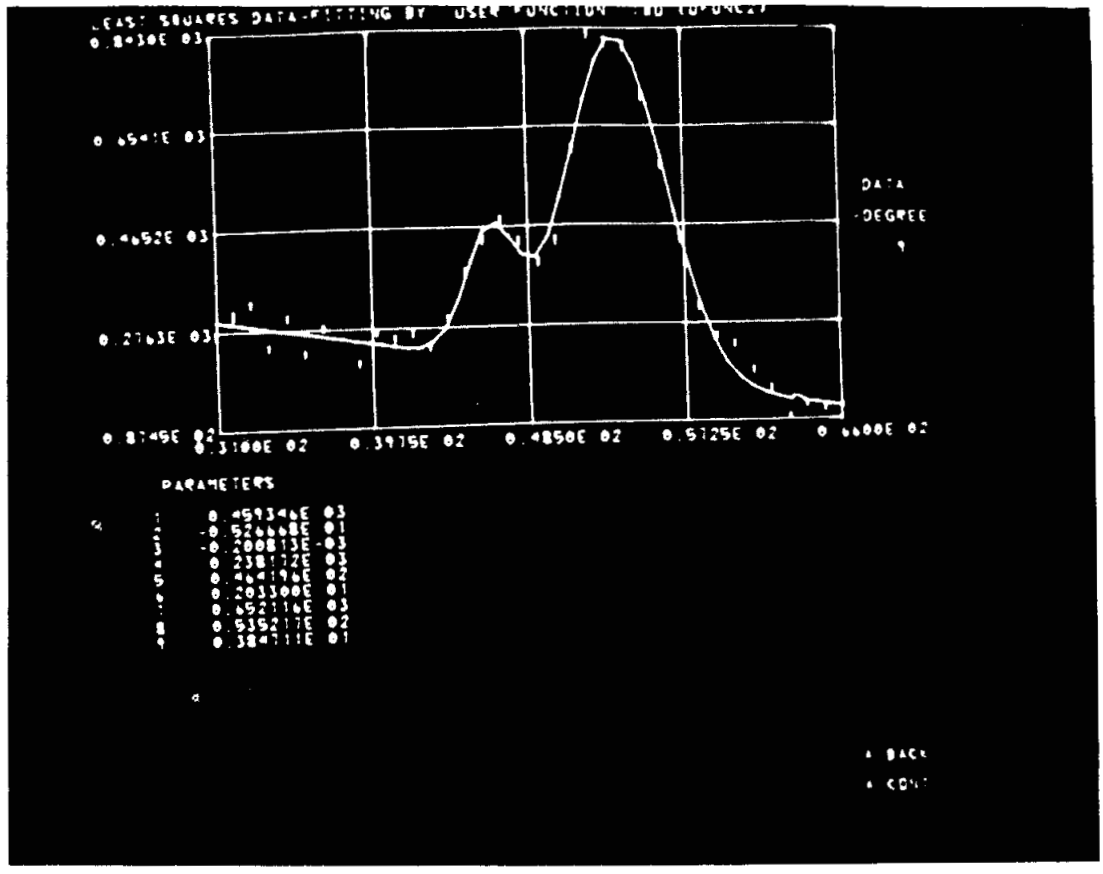

FIG. 6.4--The computed approximation to Sample 2 (with error bars).

B. Fitting With Breit-Wigner Functions

The data shown in Fig. 6.5 was obtained from Dr. Luke Mo (SLAC) (Mo [1968]) and is supposed to be approximated by a function involving a sum of Breit-Wigner functions. The approximating function is given by

$$f(x) = \sum_{i=0}^{m} a_i x^i + \sum_{j=1}^{3} b_j (BW)_j \qquad \text{(VI.2)}$$

where

$$(BW)_j = \frac{1}{(x - \hat{x}_j)^2 + \left(\frac{\gamma_j}{2}\right)^2} \quad .$$

Thus we have a polynomial background (degree 5 or so) plus a sum of three Breit-Wigner functions.

Figure 6.6 shows the results of approximating the data by (VI.2) coded as a user defined function with m = 5. The parameter correspondence is given by Table 5.

C. Knee-Action Data

Mr. Robert Potthoff, a graduate student in Mechanical Engineering at Stanford University, has performed a study of the action of the human knee (see Potthoff [1968]). By performing carefully controlled experiments, ankle and foot positions were measured while the knee was constrained to rotate about a particular axis. An example of the type of data curve obtained by this method is shown in Fig. 6.7 and in tabular form in Table 6. An analysis of the knee-action was then carried out using an analytic expression to represent the curves exemplified by Fig. 6.7. Therefore a least squares fit was needed to approximate the empirical data.

Since the curve given by the experimental points is not single-valued, the entire curve could not be approximated by the same polynomial or spline function.

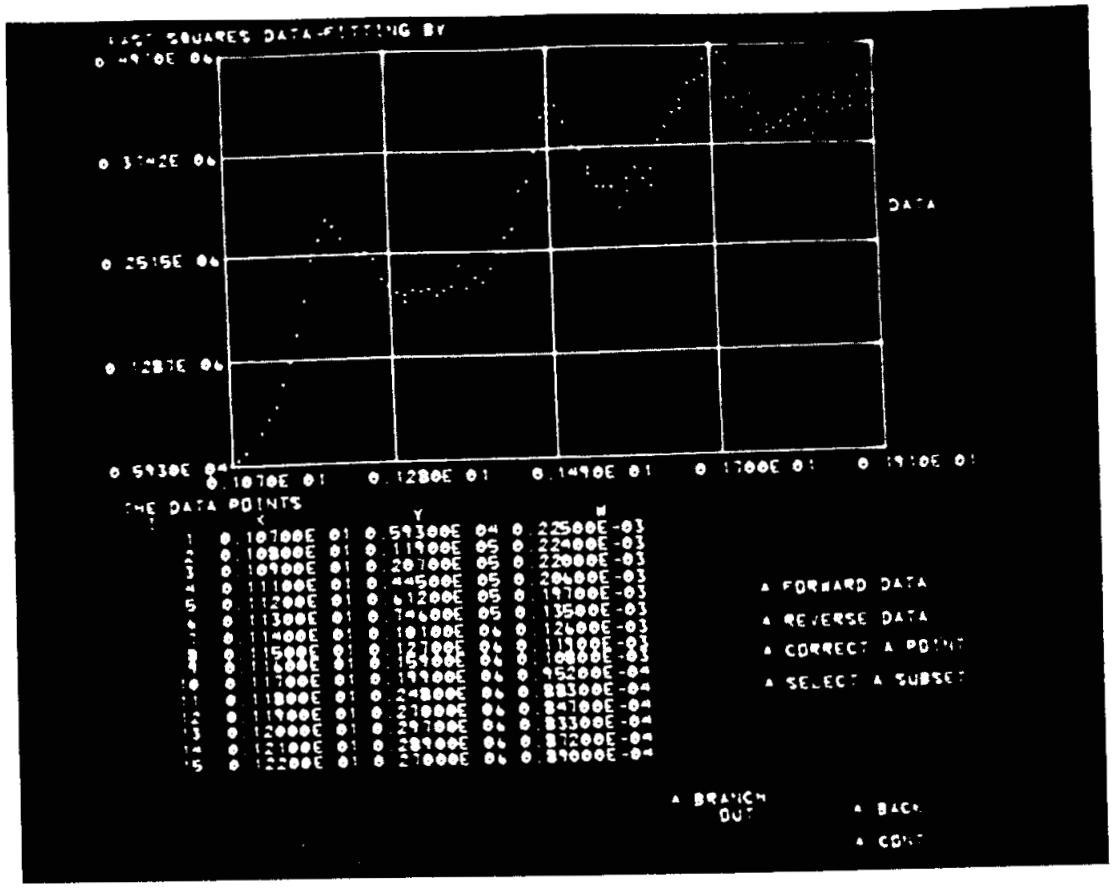

FIG. 6.5--Data to be fit by Breit-Wigner functions.

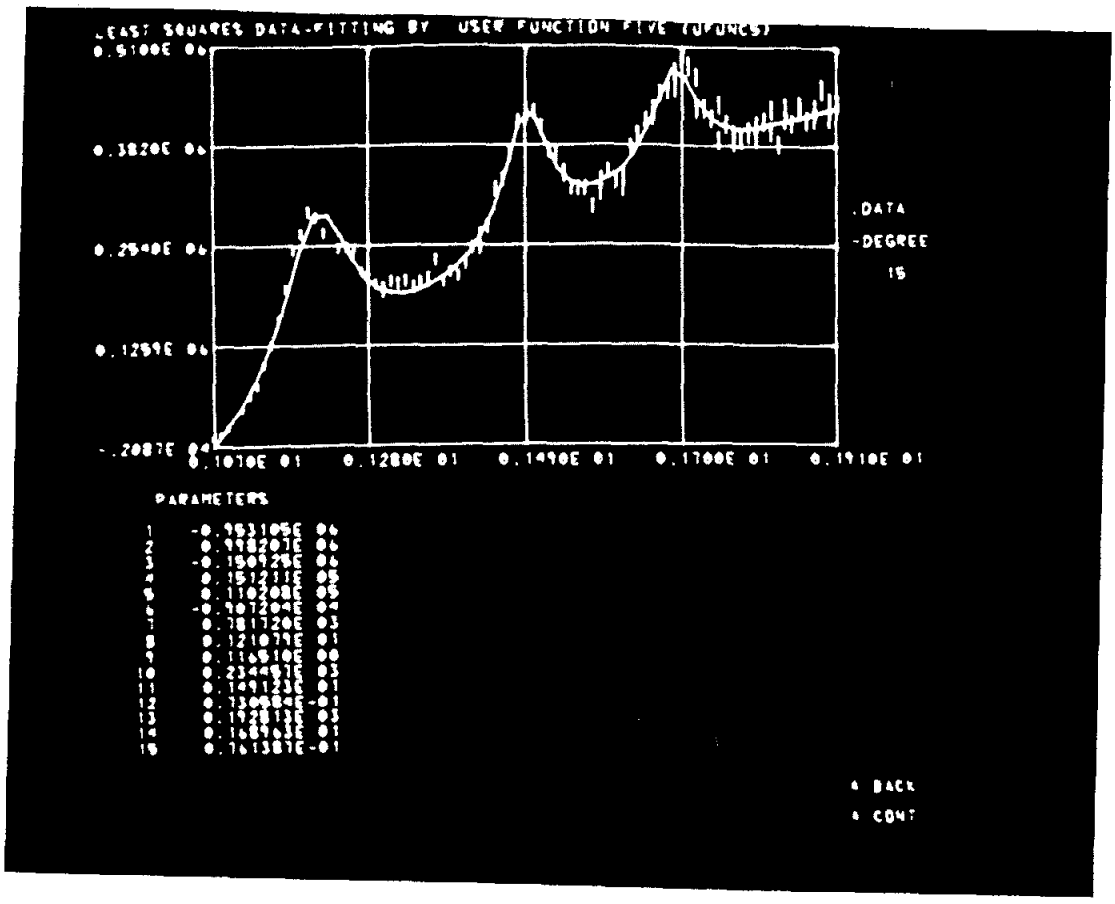

FIG. 6.6--Fit by Breit-Wigner functions.

TABLE 5

Parameter Correspondence for Breit-Wigner Fit

Parameter		
1	a_0	Polynomial background
2	a_1	
3	a_2	
4	a_3	
5	a_4	
6	a_5	
7	b_1	1st Breit-Wigner function
8	$\hat{x}_1$	
9	γ_1	
10	b_2	2nd Breit-Wigner function
11	$\hat{x}_2$	
12	γ_2	
13	b_3	3rd Breit-Wigner function
14	$\hat{x}_3$	
15	γ_3	

TABLE 6

A Knee-Action Curve

i	X_i	Y_i	i	X_i	Y_i
1	-41.20	-8.7	18	-8.50	46.60
2	-41.20	-8.4	19	-5.90	47.65
3	-41.20	-8.0	20	-2.80	48.80
4	-41.20	-7.05	21	1.30	49.90
5	-41.30	-5.15	22	5.00	50.75
6	-41.20	-0.95	23	8.50	51.25
7	-40.80	3.95	24	11.50	51.50
8	-39.90	8.55	25	14.20	51.50
9	-38.70	13.05	26	16.20	51.45
10	-37.00	17.85	27	18.40	51.35
11	-34.40	23.15	28	20.40	51.15
12	-30.90	28.50	29	22.20	50.95
13	-26.20	33.85	30	23.40	50.75
14	-22.10	37.75	31	24.40	50.60
15	-17.60	41.30	32	25.50	50.45
16	-14.40	43.60	33	26.20	50.30
17	-11.30	45.25			

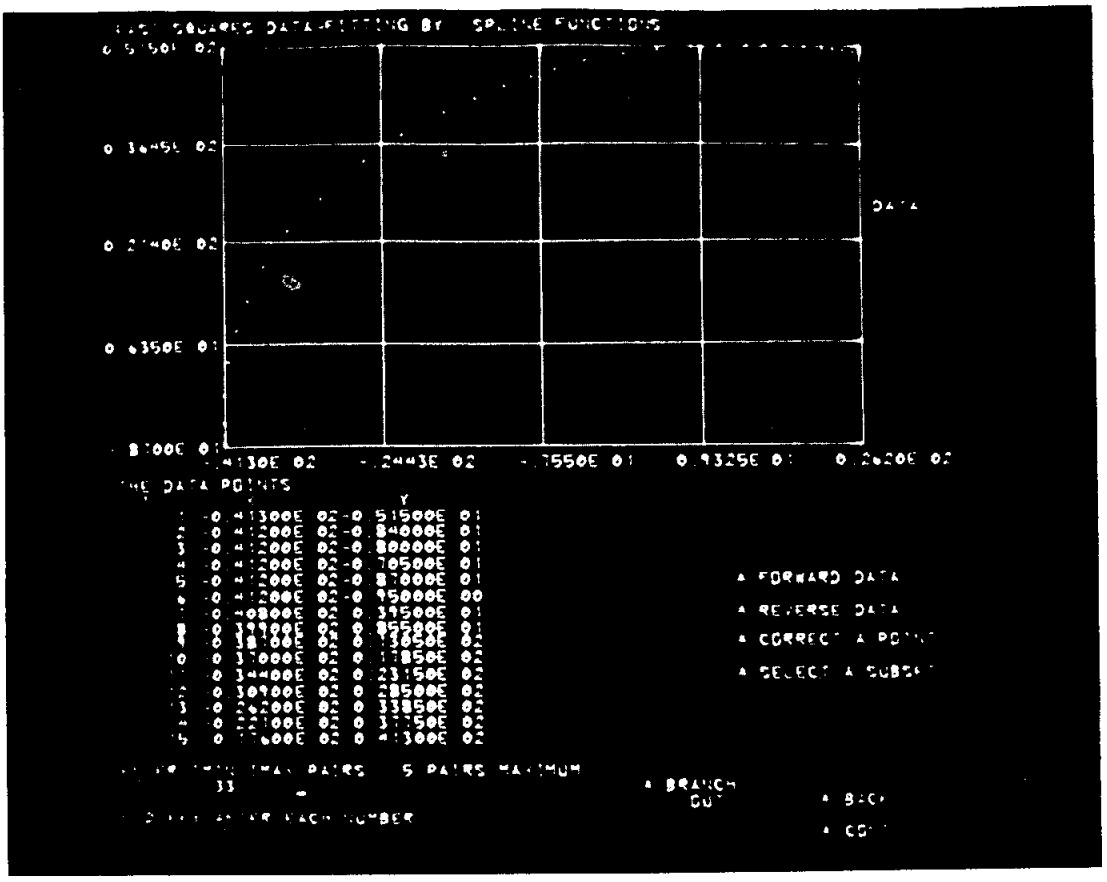

FIG. 6.7--A knee-action curve (multiple valued function).

Therefore the subset selection feature of the interactive data-fitting program was used to select a single-valued portion of the curve. For example, points 7 through 33 of the data given in Table 6 could be selected and then fit by a polynomial as shown in Fig. 6.8.

A second polynomial could then be fit to the remainder of the curve, points 1 through 6 using a different orientation of the axes. These two polynomials could then be used to represent the curve in the further analysis.

Figure 6.9 shows a spline function approximation to the same portion of the curve that is approximated by a polynomial in Fig. 6.8. This illustrates the fact that often a spline function of low degree can obtain an approximation which is as good as (or better than) a polynomial approximation of higher degree.

D. Fitting an Ellipse

The problem of fitting an ellipse to experimentally determined data arises in the area of accelerator physics. A particular problem is that of determining the phase space contours in an accelerator beam. If the phase space contour is known and is elliptical, then the particle optics downstream from the measuring point can be computed. The data is obtained by reducing digital counts to beam divergence and position. An actual set of data obtained from beam measurements is shown in Fig. 6.10. The data was obtained from Dr. Robert Allison (Allison [1968]) of the Radiation Laboratory at Berkeley.

1. Fitting an Ellipse by Linear Least Squares

There are various ways to solve a least squares problem and there are also various ways a least squares ellipse can be determined. A linear least squares problem can be solved by consideration of the normal equations or by consideration of the singular values decomposition of the rectangular matrix obtained by

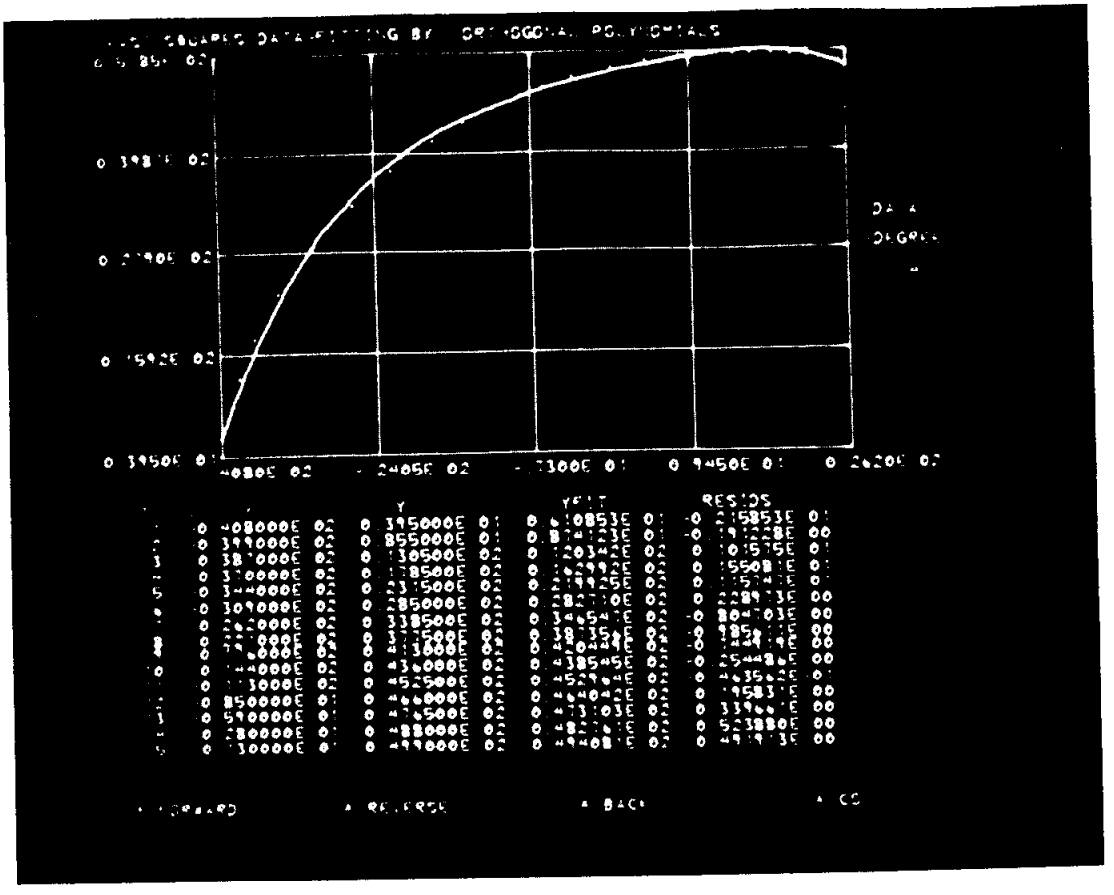

FIG. 6.8--Polynomial fit to subset of knee-action data (degree 4).

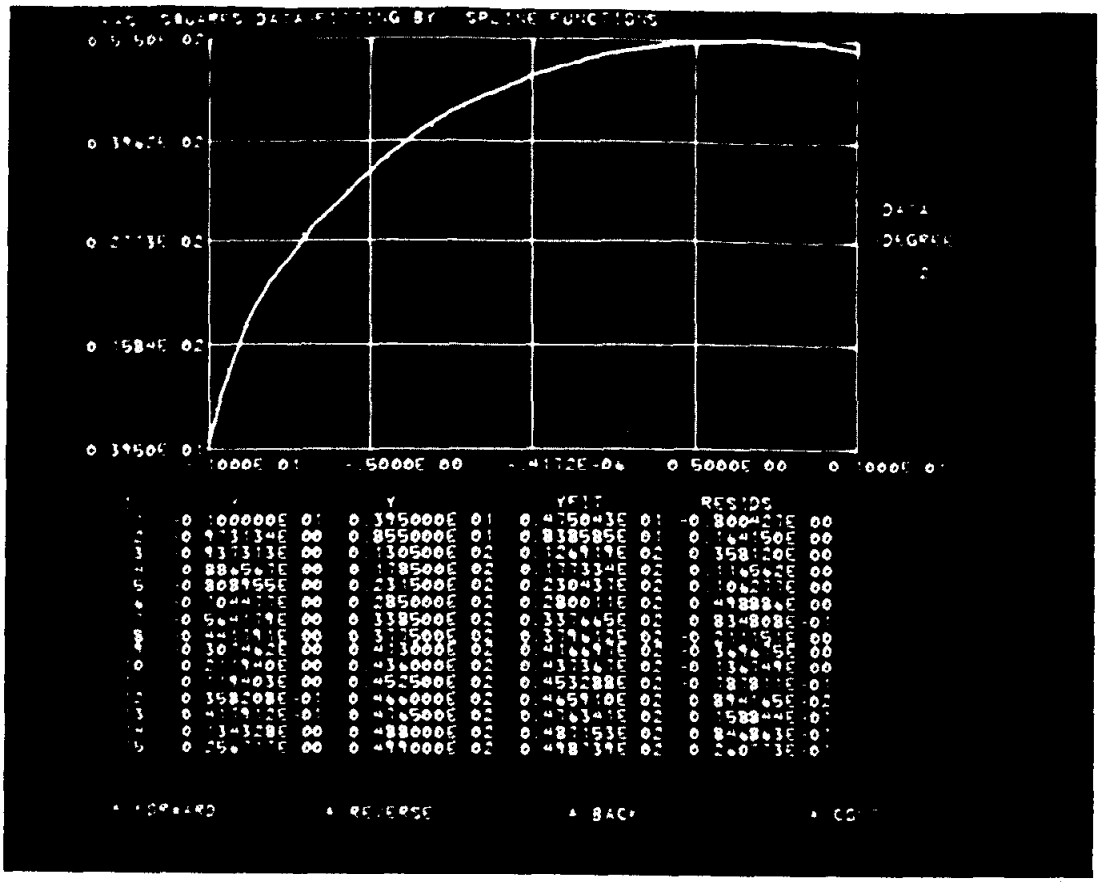

FIG. 6.9--Spline function fit to subset of knee-action data (degree 2).

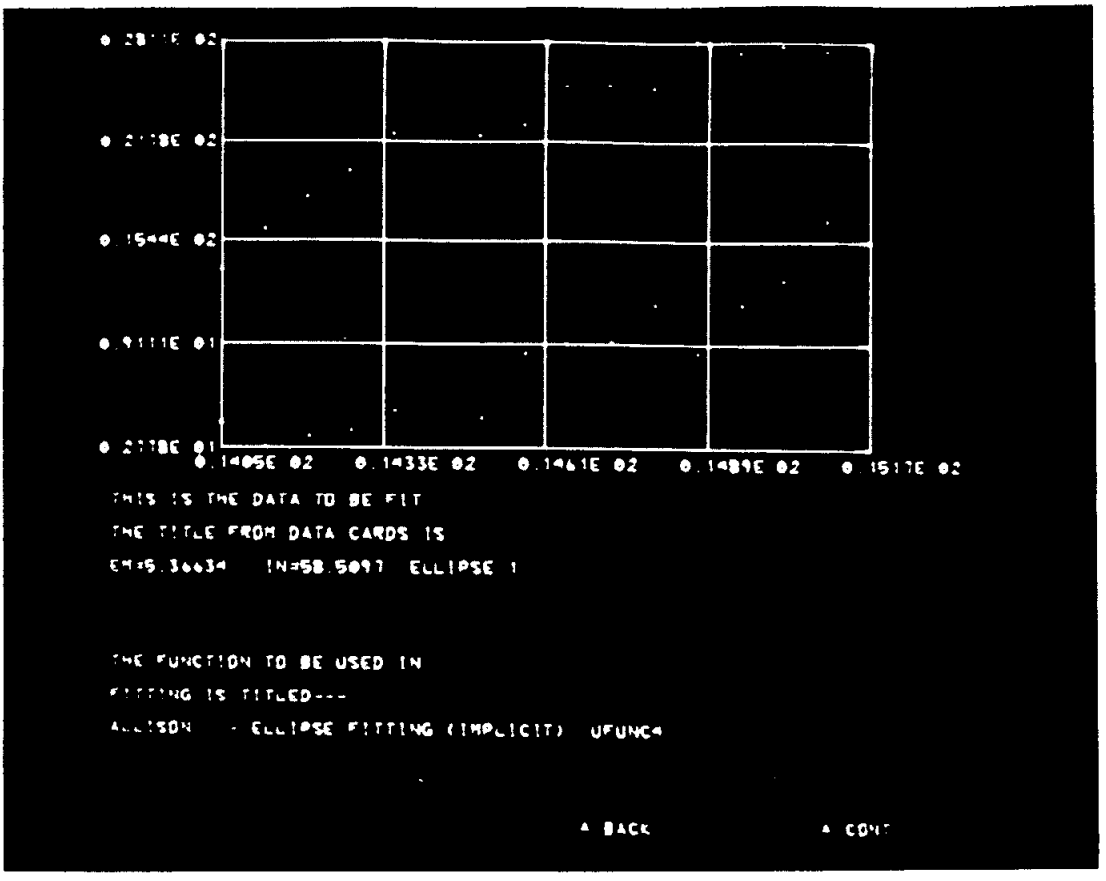

FIG. 6.10--Data representing an ellipse.

generating one row for each data point. A non-linear least squares problem must be solved iteratively. The least squares problem for an ellipse can be formulated either as a linear problem or as a non-linear problem. First we approach it as a linear problem.

The equation of an ellipse can be written as

$$Ax^2 + B\,xy + Cy^2 + Dx + Ey + F = 0 \qquad \text{(VI.3)}$$

with the restriction that $B^2 - 4\,AC < 0$ (for a general discussion of (VI.3), see Georges and Kinney [1938]). In this form we have 5 independent parameters since we may set $F = 1$ without loss of generality. We also notice that the coefficients all occur linearly so we formulate the least squares problem as a linear problem. To do this, assume we are given n data points $\{(x_i, y_i)\}_{i=1}^{n}$ which are supposed to approximate an ellipse. At each point, (x_i, y_i), we evaluate the expression (VI.3) assuming $F = 1$ and we have

$$Ax_i^2 + B\,x_iy_i + Cy_i^2 + Dx_i + Ey_i + 1 = r_i \quad . \qquad \text{(VI.4)}$$

In general $r_i \neq 0$ since the point (x_i, y_i) will not lie exactly on the ellipse determined by the coefficients A, B, C, D, and E. The least squares problem can now be formulated for the normal equation approach. The sum of squares to be minimized will be

$$S = \sum_{i=1}^{n} \left[Ax_i^2 + B\,x_iy_i + Cy_i^2 + Dx_i + Ey_i + 1\right]^2 \quad . \qquad \text{(VI.5)}$$

Taking the partial derivations of S with respect to the five coefficients A, B, C, D, and E and setting the derivatives equal to zero we obtain the normal equations

$$\begin{bmatrix} \sum x_i^4 & \sum x_i^3 y_i & \sum x_i^2 y_i^2 & \sum x_i^3 & \sum x_i^2 y_i \\ \sum x_i^3 y_i & \sum x_i^2 y_i^2 & \sum x_i y_i^3 & \sum x_i^2 y_i & \sum x_i y_i^2 \\ \sum x_i^2 y_i^2 & \sum x_i y_i^3 & \sum y_i^4 & \sum x_i y_i^2 & \sum y_i^3 \\ \sum x_i^3 & \sum x_i^2 y_i & \sum x_i y_i^2 & \sum x_i^2 & \sum x_i y_i \\ \sum x_i^2 y_i & \sum x_i y_i^2 & \sum y_i^3 & \sum x_i y_i & \sum y_i^2 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} = \begin{bmatrix} -\sum x_i^2 \\ -\sum x_i y_i \\ -\sum y_i^2 \\ -\sum x_i \\ -\sum y_i \end{bmatrix} \qquad \text{(VI. 6)}$$

where $\sum$ denotes $\sum_{i=1}^{n}$, a summation over all data points. The normal equations can then be solved by any standard method for solving a system of linear equations and the results tested to insure satisfaction of the inequality $B^2 - 4AC < 0$. However, this approach is not very satisfactory. In several cases of actual data, the results obtained by solving the normal equations in single precision on an IBM 360/75 did not represent an ellipse. However, on the same data but using a singular value decomposition routine (Golub [1967] and Businger [1967]) to solve the least squares problem, results were obtained which represented ellipses in all cases. It should be mentioned for completeness that the ellipses represented by the aforementioned data were very elongated. In some cases the ratio of major axis length to minor axis length was approximately 60 to 1.

If we let

$$r_i = Ax_i^2 + B\,x_i y_i + Cy_i^2 + Dx_i + Ey_i + F,$$

$$S = \sum_{i=1}^{n} r_i^2, \quad \text{and}$$

$$\alpha^T = (A, B, C, D, E, F),$$

the singular value decomposition gives a non-trivial solution to the problem of minimizing S. This is accomplished by finding α^T such that S is a minimum subject to the constraint $\alpha^T\alpha = 1$. This solution vector can then be tested to insure that the inequality $B^2 - 4AC < 0$ holds.

The least squares ellipse determination could also be treated as a mathematical programming problem. As such, one would minimize $S = \sum_{i=1}^{n} r_i^2$ subject to the inequality constraint $B^2 - 4AC < 0$.

Weighted linear least squares ellipses. Up to now no mention has been made of a weighting scheme for the linear least squares ellipse determination. Weights can be applied to the Eq. (VI.4) by multiplying the i^{th} equation by a weight, $w_i > 0$. The quantity to be minimized is then $\sum_{i=1}^{n} (w_i r_i)^2$. The question remains as to how to determine $\{w_i\}_{i=1}^{n}$.

In the problem involving empirically determined phase space contours, the solutions determined by the singular values approach with no weighting (i.e., constant weight, $w_i = 1$, $i = 1, 2, \ldots, n$) did not always seem to give a "good" representation of the ends of the ellipse. A weighting scheme was devised which improved the fit in the sense that the length of the computed ellipse was more nearly equal to the length intuitively indicated by the data points. There was no statistical or physical justification for the weighting scheme since it was not known if the data points near the ends of the ellipse were more accurately determined. Figure 6.11 shows a non-weighted singular value ellipse fit to empirical data where the computed ellipse appears intuitively to be too long.

The weighting scheme which was applied to the ellipse fit shown in Fig. 6.11 involved the curvature of the approximating ellipse. The weight at the i^{th} point was calculated to be equal to the curvature at the point on the ellipse where the radial line from the center of the ellipse to the i^{th} data point intersected the ellipse. Thus points toward the ends of the ellipse were weighted more heavily.

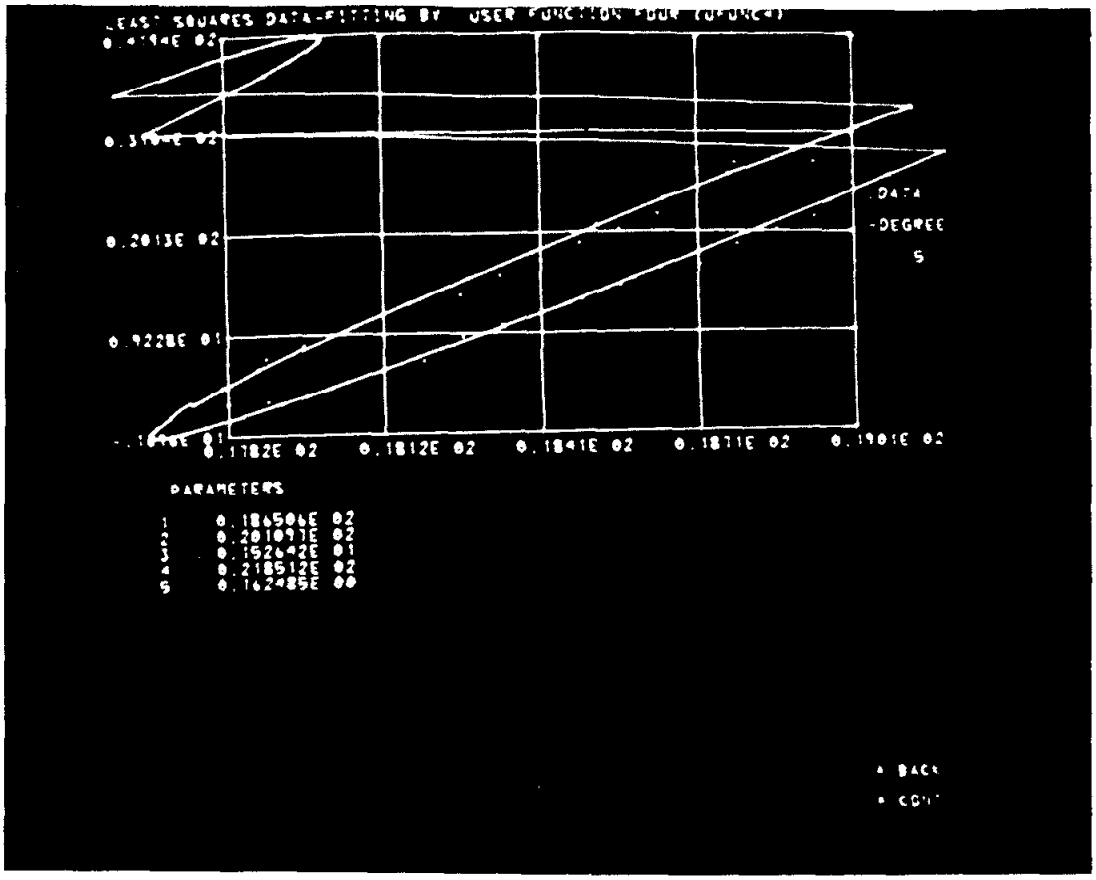

FIG. 6.11--Non-weighted ellipse fit (points plotted off-scale on the 2250 come in on the other side of the CRT screen).

Figure 6.12 illustrates the determination of points, p_i, on the ellipse at which the curvature is calculated for use as a weight. Given an existing approximation for the least squares ellipse, a weighted least squares ellipse is computed by calculating the curvature at $p_i, i = 1, 2, \ldots, n$ (see Fig. 6.12), and using these values as weights applied to Eq. (VI.4). This process can be iterated until there is relatively little change in the approximating ellipse. This is accomplished by using curvature-weights calculated at the $\{p_i\}$ on the previous approximation to obtain a new approximation.

The points $\{p_i\}$ can be calculated easily if the center, (x_c, y_c), the tilt, θ, and the semiaxes a and b of an approximating ellipse are known. Given the i^{th} data point, (x_i, y_i), we want to know the coordinates of the point p_i, say $\left(x_{p_i}, y_{p_i}\right)$. Let r denote the distance from (x_c, y_c) to $\left(x_{p_i}, y_{p_i}\right)$. Then

$$r = \frac{ab}{\left(b^2 \cos^2 \phi + a^2 \sin^2 \phi\right)^{1/2}}$$

where

$$\phi = \arctan\left(\frac{y_i - y_c}{x_i - x_c}\right) - \theta \quad .$$

Given r we obtain x_{p_i} and y_{p_i} by

$$x_{p_i} = x_c + r \cos \gamma$$

$$y_{p_i} = y_c + r \sin \gamma$$

where $\gamma = \arctan\left(\frac{y_i - y_c}{x_i - x_c}\right)$.

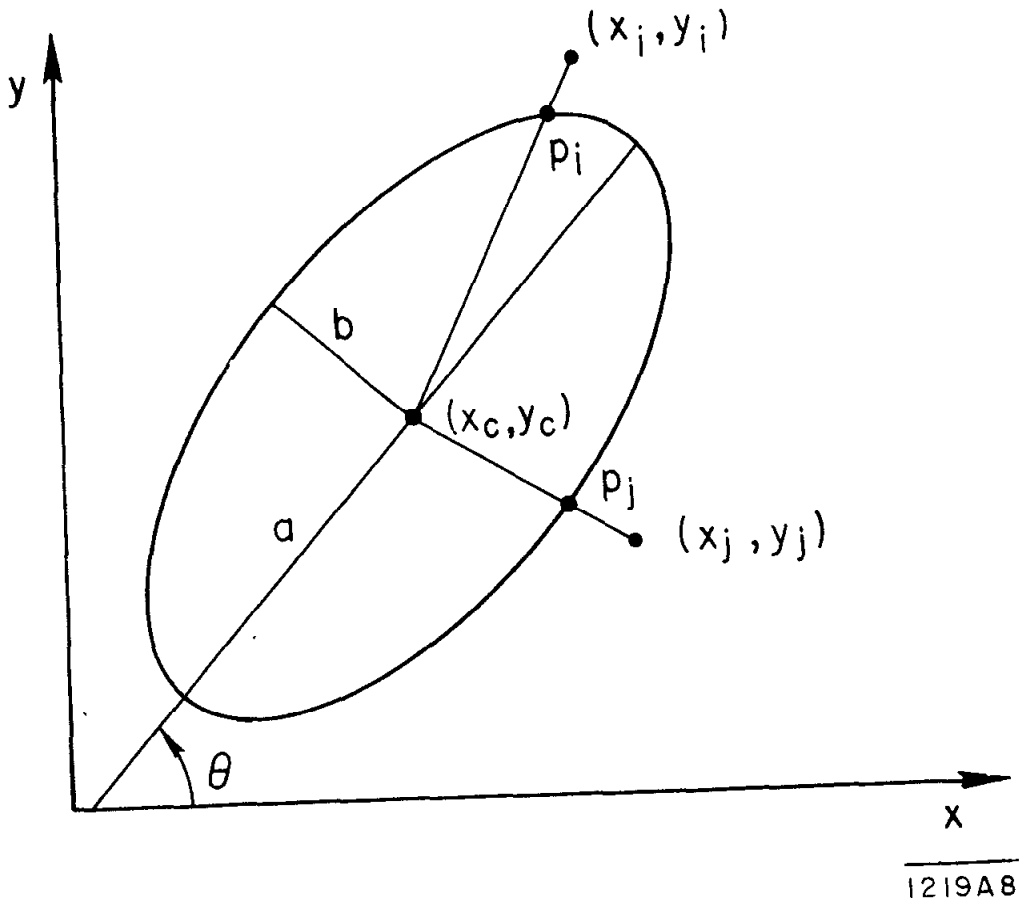

FIG. 6.12--Choosing points to calculate curvature weights.

Now the curvature can be calculated at $\left(x_{p_i}, y_{p_i}\right)$. The curvature K is defined to be 1/R where R is the radius of curvature. For a function, f(x, y), the radius of curvature is given by

$$R = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\frac{d^2y}{dx^2}} \quad .$$

Using the implicit function form of the ellipse,

$$Ax^2 + B\,xy + Cy^2 + Dx + Ey + F = 0 \quad ,$$

we have (see Georges and Kinney [1938], p. 405)

$$\frac{dy}{dx} = -\frac{2Ax + By + D}{Bx + 2Cy + E} \quad .$$

Similarly we can derive the second derivative as

$$\frac{d^2y}{dx^2} = -\frac{2C\left(\frac{dy}{dx}\right)^2 + 2B\frac{dy}{dx} + 2A}{2Cy + Bx + E} \quad .$$

We can now compute the curvature at any point (x_0, y_0) by

$$K(x_0, y_0) = \frac{\left.\frac{d^2y}{dx^2}\right|_{(x_0, y_0)}}{\left[1 + \left(\left.\frac{dy}{dx}\right|_{(x_0, y_0)}\right)^2\right]^{3/2}} \quad .$$

Figure 6.13 shows the results of applying the curvature weighting scheme to the first approximation ellipse shown in Fig. 6.11.

In Fig. 6.14 the ellipse of Fig. 6.13 has had the same weighting scheme re-applied four more times to stabilize the results.

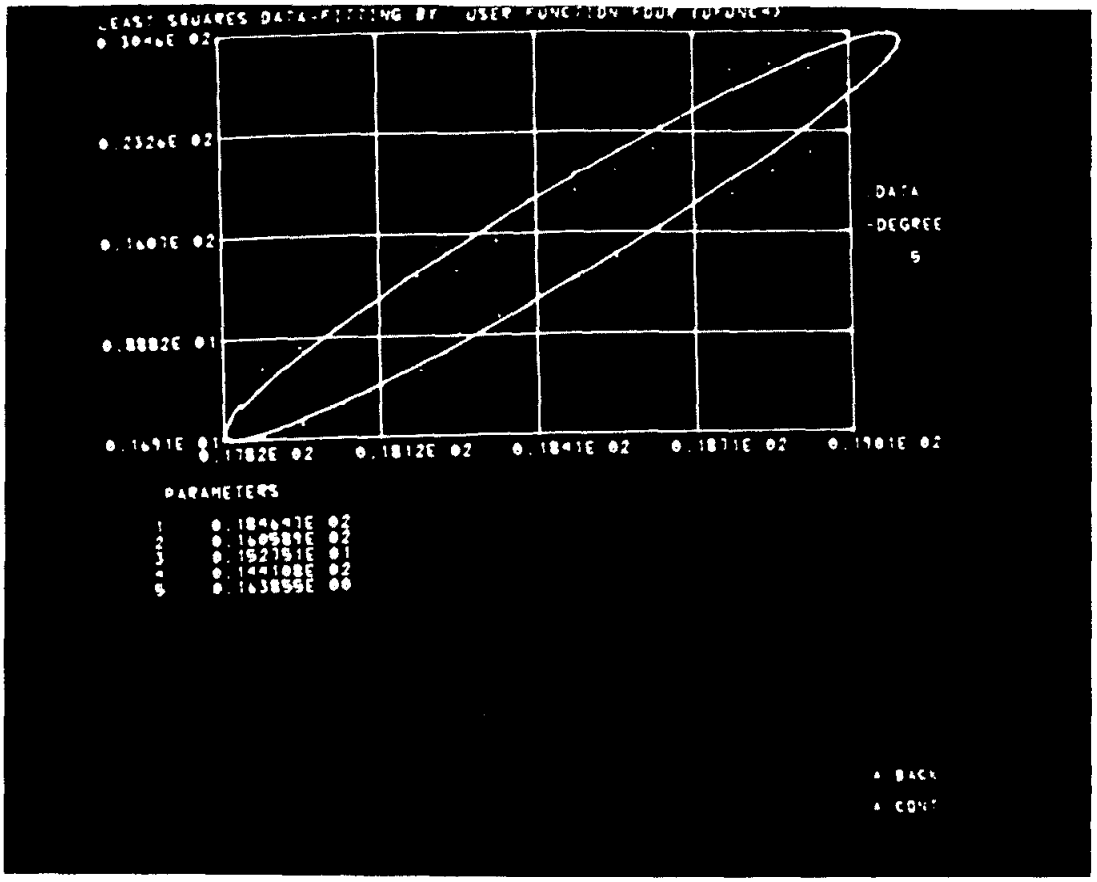

FIG. 6.13--First application of curvature weights.

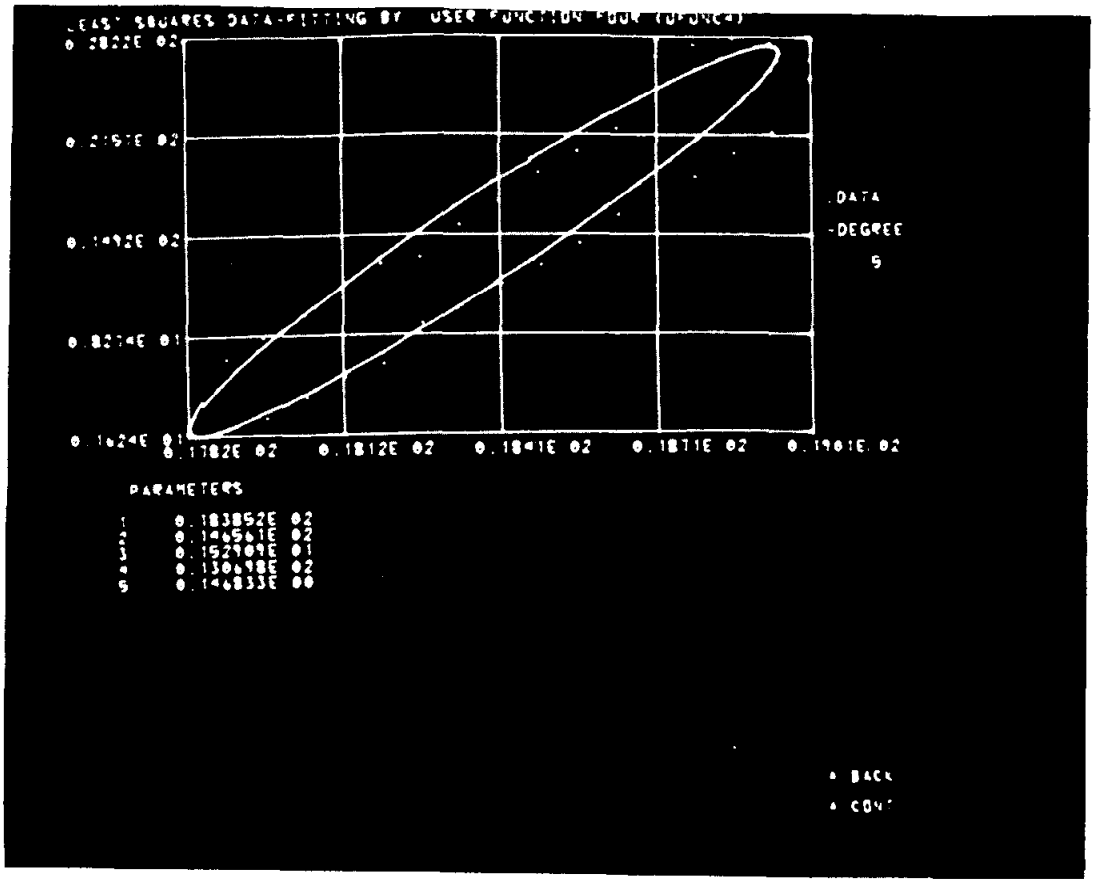

FIG. 6.14--Fifth application of curvature weights.

The changes in the elliptical parameters as the curvature weighting scheme is applied and re-applied are shown in Table 7.

TABLE 7

Change in Elliptical Parameters as Weighting Scheme is Iterated

	x_c	y_c	θ(rad)	a	b	area
no weights	18.65	20.11	1.526	21.85	.1625	11.154
iteration 1	18.46	16.06	1.528	14.41	.1639	7.418
iteration 2	18.40	14.79	1.528	13.15	.1679	6.937
iteration 3	18.40	14.71	1.528	13.09	.1603	6.595
iteration 4	18.39	14.68	1.529	13.07	.1542	6.331
iteration 5	18.39	14.66	1.529	13.07	.1468	6.029

Other weighting schemes could be applied to this problem. In particular, if there are known experimental errors associated with the data points $\{(x_i, y_i)\}_{i=1}^{n}$ these errors should be incorporated in any weighting scheme.

Converting the linear coefficients to usable quantities. Having obtained values for A, B, C, D, and E, assuming F = 1 in (VI.3), they must be checked for the property of indeed representing an ellipse. The requirement is

$$B^2 - 4AC < 0 \quad . \tag{VI.7}$$

If the inequality (VI.7) is satisfied the solution represents an ellipse and may be transformed into a representation of an ellipse from which the center, the lengths of the semiaxes, and the angle of tilt of the major axis are easily obtainable. Equation (VI.3) can be reduced to the form (see Georges and Kinney [1938])

$$A'x'^2 + C'y'^2 + D'x' + E'y' + F' = 0 \tag{VI.8}$$

by the equations of rotation

$$x = x' \cos \theta - y' \sin \theta$$
$$y = x' \sin \theta + y' \cos \theta \qquad \text{(VI. 9)}$$

where

$$\tan \theta = \frac{-A + C \pm \sqrt{(A-C)^2 + B^2}}{B} \quad . \qquad \text{(VI. 10)}$$

Either sign may be chosen in (VI. 10) since the two angles differ by 90° and the major and minor axes will be interchanged giving the same ellipse in either case. So given θ from (VI. 10) the coefficients in (VI. 8) are given by the equations

$$A' = A \cos^2 \theta + C \sin^2 \theta + B \sin \theta \cos \theta$$
$$C' = A \sin^2 \theta + C \cos^2 \theta - B \sin \theta \cos \theta$$
$$D' = D \cos \theta + E \sin \theta \qquad \text{(VI. 11)}$$
$$E' = E \cos \theta - D \sin \theta \quad .$$

Now the center coordinates, (x'_c, y'_c), in this rotated frame of reference are given by

$$x'_c = - D'/[2A'] \qquad \text{(VI. 12)}$$
$$y'_c = - E'/[2C']$$

and the squared lengths of the semiaxes, a and b, are given by

$$a^2 = \frac{D'^2}{4A'^2} + \frac{E'^2}{4A'C'} - \frac{1}{A'} \qquad \text{(VI. 13)}$$

$$b^2 = \frac{D'^2}{4A'C'} + \frac{E'^2}{4C'^2} - \frac{1}{C'} \quad .$$

To locate the coordinates of the center, (x_c, y_c), of the ellipse the point given by (VI. 12) needs to be rotated back to the original frame of reference by

$$\begin{aligned} x_c &= x'_c \cos\theta - y'_c \sin\theta \\ y_c &= x'_c \sin\theta + y'_c \cos\theta \quad . \end{aligned} \tag{VI. 14}$$

The interesting properties of the ellipse are all available at this point. The center is given by (VI. 14). The lengths of the axes are given by (VI. 13), and the angle of tilt of the major axis is given by (VI. 10).

2. Fitting an Ellipse by Non-Linear Least Squares

As we mentioned previously, there are various ways to represent an ellipse and various methods of performing a least squares fit to an ellipse. Here we consider the implicit function representation and its use in the least squares problem and secondly a geometric approach to determination of the least squares fit.

We consider two implicit functions, $f(x, y) = 0$, that can be used to represent an ellipse. These are

$$f_1(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \quad \text{with } B^2 - 4AC < 0,$$

and

$$f_2(x,y) = \frac{r^2 \cos^2(\theta - \theta_0)}{a^2} + \frac{r^2 \sin^2(\theta - \theta_0)}{b^2} - 1 = 0 \ ,$$

where

$$r = \left[(x - x_0)^2 + (y - y_0)^2\right]^{1/2} \quad , \text{ and}$$

$$\theta = \arctan\left(\frac{y - y_0}{x - x_0}\right) \ .$$

θ_0 is the angle of tilt of the major axis, and a and b are the semi-major and semi-minor axes respectively. x_0 and y_0 are the coordinates of the center of the ellipse.

The function $f_1(x,y)$ is discussed in paragraph VI.D.1 and the conversion of the coefficients to more meaningful quantities is also discussed in paragraph VI.D.1. The representation given by $f_2(x,y)$ is more familiar since the geometrical parameters of the ellipse occur explicitly. However in $f_2(x,y)$ the parameters occur non-linearly so the function is not amenable to treatment by the singular values decomposition as is $f_1(x,y)$.

To obtain a least squares fit of given data points, $\{(x_i, y_i)\}_{i=1}^{n}$, to an implicit function, say $f_2(x,y)$, we ask for the minimum of the sum of squares of the residuals where the residual, r_i, at each point is given by

$$r_i = f_2(x_i, y_i) \quad . \tag{VI.15}$$

If the data were a true representation of an ellipse the $\{r_i\}$ could all be made zero by the proper choice of the elliptical parameters. However in practice this will not be so.

The least squares solution is obtained by finding the minimum of the sum of squares of the residuals, S, where

$$S = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} \left[f_2(x_i, y_i)\right]^2 \quad . \tag{VI.16}$$

S is a function of the elliptical parameters, (x_c, y_c), the center, a and b, the semi-axis, and θ_0, the angle of tilt of the major axis. Therefore $S(x_i, y_c, \theta_0,$ a, b) can be minimized with respect to its five parameters by some appropriate method to obtain the least squares fit. In Chapter V is a discussion of the minimization methods which are incorporated in the data-fitting program described

in Chapter III. The direct search method has been successfully used to minimize (VI. 16) in an interactive environment.

A geometric approach to least squares ellipses. An intuitive approach to solving the least squares data-fitting problem for an ellipse involves geometric considerations. In the more common least squares problem involving a single valued function used to approximate empirical data, the residuals are the geometric distances from each of the data points to the approximating curve. These distances are usually measured parallel to one axis as illustrated in Fig. 6.15 where the sum of squares is $\sum_{i=1}^{n} r_i^2$.

In the case of an ellipse the usual residual measurement is not meaningful for two reasons. First, we have a double-valued function and the question arises as to which side of the ellipse to measure from. And, secondly, at each end (or side) of the ellipse the curve is nearly parallel to one axis so measurements parallel to that axis will probably not be meaningful. Figure 6.16 illustrates these problems.

Point (1) in Fig. 6.16 will have no residual if the measurement parallel to the y axis is used. Point (2) brings up the question of whether to measure toward the portion of the ellipse above it or below it, and point (3), near the end of the ellipse, has a large residual measured in the y-direction but the point is actually quite near the approximating ellipse as can be seen by measuring in the x direction.

The necessity for something other than the usual residual measurement is quite clear. Other alternatives available are measurements in the normal direction to the ellipse and measuring along a radial line. Figure 6.17 illustrates these two methods geometrically. At each of the four points shown, r_i represents the radially measured residual while n_i denotes the normally measured residual.

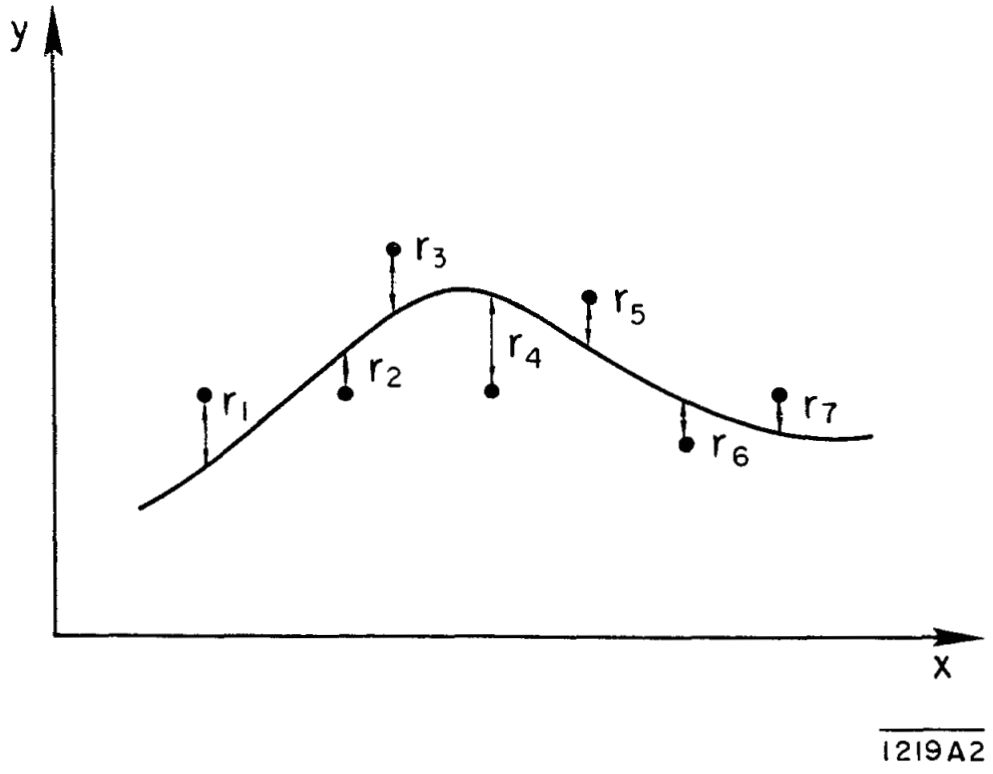

FIG. 6.15--The usual least squares problem.

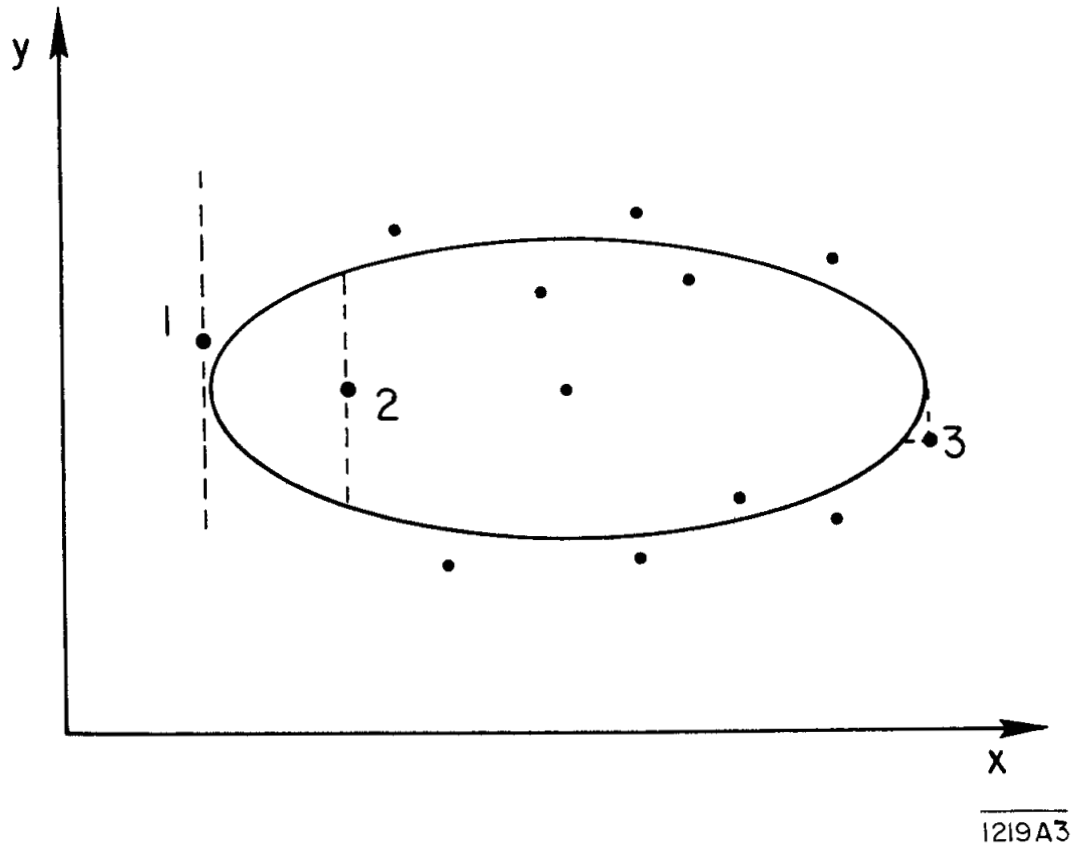

FIG. 6.16--Measuring residuals for an ellipse.

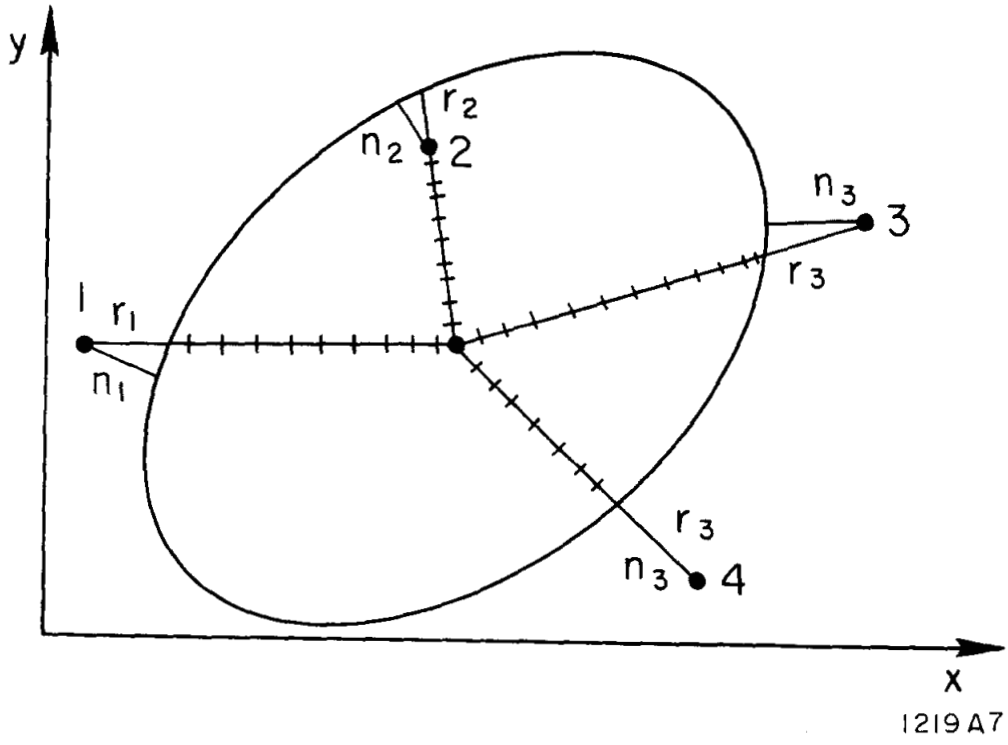

FIG. 6.17--Geometrically calculated residuals for an ellipse.

When a data point falls on a line coincident to one of the axes of the ellipse, the normal and radial residuals are identical. In all other cases, the normal residual is smaller than the radial, so we have that $n_i \leq r_i$. For ellipses with eccentricity near zero (nearly circular) the use of either one of these methods for residual measurement will produce nearly identical results. However, for ellipses with eccentricity near one (very elongated) the radial method is intuitively poor (provided there is sufficient data at the small ends of the ellipse to determine the length). This is particularly obvious for a point inside the approximating ellipse as shown in Fig. 6.18, where for example, $r_1 \gg n_1$.

The normally measured residuals appear to be more meaningful and should probably be used in a geometrical approach to computing a least squares ellipse. The solution would then be obtained by minimizing

$$S(x_c, y_c, \theta_0, a, b) = \sum_{i=1}^{n} (n_i)^2 \qquad \text{(VI.17)}$$

with respect to the five parameters indicated. The normally measured residual method has not yet been tried with PEG, however, the radial residual method has been used quite successfully.

Weighted non-linear least squares ellipses. So far we have not discussed the use of errors associated with the data points representing an ellipse. Often in least squares problems an error, ϵ_i, or weight, w_i, is associated with the dependent variable at each point. If we define the weight, w_i, to be the inverse of the error, $w_i = 1/\epsilon_i$, and the approximating function to be $y = f(x)$, the sum of squares to be minimized is

$$S = \sum_{i=1}^{n} \left[w_i \left(f(x_i) - y_i \right) \right]^2 \quad . \qquad \text{(VI.18)}$$

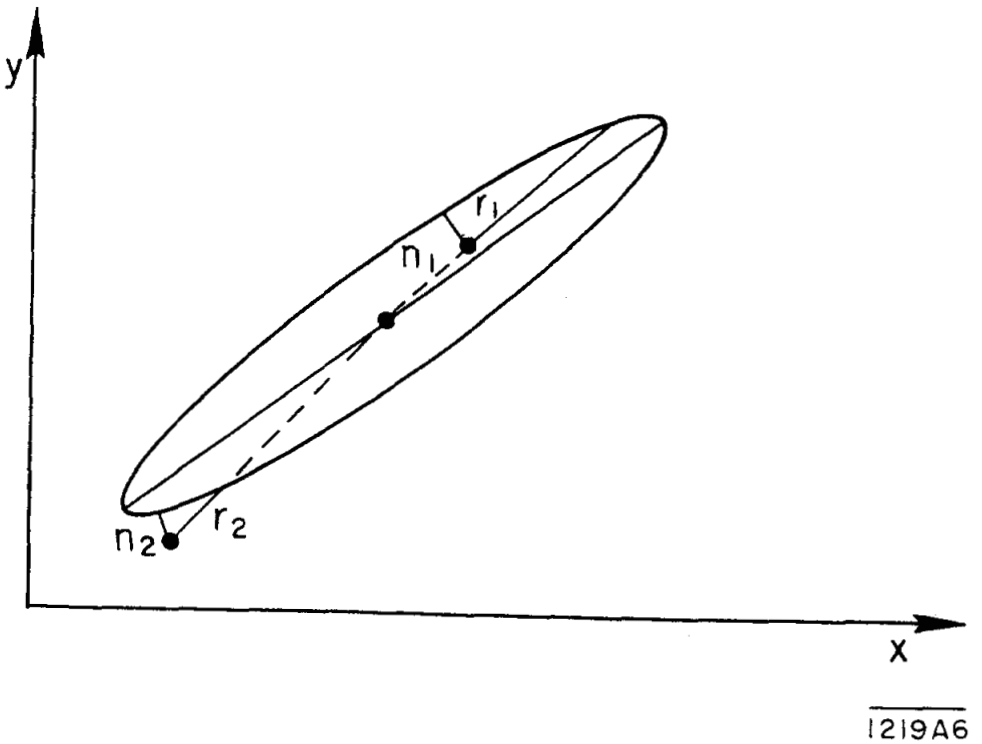

FIG. 6.18--Radial and normal residuals for an elongated ellipse.

However, in the case of data representing an ellipse we must consider the possibility of having errors associated with both coordinates of each data point. The data would then consist of $\{(x_i, y_i)\}_{i=1}^{n}$ and the errors $\{(\delta x_i, \delta y_i)\}_{i=1}^{n}$. The values for the errors would be determined, for example, by the experimentalist who collected the data. One way to utilize this information in computing the least squares ellipse is the following. Consider the implicit function form of an ellipse, $f(x,y) = 0$. Without weights we computed a least squares fit by minimizing the quantity

$$S = \sum_{i=1}^{n} f(x_i, y_i)^2 \quad . \tag{VI.19}$$

To weight the sum of squares so more accurate points play a larger role in determining the ellipse we minimize the quantity

$$S_w = \sum_{i=1}^{n} \left[\frac{f(x_i, y_i)}{\sigma_i}\right]^2 \tag{VI.20}$$

where one method (Howry [1968]) of determining σ_i^2 is

$$\sigma_i^2 = f_{x_i}^2 \, \delta x_i^2 + f_{y_i}^2 \, \delta y_i^2 \ ,$$

with

$$f_{x_i} = \left.\frac{\partial f}{\partial x}\right|_{x=x_i} , \quad f_{y_i} = \left.\frac{\partial f}{\partial y}\right|_{y=y_i} \ .$$

This form of weighting assumes the errors in x and y are independent and that the true point approximated by a data point, (x_i, y_i), lies in an ellipse which encloses the (x_i, y_i) and has axes parallel to the given coordinate axes. A more complex form of σ_i^2, assuming x and y are not independent, would involve mixed partial derivatives of $f(x,y)$, and the error ellipse would be tilted in some fashion.

Other weighting schemes could also be devised to use in the non-linear least squares ellipse fitting problem. No weighting schemes involving errors in the data have been tried with PEG.

3. Plotting an Ellipse

Given a fixed number of points there is no doubt as to the best distribution of those points to graphically represent a circle. Equal angle increments with a point on the perimeter for each angle will nicely depict a circle (by approximating it with a regular polygon). An ellipse, however, is a different matter, particularly if the eccentricity is large (near 1). Equal angle increments used to pick points for display can easily fail to represent the small ends of an ellipse as shown in Fig. 6.19 even though the sides appear quite smooth.

The equal angle approach does not provide a good method for plotting an ellipse, since at the ends the points for plotting are too far apart with respect to the curvature. Therefore the obvious method would seem to use equal perimeter lengths to separate the representative points. However, this would use more points than necessary on the sides in order to have points close enough together on the ends. Another drawback to the use of equal perimeter is the fact that their computation involves an elliptic integral, an undesirably complex operation in a computer environment.

A third scheme which produces faithful representations of ellipses (including those with large eccentricity) with a relatively small amount of computation has been devised. Consider an ellipse centered at the origin with major axis 2a and minor axis 2b. The radial line from the origin to the point (a,b) intersects the ellipse at the point $\left(\frac{\sqrt{2}}{2}a, \frac{\sqrt{2}}{2}b\right)$. If we are given N points with which to represent the ellipse at hand, the scheme is as follows (see Fig. 6.20) for the representation in the first quadrant. The other three quadrants will be handled

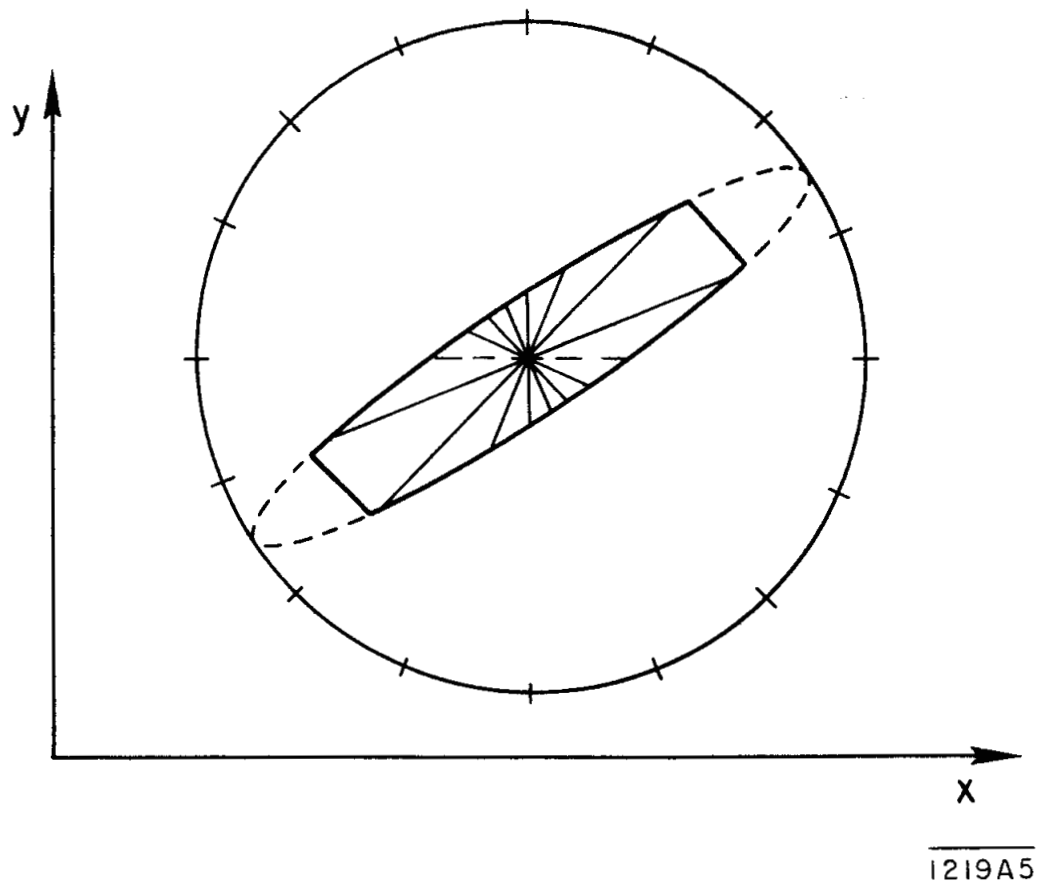

FIG. 6.19--Equal angle representation of an ellipse.

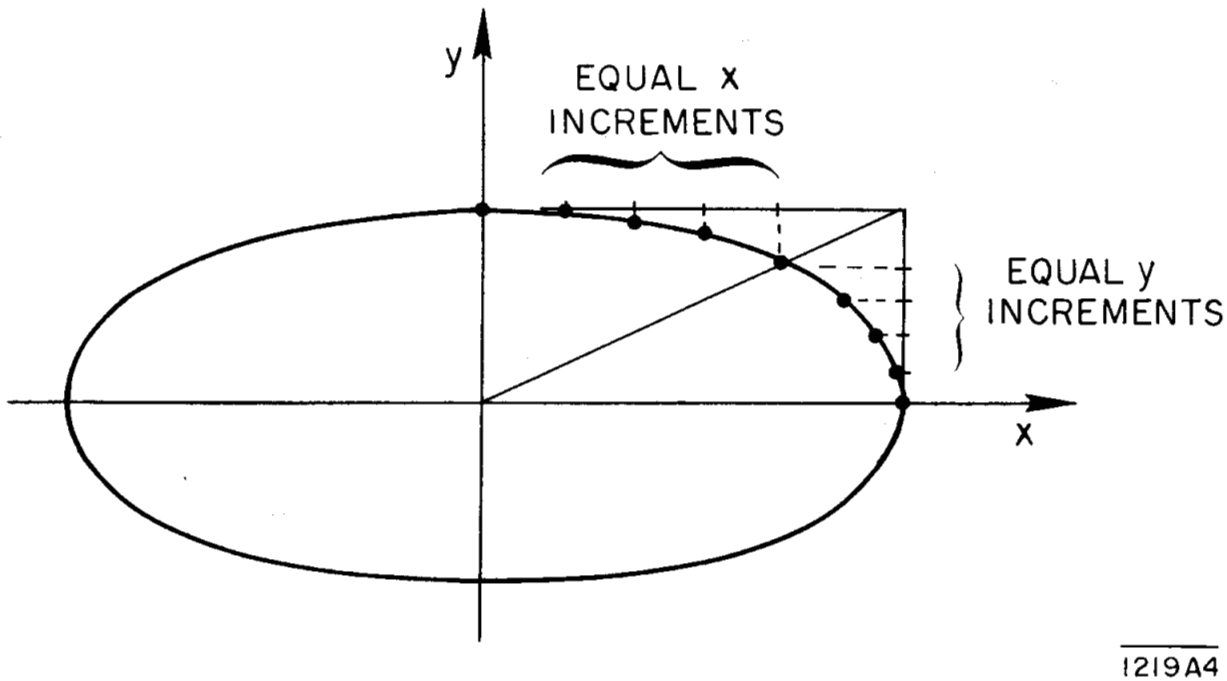

FIG. 6.20--A successful ellipse plotting scheme.

similarly. Let (x_p, y_p) be the next point on the ellipse for plotting purposes.

(a) Compute

$$dx = \left(\frac{\sqrt{2}}{2}a\right)/(N/8)$$

$$dy = \left(\frac{\sqrt{2}}{2}b\right)/(N/8)$$

(b) Set $x_p = a$, $y_p = 0$

(c) Plot (x_p, y_p)

(d) If $y < \frac{\sqrt{2}}{2}b$, set $y_p = y_p + dy$, compute x_p, go to (c)

(e) If $y \geq \frac{\sqrt{2}}{2}b$, set $x_p = x_p - dx$, compute y_p,

if $x_p \geq 0$, go to (c), if $x_p < 0$, go to (f)

(f) That's the end of the first quadrant.

If N does not have the right relation to 8 some adjustment in the values of dx and dy may be necessary to "fill out" the whole ellipse.

A fourth (and best known to the author) scheme for graphical representation of ellipses uses the parametric representation of an ellipse with axes 2a and 2b,

$$\begin{aligned} x &= a \cos \phi \ , \\ y &= b \sin \phi \ . \end{aligned} \qquad \text{(VI. 21)}$$

Using (VI. 21) as the parameter ϕ is incremented from 0 to 2π, the points (x, y) trace the desired ellipse. It is easy to use a fixed number of points, say N, by using increments of $2\pi/(N-1)$ to vary ϕ over the interval $[0, 2\pi]$.

We see that this method automatically gives the desired changes in perimeter increments, namely relatively small changes at the ends and relatively large changes along the sides which is necessary for ellipses with eccentricity

near one. We have

$$dy = b \cos \phi \, d\phi$$
$$dx = - a \sin \phi \, d\phi \qquad \text{(VI. 22)}$$

so that for small values of ϕ (or near π) we have $dy \approx b\,d\phi$ and dx near zero. For ϕ near $\pi/2$ or $3\pi/2$ we have $dx \approx a\,d\phi$ and dy near zero. Thus the ratio of perimeter increment size at the small ends to that on the sides is approximately b/a as desired for a faithful representation of an ellipse with eccentricity near one ($b/a \ll 1$). We also see that this method gives equal perimeter increments (a regular polygon) if a = b for the best possible representation in the case of a circle.

Another advantage of the use of the parametric equations for plotting an ellipse is the savings in computer time to calculate the points. The previous method, using equal y increments along the sides involves a SIN, a COS and a SQRT calculation plus considerable testing to determine each point to be plotted. The parametric method requires only a few arithmetic operations to calculate each point since we can use the multiple angle trigonometric identities as we are incrementing ϕ by an equal amount for each point. Given (x_c, y_c) the center of the ellipse, θ, the tilt of the major axis, and the major and minor semi-axes a and b we compute the points to be plotted $\{(x_i, y_i)\}_{i=1}^{N}$ as follows:

Let

$$\begin{cases} d\phi = 2\pi/(N-1) \\ CT = \cos(\theta) \\ ST = \sin(\theta) \\ CDP = \cos(d\phi) \\ SDP = \sin(d\phi) \\ CNDP = 1.0 \text{ (initially)} \\ SNDP = 0.0 \text{ (initially)} \end{cases}$$

Then repeat the following for i= 1, 2, ..., N

$$
\begin{cases}
x' = a \cdot CNDP \\
y' = b \cdot SNDP \\
\\
x_i = x_c + x' \cdot CT - y' \cdot ST \\
y_i = y_c + x' \cdot ST + y' \cdot CT \\
\\
TEMP = CNDP \cdot CDP - SNDP \cdot SDP \\
SNDP = SNDP \cdot CDP + CNDP \cdot SDP \\
CNDP = TEMP \quad .
\end{cases}
$$

The Fortran code to accomplish this ellipse plot calculation is shown in Fig. 3.36.

Pitteway [1967] describes an algorithm for drawing ellipses or hyperbolae with a digital plotter. However his method is not as well suited to the application at hand as is the method just described.

4. Interactive Ellipse Fitting

The least squares ellipse fitting problem has been worked on using the interactive data fitting program described in Chapter III. The implicit function fitting capability was used for the ellipse problem so various schemes were easily tried for comparison. The two principal methods used were minimization of the sum of squares of radial distances and the use of an implicit function representation of the ellipse. Starting values for the elliptical parameters were obtained from the linear least squares approach to the problem discussed in paragraph VI.D.1. Crude starting values for the parameters obtained from a hand plot of the data were also tried and were close enough to the least squares values to insure convergence of the minimization routine.

An option that is often very desirable in fitting experimental data is that of deleting points which the user may feel are "bad" or at best questionable. Even

without statistical justification for eliminating certain points, a person doing curve fitting is often interested in knowing what the fit would be if those points were eliminated. The provision for point deletion in the interactive data-fitting program allows such experimentation to be easily accomplished.

Another facility which could be added to the data fitting program that could be used in working on the ellipse fitting problem is that of inserting weights in a problem that previously had no weights associated with the data points. This would allow a data set to be approximated with no weighting and then after weights are entered for each point, a weighted approximation could be computed. This, usually quite subjective, post facto association of weights (errors) with the data points often has no statistical justification but it can provide considerable insight into a given problem. For example, a particular weighting scheme which gave an approximation to theoretically predicted results might point to a new or altered experimental approach to a particular problem.

Figures 6.21 and 6.22 illustrate the difference in approximating function that can be achieved by deleting certain "doubtful" data points from the least squares calculation. The approximating ellipses in these figures were calculated by the method of Eq. (VI.16).

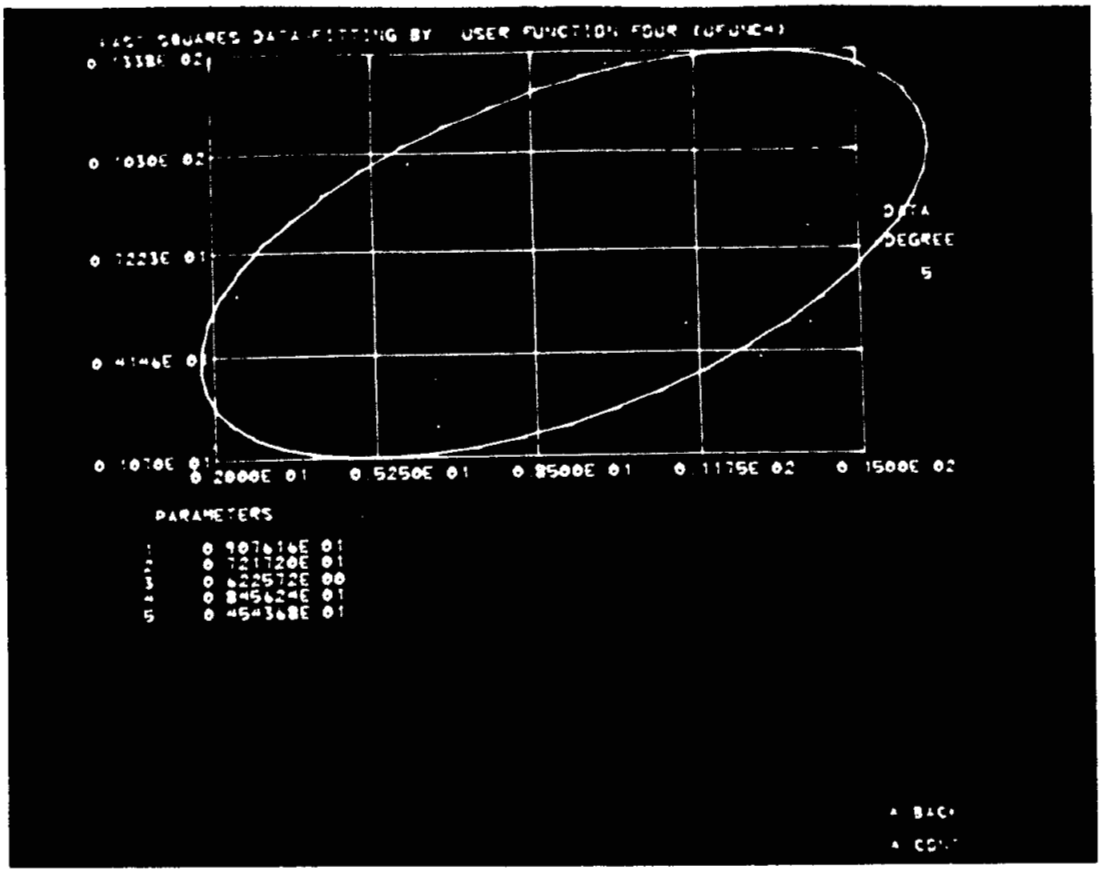

FIG. 6.21--Least squares ellipse on original data.

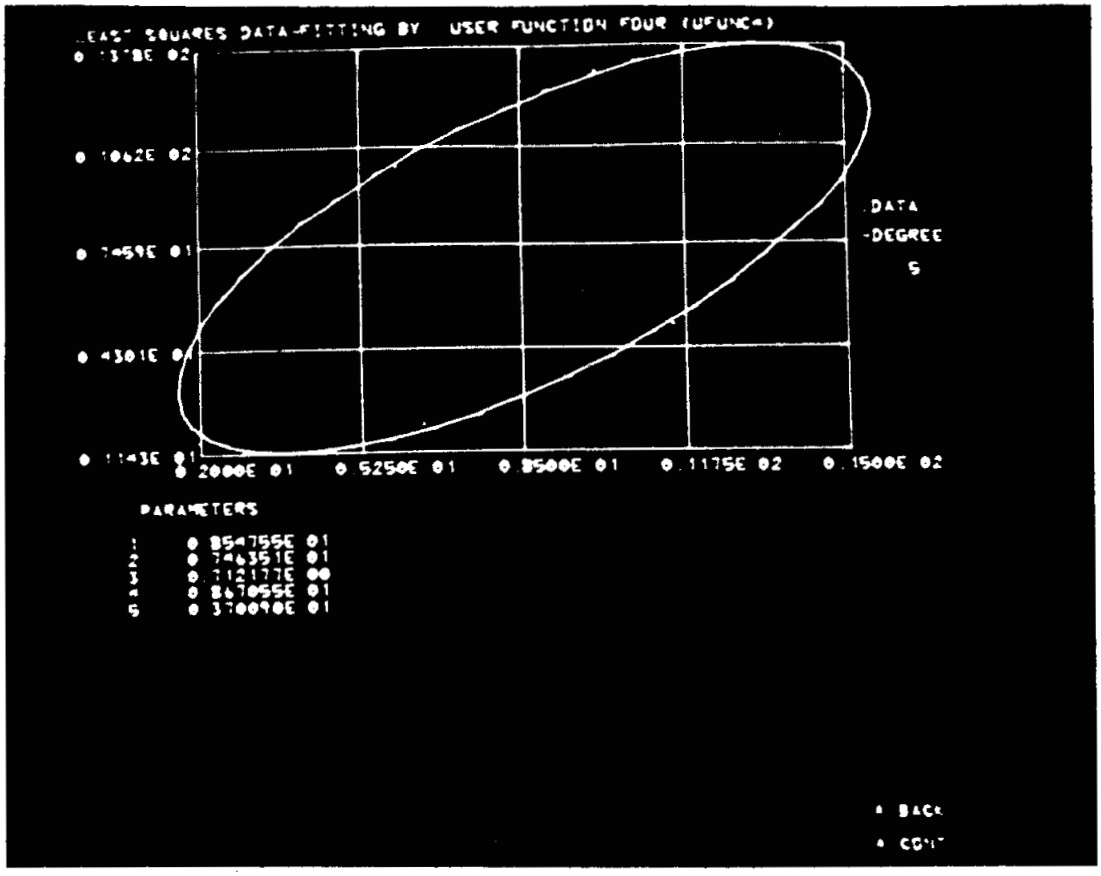

FIG. 6.22--Least squares ellipse with points deleted (Compare to Fig. 6.21).

CHAPTER VII

MAXIMUM LIKELIHOOD BY INTERACTIVE MINIMIZATION

An increasingly more popular method of data-fitting among physicists is the maximum likelihood method. This method is sometimes useful in cases where a least squares treatment of the data fails to yield values of a parameter of interest whereas a maximum likelihood approach does give values for the recalcitrant parameter. An example of such a problem is given in Section VII.D.

In Section VII.A we detail the maximum likelihood method and in the following sections we describe the application of the interactive minimizer (see Chapter V) to the method and discuss the use of interactively produced contour plots to examine the results.

A. The Maximum Likelihood Method

For the purposes of this discussion let us assume we have data points $\{x_i, y_i\}_{i=1}^{n}$ which have been determined experimentally and which lie somewhere in a plane. The method can be applied to problems in a higher dimensioned space but limiting the dimension to two simplifies the presentation. If there is known a probability function, $P(x,y)$, which associates a probability with each point in the plane, then we may apply the maximum likelihood method to determine values of any parameters that appear in $P(x,y)$.

Let us write $P(x,y)$ as

$$P(x,y) = P(x,y, a_1, \ldots, a_m) \qquad \text{(VII.1)}$$

where $\{a_j\}_{j=1}^{m}$ are the parameters of interest (to be determined by data-fitting). We may then construct the likelihood function

$$L(a_1, \ldots, a_m) = \prod_{i=1}^{n} P(x_i, y_i, a_1, \ldots, a_m) \ . \qquad \text{(VII.2)}$$

Given values for $\{a_j\}_{j=1}^m$, Eq. (VII.2) gives the probability that the data points lie where in fact they were observed to lie. The aim of the maximum likelihood method is to find values $\{a_j^*\}_{j=1}^m$ for the parameters which maximize the likelihood function L. That is, we want to find $\{a_j^*\}_{j=1}^m$ such that

$$L(a_1^*, \ldots, a_m^*) \geq L(a_1, \ldots, a_m), \quad \text{for any } \{a_j\}_{j=1}^m \tag{VII.3}$$

Thus the maximum likelihood method reduces to a maximization problem. Equation (VII.2) defines a function with m parameters and the problem is to find values of those parameters which maximize L. Since $L(a_1, \ldots, a_m)$ is a product of probabilities, we have $L \geq 0$, and by simply considering -L we have a minimization problem. In the next section we discuss a way to solve the maximum likelihood problem.

B. Using the Interactive Minimizer

As shown above, a minimization routine can be used to solve the maximum likelihood problem by consideration of $-L(a_1, \ldots, a_m)$. Here we discuss the use of the interactive minimizer described in Chapter V. We will then show how the interactive minimizer can be adapted to also produce error estimates on the parameters.

1. Minimization of $-L(a_1, \ldots, a_m)$

In theory any appropriate minimization technique could be applied to the function $-L(a_1, \ldots, a_m)$ to determine the values $\{a_j^*\}_{j=1}^m$ which maximize the likelihood function. We have chosen to use the interactive Gauss-Seidel type minimizer since it will handle non-linear functions without requiring derivative calculations and has the side benefit of easily producing error estimates.

Given an initial guess for the parameters, $\{a_j^o\}_{j=1}^m$, the method can be outlined as follows. We denote by L_i the likelihood function as a function of only the

i^{th} parameter; all other parameters held fixed.

Interactive Minimization of $-L(a_1, \ldots, a_m)$

a. Set $k = 1$.

b. Set $a_j^k = a_j^{k-1}$, $j = 1, 2, \ldots, m$.

c. Set $i = 1$.

d. Locate interactively the value, a_i', which minimizes $-L_i(a_1^k, a_2^k, \ldots, a_m^k)$.

e. Set $a_i^k = a_i'$.

f. Set $i = i + 1$. If $i \leq m$, go to step d.

If $i > m$, go to step g.

g. Decide if method has converged. For example, by examining most recent changes in all m parameters relative to parameter sizes. If converged, go to step h. If not converged, set $k = k + 1$ and go to step b.

h. Set $a_j^* = a_j^k$, $j = 1, 2, \ldots, m$. Finished!

Use of another minimization method such as the direct search method (see Chapter V) would be simple to implement. The function $-L(a_1, \ldots, a_m)$ would be minimized with respect to the parameters $\{a_j\}_{j=1}^m$.

2. Handling Products of Small Numbers

The evaluation of the likelihood function can sometimes lead to numerical difficulties in a computer environment. As given in (VII.2) L is a product of probabilities, each less than one, so that if n is large, say 500, and each factor is small, say $P(x_i, y_i, a_1, \ldots, a_m) \leq 0.1$, we would have $L \leq 10^{-500}$. The floating point hardware of most computers is not built to handle numbers of that magnitude, so some logic is necessary in the program to maintain the significance in evaluating the likelihood function.

To maintain significance in the evaluation of likelihood functions a scheme which handles exponents (or powers of 10) must be utilized. The program described in Section VII.D employs the following method. As the likelihood function is evaluated, by repeated multiplication of the partial product by the probability at the next data point, the magnitude of the partial product is monitored. Any time the partial product becomes less than 10^{-10}, it is multiplied by 10^{10} and a record is kept of how many times this occurs. Therefore the result of each evaluation is given as pair of numbers, $L'(a_1, \ldots, a_m)$, t, where $L = L' \cdot 10^{-10t}$.

In the interactive minimizer a plot is made as one of the parameters of L is varied over some interval. Since each evaluation of L for this plot could have a different value of t associated with it, a normalization is performed before plotting. The normalization consists of multiplying each value by $10^{+10\tau}$ where τ is determined so as to make all plotted values have the same t value. To illustrate, suppose the computed values for a particular plot consists of $\{(L'_\rho, t_\rho)\}_{\rho=1}^{R}$ where ρ is the index of the R points depicting the curve. Let $t_{max} = \max_\rho t_\rho$. Then the normalized values to be plotted will be

$$\left\{L'_\rho \cdot 10^{-10(t_{max} - t_\rho)}, t_{max}\right\}_{\rho=1}^{R}.$$

3. Parameter Error Estimates

A statistical estimate of the error in the i^{th} parameter, δa_i, is given by

$$\delta a_i = \frac{\int_{-\infty}^{\infty} (a_i - a_i^*)^2 L_i \, da_i}{\int_{-\infty}^{\infty} L_i \, da_i} \qquad \text{(VII.4)}$$

where as before L_i denotes the likelihood function as a function of only the i^{th} parameters while, in this case, all other parameters are held fixed at their maximized values, $\{a_j^*\}_{j=1}^m (j \neq i)$.

In the interactive minimizer environment, the evaluation of (VII.4) is easily accomplished by approximating the integrals by some quadrature formula such as Simpson's rule. After convergence has been attained, the plot for each parameter can be adjusted interactively until the likelihood function goes to zero at both ends of the interval over which the parameter at hand is being varied. The plotted values can then be used to evaluate (VII.4) to determine the error estimate. Figure 7.1 illustrates the type of curve that is involved in this process. In the current implementation, selection of "PRINT GRAPH VALUES" by lightpen will cause printing of the displayed values and also cause calculation and printing of the associated error estimate.

C. Contour Plots to Examine Parameter Correlation

Given a functional which depends on several parameters, the interdependence or correlation between two of the parameters in the neighborhood of a point can be determined by an examination of the shape of the level curves (or a contour plot) as the two parameters are varied around their values at the point of interest. If the functional is $f(a_1, \ldots, a_m)$ and the two parameters of interest are a_i and a_j, a level curve is defined to be that curve in the a_i, a_j plane such that $f(a_1, \ldots, a_m) = c$(constant) for points on the curve. A plot in the a_i, a_j plane of level curves for various values of c, is called a contour plot.

The maximum likelihood program includes the facility to display and examine contour plots involving any pair of parameters in the neighborhood of given point, say $(a_1^*, a_2^*, \ldots, a_m^*)$. The method of producing these displays is as follows. First, the two parameters must be selected and an increment or

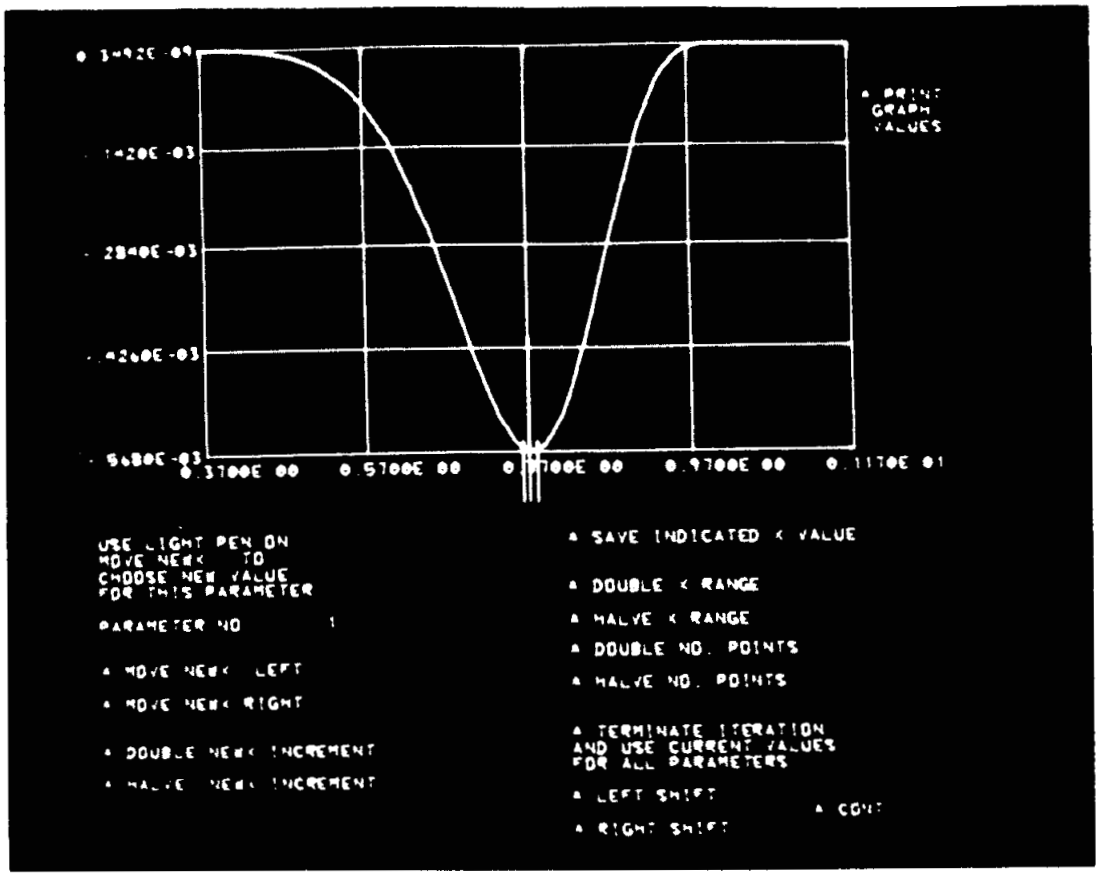

FIG. 7.1--Plot of L_i used to compute error estimates (plot is actually - L_i).

half-plot width associated with each. Then, given the parameter pair, (a_i, a_j), and the half-plot widths, $(\delta a_i, \delta a_j)$, the functional is evaluated on a grid of points in the rectangle defined by $a_i^* - \delta a_i \le a_i \le a_i^* + \delta a_i$ and $a_j^* - \delta a_j \le a_j \le a_j^* + \delta a_j$. The grid with I points horizontally and J points vertically would consist of points $(a_{i\sigma}, a_{j\rho})$ where

$$a_{i\sigma} = a_i^* - \delta a_i + \sigma \frac{2\delta a_i}{I-1}, \qquad \sigma = 0, 1, \ldots, I-1$$

$$a_{j\rho} = a_j^* - \delta a_j + \rho \frac{2\delta a_j}{J-1}, \qquad \rho = 0, 1, \ldots, J-1 \tag{VII.5}$$

If the functional is $f(a_1, a_2, \ldots, a_m)$ we then evaluate it with $a_\ell = a_\ell^*$, $\ell \neq i$, $\ell \neq j$ and a_i, a_j determined by (VII.5). This gives us an array of values $F = \left[f_{\rho\sigma}\right]$, where

$$f_{\rho\sigma} = f(a_1^*, a_2^* \ldots, a_{i\sigma}, \ldots, a_{j\rho}, \ldots a_{m-1}^*, a_m^*) . \tag{VII.6}$$

The maximum and minimum entries in the $I \times J$ array, F, determine an interval covering all values of $f_{\rho\sigma}$. This interval is then divided into K equal subintervals where K is the number of different symbols to be used in the contour plot. Then at each grid point, $f_{\rho\sigma}$ lies in one of the K subintervals of functional values, say the k^{th}. The k^{th} symbol of those chosen is then displayed at the (ρ, σ) grid point. In this manner, level curves are approximated by a band of the same symbol.

Figure 7.2 illustrates the use of this method of producing contour plots. As shown in Fig. 7.2 a user may interact with the contour plot in several ways. Assuming that the contour plot is originally given for $a_\ell = a_\ell^*$, $\ell = 1, 2, \ldots, m$, $\ell \neq i$, $\ell \neq j$ with $\delta a_i = \delta a_i^o$, $\delta a_j = \delta a_j^o$ and that the plot is centered at $(a_i, a_j) = (a_i^*, a_j^*)$, the function of the lightpen commands is as follows (a_i changes

FIG. 7.2--A contour plot using numbers.

horizontally and a_j changes vertically):

* DOWN — Changes interval of a_j to $a_j^* - 2\,\delta a_j \le a_j \le a_j^*$.
* UP — Changes interval of a_j to $a_j^* \le a_j \le a_j^* + 2\,\delta a_j$.
* RIGHT — Changes interval of a_i to $a_i^* \le a_i \le a_i^* + 2\,\delta a_i$.
* LEFT — Changes interval of a_i to $a_i^* - 2\,\delta a_i \le a_i \le a_i^*$.
* EXPAND HØRIZ — Changes interval of a_i to $a_i^* - 2\,\delta a_i \le a_i \le a_i^* + 2\,\delta a_i$.
* EXPAND VERT — Changes interval of a_j to $a_j^* - 2\,\delta a_j \le a_j \le a_j^* + 2\,\delta a_j$.
* CNTRCT HØRIZ — Changes interval of a_i to $a_i^* - \frac{\delta a_i}{2} \le a_i \le a_i^* + \frac{\delta a_i}{2}$.
* CNTRCT VERT — Changes interval of a_j to $a_j^* - \frac{\delta a_j}{2} \le a_j \le a_j^* + \frac{\delta a_j}{2}$.
* PRINT THIS — Causes printing on the line printer of the contour plot as displayed.
* USE LETTERS — MIN=A, MAX=Z — Sets the number of subintervals of the range of functional values to 26 and uses the letters of the alphabet in the plot. "A" is used for values in the lowest subinterval up through "Z" for the highest subinterval.

* USE NUMBERS — MIN=0, MAX=9

Sets the number of subintervals of the range of function values to 10 and uses the decimal digits in the plot. "0" is used for values in the lowest subinterval and "9" for the highest subinterval.

CHOOSE GRID SIZE

Here there are four options specifying various plot sizes. The largest, requires $33 \times 51 = 1683$ functional evaluations, where as the smallest requires $15 \times 21 = 315$ evaluations. If the small size plot is sufficient to show the desired features, considerable savings in computer time may be made.

* FINISHED WITH THIS — This is a signal to exit from this display and go on to the next step in the program.

The lightpen options provide considerable versatility in rough scanning of a surface as well as minute examination of some point of interest. In practice, the use of the letters often gives a more "pleasing" picture, in that it shows more detail.

Figure 7.3 shows the use of letters for the contour symbols as well as illustrating the largest choice of plot size.

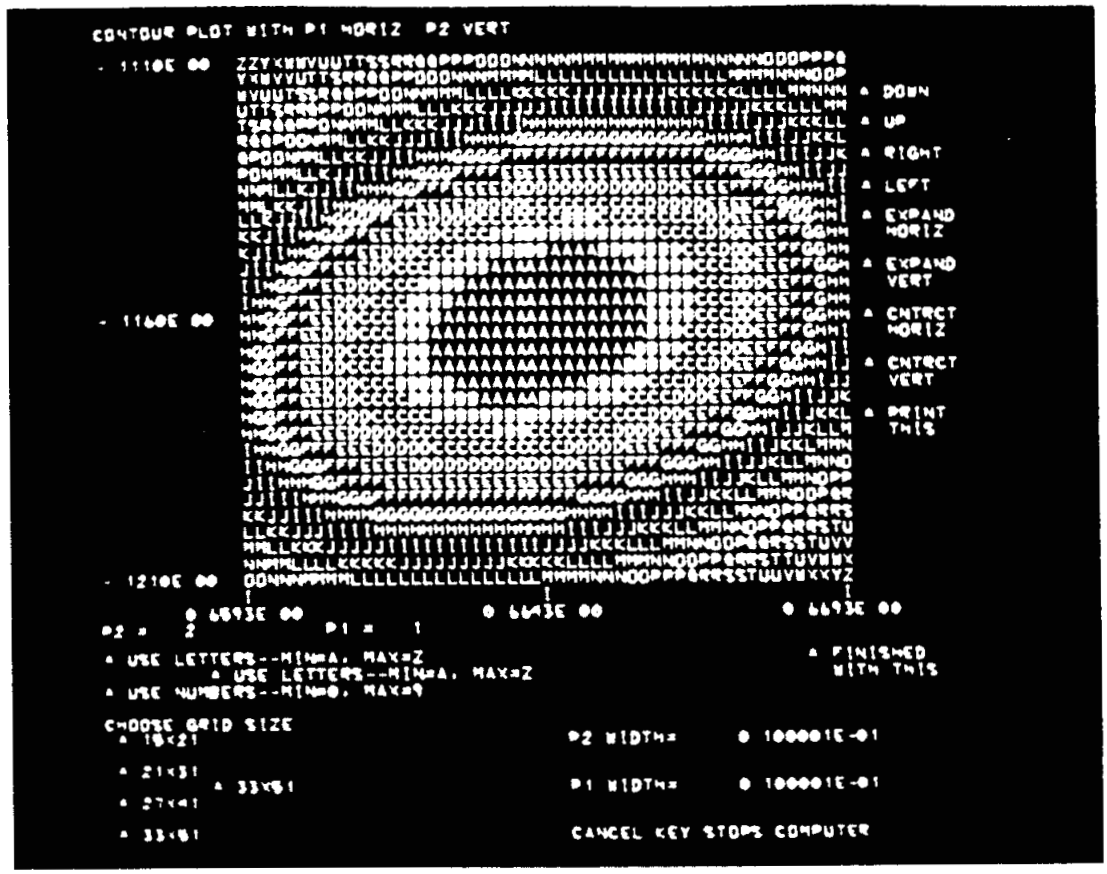

FIG. 7.3--A contour plot using letters.

D. A Program to Solve Maximum Likelihood Problems

A program has been written to solve problems by the maximum likelihood method as previously described. The program employs the interactive minimizer described in Chapter V to minimize the negative of the likelihood function. A flowchart showing an overview of the program is given in Fig. 7.4.

The numbers on each block of the flowchart in Fig. 7.4 correspond to the paragraphs to follow which describe the displays and the computations associated with each block. First we shall discuss the coding of the probability function which must be provided for each problem to be solved by this maximum likelihood function.

1. Coding the Probability Function

As discussed in Section VII.A the maximum likelihood problem involves a likelihood function L, given by

$$L(a_1, a_2, \ldots, a_m) = \prod_{i=1}^{n} P(x_i, y_i, a_1, \ldots, a_m) \quad .$$

The probability function P must, of course, be defined anew for each problem to be solved, but this is all that must be changed for the program to solve a new problem. In practice the data input subroutine will probably be changed to some extent for reasons of computational efficiency, but first let us give the required form of the probability function. To be used by the existing program a function coded as shown in Fig. 7.5 should be compiled, and the object deck included within the run deck for the maximum likelihood program.

There will often be certain quantities in the expression for the probability at (x_i, y_i) that can be evaluated independently of the values of the parameters $\{a_j\}_{j=1}^{m}$. Such quantities should be evaluated only once and not recomputed for

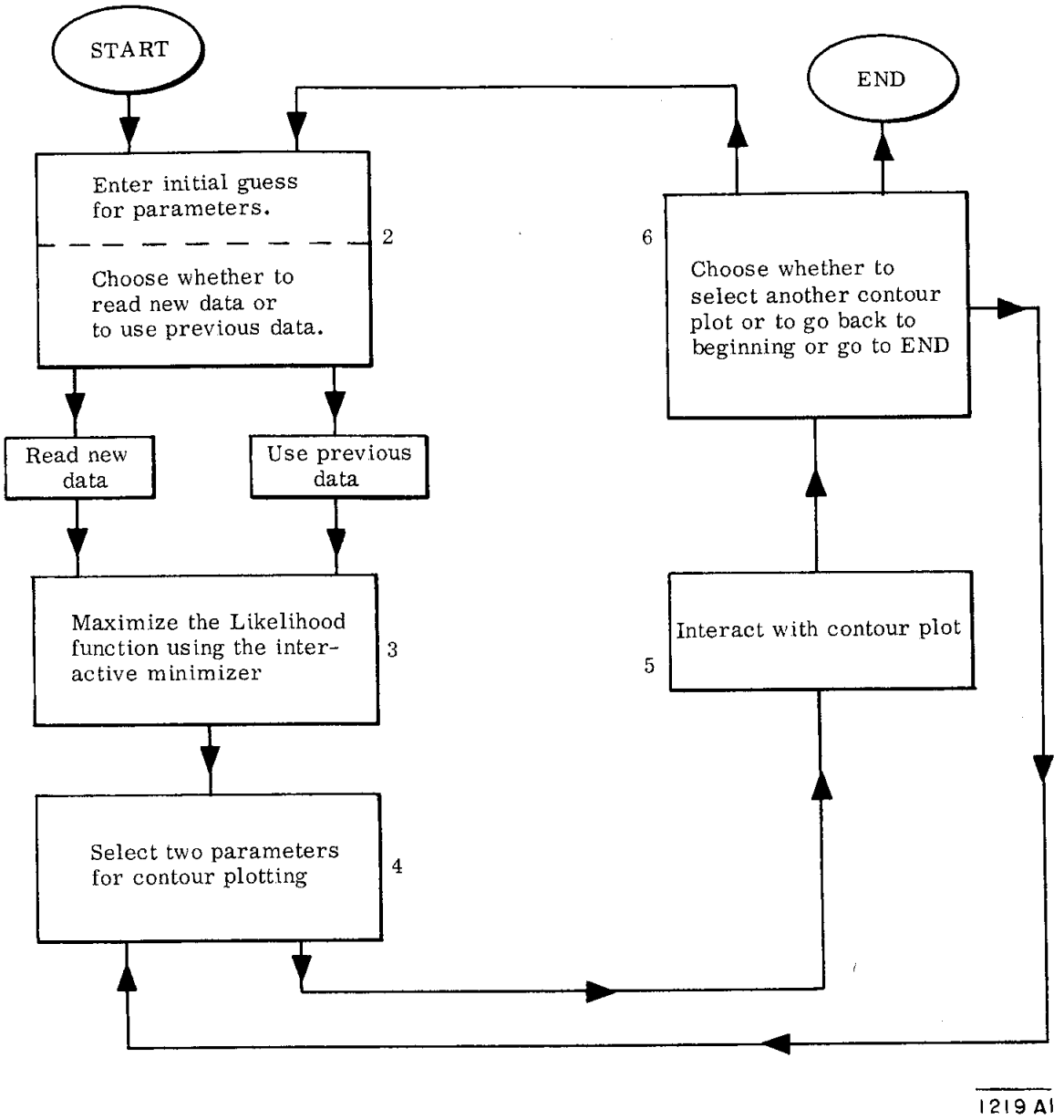

FIG. 7.4--Flowchart — Maximum likelihood program.

FORM FOR CODING THE PROBABILITY FUNCTION

```
C
      REAL FUNCTION PROB(XI,YI,A,M)
C
      INTEGER M
      REAL    XI,YI,A(M)
C
C     (XI,YI)  IS THE POINT AT WHICH THE PROBABILITY
C              IS TO BE CALCULATED.
C
C     A(M)   CONTAINS VALUES FOR THE M PARAMETERS
C              INVOLVED.
C
C     COMPUTE THE PROBABILITY AT (XI,YI) AND STORE
      THE RESULT IN PROB.
C
C
      PROB = (...)
C
C
      RETURN
C     END PROB
      END
```

FIG. 7.5--Form for coding the probability function.

every evaluation of $L(a_1, \ldots, a_m)$. Thus for example the probability at (x_i, y_i) might have the form

$$P(x_i, y_i, a_1, \ldots, a_m) = \sum_{j=1}^{m} a_j g_j(x_i, y_i) \quad . \qquad \text{(VII. 7)}$$

In such cases the functions $\{g_j\}_{j=1}^{m}$ should be evaluated once for each data point (x_i, y_i), the results saved (in COMMØN, for example) and the evaluation of (VII. 7) would simply involve m - 1 additions and multiplications. This can save large amounts of computer time if the functions $g_j(x_i, y_i)$ are of any complexity.

The advance calculation of the parameter independent terms in (VII. 7) should be carried out during the initial input of the data points. In this way an area of memory common to the input subroutine and the probability function (see Fig. 7.5) can be used to store the needed terms. Therefore in addition to the coding of a probability function, an input routine should also be coded for each maximum likelihood problem. This pair of routines would be coded as outlined in Fig. 7.6 and Fig. 7.7.

Some saving in storage space may be realized if the coordinates of the data points are not needed once the parameter independent quantities are computed. In such a case the arrays x(500) and y(500) could be eliminated (see Figs. 7.6 and 7.7).

Constraints on the parameter values may be introduced easily in this treatment of the maximum likelihood data-fitting problem. In the evaluation of the probability function as shown in Fig. 7.5 or Fig. 7.7, if the values of the parameter fails the constraint test, the probability can be set to zero, thus causing the minimization routine to avoid that value of the parameter (remember we are minimizing -L where $L \geq 0$).

FORM FOR CODING AN INPUT ROUTINE

```
C
      SUBROUTINE READIN(NPTS)
C
      INTEGER NPTS
C
C     NPTS WILL BE THE NUMBER OF DATA POINTS
C
      COMMON X(500),Y(500),G1(500),G2(500),G3(500)
      REAL X,Y,G1,G2,G3
C
      INTEGER I
C
      READ(5,101) NPTS
  101 FORMAT( I5)
C
      DO 200 I=1,NPTS
          READ(5,111) X(I),Y(I)
  111     FORMAT( 2F10.5 )
C
C         NOW COMPUTE THE PARAMETER INDEPENDENT
C         QUANTITIES FOR EACH POINT
C
          G1(I) = ...G1(X(I),Y(I))...
          G2(I) = ...G2(X(I),Y(I))...
          G3(I) = ...G3(X(I),Y(I))...
C
  200 CONTINUE
      RETURN
C     END READIN
      END
```

FIG. 7.6--Form for coding an input routine.

FORM FOR CODING PROBABILITY FUNCTION

```
C          USING COMMON STORAGE
C
      REAL FUNCTION PROB(I,A,M)
C
      INTEGER  I,M
      REAL     A(M)
C
C     I  IS THE INDEX OF THE POINT AT WHICH THE
C        PROBABILITY IS TO BE COMPUTED.
C     A  CONTAINS VALUES FOR THE M PARAMETERS.
C
      COMMON X(500),Y(500),G1(500),G2(500),G3(500)
      REAL X,Y,G1,G2,G3
C
      PROB = A(1)*G1(I) + A(2)*G2(I) + A(3)*G3(I)
C
      RETURN
C     END  PROB
      END
```

FIG. 7.7--Form for coding probability function.

2. Enter Initial Parameter Values and Choose Input Mode

The first display to appear on the screen when using the maximum likelihood program is shown in Fig. 7.8. This display requests initial values for the parameters and also an increment for each parameter. The increments are used as initial half plot widths by the interactive minimizer. That is, if the initial values are given as $\{a_i\}_{i=1}^{m}$ and $\{\delta a_i\}_{i=1}^{m}$, the initial curve shown by the interactive minimizer for the i^{th} parameter will be a plot of -L as the i^{th} parameter varies over the interval $\left[a_i - \delta a_i,\ a_i + \delta a_i\right]$.

Also requested by the display is a user decision on input mode. A new data set may be requested (*READ DATA CARDS) or the previous data set can be chosen. In this manner several different starting values may be tried for the same data.

3. Maximize L Using the Interactive Minimizer

The interactive minimizer described in Chapter V is used to maximize the likelihood function by minimizing - L. The minimization is accomplished by iteratively adjusting each parameter so as to force the partial derivative of - L with respect to that parameter to be zero while all other parameters are held fixed. The details of this process are given in paragraph VII.B.1. Figure 7.9 shows a display typical of those involved in the minimization process. The minimum point on the curve plotted in Fig. 7.9 is the point at which the partial derivative is zero and the lightpen commands shown allow interactive selection of that point. The results of the minimization are printed on the line printer.

Following convergence of the minimization (maximization) process the error estimates for each parameter can be obtained by cycling through the interactive minimizer one more time. During this cycle instead of reducing the parameter interval to determine its value more accurately, the interval will be

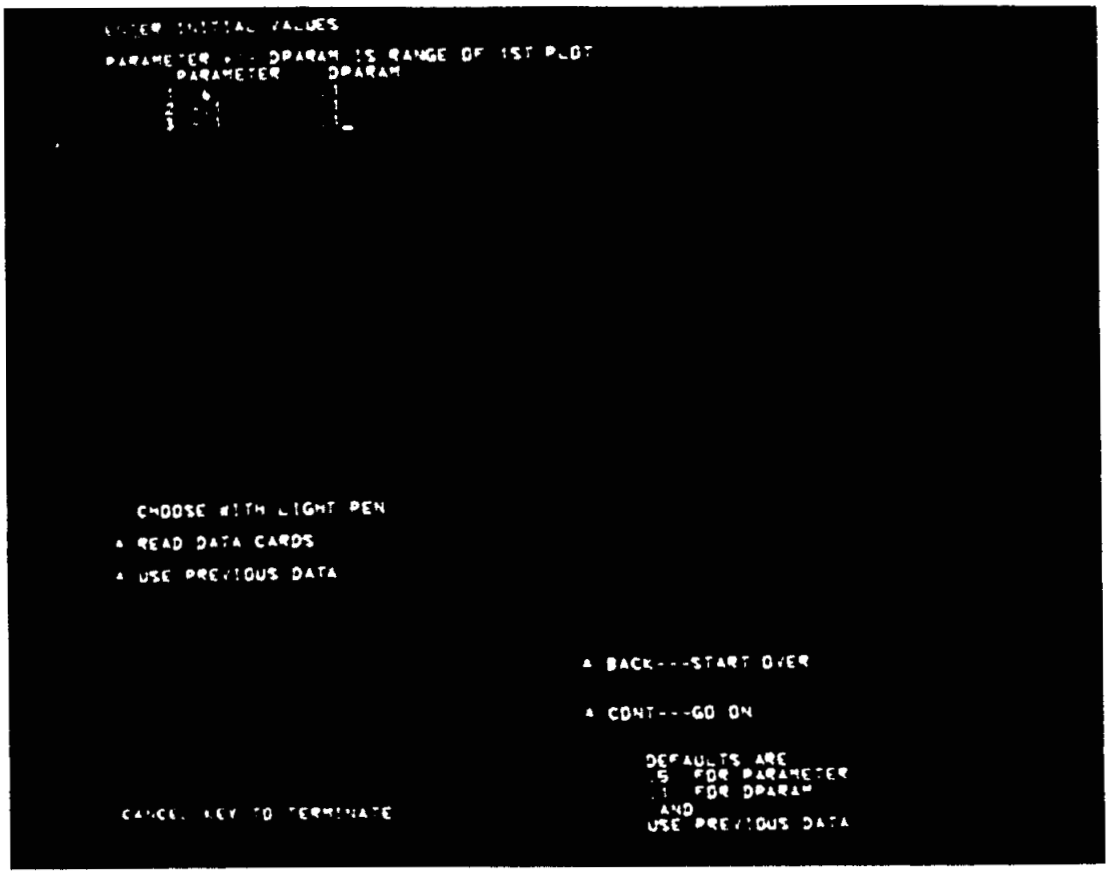

FIG. 7.8--Specify initial parameter values and choose input mode.

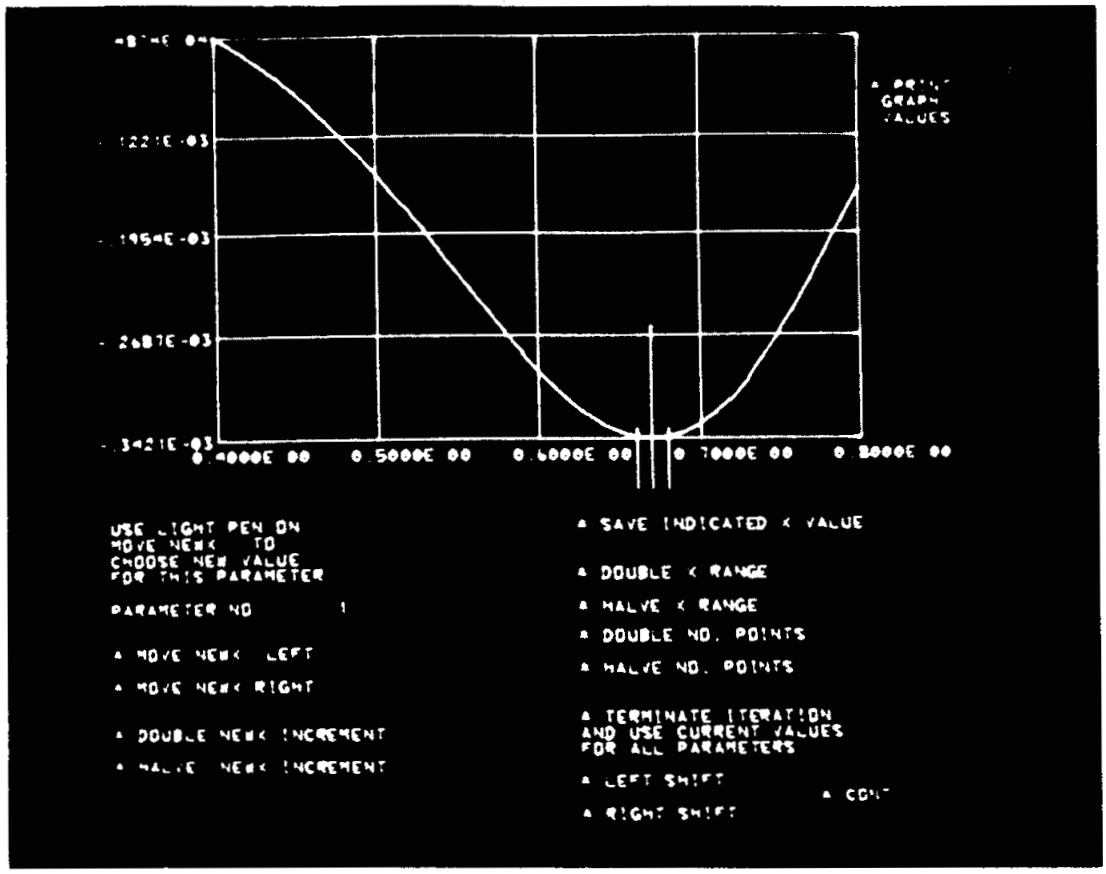

FIG. 7.9--Typical display during interactive minimization.

increased until L = 0 at both ends of the interval. The parameter value selected during the minimization must be kept at the center of the interval. When the interval has been widened sufficiently, the error estimates are calculated (as discussed in paragraph VII.B.3 by lightpen command. The results are printed on the line printer.

4. Select Two Parameters for Contour Plot

Following the minimization and error estimate calculations, a display appears which allows user selection of two parameters for which a contour plot will be produced to allow inspection of the interdependence of those parameters as discussed in Section VII.C. This display, shown in Fig. 7.10, allows entry of the quantities $\{\delta a_i\}_{i=1}^{m}$ which control the parameter intervals used in producing the contour plots. The lightpen is used to select any two of the parameters.

5. Interaction with Contour Plot

Once two parameters have been selected for contour plotting the initial contour plot is produced and interactive modification and examination of the plot can proceed. The details of the available lightpen orders available for contour plot modification are given in Section VII.C; their purpose is to allow overall as well as detailed examination of the contour plot.

6. Branching Display

When examination of a contour plot is terminated, the display shown in Fig. 7.11 allows one to choose either to select another pair of parameters for another contour plot or to return to the beginning of the program where another data set can be read in or another set of starting values can be entered. A third choice allows termination of the program.

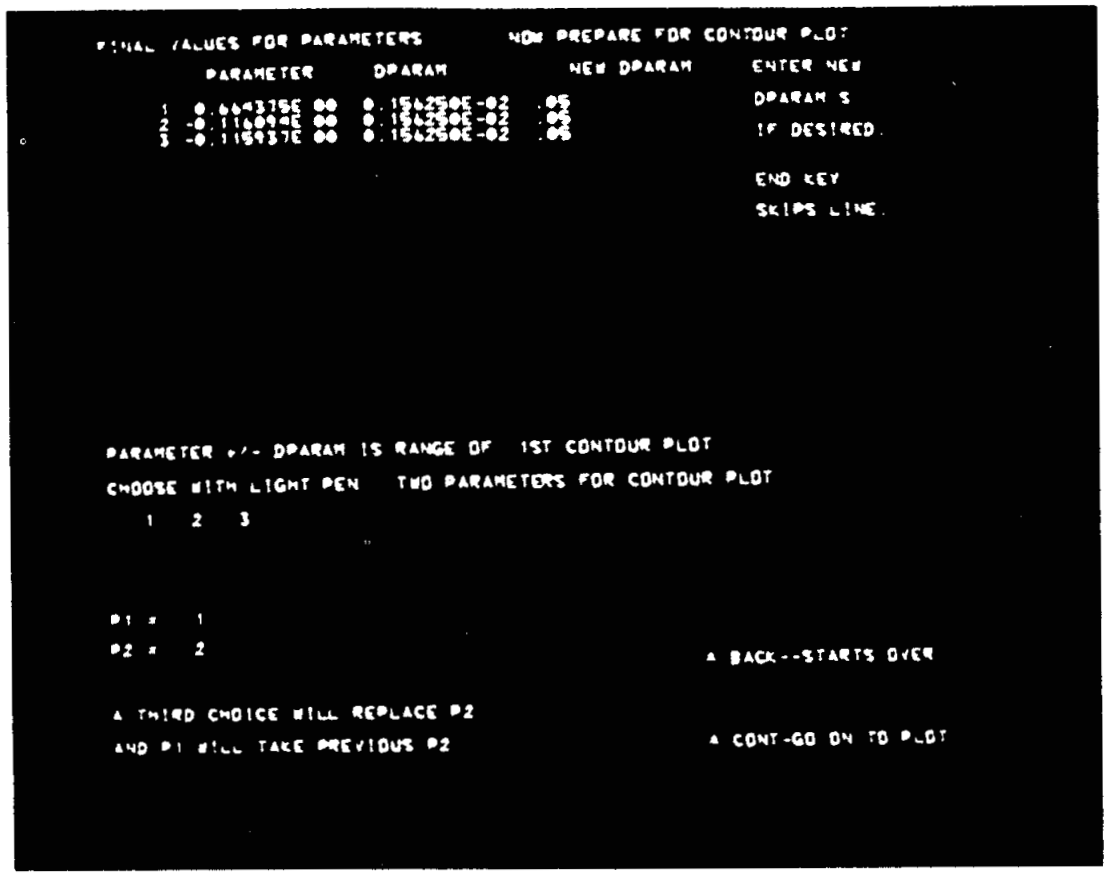

FIG. 7.10--Choosing parameters for contour plot.

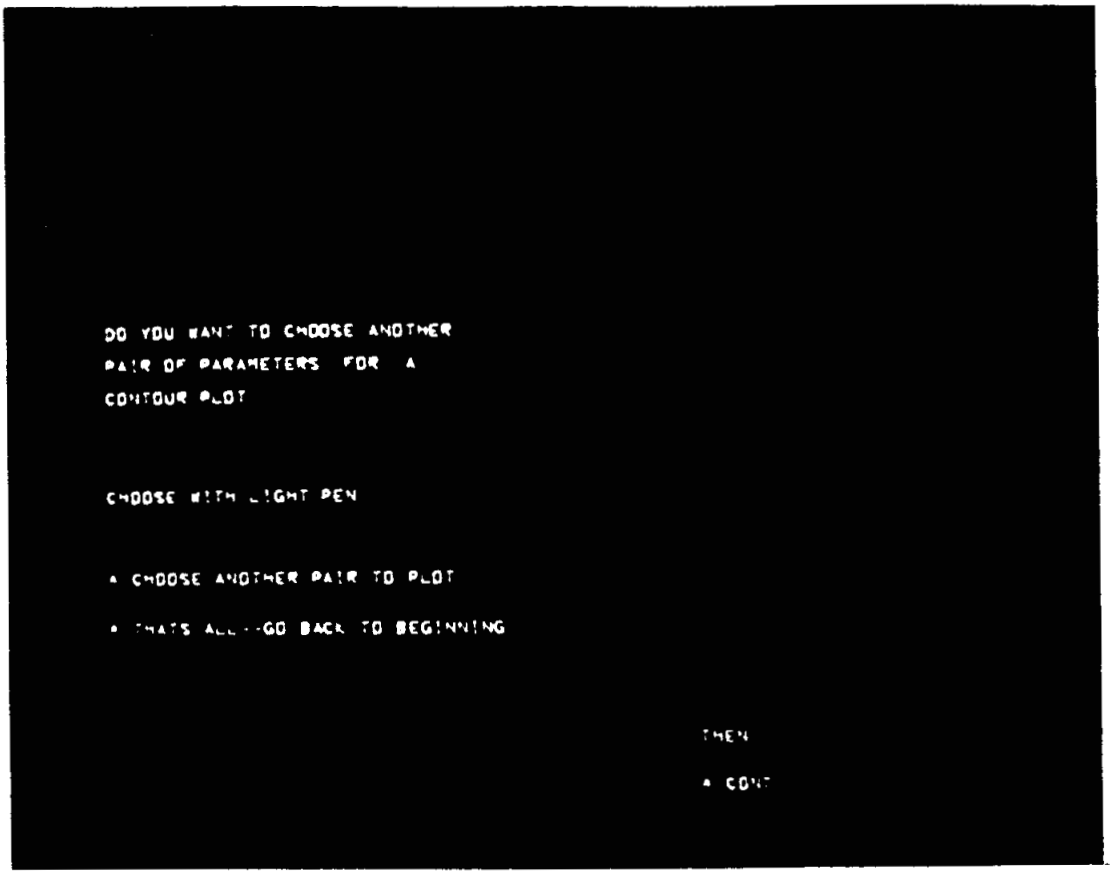

FIG. 7.11--Branching display.

E. Example Maximum Likelihood Problem

A maximum likelihood problem arising from the area of high energy physics is the following (Dr. William B. Johnson, SLAC, Johnson [1968]).

Results obtained from experiments involving the linear accelerator give data points (θ_i, ϕ_i). The physics of the problem then allow a rewriting of the problem with $u = \cos \theta_i$ and

$$v = \begin{cases} 180 + \phi_i, & \text{if } \phi_i < 0 \\ \phi_i, & \text{if } \phi_i \geq 0 \end{cases} .$$

A probability, $P(u,v)$, is then associated with each point (u_i, v_i) and is given by

$$P(u,v) = \frac{3}{\pi}\left[\frac{1}{2} + A\left(\frac{3u^2-1}{2}\right) - B(1-u^2)\cos 2v - c\left(2u\sqrt{1-u^2}\cos v\right)\right].$$

If we let

$$g_1(u,v) = \frac{3u^2-1}{2},$$

$$g_2(u,v) = -(1-u^2)\cos 2v, \text{ and}$$

$$g_3(u,v) = -\left(2u\sqrt{1-u^2}\cos v\right),$$

we can economize during the evaluation of L, where L is given by

$$L(A,B,C) = \prod_{i=1}^{n} P(u_i, v_i) .$$

As discussed in paragraph VII. D. 1, the quantities $g_1(u_i, v_i)$, $g_2(u_i, v_i)$, and $g_3(u_i, v_i)$, $i = 1, 2, \ldots, n$ can be evaluated once and saved for later use. The evaluation of L is then simply

$$L(A,B,C) = \prod_{i=1}^{n} \frac{3}{\pi}\left[\frac{1}{2} + g_1(u_i, v_i)\cdot A + g_2(u_i, v_i)\cdot B + g_3(u_i, v_i)\cdot C\right] .$$

Figure 7.12 shows the interactive minimizer results when solving this example problem for a particular set of data. Note that the change in the function value (- L) was very small on the 4^{th} iteration. Thus convergence (maximization of L) is assumed.

Figures 7.13, 7.14, and 7.15 show contour plots comparing the three coefficients of this example problem.

F. Extension to Higher Dimensions

So far the discussion of maximum likelihood problems has been confined to problems with data points taken from a two-dimensional space (a plane). Given an appropriate probability function, the method is applicable to problems of any dimension. The probability function, $P(x_1, x_2, \ldots, x_d)$, is required to satisfy (in a d-dimensional space)

$$\int \cdots \int P(x_1, \ldots, x_d)\, dx_1 \cdots dx_d = 1 .$$

The likelihood function for n data points is, as before, defined to be

$$L(a_1, \ldots, a_m) = \prod_{i=1}^{n} P(x_{1_i}, \ldots, x_{d_i}) ,$$

and can be treated in the same manner as before.

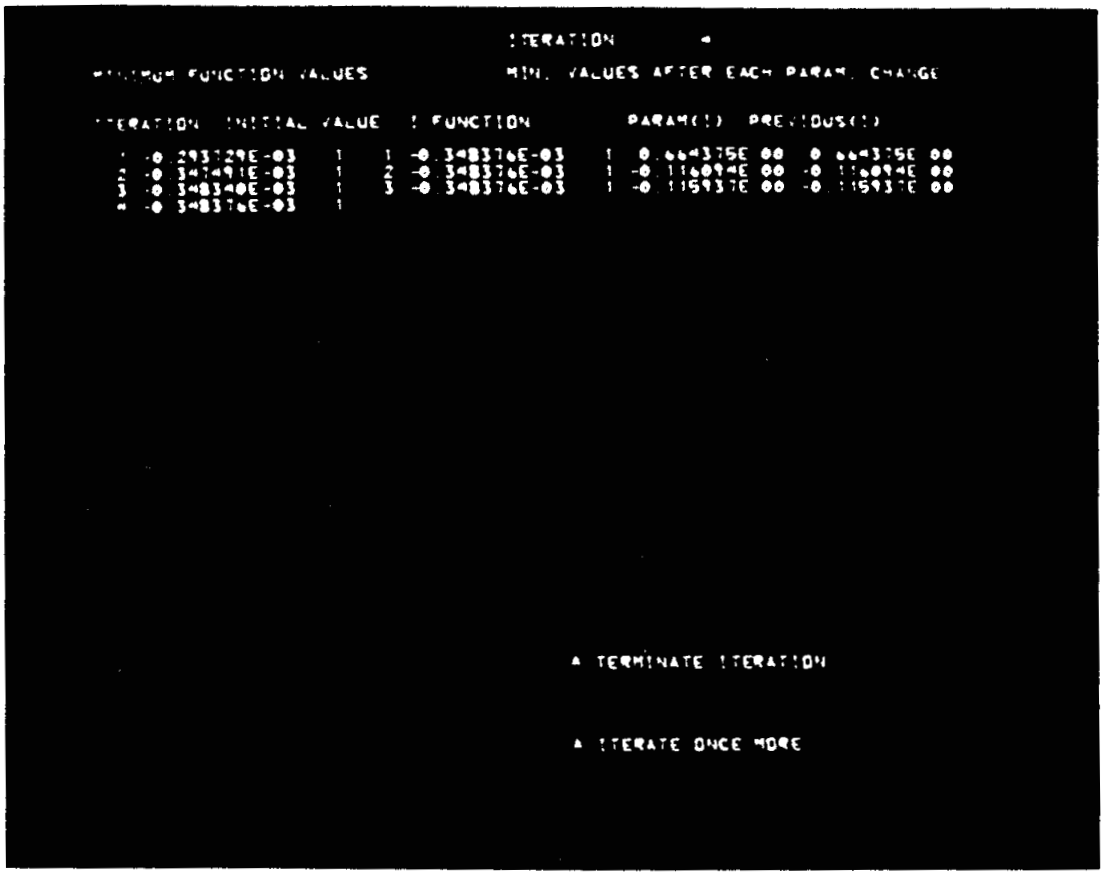

FIG. 7.12--Final iteration during maximization.

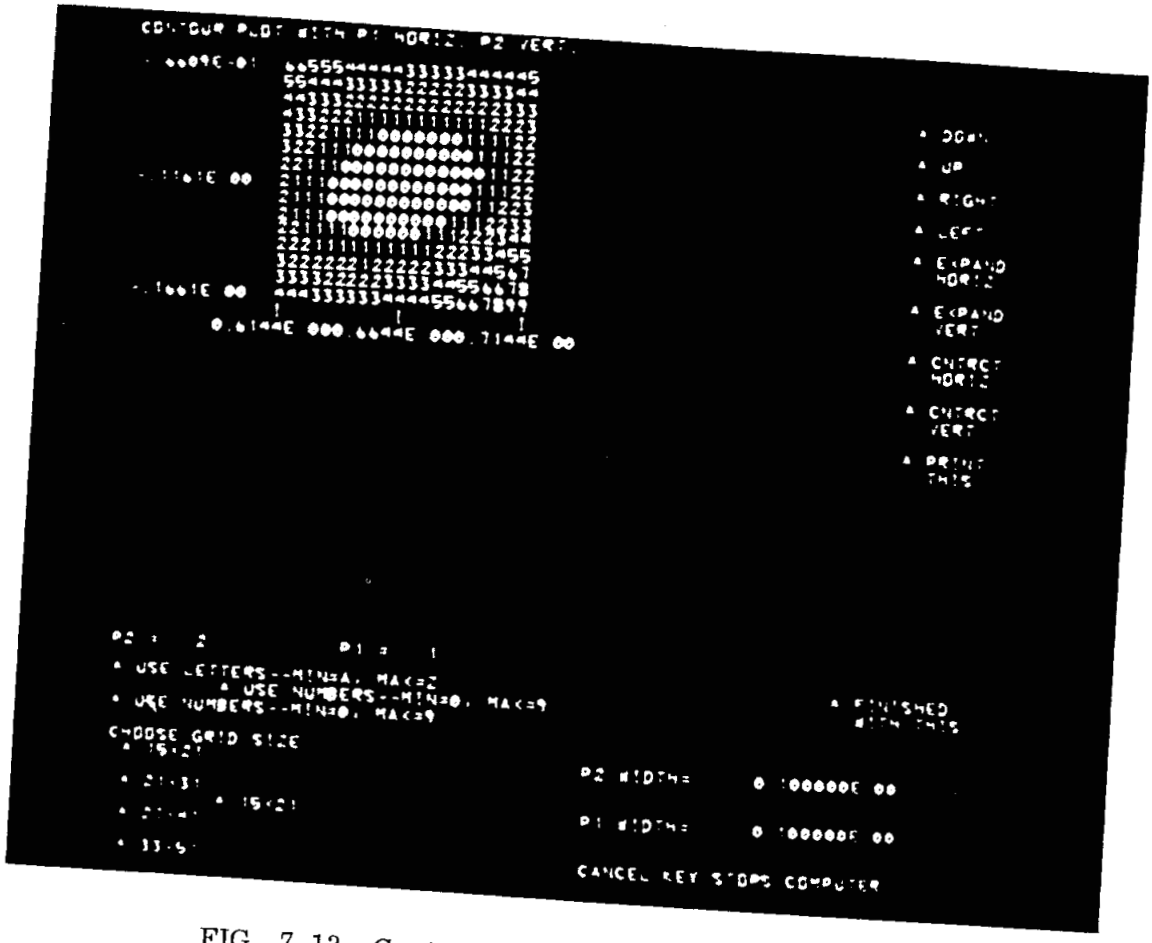

FIG. 7.13--Contour plot with P1 = A and P2 = B.

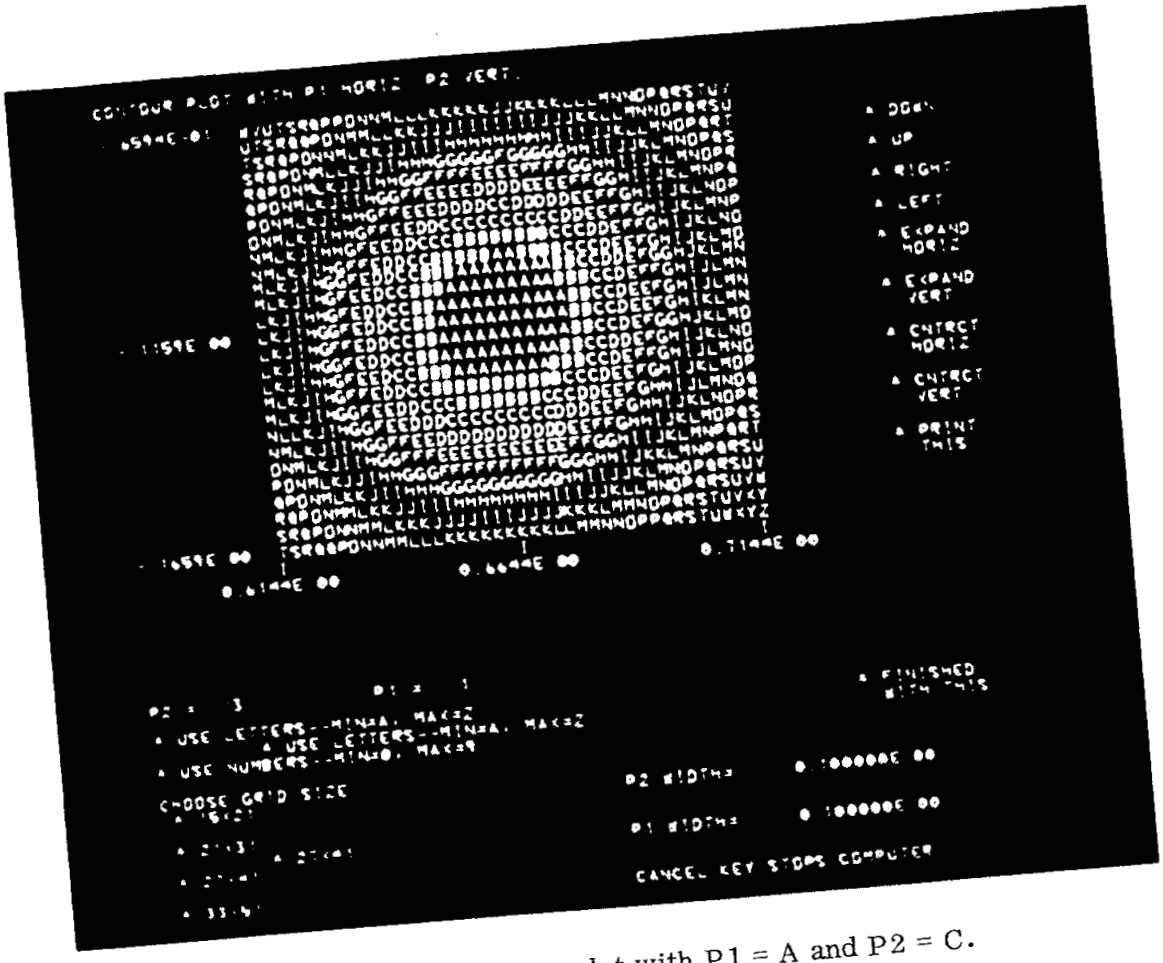

FIG. 7.14--Contour plot with P1 = A and P2 = C.

FIG. 7.15--Contour plot with P1 = B and P2 = C.

CHAPTER VIII

CONCLUSIONS

A. Summary

We have given a detailed account of an interactive problem solving system, PEG. PEG solves data-fitting problems and is oriented toward the non-programmer. The ability to select from the standard functions, orthogonal polynomials, Fourier approximation and spline functions provides the capability of getting a "good" fit to arbitrary data. The availability of user defined functions allows a scientist who wants to approximate his data by a particular function to be easily accommodated. The "easy-to-use" design of the interaction embodied in PEG, and the inclusion of the ability to use an arbitrary approximating function give PEG the combination of simplicity and effectiveness necessary to a useful interactive problem solving system.

In addition to showing the details of how PEG appears to a user and how various data-fitting problems are handled mathematically, we have shown examples of problems already successfully treated by PEG. Of particular interest was the problem of fitting an ellipse. This problem was of interest in itself but even more because it exemplified the use of implicit functions in the data-fitting problem area. The actual problems already solved by PEG show that the flexibility designed into the system is a valuable feature.

The interactive minimizer discussed in Chapter V is an example of the kind of interactive program that we will see, use and write more of, now that hardware capabilities allow on-line graphical interaction. This routine allows a close man-machine relationship during a problem solving session. The display of the curves as one parameter changes gives some graphical insight into the

behavior of the function being minimized. The ability to interact with these curves provides a way of "feeling out" a problem not available in a batch processing mode.

Maximum likelihood solutions to data-fitting problems as discussed in Chapter VII illustrate a method, other than least squares, that can be used effectively in an on-line environment. In this program we use contour plots to examine the correlation between various parameters of the approximating function. These contour plots, and the ability to interact with them on-line, also illustrate the type of interaction that will become more a part of numerical problem solving as we learn how to use on-line systems more effectively. Intuitively, the graphical representation of correlation as a contour plot provides more insight into a problem than the display or printout of a correlation coefficient given to six significant figures.

B. A Machine Independent Problem Solving Tool?

The stated goals of this dissertation were to provide a user-oriented scientific problem solving tool in an on-line environment and to make it as machine independent as practically possible. Have we reached these goals? That we have designed and implemented an effective least squares data-fitting tool is testified to by the example problem solutions and the elliptical fits given in Chapter VI. The maximum likelihood program in Chapter VII is a second data-fitting tool which uses another method; it too has been used to solve an actual problem. The question of machine independence requires further discussion which is given below.

The design goal of "simplicity-of-use" has been attained by using the principles listed in Chapter II. Some problems have been solved by people other than this author without much effort. This testifies to the usability of the system.

PEG is machine-independent to the degree afforded by having almost all of the code in Fortran IV. However, the routines that communicate with the CRT for generation of displays and handling lightpen and keyboard interrupts are written in assembly language and are peculiar to the IBM/360 and IBM 2250 hardware. It is hoped that by replacing the assembly language CRT handling routines by externally identical routines for different hardware, the PEG system could be run on another machine. These CRT handling routines are called by Fortran CALL statements in the PEG code, so new routines need only assume the same calling sequence to provide the same service. The routines involved at this level are, for example:

GVECT(X,Y):	Insert a vector with endpoint (X,Y).
GPNTAR(X,Y,N):	Insert N points from arrays X and Y.
GWTBUF(I,N,A):	Write data into buffer.
GPOLL(KEY,I,ITIME):	Poll for attentions.

The list of routines used at this level in PEG was determined by the package of IBM 2250 routines for Fortran use that was available when development of the interactive system was initiated. An area of future research which will do much to increase the machine-independent qualities of PEG and related efforts is the selection of basic interactive capabilities that can be expressed as Fortran callable routines similar to those in the above list. A study of the capabilities of various manufacturers' display equipment could possibly produce a list of functions which would lie in the intersection of the capabilities of the various display units. Systems such as PEG could then be coded to use only those capabilities for which the required machine dependent routines could be easily provided.

C. Future Work

What is the next step? We have presented working on-line data-fitting programs to help solve numerical problems in one area. The next step is actually two steps. First, extensions and improvements can be added to the existing work, and secondly, new numerical problem areas can be explored.

Let us enumerate some possible extensions and improvements that are applicable to the PEG system. The modularity in the design of PEG facilitates the addition or substitution of a new code.

1. Add more "built-in" functions. Already under consideration are rational functions and exponential functions.
2. Add more norms. For example, instead of just the least squares norm, the minimax norm and others could be added as options.
3. Add more display modes. There might be some additional display modes that would aid user interpretation of the results.
4. Implement "zoom" and variable extrapolation. These features would allow more detailed examination of graphical features.
5. Add more minimization methods. The current version of PEG allows a choice between the interactive minimizer and direct search in the case of user defined function fitting. It might be of value to add other methods to this list. Methods requiring derivative evaluation could be included by using either numerical approximations to the derivatives or including user written code to evaluate the derivative.

6. Include on-line entry of code for user functions. If an on-line text editor and compiler were made a part of the system so that Fortran code could be keyed in, edited, and compiled while still in contact with PEG, it would make PEG more convenient for those desiring to enter user defined functions. This facility could also be used to enter derivative functions if they were needed for a particular minimization method (see 5).

With some thought and imagination the reader can probably extend this list almost indefinitely. So, rather than do that here, let us go on to the next step.

The data-fitting problem area was chosen for this initial effort because of its obvious benefits from graphic display and interaction and because of this author's long standing interest in the area. Other classes of numerical problems can also benefit from this same kind of treatment. We will list a few such problem areas which we feel should be considered for treatment by interactive graphical programs.

1. Function minimization. This problem of locating optimum values has already received considerable mention in this dissertation; however, we believe it is an area that merits even more work. Useful results would come from an investigation that compares existing methods and explores the possibility of developing new on-line methods. New methods might use interaction to improve the effectiveness of existing algorithms or a combination of methods might prove to be most effective. Some entirely new approach using interaction could also be discovered.

2. Ordinary differential equations. Some work in this area has already been done by Gear [1966]; however, that work uses teletype terminals with correspondingly slow and crude graphs of results. An interactive ordinary differential equation solver with fast graphics as given by a CRT display would be a useful problem solving tool.

3. Partial differential equations. Research on interactive graphical systems to solve partial differential equations would be very rewarding. Regions could be displayed and modified interactively and the corresponding solutions indicated graphically as well. At the very least such a system could be a useful pedagogical tool.

4. Solution of simultaneous linear equations. This area exemplifies problems whose intermediate results can be graphically presented for quick comprehension by a user. The condition of the matrix involved in solving a system of linear equations is a critical factor in the ability of various algorithms to obtain a satisfactory solution. There are various ways to examine the condition of a matrix. Historically these examinations have most often produced a number whose size relates to the condition of the matrix. Geometrical consideration yields a hyperellipsoid whose axes are inversely proportional to the eigenvalues of the matrix. Thus an elongated hyperellipsoid indicates bad conditioning whereas a near hypersphere indicates a well conditioned matrix. It is felt that an on-line user will be

able to recognize and interpret a graphically presented hyperellipsoid more readily than a list of numbers.

The reader can probably extend this list easily by adding his own particular area of interest. Also, as interactive programs are developed for the various areas mentioned, the knowledge gained from experience will undoubtedly lead to the consideration of other problems that will benefit from on-line solution. Corresponding to the hyperellipsoids which occur in the linear systems problem, there are probably intermediate quantities in other problems that, if graphically displayed, could aid in problem solution and/or understanding.

In addition to the foregoing specific suggestions of future work, it is appropriate here to mention some general questions which have come to mind during the planning and implementation of the PEG system. Some of these have been answered, at least partially, in this dissertation; some are as yet unanswered. Interesting avenues of future research are suggested by many of these questions.

How should we design programs to make use of an on-line system? In Chapter II several suggestions as to the design of interactive systems are made. However, this is an area that will expand and develop with use. Once we have an interactive system in use we will say — "Wouldn't it be nice to be able to do such-and-such!" After adding that capability we will then see a need for some further addition. This growth of the "design tree" will expand into areas we have not yet imagined.

Once an interactive program to solve a class of problems exists, how can we compare the cost of problem solution to the cost of solving the same problem in a batch environment? Do we save money using on-line systems? Do we save man-hours? These questions must be answered to help determine the economic feasibility of solving certain problems on-line. There will be some problems

that should not be solved on-line. There may also be problems that can only be solved on-line. There may be interesting psychological issues involved. For example, is it enough to say that a problem can be solved in fewer (or more) man-hours with an on-line system or are there psychological advantages (or disadvantages) to having on-line problem solution? A further question to be answered might then be what economic implications the psychological effects might have. This problem of determination of the "value" of a certain type of programming system or environment is an interesting but complicated issue.

What differences in algorithm design should be made when developing algorithms for use in an interactive environment? Should we use existing algorithms and simply add interaction in some manner, or should a radically new design be considered? Minimization routines, for example, are a candidate for research on this question. Perhaps by graphical display of intermediate results and interaction between a numerical analyst and several known methods of minimization a new method specially tailored to on-line use can be developed. It could be a combination of known methods or a completely new approach spawned by the on-line environment.

What problems can be solved by use of interactive systems that could not otherwise be solved? Can we prove that any such problems exist? At first, it is hard to conceive of a candidate for such a problem since our first reaction to having on-line computing is to use it to give us instant turnaround. However, as we pause to consider the advantages of on-line computing in more depth, we realize that we do indeed have a new dimension to our computing power. The speed with which the computer can respond to our thoughts does, I believe, provide the capability to enter new problem solving areas. Perhaps we cannot prove theoretically that we can solve any new problems, but we can undoubtedly

prove in a practical or economical sense that previously unsolvable problems can be solved. Take a human with his (her) intuition, analog capabilities and "flashes of insight" and add to his abilities the problem solving system that can be programmed for a computer. The combination makes a tool more powerful than either alone.

APPENDIX A

FORTRAN CODE FOR LEAST SQUARES BY FORSYTHE ORTHOGONAL POLYNOMIALS

Here we give a FORTRAN IV subroutine which computes the orthogonal polynomial least squares fit to a set of data by the method of Forsythe. The method is described in Chapter IV. A subroutine to evaluate the orthogonal polynomial fit at a set of abscissa values is also given.

It is suggested that the abscissa values of the data be scaled to the interval $[-2, 2]$ to improve the computational accuracy of the solution. If this is done, the input to the evaluation subroutine should also be so scaled.

ORTHOGONAL POLYNOMIAL LEAST SQUARES FIT

```
C
      SUBROUTINE GEFOP1(N,M,X,Y,W,K,GEFTST,A,B,SO,S,DSO,DS,
     #                  SIGMA2,OK,NN)
C     COMPUTES ORTHOGONAL POLYNOMIAL LEAST SQUARES FIT
C     A LA  G.E.FORSYTHE IN J.SIAM, JUNE 1957,PP.74-88
C
C TYPE  INPUTS
CINTEGER  N     MAXIMUM DEGREE TO COMPUTE.
CINTEGER  M     NUMBER OF DATA POINTS (X(I),Y(I)) I=1,M
CREAL     X(M) THE ABSCISSA VALUES OF THE DATA POINTS.
CREAL     Y(M) THE ORDINATE VALUES OF THE DATA POINTS.
CREAL     W(M) THE POSITIVE WEIGHTS ASSOCIATED WITH
C              EACH DATA POINT---W(1).EQ.0.0 MEANS
C              NO WEIGHTS.
CINTEGER  K     VALUE OF N ON PREVIOUS CALL OF THESE SAME
C              DATA POINTS---THIS ALLOWS SUBSEQUENT CALLS
C              TO COMPUTE HIGHER DEGREE FITS WITHOUT ANY
C              RECOMPUTING.
CLOGICAL  GEFTST     IF .TRUE. PROGRAM DETERMINES BEST
C                              DEGREE .LE. N (NEW N PUT IN NN)
C                    IF .FALSE.  PROGRAM COMPUTES ALL DEGREES
C                              INCLUDING N AND LETS USER TEST
C                              VALUES IN SIGMA2 FOR BEST FIT.
C
C       OUTPUTS
CREAL     A(N) THE COEFFICIENTS ALPHA IN THE RECURRENCE
C              RELATION DEFINING THE ORTHOGONAL POLYNOMIALS.
CREAL     B(N) THE COEFFICIENTS BETA  IN THE RECURRENCE
C              RELATION DEFINING THE ORTHOGONAL POLYNOMIALS.
CREAL     SO AND S(N)  THE COEFFICIENTS OF THE LEAST SQUARES FIT.
CREAL     DSO AND DS(N)  ESTIMATES OF THE STANDARD DEVIATIONS OF
C              THE ERROR IN THE COEFFICIENTS.
CREAL     SIGMA2(N) THE STATISTIC  SIGMA**2 WHICH DECREASES
C                   AS THE DEGREE OF FIT INCREASES--UNTIL
C                   THE BEST DEGREE IS REACHED, AT THAT
C                   POINT SIGMA2 WILL LEVEL OFF OR INCREASE.
CLOGICAL  OK    IF .TRUE. PROGRAM HAS SUCESSFULLY COMPUTED
C                        A LEAST SQUARES FIT
C               IF .FALSE. ONE OF THE FOLLOWING HAS OCCURRED:
C                     (1) N .GE. M       (2) M .LE. 1
C                     (2) SOME W(I).EQ.0.0 WHEN W(1).NE.0.0
CLOGICAL  NN    THE BEST DEGREE .LE. N IF GEFTST =.TRUE.
C     THE LEAST SQUARES FIT OF DEGREE N IS GIVEN BY
C
C                              I=N (            )
C          SO*PO(X) + SUMMATION ( S(I)*PI(X))
C                              I=1 (            )
C
```

```
C      WHERE
C           P0(X) = 1
C           P1(X) = X*P0(X) - ALPHA(1)*P0(X)
C           P2(X) = X*P1(X) - ALPHA(2)*P1(X) - BETA(1)*P0(X)
C           P3(X) = X*P2(X) - ALPHA(3)*P2(X) - BETA(2)*P1(X)
C           ...
C
      INTEGER N,M,K
      LOGICAL GFFTST,OK
      REAL X(M),Y(M),W(M),A(N),B(N),S(N),SG,SIGMA2(N),DS0,DS(N)
C
C     THE DIMENSION OF PIM1,PI,PIP1 MUST BE
C          .GE. M  (THE NUMBER OF DATA POINTS)
C
      REAL PIM1(100),PI(100),PIP1(100)
      REAL WII,WIP1,DELTA,WI,DELTAI,GAMMAI,DSS
      REAL DENOM,SI,SIGI,DSIG1,DSIG2,MINSIG
      INTEGER I,J
      LOGICAL NOWTS
C
      NN = N
      OK = .TRUE.
      NOWTS = .TRUE.
      IF(W(1) .GT. 0.0) NOWTS = .FALSE.
      IF(M .LE. 1) GO TO 255
      IF(NOWTS) GO TO 102
      DO 101 I=2,M
           IF(W(I) .LE. 0.0) GO TO 255
  101 CONTINUE
  102 IF(K .EQ. 0) GO TO 104
      I = K
      GO TO 175
C
  104 DO 105 I=1,M
           PIM1(I)= 0.0
           PI(I)  = 1.0
  105 CONTINUE
      WII = M
      IF(NOWTS) GO TO 107
      WII = 0.0
      DO 106 I=1,M
           WII = WII + W(I)*W(I)
  106 CONTINUE
  107 SI = 0.0
      DELTA = 0.0
      DO 120 I=1,M
           IF(NOWTS) GO TO 118
           DELTA = DELTA + Y(I)*Y(I)*W(I)*W(I)
           GO TO 120
  118      DELTA = DELTA + Y(I)*Y(I)
  120 CONTINUE
C
```

```
      I = 0
  130 WI = 0.0
      GAMMAI = 0.0
      DO 135 J=1,M
          IF(NOWTS) GO TO 133
          GAMMAI = GAMMAI + PI(J)*PI(J)*W(J)*W(J)
          WI = WI + Y(J)*PI(J)*W(J)*W(J)
          GO TO 135
  133     GAMMAI = GAMMAI + PI(J)*PI(J)
          WI = WI + Y(J)*PI(J)
  135 CONTINUE
  140 SI = WI / WII
      DELTAI = DELTA - SI*SI*WII
      DENOM  =  M - I - 1
      IF(DENOM .LE. 0) GO TO 150
      SIGI   =  DELTAI / DENOM
      GO TO 160
  150 SIGI   =  0.0
  160 DSS = SQRT(SIGI / GAMMAI)
C
      IF( I .LE. 0) GO TO 163
          S(I) =  SI
          DS(I) = DSS
          SIGMA2(I) = SIGI
          IF(I .EQ. 1) MINSIG = SIGI
          GO TO 164
  163 S0 = SI
      DS0= DSS
  164 CONTINUE
C
      IF(.NOT. GEFTST) GO TO 170
      IF(I .LE. 0) GO TO 165
      IF(SIGMA2(I) .GE. MINSIG) GO TO 165
      MINSIG = SIGMA2(I)
      NN = I
  165 CONTINUE
C     TEST FOR BEST DEGREE .GE. 3
      IF(I .LT. 3) GO TO 170
      DSIG1 = SIGMA2(I-1) - SIGMA2(I)
      DSIG2 = SIGMA2(I-2) - SIGMA2(I-1)
      IF( ((DSIG1.LT.0.0).OR.(DSIG1.LE. 0.01*SIGMA2(I)))
     *   .AND.
     *    ((DSIG2.LT.0.0).OR.(DSIG2.LE. 0.01*SIGMA2(I-1))) )
     *   GO TO 166
      GO TO 170
  166 NN = I-2
      GO TO 260
C
  170 IF(I .GE. N) GO TO 260
  175 DO 180 J=1,M
          PIP1(J) = X(J)*PI(J)
  180 CONTINUE
```

```
C
      A(I+1) = 0.0
      DO 190 J=1,M
          IF(NOWTS) GO TO 185
          A(I+1) = A(I+1) + PIP1(J)*PI(J)*W(J)*W(J)
          GO TO 190
  185     A(I+1) = A(I+1) + PIP1(J)*PI(J)
  190 CONTINUE
      A(I+1) = A(I+1) / WII
C
  200 DO 205 J=1,M
          PIP1(J)= PIP1(J) - A(I+1)*PI(J) - B(I)*PIM1(J)
  205 CONTINUE
      WIP1 = 0.0
      DO 220 J=1,M
          IF(NOWTS) GO TO 215
          WIP1 = WIP1 + PIP1(J)*PIP1(J)*W(J)*W(J)
          GO TO 220
  215     WIP1 = WIP1 + PIP1(J)*PIP1(J)
  220 CONTINUE
C     NOW INCREASE I BY ONE
C
      I = I + 1
      DO 235 J=1,M
          PIM1(J) = PI(J)
          PI(J)  = PIP1(J)
  235 CONTINUE
C
      B(I) = WIP1 / WII
      WII  = WIP1
      DELTA= DELTAI
C     END OF LOOP ON I
      GO TO 130
C
C     ERROR EXIT
  255 OK = .FALSE.
C
C     NORMAL EXIT
  260 CONTINUE
      RETURN
C     END GEFOP1
      END
```

SUBROUTINE TO EVALUATE ORTHOGONAL POLYNOMIAL FIT

```
      SUBROUTINE EVALXS(N,M,X,A,B,S,Y,SO)
C     EVALUATES ORTHOGONAL POLNOMIAL FIT AT M POINTS
C     N = DEGREE OF FIT
C     M = NUMBER OF POINTS
C     X = ARRAY CONTAINING THE M POINTS
C     A = THE ALPHAS FOR RECURRENCE
C     B = THE BETAS FOR RECURRENCE
C     S = THE COEFFICIENTS OF THE ORTHOGONAL POLYNOMIALS
C     Y = VALUE OF FIT AT X()
      INTEGER N,M
      REAL X(M),A(N),B(N),S(N),Y(M)
      REAL SO
C
      INTEGER I,J
      REAL PI(100),PIM1(100),PIP1(100)
      REAL SI
C
  100 DO 105 J=1,M
          Y(J)= 0.0
          PIM1(J)= 0.0
  105     PI(J) = 1.0
C
  110 I = 0
  111 SI = SO
  115 DO 120 J=1,M
  120     Y(J) = Y(J) + SI*PI(J)
  125 IF(I .GE. N) GO TO 160
  130 DO 135 J=1,M
  135     PIP1(J) = X(J)*PI(J)
  140 DO 145 J=1,M
  145     PIP1(J) = PIP1(J)- A(I+1)*PI(J)- B(I)*PIM1(J)
      DO 144 J =1,M
          PIM1(J) = PI(J)
  144     PI(J) = PIP1(J)
  150 I = I + 1
  151 SI = S(I)
  155 GO TO 115
  160 RETURN
      END
```

APPENDIX B

FORTRAN CODES FOR LEAST SQUARES BY FOURIER APPROXIMATION (GOERTZEL'S ALGORITHM)

There are three FORTRAN subroutines given here.

1. CAB — Computes the Fourier coefficients of the least squares approximation.
2. ERRAB — Computes the probable error in the coefficients of the fit. ERRAB also computes the statistic SIG(σ) which is used to determine the "best" fit.
3. FEVAL — Evaluates the Fourier approximation either at the given data points or at another set of equally spaced points on the same interval.

```
C      FORTRAN CODE TO COMPUTE FOURIER APPROXIMATIONS TO
C      EQUALLY SPACED DATA POINTS (NO WEIGHTING)
C
C
       SUBROUTINE CAB(N,Y,M,L,A0,A,B,J0)
C
C      THIS SUBROUTINE COMPUTES THE FOURIER COEFFICIENTS
C      OF THE APPROXIMATION TO Y.
C      Y() CONTAINS THE ORDINATS OF DATA POINTS WHERE WE (N POINTS)
C          ASSUME THAT THE ABCISSAE ARE EQUALLY SPACED
C          AND START AT 0.
C      M   IS THE NUMBER OF SIN (AND COS) TERMS WE ARE
C          USING IN THE FIT.
C      L   CORRESPONDS TO THE NUMBER OF POINTS WHICH IS
C          ASSUMED TO BE ODD =2L+1=N  OR EVEN =2L=N
C      A0  IS THE CONSTANT TERM OF THE APPROXIMATION
C      A() ARE THE COEFF. OF THE COS TERMS
C      B() ARE THE COEFF. OF THE SIN TERMS
C
C      THE FIT IS =  A0/2 + SUMMATION(J=1,M) OF
C                    A(J)COS(J*X) + B(J)SIN(J*X)
C
C      LYLE B. SMITH
C      STANFORD COMPUTER SCIENCE DEPARTMENT
C      AUGUST 1967
C
C      J0  IS LOWER LIMIT ON J (USED .NE. 0 ON SUBSEQUENT ENTRIES)
C
       INTEGER N,M,L
C
       REAL Y(N),A0,A(M),B(M)
C
       INTEGER J,K
C
       REAL COSXJ,SINXJ,COSX,SINX,U1,U2,U3,CONST,PI,T1
       LOGICAL EVEN
       INTEGER ULIM
C
C
       L = N/2
  100  EVEN = .FALSE.
       IF((N/2)*2 .EQ. N) EVEN = .TRUE.
       IF(EVEN)       ULIM = 2*L
       IF(.NOT. EVEN) ULIM = 2*L+1
C
       IF( EVEN     ) CONST = 2.0 / (2.0*L)
       IF(.NOT. EVEN) CONST = 2.0 / (2.0*L + 1.0)
C
```

```
      PI = 3.14159265358979
      COSX = COS(PI*CONST)
      SINX = SIN(PI*CONST)
C
  105 J = J0 - 1
C
  110 J = J + 1
      IF(J .GT. M) GO TO 180
  115 IF(J)125,120,125
  120     COSXJ = 1.0
          SINXJ = 0.0
          GO TO 140
  125 IF(J-1)135,130,135
  130     COSXJ = COSX
          SINXJ = SINX
          GO TO 140
  135 T1 = COSXJ
      COSXJ = T1*COSX - SINXJ*SINX
      SINXJ = SINXJ*COSX +  T1  *SINX
C
  140 U3 = 0.0
      U2 = 0.0
C
  145 K = ULIM
C
  150 U1= Y(K) + 2.0*COSXJ*U2 - U3
      K = K-1
      IF(K-1)160,160,155
  155     U3 = U2
          U2 = U1
          GO TO 150
  160 IF(J)170,165,170
  165     A0 = CONST*(Y(1) + U1*COSXJ - U2)
          GO TO 175
  170 B(J) = CONST*U1*SINXJ
      A(J) = CONST*( Y(1) + U1*COSXJ - U2)
  175 GO TO 110
  180 RETURN
      END
```

```
      SUBROUTINE ERRAB(Y,A0,A,B,M0,M,N,L,DA0,DA,DB,SIG,BEST)
C
C     THIS ROUTINE COMPUTES THE PROBABLE ERROR IN THE FIT COEFFICIENTS.
C     ALSO THE STATISTIC SIGMA WHICH CAN BE USED TO DETERMINE 'BEST'FIT.
C
C     LYLE B SMITH
C     STANFORD COMPUTER SCIENCE DEPARTMENT
C     AUGUST 1967
C
C     Y() = THE DATA  (ASSUMED ON EQUALL SPACED ABCISSAE)
C     A0,A(),B() = THE FIT COEFFICIENTS
C     M0  = 1 OR 1 ABOVE M FROM LAST CALL OF ERRAB
C     M   = DEGREE (MAX)
C     N   = NUMBER OF DATA POINTS = 2L+1 (OR 2L)
C     L   = SEE N
C     DA0(),DA(),DB()=THE PROBABLE ERRORS
C     SIG() = THE STATISTIC FOR DECIDING'BEST'FIT
C     BEST = THE DEGREE OF 'BEST' FIT
C
      INTEGER M0,M,N,L,BEST
      REAL Y(N),A0,A(M),B(M),DA0(M),DA(M),DB(M),SIG(M)
C
      REAL ROOT2,DMSQ,SIGT
      LOGICAL LASTOK
      LOGICAL EVEN
C
      INTEGER I
C
      L = N/2
      EVEN = .FALSE.
      IF((N/2)*2 .EQ. N) EVEN = .TRUE.
  100 ROOT2 = SQRT(2.0)
      IF(M0 .GT. 1) GO TO 120
      DMSQ = 0.0
  105 DO 110 I=1,N
          DMSQ = DMSQ + Y(I)*Y(I)
  110 CONTINUE
C
C
      DMSQ = DMSQ - N*( 0.5*A0*A0 )  /2.0
C
  120 I= M0
  125 DMSQ = DMSQ - N*( A(I)*A(I) + B(I)*B(I) )  /2.0
C
  130 IF(EVEN)       SIGT = (0.5*DMSQ) / (L-I -0.5)
      IF(.NOT.EVEN) SIGT = (0.5*DMSQ) / (L-I)
C
      SIG(I) = SIGT
C
```

```
      IF(SIGT .LE. 0.0) SIGT = 0.0
  135 DA(I) = 0.6745*SQRT(2.0*SIGT / N )
      DAO(I)= ROOT2 * DA(I)
      DB(I) = DA(I)
  140 I = I+1
      IF(I .LE. M) GO TO 125
C
C     NOW ERRORS AND SIG HAVE BEEN CALCULATED
C     SCAN SIG() FOR BEST FIT
C
  150 LASTOK = .FALSE.
      IF(M-1)155,155,160
  155     BEST = 1
          GO TO 200
  160 DO 195 I=2,M
          IF(ABS(SIG(I)-SIG(I-1))
     *      -ABS(SIG(I))*0.01  )165,165,175
  165     IF(LASTOK) GO TO 180
          LASTOK = .TRUE.
  170     GO TO 195
  175     IF(SIG(I)-SIG(I-1))190,190,180
C         SIG HAS INCREASED SO SAVE I-1
  180     BEST = I-1
  185     GO TO 200
  190     LASTOK = .FALSE.
  195 CONTINUE
C
      COMING THRU THE LOOP SAYS M IS BEST BUT COULD BE BETTER
C
      BEST = M
  200 CONTINUE
C
C     WRITE RESULTS TO DATE
C
  205 WRITE(6,900)
  900 FORMAT(14HODEGREE  SIGMA,10X,6HP.E.A0,10X,6HP.E.A ,10X,5HP.E.B )
      DO 210 I=1,M
          WRITE(6,901) I,SIG(I),DAO(I),DA(I),DB(I)
  901     FORMAT( I4,4E16.6)
  210 CONTINUE
C
  215 IF(BEST.EQ.M)GO TO 225
  220 WRITE(6,902) BEST
  902 FORMAT(20HOFROM IN HERE DEGREE,I4,15HLOOKS BEST.)
      GO TO 230
  225 WRITE(6,903) BEST
  903 FORMAT(22HOBEST DEGREE SO FAR IS,I4,34H,A HIGHER WOULD NO DOUBT BE
     *BETTER.)
  230 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FEVAL(A0,A,B,N,L,M,RESIDS,YF,YF2,N2)
C
C     THIS SUBROUTINE EVALUATES THE FOURIER APPROXIMATION
C     EITHER AT THE GIVEN DATA POINTS
C     OR AT ANOTHER SET OF EQUALLY SPACED POINTS (SAME INTERVAL).
C
C     A0,A(),B()= THE FIT COEFFICIENTS
C     N,L = THE NUMBER OF DATA POINTS ( N= 2L+1)
C     CHANGES FOR FEVAL TO INCLUDE  N EVEN.
C     NONE SEEM NECESSARY
C     M = THE DEGREE (NUMBER OF COS (SIN) TERMS )
C     RESIDS = TRUE FOR COMPUTING RESIDUALS
C            = FALSE FOR OTHER POINTS
C     YF() = THE RESIDUALS
C     YF2() = THE OTHER POINTS
C     N2 = NUMBER OF OTHER POINTS ( N2 ODD )
C
C     LYLE B. SMITH
C     STANFORD COMPUTER SCIENCE DEPARTMENT
C     AUGUST 1967
C
      INTEGER N,L,M,N2
      REAL A0,A(M),B(M),YF(N),YF2(N2)
      LOGICAL RESIDS
C
C
      INTEGER I,J,LIM

      REAL COSXJ,SINXJ,COSX,SINX,T1,PI,ARG,Y,COSIX,SINIX


      L = N/2
  100 PI = 3.14159265358979
  105 IF(RESIDS) GO TO 110
      LIM = N2
      ARG = (N-1)*PI*2.0
      ARG = ARG/(N*(N2-1))
      GO TO 115
  110 LIM = N
      ARG =(PI*2.0)/ N
C
  115 COSX = COS(ARG)
      SINX = SIN(ARG)
C
```

```
120 DO 145 I=1,LIM
    IF(I .GT. 1) GO TO 121
    COSXJ = 1.0
    SINXJ = 0.0
    COSIX = 1.0
    SINIX = 0.0
    GO TO 123
121 CONTINUE
        T1 = COSIX
        COSIX = T1 * COSX - SINIX * SINX
        SINIX = SINIX * COSX + T1 * SINX
122     COSXJ = COSIX
        SINXJ = SINIX
123     Y = 0.5*A0
125     DO 130 J=1,M
            Y = Y + A(J)*COSXJ + B(J)*SINXJ
            T1= COSXJ
            COSXJ = T1*COSIX- SINXJ*SINIX
            SINXJ = SINXJ*COSIX+ T1    *SINIX
130     CONTINUE
135     IF(RESIDS) GO TO 140
        YF2(I) = Y
        GO TO 145
140     YF(I) = Y
145 CONTINUE
    RETURN
    END
```

APPENDIX C

PEG — PROGRAM ORGANIZATION

In this appendix a listing of the driver program for the PEG system is given. With this listing and an awareness of what variables are stored in COMMON, it is easy to substitute alternative methods for computing the least squares fits for the various built-in functions. It is also easy to see how additional functions can be added to the system. In the following discussion we will point out how new minimization methods could be implemented. The modular structure of the PEG systems facilitates the addition or substitution of new code.

1. Overall Organization

The flow of the program corresponds to the flowchart shown in Fig. 3.1. The driver program (see listing) consists of calls on various subroutines imbedded in logic that controls the program as indicated by the flowchart.

Overlay of various routines has been used extensively in order to make the program fit into the available region of memory. On the IBM 360/91 at SLAC 150 K bytes of memory have been made available to interactive programs such as PEG which run with a high priority but require only occasional bursts of CPU usage. The overlay structure has been designed to try to have all the code needed for fitting with a particular function in core at the same time. The routines that generate the various interactive displays for data manipulation and decision making are overlayed in the same region. Thus each new display requires only one overlay. The driver program, the CRT handling routines and certain other frequently used routines are held in the root segment and never overlayed.

2. Substitution of Methods

The modular design of PEG allows substitution of numerical methods in a straightforward manner. For polynomial fitting a subroutine, called FIT1,calculates the least squares fit by the method described by Forsythe [1957]. Substituting a different subroutine with the same name and parameter list and with appropriate attention to the use of COMMON would allow the use of a different algorithm. The displays which show the parameters of the calculated fit could be changed to correspond to the parameters of the new method. Changes would also have to be made to the subroutine which evaluates the computed fit for plotting purposes.

For the Fourier approximation, a change of method would be handled similarly to the polynomial case. The routine called by the driver program in this case is named FIT3. Substitution of a new subroutine with the same name and parameter list, again with appropriate attention to COMMON usage would allow a different algorithm to be utilized. Subroutines for function evaluation and display of parameters would also be required.

The spline function fitting is handled somewhat differently from polynomials and Fourier approximation since it is either a linear or non-linear problem depending on whether the joints are allowed to vary. To implement a new method the function named SUMSQ would be changed.

SUMSQ computes the least squares fit for a set of fixed joints (the linear problem) and computes the sum of squares of the residuals for that fit. For the variable joint case a subroutine named DIRECT is called with SUMSQ as a parameter. DIRECT employs the method of "direct search" (see Chapter V) to minimize the sum of squares given by SUMSQ as the joint positions are varied. Thus substitution of new routines for SUMSQ, DIRECT and the associated

routines for function evaluation could cause completely different methods to be used for spline function fitting.

Fitting by user defined functions is accomplished by minimization of the sum of squares of residuals as the parameters to the function are varied. The minimization is carried out by using the routine named DIRECT or by use of the interactive minimizer (see Chapter V). As we mentioned in the case of spline fitting with variable joints, DIRECT could be replaced by any other minimization routine with appropriate attention to the parameter list and COMMON usage. DIRECT, as implemented in PEG, checks for interrupts and updates a display of the computed fit at the end of each iteration to enable interactive control of the process. Similar display and checking should be imbedded in any routine that is substituted for DIRECT.

3. Addition of Fitting Functions

The modularity of the PEG system simplifies the task of adding new functions to the repertoire of available functions. To accomplish such an addition the new function would be added as an option on the display allowing function choice, the integer variable CHOICE would have another legitimate value, and several computed GO TO statements would have another branch added for the new function. The new branches would include calls on a routine to calculate the least squares fit similar to FIT1 and FIT3 mentioned before, and a routine to display the computed fit with its associated coefficients, table of residuals, etc., similar to the subroutines DISP7, D2SP7 and D3SP7 which handle the three currently built-in functions.

4. Adding New Minimization Methods

User defined functions are handled by the subroutine called DRIVP1. In DRIVP1 the display asking for the user's decision as to which minimization

routine to use is called and that is followed by a call on DIRECT or the interactive minimizer (see Chapter V) depending on the choice just made. To implement additional methods they should just be added to the display asking which method to use. Then the code in DRIVP1 which branches to either of the existing methods could be made to include branches to calls on the new methods appropriately added. Codes for new methods could be overlayed in the same region that DIRECT is in since no two methods will be called for at the same time.

5. Driver Program for PEG

The following listing of the driver program for PEG is given to illustrate the flow of the program and to emphasize the modularity of the structure. This program contains primarily the logic controlling the flow of the program as well as some bookkeeping functions. Each call statement is followed by a comment card which briefly describe the function of the subroutine being called. It is hoped that the variable names contain sufficient mnemonic value to facilitate reading the program.

```
C      DRIVER FOR LEAST-SQUARES DATA FITTING
C      LYLE B. SMITH
C      COMPUTER SCIENCE DEPT.
C      STANFORD UNIVERSITY
C      NOV. 1967 -- DEC. 1968
C
C**** THE NAMED COMMON AREAS HOLD THE FOLLOWING INFORMATION
C      FCOMP----FIT COMPARISON INFO
C      INTGRS---SEVERAL INTGER VARIABLES USED THROUGHOUT
C      ORTHOG---ORTHOGONAL POLYNOMIAL FIT INFO
C      SPLNS----SPLINE FIT COEFFICIENTS(JOINTS AND POLY COEFF)
C      PERDIC---PERIODIC FIT COEFFICIENTS
C      DATA-----THE DATA BEING FIT
C      LOGIC----LOGICAL VARIABLES USED THROUGHOUT
C      TITL-----DATA AND FUNCTION TITLES
C      SPLIN----SPLINE COEFICIENTS (BEST,CURRENT,NEXT)
C**** MORESP---MORE VARIABLES USED IN SPLINE WORK
C
       COMMON/FCOMP/BESD(10),BSMSQ(10),BMAXR(10),BESNJ
       REAL BSMSQ,BMAXR
       INTEGER BESD,BESNJ
       LOGICAL SAVPF
C
       COMMON/INTGRS/CHOICE,WHICH,DEGPLT,DEG,BESTD,NJ,N,NJP1,DEGP1
       INTEGER CHOICE,WHICH,DEGPLT,DEG,BESTD,NJ,N,NJP1,DEGP1
       COMMON/ORTHOG/ALPHA(20),BETA(20),S(20),SIGMA(20),MAXRES(20),SO
       REAL ALPHA,BETA,S,SIGMA,MAXRES,SO
       COMMON/SPLNS/XJ(10),COEFF(11,21)
       REAL XJ,COEFF
       COMMON/PERDIC/A(20),B(20),DA0(20),DA(20),A0
       REAL A,B,DA0,DA,A0
       COMMON/DATA/X(100),Y(100),WTS(100)
       REAL X,Y,WTS
       COMMON/LOGIC/WEIGHT,BBACK,BRANCH
       LOGICAL      WEIGHT,BBACK,BRANCH
       INTEGER RETRN
C  *** TITLES FOR DATA SETS ***
C   IN DRIVER
       COMMON/TITL/DTITL(10),FTITL(5,15)
       INTEGER DTITL,FTITL
       REAL TEMPPP
C
C *** IN UDRIVE FOR FUNCTION TITLES ***
       INTEGER TITUFN( 7)/'USER DEFINED FUNCTION--UFUNC'/
       INTEGER TITUFI( 5)/'  1   2   3   4   5 '/
       REAL TITP
       INTEGER KBTITL(5)/'DATA FROM KEYBOARD'/
C
       INTEGER DMODE,I,NEWN,OLDN,LASTD
       LOGICAL HANDJ,FOK
       LOGICAL FOK2
C
       REAL X1(100),Y1(100),W1(100),RESIDS(100)
       REAL YFITS(100,20)
       INTEGER JO,DEGO
C
```

```
      INTEGER NBACK
      INTEGER NR
C
C
C     ADDITIONAL DECLARATIONS FOR SPLINE WORK
C
      COMMON/SPLIN/BSPCOE(11,21),BXJ(20),BSUMSQ,CSUMSQ,NEXTXJ(20),
     *             NSUMSQ,CSPCOE(11,21)
      REAL BSPCOE,BXJ,BSUMSQ,CSUMSQ,NEXTXJ,NSUMSQ,CSPCOE
      COMMON/MURESP/SAVEXJ,CONV,WRITE,D5P5L,J
      LOGICAL SAVEXJ,CONV,WRITE,D5P5L
      INTEGER J
C
      REAL STEP1,RHO,EPS,FNEW
      INTEGER IMAX
      REAL XJNEW(10)
C     IN DRIVERS (TO USE RDATA2)
      LOGICAL DATAOK
C
      INTEGER TIMER1,ITIME
C
      LOGICAL DONE1C
      INTEGER TNBACK
C
      DO 9090 I=1,5
          DO 9080 J=1,7
               FTITL(I,J) = TITUFN(J)
 9080     CONTINUE
          FTITL(I,8) = TITUFI(I)
 9090 CONTINUE
C
      WRITE(6,9091)
 9091 FORMAT(25H0CALLING UFUNCS WITH N=0     )
 9092 FORMAT(24H0CALLED UFUNCS WITH N=0     )
      TITP=UFUNC1(TITP,XJ,0)
      TITP=UFUNC2(TITP,XJ,0)
      TITP=UFUNC3(TITP,XJ,0)
      TITP=UFUNC4(TITP,XJ,0)
      TITP=UFUNC5(TITP,XJ,0)
      WRITE(6,9092)
C
      ITIME = TIMER1(0)
      CALL GASBUF(1,5120)
C**** GASBUF ASSIGNS BUFFER AREA FOR THIS 2250 JOB
C
  122 DATAOK =.TRUE.
C
C
      DO 96 I=1,11
      DO 96 J=1,21
       COEFF(I,J) = 0.0
       BSPCOE(I,J)= 0.0
       CSPCOE(I,J)= 0.0
   96 CONTINUE
      DO 106 I =1,10
      BESD(I) = 0
      BSMSQ(I) = 0.0
      BMAXR(I) = 0.0
  106 CONTINUE
      BESNJ = 0
```

```
      OLDN = 0
      DONE1C = .FALSE.
  102 NR = 0
      CALL DISP0
C**** DISP0 IS THE INTRODUCTORY DISPLAY
       RETRN = 1
      IF(BRANCH) GO TO 800
  100 CONTINUE
      WRITE(6,10)
   10 FORMAT(36H1LEAST SQUARES DATA-FITING WITH 2250 )
  105 CALL DISP1(CHOICE)
C**** DISP1 IS DISPLAY FOR FUNCTION CHOICE
      IF(BBACK) GO TO 102
       RETRN = 2
      IF(BRANCH) GO TO 800
C
  110 CALL DISP2(CHOICE,DMODE,BBACK)
C**** DISP2 ALLOWS DATA MODE CHOICE
      IF(BBACK) GO TO 105
       RETRN = 3
      IF(BRANCH) GO TO 800
  115 GO TO(120,130,140,150,160),DMODE
C     DATA FROM KEYBOARD
  120 CALL DISP3(CHOICE,DMODE,N,X,Y,WTS,WEIGHT,BBACK)
C**** DISP3 ALLOWS DATA ENTRY FROM KEYBOARD
      IF(BBACK) GO TO 110
      RETRN = 3
      IF(BRANCH) GO TO 800
      DO 121 I=1,5
          DTITL(I) = KBTITL(I)
  121 CONTINUE
      GO TO 400
C
C
C     DATA FROM CARDS
  130 CONTINUE
  133 IF(DATAOK) GO TO 134
      GO TO 110
  134 CALL RDATA2(N,X,Y,WTS,WEIGHT,DATAOK)
C**** RDATA2 READS DATA CARDS
      IF( N .GT. 0) GO TO 400
      GO TO 110
C
C     DATA IS RESIDUALS OF PREVIOUS FIT
  140 IF(NR .GT. 0) GO TO 144
      DO 143 I=1,10
  143     CALL GBELL
      WRITE(6,11)
   11 FORMAT(51HOTRIED TO FIT NONEXISTENT RESIDUALS---BACK TO DISP2 )
      GO TO 110
C
  144 DO 145 I=1,N
  145     Y(I) = Y(I) - YFITS(I,BESTD)
      WEIGHT=.FALSE.
      GO TO 400
C
C     DATA FROM ONLINE FILE (DISK,TAPE,DATA CELL, ETC.)
  150 CONTINUE
C     TO BE IMPLEMENTED LATER
C     ASSUME ON CARDS
```

```
      GO TO 130
C
C     DATA OF PREVIOUS FIT
  160 CONTINUE
      IF(OLDN.EQ.0) GO TO 110
      N= OLDN
      DO 162 I =1,N
       X(I) = X1(I)
       Y(I) = Y1(I)
       WTS(I) = W1(I)
  162 CONTINUE
      CALL DATOUT(N,X,Y,WTS,WEIGHT)
C**** DATOUT PRINTS OUT THE DATA
      GO TO 400
C     READY FOR DISP4---ALLOWING  CORRECTION AND/OR SUBSET SELECTION
  400 CONTINUE
 4444 CONTINUE
C     IN UDRIVE
      IF(CHOICE .NE. 8) CALL BSORT3(X,Y,WTS,N)
C**** BSORT3 SORTS THE DATA INTO INCREASING ABSCISSA VALUES
  402 GO TO 415
  405 CALL DISDTT
C**** DISDTT DISPLAYS DATA AND FUNCTION TITLES
      IF(BBACK) GO TO 110
 4059 CALL DISP4(N,X,Y,WTS ,WEIGHT,CHOICE,NEWN,BBACK)
C**** DISP4  ALLOWS DATA CORRECTION  AND/OR SUBSET SELECTION
      IF(BBACK) GO TO 405
       RETRN = 4
      IF(BRANCH) GO TO 800
      WRITE(6,12) N,NEWN
   12 FORMAT(20HOAFTER DISP4---OLDN= I5, 6H NEWN= I5)
      IF(NEWN.GE.2) GO TO 420
      DO 406 I=1,10
  406     CALL GBELL
C**** GBELL  ACTIVATES AUDIBLE SIGNAL ON 2250
      WRITE(6,13)
   13 FORMAT(50HONOT ENOUGH DATA LEFT---BACK TO DISP4 W/O CHANGING)
      GO TO 418
C     AT 415 WE SAVE DATA
  415 CONTINUE
  416 DO 417 I=1,N
          X1(I)= X(I)
          Y1(I)= Y(I)
          W1(I)= WTS(I)
  417 CONTINUE
      OLDN = N
      GO TO 405
C
  418 DO 419 I=1,N
          X(I)=X1(I)
          Y(I)=Y1(I)
  419     WTS(I)=W1(I)
      GO TO 405
C
C
C     AT 420 WE GO ON WITH CHANGED DATA
  420 CONTINUE
      N = NEWN
      NR = N
C     NOW READY FOR DISP5---ASKS FOR DEGREE (AND JOINTS IF SPLINES)
```

```
      CALL DATOUT(N,X,Y,WTS,WEIGHT)
C**** DATOUT PRINTS OUT THE DATA
      DEG = 0
      GO TO 500
C
C     DISP5
  500 CONTINUE
 7501 BBACK = .FALSE.
      CALL TRANSF(N)
C**** TRANSF ALLOWS TRANSFORMATIONS OF THE DATA
      IF ( BBACK ) GO TO 4059
       RETRN = 5
      IF(BRANCH) GO TO 800
 8501 CALL DELPTS
      IF(BBACK) GO TO 7501
       RETRN = 6
      IF(BRANCH) GO TO 800
      IF(CHOICE .NE. 8) CALL BSORT3(X,Y,WTS,N)
C**** BSORT3 SORTS THE DATA
C
C
 9501 LASTD = DEG
C
      DEG = 0
      LASTD = 0
C
      DEG0 = DEG+1
      J0 = DEG0
      IF(DEG.EQ.0) J0=0
      IF(CHOICE .NE. 2) GO TO 505
      T1 = (X(N) + X(1))/2.0
      T2 = 2.0/(X(N) - X(1))
      DO 502 I =1,N
  502    X(I) = (X(I) - T1)*T2
  505 CALL DISP5(N,X,Y,WTS,WEIGHT,CHOICE,BBACK,DEG,NJ,XJ,HANDJ)
C**** DISP5  ALLOWS PARAMETER (DEGREE, INITIAL JOINTS ETC) ENTRY
      IF(BBACK) GO TO 8501
       RETRN = 7
      IF(BRANCH) GO TO 800
C
  506 CONTINUE
      CALL D5P2(CHOICE)
C**** D5P2   DISPLAY MESSAGE 'FIT BEING COMPUTED'
C     NOW COMPUTE FITS
      GO TO(510,520,530,540,550,560,570,580,590),CHOICE
C     ORTHOGONAL POLY. FIT
  510 CONTINUE
      CALL FIT1(X,Y,WTS,N,DEG,YFITS,MAXRES,BESTD,WEIGHT,LASTD,FOK,
     *          ALPHA,BETA,S,S0,SIGMA)
C**** FIT1   COMPUTES ORTHOGONAL POLYNOMIAL FIT
C
  511 IF( FOK) GO TO 515
      WRITE(6,14) DEG
   14 FORMAT(36HCFIT1--ORTHOG.POLYS--NOT OK FOR DEG= I5)
      GO TO 500
C
  515 GO TO 600
C
C     SPLINE FIT
  520 D5P5L =.FALSE.
```

```
      BESTD = DEG
      IF(.NOT. HANDJ) GO TO 525
  521 CSUMSQ = SUMSQ(XJ,NJ)
      BSUMSQ = CSUMSQ
      DO 1521 I=1,NJ
 1521     BXJ(I) = XJ(I)
C
      DO 2521 I=1,11
      DO 2521 J=1,21
 2521     BSPCOE(I,J)=COEFF(I,J)
C
      CALL D5P5(BXJ,BSPCOE,NEXTXJ,SAVEXJ,BSUMSQ,CSUMSQ,D5P5L,CONV,BBACK)
C**** D5P5   DISPLAYS PREVIOUS BEST AND CURRENT SPLINE FITS
      IF(BBACK) GO TO 9501
C
      IF(SAVEXJ) GO TO 522
      CSUMSQ = SUMSQ(NEXTXJ,NJ)
      DO 3521 I=1,NJ
 3521     XJ(I)= NEXTXJ(I)
C
 4521 CALL D5P5(BXJ,BSPCOE,NEXTXJ,SAVEXJ,BSUMSQ,CSUMSQ,D5P5L,CONV,BBACK)
C**** D5P5   DISPLAYS PREVIOUS BEST AND CURRENT SPLINE FITS
      IF(BBACK) GO TO 9501
      IF(BSUMSQ .GT. CSUMSQ) GO TO 6521
      IF(SAVEXJ) GO TO 522
C     BEST IS STILL BEST
C     COMPUTE NEW CURRENT AND GO DISPLAY AND READ
C
      CSUMSQ = SUMSQ(NEXTXJ,NJ)
      DO 5521 I=1,NJ
 5521     XJ(I)= NEXTXJ(I)
C
      GO TO 4521
C
 6521 CONTINUE
C     HERE WE HAVE A NEW BEST
      BSUMSQ = CSUMSQ
      DO 7521 I=1,NJ
          BXJ(I) = XJ(I)
 7521     XJ(I) = NEXTXJ(I)
C
      DO 8521 I=1,11
      DO 8521 J=1,21
 8521     BSPCOE(I,J)= COEFF(I,J)
C
C     NOW NEW BEST IS SAVED SO
C     COMPUTE NEW CURRENT AND GO DISPLAY AND READ
      CSUMSQ = SUMSQ(NEXTXJ,NJ)
      IF(SAVEXJ) GO TO 522
      GO TO 4521
C
C     HERE WE QUIT ADJUSTING THE JOINTS AND GO ON TO DISP6
  522 DO 1522 I=1,NJ
 1522     XJ(I)= BXJ(I)
      DO 2522 I=1,11
      DO 2522 J=1,21
 2522     COEFF(I,J) = BSPCOE(I,J)
      GO TO 600
C
C
```

```
C       NOW WE HANDLE JOINT ADJUSTMENT BY DIRECT
  525 CONTINUE
      D5P5L =.TRUE.
C
      CSUMSQ = SUMSQ(XJ,NJ)
      BSUMSQ = CSUMSQ
      DO 1525 I=1,NJ
 1525     BXJ(I)= XJ(I)
      DO 2525 I=1,11
      DO 2525 J=1,21
 2525     BSPCOE(I,J)=COEFF(I,J)
C
 3525 CONTINUE
C
C       SET PARAMETERS FOR DIRECT
      STEP1 =(X(N)-X(1))/ 2.0
C
C
      STEP1 = 0.2
C
      RHO = 0.1
      EPS = 0.001
      IMAX = 25
      WRITE =.TRUE.
      DO 4525 I=1,NJ
 4525     XJNEW(I) = XJ(I)
C
      CALL CHUZDS(STEP1,RHO,EPS,IMAX)
C *** CHUZDS ALLOWS ON-LINE CHANGE OF DIRECT SEARCH PARAMETERS
C
      CALL DIRECT(NJ,XJNEW,FNEW,STEP1,RHO,EPS,IMAX,WRITE,CONV,SUMSQ)
C**** DIRECT MINIMIZES THE FUNCTION SUMSQ W.R.T. THE JOINTS
C
      DO 5525 I=1,NJ
 5525     XJ(I) = XJNEW(I)
      CSUMSQ = FNEW
C
      CALL D5P5(BXJ,BSPCOE,NEXTXJ,SAVEXJ,BSUMSQ,CSUMSQ,D5P5L,CONV,BBACK)
C**** D5P5   DISPLAYS PREVIOUS BEST AND CURRENT SPLINE FITS
      IF(BBACK) GO TO 9501
C
      IF(.NOT. SAVEXJ) GO TO 526
C       DIRECT RESULTS SAVED IN XJ AND COEFF SO GO ON TO DISP6
      BSUMSQ = CSUMSQ
      GO TO 600
C
  526 DO 1526 I=1,11
      DO 1526 J=1,21
 1526     CSPCOE(I,J) = COEFF(I,J)
C
      NSUMSQ = SUMSQ(NEXTXJ,NJ)
      IF(NSUMSQ .GE. CSUMSQ) GO TO 527
      BSUMSQ = NSUMSQ
      DO 2526 I=1,NJ
          XJ( I) = NEXTXJ(I)
  252     BXJ(I) = NEXTXJ(I)
      DO 3526 I=1,11
      DO 3526 J=1,21
 3526     BSPCOE(I,J)= COEFF(I,J)
C
```

```
      GO TO 3525
C
C     HERE DIRECT RESULTS ARE BEST
  527 BSUMSQ = FNEW
      DO 1527 I=1,NJ
          XJ( I) = NEXTXJ(I)
 1527     BXJ(I) = XJNEW(I)
C
      DO 2527 I=1,11
      DO 2527 J=1,21
 2527     BSPCOE(I,J)=CSPCOE(I,J)
C
      GO TO 3525
C     THIS ENDS THE'JOINTS BY DIRECT'
C
C     PERIODIC FIT
  530 CONTINUE
      CALL FIT3(J0,DEG0)
      GO TO 600
C
C     RATIONAL FIT
  540 CONTINUE
C     CALL FIT4(...)
      GO TO 600
C
C
C     USER FUNCTION FITS
  550 CONTINUE
  560 CONTINUE
  570 CONTINUE
  580 CONTINUE
  590 CONTINUE
C
      CALL DRIVP1(HANDJ)
C**** DRIVP1 HANDLES USER DEFINED FUNCTIONS SIMILARLY TO
C****           THE ABOVE SPLINE TREATMENT
C
      IF(BBACK) GO TO 9501
C
      GO TO 600
C
C     NOW DISP6---CHOOSE DISPLAY MODE
  600 CONTINUE
      DONE1C = .TRUE.
C
  602 CALL DISP6(WHICH,DEG,DEGPLT,CHOICE,BBACK)
C**** DISP6  ALLOWS CHOICE OF DISPLAY MODE
C
  604 IF(BBACK) GO TO 9501
       RETRN = 8
      IF(BRANCH) GO TO 800
C
      GO TO 700
C
C
C     NOW DISP7---DISPLAY ACCORDING TO WHICH
  700 CONTINUE
  701 DEGP1 = DEG + 1
      NJP1 = NJ + 1
C
```

```
      GO TO(1001,1002,1003,1014,1005,1006,1007,1008,1009),CHOICE
 1001 CALL DISP7(YFITS)
C**** DISP7  DISPLAY ORTHOGONAL POLYNOMIAL FIT
      GO TO 1004
 1002 CALL D2SP7(YFITS)
C**** D2SP7  DISPLAY SPLINE FIT
      GO TO 1004
 1003 CALL D3SP7(YFITS)
C**** D3SP7  DISPLAY FOURIER APPROXIMATION FIT
      GO TO 1004
 1014 CONTINUE
C     RATIONALS NOT IMPLEMENTED
      GO TO 1004
 1005 CONTINUE
 1006 CONTINUE
 1007 CONTINUE
 1008 CONTINUE
 1009 CONTINUE
      CALL DUSP7(YFITS)
C**** DUSP7  DISPLAY USER DEFINED FUNCTION FIT
      GO TO 1004
C
 1004 CONTINUE
C
  705 IF(BBACK) GO TO 600
C
      GO TO 800
C
C
C     NOW DISP8---CHOOSE WHETHER TO BACKUP
C                                 OR START NEW FIT
C                                 OR CANCEL THE JOB.
  800 CONTINUE
C
      SAVPF = .FALSE.
  802 CALL DISP8(CHOICE,NBACK,SAVPF)
C**** DISP8  BRANCHING DISPLAY
      IF(.NOT. SAVPF) GO TO 805
      BESD(CHOICE) = DEGPLT
      IF(CHOICE .EQ. 2)
     *BESNJ = NJ
      IF(CHOICE .EQ. 1)
     *BMAXR(1) = MAXRES(DEGPLT)
      IF(CHOICE .EQ. 2)
     *BSMSQ(2) = BSUMSQ
C
  805 IF(NBACK .EQ. 4) GO TO 810
      IF(NBACK .GE. 10) GO TO 810
      IF(DONE1C) GO TO 810
      TNBACK = NBACK
      IF(NBACK .GT. 4) TNBACK = NBACK - 1
      IF( TNBACK .GT. RETRN)  GO TO 874
  810 GO TO(102,105,110,876,4444,7501,8501,9501,600,872,870,874,900),
     $ NBACK
C
  870 CALL DISP9
C**** DISP9  FIT COMPARISON DISPLAY
      GO TO 800
  876 CALL DISDTT
C**** DISDTT DISPLAY DATA AND FUNCTION TITLES
```

```
      GO TO 80
  872 CALL TUTORB
C**** TUTORB ALLOWS CALLING OF INDIVIDUAL TUTORIAL DISPLAYS
      GO TO 80
  874 GO TO(102,105,110,4444,7501,8501,9501,600),RETRN
C
C     END OF JOB
  900 CONTINUE
      WRITE(6,15)
   15 FORMAT(16H0THIS IS THE END)
      DO 905 I=1,10
  905     CALL GBELL
C
      CALL GHALT(0)
C**** GHALT  STOPS 2250 DISPLAY (HALTS BUFFER EXECUTION)
C
      ITIME  = TIMER1(1)
      WRITE(6,8888) ITIME
 8888 FORMAT(23H0TOTAL TIME(MILLISEC)=  I10 )
      CALL GCLOSE
C**** GCLOSE CLOSES OUT 2250 BUFFER ASSIGNMENT
      STOP
C     END DRIVER
      END
```

APPENDIX D

A NEW USER TRIES PEG

The following is a chronological account of a session with PEG by a new user. Pictures of the displays referred to are shown in Chapter III. The user arrived at the IBM 2250 console with a plot on graph paper of a 17 point data set and was asked to use PEG to obtain a fit to the data. The author sat behind him to take notes and to answer questions if necessary. Paraphased remarks by the user are labeled "U" and remarks by the author are labeled "A". The numbers given to identify various displays refer to the figure numbers in Chapter III. Comments on suggested improvements or changes to the PEG system are labeled "C".

Time Action

1:47 User sits down and begins to read introductory display (3.2).

1:48 Picks up lightpen and continues to read.

U: "What does *BRANCH OUT do?"

A: "Transfers to a branching display which includes a branch to some tutorial displays."

C: Some explantion of the *BRANCH OUT option could be included in display (3.2).

1:49 Selects *BRANCH OUT.

Selects *TUTORIAL DISPLAYS from branching display (3.28).

An arrow marks his choice but he puzzles.

A: "You need to select *CONT with lightpen after making a choice."

C: Some explanation of the need for *CONT after making a choice could be included in display (3.28) and perhaps other places as well.

Branches to Tutorial Displays (3.30)

Time Action

Selects *ON HOW TO ENTER NUMBERS FROM THE KEYBOARD.

Reads the tutorial display then returns to the introductory display.

1:51 Selects *CONT on (3.2) and obtains display to choose function (3.3).

Goes back and forth between (3.2) and (3.3) twice.

U: "I want to enter my data."

A: "The program expects a function choice first and then the entry of data."

C: Perhaps the function choice display (3.3) and the data mode choice display (3.4) should appear in reverse order. Either that or some explanation of their order could be displayed. One method of alleviating this problem would be to have the *CONT on each display include a brief message as to what the next display will be.

Selects Orthogonal Polynomials on display (3.3).

Selects *KEYBOARD on display (3.4).

Selects *NO on display (3.5) to indicate no weighting.

1:53 User puts lightpen down and beings to key in the data. After typing the first number he acts puzzled.

A: "Notice that 'END KEY AFTER EACH NUMBER' is specified on the CRT."

1:56 Seventeen data points are entered. The program displays the data tabularly with *FORWARD and *REVERSE options.

U: "What does *FORWARD and *REVERSE mean?"

A: "These allow the roll-by of tabular information past the viewing window on the CRT".

Time Action

Goes on to display (3.7) which allows point correction and/or subset selection. Examines it briefly and goes on to transformation display (3.8).

Chooses *SQRT transformation for X --- looks puzzled.

U: "Why didn't anything happen?"

A: "It expects a choice for both coordinates before actually plotting the transformed data."

C: More explanation could be added to this display (3.8). As currently implemented the word CHOOSE indicates to which coordinate the next choice of transformation will apply. Perhaps an explanation of this and a comment that after choosing transformations for both coordinates the transformed plot is available would be appropriate.

1:59 Exits from transformation display (3.8) and point deletion display (3.10) appears.

U: "What is this? I thought I was going to do the fitting now?"

A: "That is the program's fault. The next display is for fitting."

C: The transformation display has a comment "go on to fit transformed data." This comment should be altered to say "go on to the point deletion display." The point deletion display (3.10) could then include a comment that the fitting (parameter entry) display follows.

Goes on to display (3.11) for parameter (degree) entry.

Time Action

2:01 Enters 4 for degree followed by end key. The numbers is rejected by PEG.

U: "Even integers require decimal points?"

A: "Yes, that is how PEG is currently coded."

C: A more sophisticated input routine would make number entry more simple.

Enters 4.0 for degree followed by end key.

U: "How do I know it has accepted that?"

A: "When the cursor disappears it has accepted the number."

2:03 Goes on to display (3.18) for selecting display mode.

U: "What is this about entering a degree less than or equal to 4?"

A: "For orthogonal polynomials or Fourier approximations with equally spaced points the display of any degree fit less than or equal to that calculated is simplified since the coefficients are already calculated. If you don't specify a degree PEG defaults to the degree just calculated."

C: A comment explaining the default option could be added.

User chooses *PLOT DATA AND FIT WITH FIT DISPLAYED. The fit appears graphically.

U: "Fourth degree is not enough!"

The *FORWARD and *REVERSE options are used to examine the table of residuals.

U: "This 'forward and reverse' is really a paging system rather than a 'roll-by'."

Goes on to branching display (3.28) and selects *RETURN TO WHERE CAME FROM.

Time Action

U: "What happened? Oh."

Selects full scope plot.

U: "There we go. Beautiful!"

2:06 Goes back ot entry display (3.11).

Enters 5 for degree.

Selects full scope plot and examines the fit.

Goes back to degree entry display (3.11).

Enters 10 for degree.

Selects full scope plot and sits back to admire the fit (a tenth degree polynomial fits the data well).

Backs up twice to the point deletion display (3.10).

Tries to delete a point by pointing the lightpen at the point. Nothing happens.

A: "Try pointing the lightpen at the table of numbers."

C: The displayed instruction "LIGHTPEN ON NUMERIC DATA TO SELECT A POINT FOR DELETION" could be reworded or placed in a more obvious place on the display to clarify the operation of this display.

One point is deleted. Goes on to enter degree.

2:11 Enters 10 for degree.

Selects full scope plot.

U: "Can I get back the deleted point?"

A: "Yes. Go back to select data mode and choose *DATA OF PREVIOUS FIT."

Uses *BACK options to back up transformation display (3.8).

Can not use *BACK to get out of display (3.8).

Time Action

A: "That is the program's fault. You'll have to use the branching display to get to the data mode selection display."

C: The *BACK option should be fixed in the transformation display to restore the data to its original coordinate system if any transformation has been done or to back up to the data mode selection display if the data is in its original form.

Goes to data mode selection display (3.4) by way of the branching display.

Selects data of previous fit to get deleted point back and gets a plot of data.

U: "Good! There it is!"

2:14 Backs up to choose function (3.3).

Selects spline functions.

Goes on to spline parameter entry display (3.12).

U: "What is a good degree?"

A: "Two ... Three ..."

U: "Oh yes. I'll try 3."

U: "What does automatic joint adjustment mean?"

A: "The positions of the joints will be automatically varied to minimize the sum of squares. Also notice that the data has been scaled to be in the interval -1 to + 1 for the spline computation. This improves the numerical properties of the algorithm."

Time Action

U: "Does it want a data point for a joint?"

A: "No. Anything between -1 and +1."

User enters 3 joint values after selecting automatic joint adjustment.

C: Some explanation could be added to this display. Comments such as those just made by "A" above could be displayed in the same area that is used for joint entry. The comments would then disappear when the numerical values are to be entered.

2:17 The display allowing change of direct search parameters (3.16) is reached.

U: "Ok. I'm snowed."

A: "This just allows on-line alteration of the convergence criteria for direct search minimization. You can simply use the preset values."

C: This display (3.16) could be put one level down by having a simpler display which gives an option to call it if desired.

2:19 Selects *USE PRESET VALUES AND CONTINUE.

Iteration O appears similar to display (3.17).

A: "See the joint markers?"

Continues iterating until the display of spline fit allowing selection of new parameters and comparison of previous best and current fit (3.15) appears.

2:21 U: "What did they do to me?"

A: "This display allows comparison between the fit with the initial joints and the fit found by direct search."

C: This display (3.15) seems to confuse people. Perhaps the display of more explanation would help. Another approach might be to

Time Action

simply display the final fit and then give an explanation of this comparison display with an option to invoke it.

2:23 Backs up to the point deletion display and deletes a point (the third (point).

U: "That wasn't the point I wanted to delete. I forgot there is one point plotted right on the line."

After some manuvering (getting back to the data mode selection — choosing data of previous fit — going back to the point deletion display) the fourth point is deleted instead of the third.

C: The previous step could have been eliminated in two ways. Either a better recovery option on the point deletion display or by using a plotting routine that adjusts the origin and scale so all data points are inside the boundary grid lines of the plot.

2:28 Enters degree 3 and 3 joints for a spline fit. Selects automatic joint adjustment.

U: "But I already entered the initial joints."

A: "That was for the last fit. You must enter new values each time."

C: A more sophisticated method of saving user inputs for later reuse could be added to PEG. This could save the time of re-entry of the same numbers.

2:29 Initial joint values are keyed in.

Iteration 0.

Iteration 1.

U: "It didn't like my initial joints at all! That's right. A third degree polynomial can have an inflection."

Time Action

2:31 Terminates iteration.

U: "There we go!"

Goes on to select display mode (3.18).

2:33 Plots residuals with coefficients displayed.

U: "It's bad over there. It's also bad in the middle."

Backs up to select a new function (3.3).

Chooses rational functions which are not yet implemented, however, PEG allows him to continue.

C: This is wrong. PEG does display a message "RATIØNAL FUNCTIØNS NOT YET IMPLEMENTED, TRY ANOTHER" but it should not allow continuation from this point.

User continues until the fit is asked to be plotted. At that point it's clear that there are no rational functions.

Goes back to choose a data mode.

Selects *RESIDUALS OF PREVIOUS FIT. Nothing happens.

U: "Could I make a criticism?"

A: "Of course!"

U: "It should display a message instead of doing nothing."

A: "You are absolutely right."

U: "What happened to my residuals?"

A: "When you tried rational functions that became the previous fit and the residuals from the spline fit were lost."

C: A more sophisticated method of storing results for later examination or reuse would be helpful to PEG users. A message should be displayed as indicated by the above conversation as well.

Time Action

2:39 Goes back to choose a periodic function fit (Fourier approximation).

Goes on to parameter entry (3.11).

U: "What does degree mean?"

A: "A degree of two means there will be a constant term, two sine terms and two cosine terms."

C: Some explanation of "degree" could be added to this display (3.11) for Fourier approximations.

U: "This looks like a fourth degree. I'll try 4."

Goes on to display a full scope plot of the fit.

U: "Not too good. Now I want to get the residuals."

A: "Go back to choose data mode."

Goes back to choose data mode and selects the residuals of the previous fit.

Goes on to data display (3.6).

C: This display should say that the data is the residuals of the previous fit. As currently implemented it does not.

2:43 Enters degree 3 (still using periodic functions) to fit the residuals.

Goes on to display the fit.

U: "What is the maximum error?"

A: "If you go on to the branching display you can select *SAVE FIT JUST DISPLAYED and *EXAMINE FIT COMPARISON. This will display the sum of squares and the maximum residual of the fit just computed."

2:46 After examining the maximum residual, user goes back to choose data mode. He selects the data of previous fit hoping to retrieve his original data, however at this point it has been lost.

End of session.

In addition to the several displays that could benefit from additional explanatory material and the other changes suggested by the above session, there is one recurrent theme. Improved data handling and retrieval of previously available information would make PEG more useful. This would include more sophisticated access to data entered from cards, data entered from the keyboard (data points as well as parameter specification), and results generated by the program. In general, however, users of PEG other than the author, have felt comfortable with the system and say that it is indeed quite self-explanatory and easy to understand and use.

REFERENCES

Allison, R. W. [1968]. Private communication.

Anselone, P. M. and Laurent, P. J. [1968]. A general method for the construction of interpolating or smoothing spline-functions. Numerische Mathematic, Vol. 12, 66-82.

Apostol, T. M. [1957]. Mathematical Analysis, Addison-Wesley Publishing Co., Reading, Massachusetts.

Ball, G. H. and Hall, D. J. [1967]. PROMENADE, an on-line pattern recognition system. Stanford Research Institute Technical Report No. RADC-TR-67-310 (September).

Bell, M. and Pike, M. C. [1966]. Remark on Algorithm 178. Communications of the ACM, Vol. 9, No. 9, (September), 684.

Berztiss, A. T. [1964]. Least squares fitting of polynomials to irregularly spaced data. SIAM Review, Vol. 6, No. 3, (July), 203-227.

Björck, A. and Golub, G. H. [1968]. Iterative refinements of linear least squares solutions by Householder transformations. Report No. CS-83, Computer Science Department, Stanford University (January).

Bowman, S. and Lickhalter, R. A. [1968]. Graphical data management in a time-shared environment. 1968 Spring Joint Computer Conference, Thompson Book Company, Washington, D. C., 353-362.

Box, M. J. [1965]. A new method of constrained optimization and a comparison with other methods. The Computer Journal, Vol. 8, 42-52.

Businger, P. [1967]. An Algol procedure for computing the singular value decomposition. Report No. CS-73, Computer Science Department, Stanford University (July).

Businger, P. and Golub, G. H. [1965]. Linear least squares solutions by Householder transformations. Numerische Mathematik 7, 269-276.

Cameron, S. H., Ewing, D., and Liveright, M. [1967]. DIALØG: A conversational programming system with a graphical orientation. Communications of the ACM, Vol. 10, No. 6 (June), 349-357.

Carasso, C. and Laurent, P. J. 1968 . On the numerical construction and the practical use of interpolating spline-functions. Proceedings of the IFIP congress 1968, A6-A9.

Clapp, L. C. and Kain, R. Y. [1963]. A computer aid for symbolic mathematics. 1963 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 509-517.

Clenshaw, C. W. and Hayes, J. G. [1965]. Curve and surface fitting. J. Inst. Maths Applics, Vol. 1, 164-183.

Conn, R. W. and vonHoldt, R. E. [1965]. An online display for the study of approximating functions. Journal of the ACM, Vol. 12, No. 3 (July), 326-349

Culler, G. J. and Fried, B. D. [1963]. An on-line computing center for scientific problems. Thompson Ramo Wooldridge Computer Division Report (now Bunker-Ramo Corp.) Canoga Park, California.

Culler, G. J. and Fried, B. D. [1965]. The TRW two-station, on-line scientific computer: general description. Computer Augmentation of Human Reasoning, Spartan Books, Washington, D.C.

deBoor, C. and Rice, J. R. [1968a]. Least squares cubic spline approximation I-fixed knots. Report No. CSD-TR-20, Computer Sciences Department, Purdue University (April).

deBoor, C. and Rice, J. R. [1968b]. Least squares cubic spline approximation II-variable knots. Report No. CSD-TR-21, Computer Sciences Department, Purdue University (April).

deMaine, P.A.D. [1965]. The self-judgment method of curve fitting. Communications of the ACM, Vol. 8, No. 8, (August), 518-526.

DeVogelaere, R. [1968]. Remark on Algorithm 178. Communications of the ACM, Vol. 12, No. 7, (July), 498.

Dixon, W. J. [1967]. Use of displays with packaged statistical programs. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 481-484.

Dixon, W. J. and Kronmal, R. A. [1965]. The choice of origin and scale for graphs. Journal of the ACM, Vol. 12, No. 2, (April), 259-261.

Elkin, R. M. [1968]. Convergence theorems for Gauss-Seidel and other minimization algorithms. University of Maryland, Computer Science Center, College Park, Maryland (January).

Engelman, C. [1965]. MATHLAB: A program for on-line machine assistance in symbolic computations. 1965 Fall Joint Computer Conference, Part II, Spartan Books, Washington, D.C., 117-126.

Fletcher, R. [1965]. Function minimization without evaluating derivatives — a review. The Computer Journal, Vol. 8, 33-41.

Fletcher, R. [1968]. Generalized inverse methods for the best least squares solution of systems of non-linear equations. The Computer Journal, Vol. 10, No. 4, (February), 392-399.

Forsythe, G. E. [1957]. Generation and use of orthogonal polynomials for data-fitting with a digital computer. SIAM Journal, Vol. 5, No. 2, (June), 74-88.

Gear, C. W. [1966]. Numerical solution of ordinary differential equations at a remote terminal. Proceedings — 1966 ACM National Conference, Thompson Book Company, Washington, D.C., 43-49.

Georges, J. S. and Kinney, J. M. [1938]. Introductory Mathematical Analysis, The Macmillan Company, New York.

Goertzel, G. [1958]. An algorithm for the evaluation of finite trigonometric series. American Mathematical Monthly, Vol. 65, 34-35.

Golub, G. H. [1967]. Least squares, singular values and matrix approximations. Report No. CS-73, Computer Science Department, Stanford University, (July).

Golub, G. H. [1965]. Numerical methods for solving linear least squares problems. Numerische Mathematik 7, 206-216.

Goodenough, J. B. [1965]. A lightpen-controlled program for online data analysis. Communications of the ACM, Vol. 8, No. 2, (February), 130-134.

Greenberger, M., Jones, M. M., Morris, J. H. (Jr.), and Ness, D. N. [1965]. On-line computation and simulation: The OPS-3 system. The M.I.T. Press, Cambridge, Massachusetts.

Hooke, R. and Jeeves, T. A. [1961]. "Direct search" solution of numerical and statistical problems. Journal of the ACM, Vol. 8, No. 2, 212-229.

Howry, S. [1968]. Private communication.

Hughes, B. [1968]. Private communication.

Johnson, W. B. [1968]. Private communication.

Joyce, J. D. and Cianciolo, M. J. [1967]. Reactive displays: improving man-machine graphical communication. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 713-721.

Kaplow, R., Brackett, J., and Strong, S. [1966]. Man-machine communication in on-line mathematical analysis. 1966 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 465-477.

Kaplow, R. Strong, S., and Brackett, J. [1966a]. MAP, A system for on-line mathematical analysis. Report No. MAC-TR-24, Massachusetts Institute of Technology (January).

Kaupe, A. F., Jr. [1963]. Algorithm 178, direct search. Communications of the ACM, Vol. 6, No. 6, (June), 313.

Klerer, M. and May, J. [1964]. An experiment in a user-oriented computer system. Communications of the ACM, Vol. 7, No. 5 (May), 290-294.

Marchuk, G. I. and Yershov, A. P. [1965]. Man-machine interaction in solving a certain class of differential equations. Proceedings of the IFIP Congress 65, Part II, 550-551.

Mo, L. [1968]. Private communication.

Moore, D.W.G. and Erickson, M. J. [1966]. The display as a research tool. Proceedings of Australia Computer Conference. Canberra (May), 16/1/1-16/1/4.

Ostrowski, A. M. [1964]. Contributions of the theory of steepest descent I. MRC Report No. 615, Mathematics Research Center, Madison, Wisconsin.

Pitteway, M. L. V. [1967]. Algorithm for drawing ellipses or hyperbolae with a digital plotter. The Computer Journal, Vol. 10, No. 3 (November), 282-289.

Potthoff, R. E. [1968]. An analysis of the biomechanics of the human knee. Engineer Degree Thesis, Department of Mechanical Engineering, Stanford University (June).

Pyle, I. C. [1965]. Data input by question and answer. Communications of the ACM, Vol. 8, No. 4 (April), 223-226.

Ralston, A. [1965]. A First Course in Numerical Analysis, McGraw-Hill Book Co., New York.

Reinfelds, J., Flenker, L. A., and Seitz, R. N. [1966]. AMTRAN, a remote-terminal, conversational-mode computer system. Proceedings 1966 ACM National Conference, Thomspon Book Company, Washington, D. C., 469-477.

Rice, J. R. and Rosen, S. [1966]. NAPSS — a numerical analysis problem solving system. Proceedings — 1966 ACM National Conference, Thompson Book Company, Washington, D. C., 51-56.

Ruyle, A [1967]. Project TACT. Presented at the ACM Symposium on Interactive Systems for Experimental Applied Mathematics, August 26-28, 1967 (Proceedings in Press).

Ruyle, A., Brackett, J. W., and Kaplow, R. [1967]. The status of systems for on-line mathematical assistance. Proceedings — 1967 ACM National Conference, Thompson Book Company, Washington, D. C., 151-167.

Schlesinger, S. and Sashkin, L. [1967]. POSE: A language for posing problems to a computer. Communications of the ACM, Vol. 10, No. 5 (May), 279-285.

Schoenberg, I. J. and Whitney, A. [1953]. On Pólya frequency functions, III. The positivity of translation determinants with an application to the interpolation problem by spline curves. Trans. Amer. Math. Soc., 74, 246-259.

Shaw, J. C. [1964]. JOSS: A designer's view of an experimental on-line computing system. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D. C., 455-464.

Simonsen, R. H. and Anketell, D. L. [1966]. Mechanization of the curve fitting process: DATAN. Communications of the ACM, Vol. 9, No. 4 (April), 299-304

Smith, L. B. [1968]. Remark on Algorithm 178 (submitted for publication).

Smith, L. B. [1969]. A Survey of Interactive Graphical Systems for Mathematics. SLAC-PUB-540, Stanford Linear Accelerator Center, Stanford University, Stanford California.

Stowe, An. N., Wiesen, R. A., Untema, D. B., and Forgie, J. W. [1966]. The Lincoln Reckoner: An operation-oriented, on-line facility with distributed control. 1966 Spring Joint Computer Conference, Spartan Books, Washington, D.C., 433-444.

Traub, J. F. [1968]. Private communication.

Witte, B. F.W. and Holst, W. R. [1964]. Two new direct minimum search procedures for functions of several variables. 1964 Spring Joint Computer Conference, Spartan Books, Washington, D.C., 195-209.

Zangwill, W. I. [1967]. Minimizing a function without calculating derivatives. The Computer Journal, Vol. 10, No. 3 (November), 293-296.