SLAC-PUB-540
January 1969
(MISC.)

A SURVEY OF INTERACTIVE GRAPHICAL SYSTEMS FOR MATHEMATICS*

Lyle B. Smith

Computation Group
Stanford Linear Accelerator Center
Stanford University, Stanford, California

ABSTRACT

Existing and proposed systems for performing interactive mathematics are surveyed with special attention to those systems with graphical output. The systems are grouped as general purpose, special purpose and other systems of interest. The solution of a least squares data-fitting problem by the various general purpose systems is then compared to the solution of the same problem by PEG, a special purpose system written at the Stanford Linear Accelerator Center for interactive least squares data-fitting. A summary includes a discussion of many of the references that appear in the fairly comprehensive bibliography.

(Submitted to ACM Computing Surveys)

*Work supported by the U. S. Atomic Energy Commission.

TABLE OF CONTENTS

I. INTRODUCTION

Several authors have recently discussed the possibility of man-computer interaction and its feasibility, promise, and current efforts. In Licklider and Clark [1962] an examination of then current on-line capabilities including some graphical capability is followed by a discussion of basic problems to be solved in improving man-computer communication. The five short term problems they list are:

1) The development of adequate time-sharing systems.
2) To devise an electronic I/O surface for two-way communication.
3) The development of a programming system that allows on-line, real-time selection and shaping of information processing procedures.
4) The development of systems for storage and retrieval of the large amounts of data required to support several on-line users.
5) To solve the problem of human cooperation in the development of large programs and systems.

Some progress has been made in these areas since 1962. In answer to problem 1), for example, Schwartz et al. [1964] discuss a working time-sharing system and Anderson et al. [1968] discuss a proposed time-sharing system which will allow interactive graphical consoles as well as typewriter-like consoles. Problem 2) is answered by Davis and Ellis [1964] who describe the Rand Tablet, an electronic surface for communication with a computer. Also in answer to 2), Haring [1965] describes the Beam Pen, an input/output device for CRT display systems and Lewin [1965] discusses a magnetic device for computer graphic input.

The third problem has been answered, to some extent, by on-line text editors, and various other capabilities that are currently implemented. The problem of

storing and retrieving vast quantities of information in support of several simultaneously active user stations is partially solved by the currently available disk storage units, drums, bulk core, and large main memories, used together in an hierarchical manner. Problem 5) may not be so formidable as more sophisticated programming systems and languages are developed. For example, PEG, the interactive data-fitting system written in FORTRAN by this author alone was operational in somewhat less than one man-year of programming and design effort. In other words, if fewer people are necessary for the completion of a programming project, there is less chance of a problem in human cooperation.

Licklider and Clark [1962] also pose four long term problems whose solution will enhance man-computer partnerships. These four problems are:

1) Computer appreciation of natural written languages.

2) Computer recognition of spoken words.

3) The theory of algorithms – their discovery and simplification.

4) Heuristic programming.

The first of these problems has been studied in some detail. It is directly related to the problem of natural language translation by computer. However, no completely satisfactory solution has, as yet, been found. Some of the current efforts toward solving the second problem are described by Lee [1968] and Allen [1968]. One approach toward solving a related pattern recognition problem, automatic recognition of handwritten words, is discussed by Mermelstein and Eyden [1964]. The third and fourth of these problems are more abstract and thus it is more difficult to illustrate progress in these areas.

In a later paper, Licklider [1965], Licklider makes the statement, "It is now feasible – and practical, too – for a very creative man to think in direct

interaction with a computing machine." We shall show by our work and that of others that it is also feasible (and practical) for everyone who has a problem to solve – to interact with a computing machine while solving problems.

Some other papers of interest that discuss the idea of man-machine interaction are Yershov [1965], Uncapher [1965], and Whiteman [1966]. Much of the interest in interactive graphical systems has been in the area of computer aided design. Some articles of interest in this area are Mann [1965], Roos [1965], Chasen [1965], Jacks [1964], Cole et al. [1964], Hargreaves et al. [1964], Allen and Foote [1964], Krull and Foote [1964] and Newman [1966].

It is, I believe, an accepted fact that man-machine interaction both with graphical capability and with only textual input/output – is a useful, nay necessary, part of our scientific problem solving repertoire of today and tomorrow. To substantiate this, consider for example, that three years ago Shaw [1965], at the conclusion of some remarks about JOSS (see Shaw [1964]), says ". . . at least for small numerical problems, direct conversation with a computing system meets computing requirements that are not well satisfied by conventional services." Since 1965 we have seen even more successful man-machine interactive systems.

In a panel discussion of promising avenues for computer research, Uncapher [1965] points out that present systems for on-line computing and problem solving assistance, are not oriented toward the thousands of practicing engineers in the United States. They may be very useful to the computer specialist and the skilled programmer but the casual user is not helped. Recently there has been more effort to make the power of an on-line system available to the casual user as proposed by Uncapher [1965]. Some efforts in this area are included in the following discussion of various on-line systems. In particular, the interactive

data-fitting system outlined in Section 4 and described in detail in Smith [1969] is designed for the use of non-specialists in computer science.

II. WHAT ARE THE VARIOUS SYSTEMS?

First let us simply list several on-line systems which include mathematical and/or graphical capabilities. Then we will give a more detailed description of each system. An interesting survey paper is that by Ruyle et al. [1967], however, the paper is confined (by choice) to four systems – AMTRAN, the Culler-Fried System, the Lincoln Reckoner, and MAP. Some of the following remarks are based on the paper by Ruyle et al. [1967].

A. A List of Various Systems

Here we list some interactive graphical systems. 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.

General Purpose Systems

1) Culler-Fried at Santa Barbara, California. Culler-Fried at TRW Systems, Redondo Beach, California.
2) TOC at Aiken Computation Laboratory, Harvard University.
3) AMTRAN at NASA Marshall Space Flight Center in Huntsville, Alabama.
4) Lincoln Reckoner at Lincoln Laboratory of Massachusetts Institute of Technology.

5) OPS-3 at Massachusetts Institute of Technology.

6) MAP at Massachusetts Institute of Technology.

7) NAPSS at Purdue University.

8) POSE at Aerospace Corporation, San Bernadino, California.

9) JOSS at the RAND Corporation, Santa Monica, California.

Special Purpose Systems

1) STATPAC at Decision Sciences Laboratory, Bedford, Massachusetts

2) Marchuk and Yershov in USSR

3) Gear at University of Illinois

4) Dixon at University of California, Los Angeles, California

5) PEG at Stanford Linear Accelerator Center (SLAC), Stanford, California

Other Systems

1) DIALOG at IIT Research Institute, Chicago, Illinois. (An on-line algebraic language with graphics).

2) MATHLAB at Massachusetts Institute of Technology and the Mitre Corporation (on-line assistance in symbolic computations).

3) MAGIC PAPER at Bolt, Beranck and Newman Inc., Cambridge, Massachusetts (primarily for on-line symbolic mathematics).

4) DISPLAY at System Development Corporation, Santa Monica, California (primarily for graphical examination of stored data – allows excursions into TINT, an Algol-type interpretor).

5) Klerer-May system at Columbia University, Hudson Laboratories, Dobbs Ferry, New York.

6) Moore et al., at the University of Western Australia. (FORDESK an on-line system and an interactive polynomial fitting program.)

7) Hall and Ball at the Stanford Research Institute (SRI), Menlo Park, California. (On-line systems for statistical and data analysis.)

B. General Purpose Systems – Operation Oriented

The first five systems to be described are operation oriented as opposed to language oriented. This means that commands to the system are generated by involking operators with appropriate operands. Later we will discuss several language oriented systems.

1. The Culler-Fried Systems

The Culler-Fried system was first developed at Thompson Ramo Wooldridge, Canoga Park, California, beginning in 1961, see Culler and Fried [1963]. Since then a similar system, operating on the same RW-400 computer (AN/FSQ-27), has been implemented at the University of California at Santa Barbara. This system is used for teaching and research on the Santa Barbara campus as well as from remote terminals at UCLA and Harvard, see Winiecki [1966]. An expanded version of the original system has also been implemented at TRW Systems (formerly Space Technology Laboratories) in Redondo Beach, California on a Bunker-Ramo 340 computer. In 1968 the system has also been implemented on an IBM System 360/65 computer at the University of California at Santa Barbara. For more references see Culler and Fried [1965], Fried [1967], Culler and Huff [1962], and others referred to by these articles.

The Culler-Fried terminals consist of an array of pushbuttons (96 keys) and a five inch storage oscilloscope used for output. The TRW version also has a Calcomp plotter and an output typewriter shared among four terminals, but most Culler-Fried consoles have only the scope for output. The pushbuttons are used for entering instructions to the system. The 96 pushbuttons are arranged in 2 adjoining keyboards, each with 48 keys. One keyboard permits access to operators and the other permits access to operands. A user solves his problem by entering instructions and data through the keyboard and viewing his results, displayed numerically and/or graphically, on the oscilloscope.

Culler-Fried systems are operation-oriented in that they provide operators with pushbutton control and allow user definition of new operators (console programs). Some of the system supplied operators are of a lower level than some of the more recently developed systems such as MAP and NAPSS which provide complete solution algorithms (e.g., polynomial root finding) as operators. For example, some of the operators supplied by Culler-Fried systems are comparable to assembler language instructions and console programs are written at that level. On the other hand, other operators are on a higher level, for example, the forward difference operator and the running sum operator are among the operators intended to provide the basic tools of operator calculus.

There are two working registers, α and β, whose contents are affected by the various operations. For example, keying PLUS 5.0 will add 5.0 to the contents of β register and SQRT will take the square root of the number in the β register; in both cases the result is left in the β register.

To add to the limited number of pushbuttons available, levels are introduced in the Culler-Fried systems. By changing levels (a pushbutton accomplishes the level change), many keys take on different meanings. Each level is composed of operators of a particular mathematical character, for example, real numbers and matrices form different levels so that some operators work automatically on vectors, matrices, or functions depending on the level. The USER levels allow storage of console programs composed by users.

The Culler-Fried systems have been the inspiration for several other interactive on-line mathematical systems. AMTRAN, and the TOC system, for example, bear many similarities to the Culler-Fried on-line systems as does the Lincoln Reckoner.

For examples of applications of the Culler-Fried system see Culler and Huff [1962] and the chapter by B. D. Fried in Karplus [1967].

2. The TOC System

TOC (Tact On-Line Computer) was begun in 1966 at the Aiken Computation Laboratory of Harvard University; see Ruyle [1967]. TOC was developed by Project TACT (Technological Aids to Creative Thought) and is based on two earlier stages of development. The first stage, TOCS, an effort to duplicate the Culler-Fried system under CTSS at Project MAC, was started in May 1965. This initial effort differed from the Culler-Fried system in the following ways:

1) an improved display facility
2) inclusion of statistical operators
3) ability to enter numeric data directly into vectors
4) ability to enter expressions

The second version of TOCS (Version II) contained even further departures from the Culler-Fried system. The levels which allow multiple use of the push-buttons were dropped by providing for arbitrary names for console programs. The ability to designate which variables are local to a console program, and to designate variables as argument variables (parameters) was added. Edit facilities were also made available for modification of console programs. Other improvements were the ability to give multi-character names to data entities, the ability to redefine system operators, the ability to use vectors of unlimited length (to within memory size), and error messages.

TOC, the most recent system developed by project TACT uses three basic features borrowed from the Culler-Fried Systems.

1) The use of pushbuttons to invoke operators.

2) Operators which work automatically on entities larger than scalars.

3) A repertoire of operators providing the basic tools of operator calculus.

Two engineering features which were also borrowed are:

1) The use of storage oscilloscopes for console output.

2) The ability to draw on the output scope from the input keyboard. Drawings are composed using small directed straight line segments, each associated with a button.

The two versions of TOCS which preceded TOC provided many more ideas for the development of TOC, most of these ideas are listed above as departures from the Culler-Fried System. Each TOC console has 65 operator keys plus a full typewriter keyboard. The TOC system was expected to be in operation by the fall of 1967 on an IBM 360/50.

3. The AMTRAN System

AMTRAN (Automatic Mathematical TRANslation System) was developed at NASA's Marshall Space Flight Center in Huntsville, Alabama and has been available to users since early 1966; see Wood et al. [1966], Reinfelds et al. [1966] and Ruyle et al. [1967]. AMTRAN was inspired by the Culler-Fried system and retains many similarities, especially the use of function buttons. The basic goals of AMTRAN as stated by Reinfelds et al. [1966] are:

1) To use the natural language of mathematics as a programming language without any arbitrary restriction whatsoever.

2) To obtain immediate graphical output of intermediate and final results.

3) To retain a hard copy of useful results and programs.

4) To retain copies of programs in an easily reusable form.

5) To retain utmost flexibility in the system so as to allow its use from the level of existing programming language up to the level of advanced calculus.

The AMTRAN system employs remote terminals consisting of a large keyboard (with 224 pushbuttons), two 5-inch cathode ray storage oscilloscopes for graphical and alphanumeric display, and a typewriter to provide a permanent record of displayed information when desired. The AMTRAN language is multi-level in that it can be used by a systems programmer or an applied mathematician or at any intermediate level. Both an on-line conversational mode and an off-line batch mode (or a combination) are available to the users.

The language includes Algol/60 programming capabilities as well as the pushbutton operators. Among the operators which are implemented to facilitate the expression of mathematical calculations in classical notation are $\int$, $\frac{d}{dx}$, MINIMAX, etc. An array of unlabeled buttons is provided for storing a user defined sequence of button pushes (a user defined console program). Automatic array arithmetic is provided.

4. The Lincoln Reckoner

The Lincoln Reckoner system has been in routine use by the staff of the Lincoln Laboratory of the Massachusetts Institute of Technology since the spring of 1966; see Stowe et al. [1966], Wiesen et al. [1967] and Ruyle et al. [1967]. The Reckoner system, designed to operate within the APEX time-sharing system for the TX-2 computer, is for on-line use in scientific and engineering research. The system in its present state does not provide all the services to an engineer or scientist that might be envisioned, instead the development has been concentrated in the area of numerical computations on arrays of data. As of the summer of 1966 there were 64 operations available in the Reckoner public library. These include the following:

1) Basic arithmetic on arrays.

2) Other arithmetic on arrays.

3) Data shuffling in two-dimensional arrays.

4) Matrix operations.

5) Signal analysis.

6) Input and Output.

7) Miscellaneous.

The Reckoner is primarily a facility for making use of routines and not for writing them. The aim is to provide a system whereby a scientist can feel his way through the reduction of data from a laboratory experiment without previous programming experience or training. However, the capability to construct ones own routines is available in the form of a small algebraic compiler called Junior which can be utilized on-line.

The Lincoln Reckoner is an "operation-oriented" system as are AMTRAN, MAP, OPS-3, and the Culler-Fried system. That is, the system tries to provide the operations (add, divide, integrate, differentiate, plot, etc.) that a user will need to solve his problem. The user then combines operations and operands so as to solve a particular problem. For example, suppose we had already stored values in a one-dimensional array Y and a two-dimensional array X and we wished to minimize the quantity

$$\sum_i (Y_i - \sum X_{ij}\beta_j)^2 \quad .$$

To do this with the Lincoln Reckoner would require the following instructions:

TRANSPOSE	X	XPRIME	
MATMUL	XPRIME	X	L
MATMUL	XPRIME	Y	R
SOLVE	L	R	BETA

This would put the solution to this least squares problem in the array BETA.

To compare graphically the newly found approximation and the original data,

a user could type

```
MATMUL      X       BETA      XBETA
PLOT        XBETA   Y
```

This would cause a graph of the elements of XBETA to be plotted against the corresponding elements of Y.

5. OPS-3

OPS-3, on-line computation and simulation, was developed at the Massachusetts Institute of Technology, Cambridge, Massachusetts, see Greenberger et al. [1965], OPS-1 grew out of a graduate seminar on advanced computer systems during the spring of 1964. OPS-2, an improved version of the OPS-1 on-line computational facility, was programmed during the summer of 1964. The fall of 1964 saw the beginning of development of OPS-3, a larger-scale improvement, which runs under CTSS, the MIT time-sharing system. As stated by Greenberger et al. [1965], the aim of the designers of the OPS-3 system . . . is to give the researcher a complete information-processing and model-building facility without placing any artificial demands on him; that is, without having the system get in the way of the research. This goal has been partially realized. For example, vectors and matrices are operated upon algebraically and processed as whole arrays simply by symbolic reference to their names. A user need not index through subscripts and test for boundaries, although the ability to do so is retained. OPS-3 does retain some ad hoc rules and conventions which a beginning user must learn to become familiar with OPS-3. However, according to Greenberger et al. [1965], experience has shown that users like the system, especially after becoming well acquainted with it.

Communication with OPS-3 is operator-oriented. There are 60 to 70 standard operators provided by the system such as:

COMPUT (performs a computation and prints a result)

SET (allows algebraic assignment statements)

PRINT (prints information from storage)

READ (allows entry of data from the console)

FIT (fits a linear equation to a multicomponent set of observations)

The SET operator would be used, for example, to perform an assignment as follows:

```
SET   X = 5  *  (Z + 17.3) .
```

Compound operators, called KOP's, can be written in terms of the standard operators and saved for subsequent usage. A KOP is executed interpretively or it may be compiled. In either case it is referred to by name and thought of as an operator. Compound operators may call themselves (recursively) and each other to any depth. A statement to call a KOP named BETA, with parameters X, Y, 25 would be . . . CALLK BETA X Y 25.

With matrix multiplication denoted by ".M." and raising a matrix to a power by ".P." we can use the SET operator to define statistical calculations of the mean and variance for data stored in an array DATA and the covariance for data stored in arrays X and Y as follows:

```
SET     MEAN = (DATA.M. 1)/N
SET       SQ = DATA .M. DATA
SET      VAR = SQ/N - MEAN .P. 2
SET      COV = (X.M.Y)/N - (X.M.1) * (Y.M.1)/(N*N).
```

The REPEAT operator used in conjunction with the IF and IFB operators allows loops to be programmed. This capability allows iterative algorithms to be easily incorporated as KOP's.

Simulation is one of the primary used of the OPS-3 system. There are several operators and compound operators available for this purpose, however since this survey is mathematically inclined we won't discuss this capability here.

For data analysis there are operators for manipulation of stacks, ranking and counting, solving polynomial equations, and for multiple and polynomial regression. The operators for statistical analysis that are available with the OPS-3 system include linear least squares curve fitting and regression, contingency analysis, intercorrelation analysis, and about 10 other statistical tests and computations. Each operator has two forms: a guided form and a short form. The guided form essentially asks the user for the necessary parameters one-by-one. The short form allows a user to enter the parameters as he calls the operator, thus eliminating considerable interaction time waiting for the system to type out messages. OPS-3 has no provision for graphical output.

C. General Purpose Systems – Language Oriented

The following four systems are language oriented as opposed to operation oriented.

1. The MAP System

The MAP (Mathematical Analysis Program) system has been in use by researchers and for teaching at Massachusetts Institute of Technology since the middle of 1964; see Ruyle et al. [1967], Kaplow et al. [1966], and Kaplow et al. [1966a]. The system operates within the MIT Compatible Time Sharing System (CTSS) at the MIT Computation Center or at Project MAC.

MAP can be involked from any teletype or IBM 1050 terminal that can dial into CTSS, regardless of the physical location of the terminal. Graphical output can be requested if a CRT is available. Some graphical output has been obtained by using the MIT Electronic Systems Laboratory display terminal (see Ward [1965]), but these terminals are expensive. MAP with full graphics was used at Harvard with the CRT display facilities there by Ruyle [1967]. Storage oscilloscopes are now available at some terminals.

MAP does not utilize pushbuttons as do the Culler-Fried, AMTRAN, and TOC systems. Instead a language which is a combination of English and arithmetic operation symbols is utilized. According to Kaplow et al. [1966], "the user 'talks' in one or two-word phrases or in arithmetic equations, while the computer uses a passable form of English." The system is flexible in that it allows intermixing of the basic procedures of MAP and user written programs. Automatic error control has been designed into the MAP system in the sense that the procedures used will not lose any of the precision inherent in the input.

Functions are handled as arrays of values at specified values of the independent variable. To insure accuracy of the results of MAP operations on functions, methods are used which are exact (or nearly so) if three point fitting to the given function values is adequate. Thus a user need only be sure that his experimental data and calculated input functions (except those intended for least squares fitting) be tabulated such that a parabolic interpolation over three successive points specifies the function as accurately as is meaningful or required.

MAP has no logical operators. This and the absence of looping facilities mean that the console programming facilities are somewhat limited compared to the AMTRAN, Culler-Fried, and Lincoln Reckoner systems. However, the ability to code functions in various languages and have them executable from within a MAP program provides considerable flexibility to a knowledgeable programmer.

Some of the complex procedures found in the MAP language in addition to such functions as sine, cosine, exp, and abs are the following:

1) Integrate (between fixed or variable limits).

2) Convolute ("folding" of two functions).

3) Least square analysis (linear least square analysis).

4) Basis change (F(x) $\rightarrow$ F(any function of x)).

5) Fourier transformation (sine or cosine or both).

6) Minimax (changes the range of definition of a function).

7) Select (manipulation of a portion of a function).

8) Root (finds roots of an equation g(x) - c = 0).

9) Equalize (creates a function tabulated at equal intervals from two functions which contain the data and the corresponding values of the independent variable).

10) Edit (allows editing of MAP functions).

11) Matrix operations (full package).

2. NAPSS

As stated by Rice and Rosen [1966], the aim in the design of NAPSS (Numerical Analysis Problem Solving System) ". . . is to make the computer behave as if it had some of the knowledge, ability and insight of a professional numerical analyst." Other references to NAPSS are Rice [1967], Roman and Symes [1967a and 1967b] and Symes and Roman [1967]. To implement the above stated aims, the researchers at Purdue University, Lafayette, Indiana, are developing "polyalgorithms" to be the numerical analysts behind the keyboard (on-line console). These polyalgorithms will be ". . . formed by the synthesis of a group of numerical methods and a logical structure into an integrated procedure for solving a specific type of mathematical problem."

NAPSS is currently under development at Purdue. Several polyalgorithms have been developed to various levels of sophistication, see Rice [1967]. The system is planned to be run in a time sharing environment at on-line remote consoles including graphical capabilities, however NAPSS programs will be executable as batch jobs as well. The system is expected to be useful as a device for teaching as well as for solving problems in numerical analysis.

The polyalgorithms being developed as of late 1967 are shown in the following list.

1) Linear equation solver (employs LU decomposition and SOR methods depending on the size of matrix involved).
2) Solve $f(x) = 0$ (one variable).
3) Solve differential equations.
4) Perform differentiation.
5) Determine polynomial zeros.
6) Integration (may use Romberg).
7) Approximation (includes the following):
    a) Least squares polynomial.
    b) Least squares with user specified functions.
    c) Pseudo least squares by $ae^{tx} + b$.
    d) Pseudo least squares by $a + b(x + c)^P$.
    e) Minimax broken line.
    f) Pseudo minimax pricewise cubic with continuous derivatives.
    g) Least squares nonlinear cubic splines.

The NAPSS language is being developed as an interactive problem oriented language. The purpose being to state numerical problems in a mathematical-like notation with direct manipulation of arrays and functions. The language will include a procedural language for expressing user defined algorithms but efforts are being made to eliminate all unnecessary clerical operations sometimes associated with procedural language implementations.

3. POSE

POSE: A language for posing problems to a computer is being developed at the Aerospace Corporation, San Bernadino, California, by Schlesinger and

Sashkin, see Schlesinger and Sashkin [1967] and Sashkin et al. [1967]. Its initial implementation is to be on the IBM System/360 and 1800. The authors state that POSE (Processing, Organizing and Solving Equations) is very similar to NAPSS (see paragraph II. C. 2), however POSE will use general purpose methods on all problems where NAPSS polyalgorithms will attempt to tailor the methods to the problem at hand.

The POSE language will utilize FORTRAN conventions to express algebraic statements and allow assembly language instructions to be included as well. The power of the language comes from the extended capabilities that allow a user to describe his problem in equation-like form. The method of solution as well as translation of the problem from equation form to computer instructions will be provided automatically.

The extended capabilities to be included in the initial version of POSE are:

1) Solution of simultaneous ordinary differential equations (initial value problems)
2) Evaluation of multiple integrals
3) Solution of a transcendental algebraic equation
4) Solution of a system of linear algebraic equations
5) Matrix arithmetic
6) Inversion of square matrices
7) Simplified data input
8) Simplified printed output
9) Two-dimensional graphical display
10) Table lookup and Nth – order interpolation
11) Basic statistical computation
12) Function evaluation with automatic parameter variation

The authors call POSE a polymorphic language since it includes assembly language, procedural language and a declarative language for involking the extended capabilities listed above. The ability to mix Fortran statements with assembly language statements can be very useful to an experienced programmer, however, the POSE implementation of the extended capabilities looks to this author much like a reworded call of a subroutine with a list of arguments. In fact, in writing POSE programs one must be careful to spell things correctly and get parameters in some specified order. This somewhat lessens the usefulness of the language to the non-programmer in an on-line interactive environment.

4. JOSS

JOSS (Johnniac Open-Shop System) has been in daily use since January 1964 by the staff members of the RAND Corporation, Santa Monica, California. JOSS is an experimental, on-line, time-shared computing system which was designed ". . . to give the individual scientist or engineer an easy, direct way of solving his small numerical problems without a large investment in learning to use an operating system, a compiler, and debugging tools, or in explaining his problems to a professional computer programmer and in checking the latter's results." (Shaw [1964]). JOSS was supposed to demonstrate the benefits of on-line interaction.

JOSS was originally written to use the JOHNNIAC computer (a Princeton-class machine built at the RAND Corporation in 1950-53) in a time-sharing mode to provide a modest computing service to the open-shop via remote typewriters. Physically, JOSS consisted of the JOHNNIAC computer, ten remote typewriter consoles, and a multiple typewriter communication system to mediate between JOHNNIAC and the consoles. The capacity of JOHNNIAC (50 μs add time) limited the service to small numerical computations. In 1964 work began on an expanded

JOSS utilizing a PDP-6 computer, see Bryan [1967]. This PDP-6 system became operational in February, 1966, and allows storage of more and larger programs than was previously possible.

The JOSS language permits a user to direct the system in editing as well as computing and typing. According to Ruyle et al. [1967], "JOSS is a simple coherent system but the ratio of clerical detail required to power afforded is rather great." There are no graphical commands or capabilities in the system. A simple example of the on-line typing illustrates the language (U means user, J means JOSS):

U	TYPE	2 + 2
J		2 + 2 = 4
U	SET	E = 2.71828183
U	TYPE	log (E)
J		1

A numeric label (user supplied) as a prefix to a step is an implied command to JOSS to store the step in sequence according to the numeric value of the label. Steps with no numeric label are direct commands to JOSS which are to be executed immediately.

D. Special Purpose Systems

1. STATPAC

STATPAC, a lightpen-controlled program for on-line data analysis, was developed and is being used at Decision Sciences Laboratory, Hanscom Field, Bedford, Massachusetts, see Goodenough [1965]. The system is designed so that no vocabulary of a language need be learned by a user. This is accomplished

by displaying a "menu" of the vocabulary on a computer-driven scope for user selection. By displaying on the scope only those items which will make a syntactically correct command, no invalid commands can be generated.

The author claims similarity in his objectives and methods to the Culler-Fried systems, however, instead of providing general purpose mathematical operators, STATPAC is oriented toward statistical operations. While composing a command at the console, menus of operands (vector titles) and operations are displayed for lightpen selection. Some of the available operations are:

1) DISPLAY
2) TYPE
3) PUNCH
4) CORR between (X) and (Y)
5) STD DEV OF
6) MEAN OF
7) REGRESSION (X) vs (Y)
8) SUM OF SQUARES OF

2. Marchuk and Yershov's System

According to Marchuk and Yershov [1965], ". . . the problem of solving differential equations comprises at least two thirds of all problems arising in engineering and scientific calculations." Thus they proposed (in 1965) to construct a nonclosed programming system based on a continuous man-machine interaction to aid in the solution of differential equations. In this problem area they plan to have the human make such decisions as:

1) Specification of rules, variable substitution, and direction of analytical transformation to reduce the problem to a canonical form.

2) Information on partition of the operator of the equation into elementary ones.

3) Which difference scheme is preferred for a given elementary operator.

The current status of this on-line system for solving differential equations is unknown.

3. Gear's System

Gear [1966] describes a system for finding the numerical solution of ordinary differential equations at a remote terminal. Systems such as MAP represent a function by a pair of vectors of values and use interpolation when necessary. However, this method is not sufficiently accurate nor is it convenient for high accuracy problems in differential equations. Therefore, at the Department of Computer Science, University of Illinois, Urbana, Illinois, a system has been developed for the solution of differential equations. The system consists of three basic packages:

1) The numerical integrator – this contains a large number of switches which determine accuracy.

2) The equation compiler – this allows input in a natural form.

3) The dialogue program – this accepts input, corrections to input (editing), and interacts with the user to set the switches in the integrator.

A set of differential equations is entered, for example, by typing

```
10          Y0' = Y1 * Y2
20          Y1' = -Y0 * Y2
30          Y2' = -.51 * Y0 * Y1
            END
```

To simplify expressions and to make it easier to modify parameters, 23 variables named A through W are allowed in the expressions on the right hand sides.

Up to ten dependent variables Y0 through Y9 may be used, the highest order derivative of each must appear on the left hand side of an equation.

The dialogue consists of the following steps:

1) Type in equations
2) Enter initial values
3) Optionally enter
    a) step size upper limit
    b) order
    c) error tests for step control
    d) or fixed step
    e) where and what to print
    f) format
    g) when to stop

In 1966 teletype models 33 and 35 were in use as remote terminals. Typewriter plots of the solution are generated and plotted on-line. No other graphical devices are available to the system.

4. Dixon's On-Line Statistical Programs

Dixon [1967] describes the use of on-line displays with packaged statistical programs. In this paper he discusses the usefulness of being on-line with a statistical program and especially the usefulness of on-line graphical ability. He says ". . . many experts in data analysis have always used graphical methods to aid their analysis of data. One often hears directives of these experts to their assistants something like, 'go thou and plot your data.' The plots and charts frequently do not survive the process of report writing and publication, but have played an important part in the analytical process itself." An on-line interactive situation is ideal for the examination of these charts and plots which are discarded during the data analysis process.

At the Health Sciences Computing Facility, University of California at Los Angeles, California, the BMD programs (Dixon [1964]) are being rewritten for on-line usage. Programs for stepwise regression, linear regression and spectral analysis are now available for on-line use and other programs are being developed. The programming is being carried out for an IBM System 360/75 computer with an IBM 2250 display unit with lightpen and special function keyboard as a console.

5. PEG System

At the Stanford Linear Accelerator Center, Stanford University, Stanford, California, this author has been developing the PEG (On-Line Data-Fitting) system. The work was begun in the fall of 1967 on an IBM System 360/75 computer using an IBM 2250 II display unit with lightpen as the interactive console. The interactive program runs in a separate partition of memory with high priority. By the fall of 1968 a working system was available and was used by physicists for actual data-fitting problems. By October 1968 the IBM 360/75 had been replaced by an IBM 360/91 and the PEG system was operational on that computer.

As described in Smith [1969], the PEG system allows user selection of:

1) Fitting function
    a) user defined function
    b) orthogonal polynomials
    c) spline functions – fixed or variable joints
    d) Fourier approximations
2) Data mode
    a) data from cards
    b) data of previous fit

c) residuals of previous fit

d) keyboard entry

3) Display mode – after a fit has been computed there are seven different display modes.

In addition to the above, PEG allows specification of degree, initial guesses for nonlinear problems, choice of minimization method (in some cases), correction, subset selection, selective deletion, or transformation of data values.

All user actions are either light pen selections or numerical entries from the keyboard. This has been accomplished by anticipating in advance all possible (at least nearly all – hopefully) desires of a user and providing for on-line selection from the list of available options.

The PEG System was partially inspired by the DATAN System, see Simonsen and Anketell [1966]. Some other references of interest in the approximation and curve-fitting areas are Conn and vonHoldt [1965] and deMaine [1965]. Pyle [1965] describes a system for on-line data input by question and answer which is related to the method employed by PEG to obtain input from the user. PEG in many cases asks multiple choice questions which can be answered with the lightpen.

E. Other Systems

1. DIALOG

DIALOG is a conversational programming system with a graphical orientation described by Cameron et al. [1967]. The language was designed by the IIT Research Institute, Chicago, Illinois, ". . . as an experimental development intended to explore the effectiveness of an on-line graphical communication terminal as an algebraic programming tool. The system relies entirely on a

graphical stylus and a single push button to provide input and, when used on-line, does not make use of a mechanical keyboard."

The language is designed to be used as a computational aid for a casual user. To facilitate use by untrained personnel, on-line DIALOG programs are composed by selecting symbols from a list displayed on an on-line oscilloscope. The list of displayed symbols is restricted to those symbols whose choice will result in a syntactically correct program. A few lines of code will illustrate the type of statements that occur in DIALOG programs.

12.1 K = K + (B - Y) * (X↑2 + A↑2);

12.6 'WRITE' K, A, B

13.3 'PLOT' (A, B)

An interpretive processor for DIALOG programs has been coded for the UNIVAC 1105 and imbedded in a time sharing monitor that allows simultaneous operation of several terminals. Programs can be prepared off-line, run as batch programs, and can produce hard copy of results as well as the on-line mode of operation. In fact, a DIALOG compiler has been prepared for use on the IBM 7094 for strictly batch operation.

2. MATHLAB

MATHLAB: A program for on-line machine assistance in symbolic computations, is described by Engelman [1965]. The program has been developed on the time-shared system of project MAC at Massachusetts Institute of Technology and on the IBM 7030 at the MITRE Corporation, Bedford, Massachusetts. As of September 1, 1965 work was under way to provide the display of mathematical expression on scopes and to adapt MATHLAB to the AN/FSQ-32 computer at the Systems Development Corporation, Santa Monica. In 1965 MATHLAB had no graphical capabilities.

Some of the qualities of MATHLAB are listed as follows:

1) Numerical computations – these are weak since original effort was in the area of symbolic computation.
2) Symbolic computations – capabilities include
   - simplification
   - substitution
   - adding equations
   - differentiation
   - some integration
   - solution of equations, etc.
3) Simple user commands – for example to differentiate e1 with respect to e2 a user need only type "differentiate"(e1 × e2).
4) The program can be expanded by any LISP programmer.
5) MATHLAB can be extended by the user – he can "teach" it derivatives and rename system functions.
6) It is intimate – a close relationship develops between user and computer.

The following list of some of the system commands available with MATHLAB gives an indication of the kind of problems that can be worked on in the MATHLAB environment.

repeat	flip
pleasesimplify	makeequation
forget	makeexpression
substitute	makefunction
add	factor
multiply	differentiate
subtract	learnderivative

division	integrate
raise	solve
negative	rename
invert	newname

3. MAGIC PAPER 1

In 1963 Clapp and Kain [1963] described a computer aid for symbolic mathematics called Magic Paper 1. This is a computer system that is primarily for symbolic mathematics but it does allow some function evaluation and plotting on a display scope. Magic Paper 1 was developed at Bolt Beranck and Newman, Inc., Cambridge, Massachusetts,for a small time-shared computer (PDP-1).

The console consists of a typewriter, a display scope and a lightpen. A paper tape reader and punch and several magnetic tape units are available for large scale input/output and saving information off-line. Control of the calculations is handled by executive control characters which the user types in. Some typical control characters are:

I - - - enter input mode

, - - - leave input mode

P- - - display pointer on scope

D- - - display figure

EVAL- - - evaluate function

Typical manipulations are insertion, substitution, multiplication of an equation by a term, transposition, and the addition of two equations. A function can be built up and then plotted on the scope to examine its graphical characteristics. The scope plots can be "zoomed" in and out to examine certain features in detail.

Standard two dimensional mathematical notation is used on the scope while a linear typewriter notation is used for input. The following mathematical evaluation operators are included in the system:

+	addition
-	subtraction
×	multiplication
/	division
Σ	sum over an index between limits
π	product over an index between limits
↑	exponentiation
() []	parenthesisation
if then	
< > ≤ ≥	conditional expressions
when	

Notational flexibility is built into the Magic Paper system by allowing the user to define new control and data interpretation operations as he proceeds with a problem.

4. The DISPLAY System

The DISPLAY system was written at System Development Corporation, Santa Monica, California, in 1967, see Bowman and Lickhalter [1968]. DISPLAY is a system for graphical data management in a time-shared environment which has been written as a forerunner of the display component of TDMS (Time-Shared Data Management System) being implemented on the IBM System 360 family of computers. [TDMS became operational in the summer of 1968, see DATAMATION,

August 1968, p. 95.] The three goals in designing DISPLAY were

1) To provide satisfactory response within a time-shared computer.
2) To produce a system easy for the nonprogrammer to use.
3) By achieving the first two goals, gain users for the system in order to obtain feedback to improve the system.

These three goals have been achieved by the DISPLAY system implemented on the SDC Q-32 machine, running under the time-sharing system with no special consideration from the executive.

The primary function of the system is to allow graphical examination of stored data. However, excursions into TINT (a higher-order Algol-type interpreter) are allowed whereby a user can perform nonstandard operations on the data. For example, assume we are using DISPLAY and we have already specified the data to be retrieved (time for data retrieval varies from 15 seconds to 5 or 6 minutes). At this point we could execute the following steps:

1) Call TINT
2) Specify a TINT program (from a list of already written programs) or
3) Write a new TINT program on-line
4) The TINT program then operates on the data and outputs a graphic data array which is fed into DISPLAY
5) DISPLAY presents the graphic output on the scope

This capability could be used to do data-fitting similar to that provided by PEG (see paragraph II.C.5) if the appropriate TINT programs were available.

5. The Klerer-May System

Klerer and May [1964] describe a software-hardware system for the purpose of facilitating the programming and analysis of well-formulated problems. Other references to this system are Klerer and May [1965a], [1965b] and Klerer and Grossman [1967].

The Klerer-May system was originally written for a GE-225 computer at the Hudson Laboratories of Columbia University, Dobbs Ferry, New York. The system employs modified Flexowriters as input/output stations. Special characters and the ability to control platen movement by half spaces from the keyboard give the modified Flexowriter the capability of producing "natural" two-dimensional mathematical notation. In other words, the Klerer-May language employs summation signs, integral signs, superscripts and subscripts as in conventional mathematical notation. The compiler takes this two-dimensional input and produces very efficient code (2 to 4 times faster object code than other compilers for the GE-225).

The system is supposed to be self-teaching and succeeds quite well as testified by the fact that a one-page manual is all that is handed to new users of the system. After a program has been entered into the system, the computer "echos" its interpretation of the input, thus catching some ambiguities and mistakes in a program. Plans are to implement the system with on-line interactive programming capability.

6. Moore et al.

Moore et al. [1966] discuss their experiences with a remote console time shared system at the University of Western Australia. In particular they discuss FORDESK, a FORTRAN compatible on-line system which enables simultaneous editing, translation, execution and debugging from a user's console. FORDESK was first released towards the end of 1965.

Moore and Erickson [1966] describe the use of a CRT with lightpen in a time-shared environment. Of particular interest is an application involving polynomial curve fitting. The curve fitting program allows dynamic location of the axis, adjustment of scale, and choice of the degree of polynomial all by lightpen picking of displayed "light buttons."

7. Ball and Hall

Ball and Hall [1967a] discuss PROMENADE, an on-line system for data analysis using clustering techniques. Their work has been carried out at the Stanford Research Institute (SRI), Menlo Park, California. They use a high precision CRT to perform interactive statistics and data analysis. Other references of interest are Ball and Hall [1967b] and Eusebio and Ball [1968]. These discuss methods of handling multivariate data in an on-line graphical environment.

III. SOLUTION OF A LEAST SQUARES PROBLEM BY VARIOUS SYSTEMS

In this section we will take a least squares problem and discuss how it would be solved by several of the general purpose interactive systems described in the preceding section. The particular problem we will consider is a constrained linear least squares problem that arose in a physics laboratory. Several measurements related to a cross section problem gave points which were to be approximated by a polynomial in even powers only with the added restriction that the polynomial should always be non-negative. The data, with weights (inverse of errors in the points), is given in Table 1. The approximating polynomial is to be:

$$p(x) = a_0 + a_1X^2 + a_2X^4 + a_3X^6 \quad .$$

In the case at hand, restriction of a_0 to be non-negative is sufficient to satisfy the non-negativity of $p(x)$.

TABLE 1

DATA TO BE FIT BY p (x)

i	x(i)	y(i)	w(i) weight
1	.713	2.7	5.0
2	.856	8.1	1.667
3	.932	13.2	1.111
4	.988	21.5	.6667
5	.994	22.4	.625

Restating the problem we have:

given: Data $\{(x_i, y_i, w_i)\}_{i=1}^{5}$

given: A function $p(x) = a_0 + a_1X^2 + a_2X^4 + a_3X^6$ with $a_0 \geq 0$.

Find: Values of $a_0, \ldots a_3$ which give the weighted least squares fit of $p(x)$ to the given data. That is

minimize $\sum_{i=1}^{5} w_i^2 [y_i - p(x_i)]^2$.

Display: The resulting fit superimposed on the data.

This problem could also be formulated as a quadratic programming problem and solved by known quadratic programming methods.

In some of the following sample solutions to this problem we will ignore the constraint, as it complicates the solution considerably. Without the constraint we have a weighted linear least squares problem which is easily handled by some of the on-line systems under consideration. The weights are also ignored in some cases to simplify the problem even more.

A. Culler-Fried Solution

The Culler-Fried system is oriented toward handling functions defined on equally spaced points. The unequal spacing of the data points makes this problem more complex than it might otherwise be; however, a considerable amount of console programming would be required even for equally spaced points.

To illustrate the amount of programming required using the Culler-Fried system, let us consider the code necessary to evaluate only the polynomial,

$$p(x) = a_1 + \ldots + a_3 X^6 \quad .$$

The following code will type out a message "ENTER X ....." and upon entry of a value, proceed to evaluate the polynomial

$$p(X) = A + B \cdot X^2 + C \cdot X^4 + D \cdot X^6 \quad .$$

The code is:

```
TYPE RS ENTER X.....
LII REAL LOAD ENTER
SQ STORE Y SQ STORE
Z ⊙ Y ⊙ D STORE
T LOAD Z ⊙ C ⊕
T STORE T LOAD Y ⊙
B ⊕ T ⊕ A STORE
T.
```

After the above code has been executed, the value of p(X) is stored under the key T for later use. Noting the detail of the code to evaluate the polynomial we can extrapolate to say that the code to solve a nonlinear set of equations by iteration, or even the code to solve a set of simultaneous linear equations would be quite involved. The code complexity is, of course, relative. What we are comparing this code to is, for example, the code required by NAPSS to solve a set of simultaneous equations; see paragraph III. B.

B. NAPSS Solution

Thanks to Symes [1968], we have two methods of solving the least squares problem if we ignore the constraint and the weighting. With no constraint on the constant term the problem is a linear least squares problem. Assuming we already have the data read into the one-dimensional arrays X and Y we have the following methods.

Method 1. (weighting can easily be added to this method)

```
FOR I←—1, 2, ..., 4 DO
    Q[I]←SUM (Y[K]X[K] ↑ (2(I-1)), FOR K←—1, 2, ..., 5)
    FOR J←—I, I+1, ..., 4 DO
        R[I, J]←—SUM (X[K] ↑ (2(I+J-2)), FOR K←—1 TO 5)
        IF I≠J THEN [R+J, I] = R[I, J];;;
SOLVE R*A = Q FOR A;
P(X)←—A[1] +A[2] X ↑ 2 + A[3] X ↑ 4 + A[3] X ↑ 6
TABLE (f(X), Y, X)
PLOT (f(X), P(X), ON X[1] ≤ X ≤ X[5])
```

Method 2.

TABLE (f(X), Y, X)

P(X)←APPROXIMATION (f(X), WITH 1, X ↑ 2, X ↑ 4, X ↑ 6, USING LEAST SQUARES)

A←COEF(P)

PLOT (f(X), P(X), ON X[1]≤X≤X[5])

Although the authors of NAPSS have not mentioned it yet, we expect that someday they will add a polyalgorithm for optimization (locating a maximum or minimum). A call on such a polyalgorithm might be

MINIMIZE (f(A), STARTING WITH A=A0).

A solution to the weighted and constrained least squares problem might then be as follows (we assume the data has already been read into the vectors X, Y, and W, and that starting guesses for the coefficients are in A0):

Possible future method without constraint.

P(X, A)←A[1] + A[2] X ↑ 2 + A[3] X ↑ 4 + A[4] X ↑ 6

F(A)←SUM(W[I] ↑ 2 (Y[I] - P(X[I], A)) ↑ 2, FOR I=1 TO 5)

MINIMIZE (F(A), STARTING WITH A=A0)

Possible future method with constraint.

P(X, A)←A[1] + A[2] X ↑ 2 + A[3] X ↑ 4 + A[4] X ↑ 6

F(A)←SUM((W[I] ↑ 2) (Y[I] - P(X[I], A)) ↑ 2, FOR I=1 TO 5) IF A[1]≥0
OTHERWISE 10 ↑ 30

MINIMIZE (F(A), STARTING WITH A=A0)

C. POSE Solution

Since POSE is not yet fully implemented it is difficult to state a solution very precisely. POSE has been implemented on an IBM 1800 with limited capabilities; see Sashkin et al. [1967]. Plans call for implementation of the

capability to solve a system of linear algebraic equations which would allow solution to the weighted least squares problem (without constraint). This modified problem would be solved by a program similar to method 1 of the NAPSS language. After entry of the data, Fortran statements would be used to calculate the matrix of the normal equations; then the solution vector would be found by the built-in subroutine for solving simultaneous linear equations. An approximation to what the POSE program would look like is:

```
POSE program (without constraints)
S.0     CALCULATION SEQUENCE
        READ DATA
        EXECUTE S.20
        PRINT RPT.1 (X(1), X(2), X(3), X(4))
        PROBLEM END
S.20    RANGE OF I = 1(1)4
        RANGE OF J= 1(1)4
        A(I, J) = 0.0
              RANGE OF K = 1(1)N
              A(I, J) = A(I, J) + W(K)*W(K)*X(K)**(2(I+J-2)
        RANGE OF I=1(1)4
        B(I) = 0.0
              RANGE OF K = 1(1)N
              B(I) = B(I) + W(K)*W(K)*Y(K)*X(K)**(2(I-1))
        X = SOLVE (A, B)
```

To plot a graph of the resulting computed fit would involve first evaluation of the polynomial at a series of points for plotting purposes (call these values YY

and the independent variable XX). Once the points to be plotted are computed we would then use the following POSE statements to obtain the graph:

PLOT GRAPH: 1(XX: XMIN: XMAX, YY: YMIN: YMAX)

TITLE (POLYNOMIAL(X), X)

The statement, DISPLAY GRAPH:1, in the calculation sequence would then cause the display to take place.

D. OPS-3 Solution

To solve the constrained least squares problem using OPS-3 would involve writing a compound operator to solve the non-linear problem. Therefore let us describe how to solve only the weighted linear least squares problem. There is an operator in OPS-3 which performs linear least squares curve-fitting. A linear equation is fitted to the observations with certain optional constraints. The allowable constraints, however, do not include the constraint of interest in this problem. The linear equation used is

$$Y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n \quad ,$$

where the $x_1, \ldots, x_n$ are considered as n independent variables. To use this operator on the problem under consideration, we define four independent variables:

$$x_1 = 1$$
$$x_2 = X^2$$
$$x_3 = X^4$$
$$x_4 = X^6$$

By defining four independent variables (including the constant, 1) and by selecting the option which suppresses the term a_0 in the above equation, we can solve the

least squares problem with weighting. First we consider the equations

$$y_i w_i = w_i a_1 + w_i a_2 X_i^2 + w_i a_3 X_i^4 + w_i a_4 X_i^6 \quad , \quad \text{for } i=1, 2, \ldots, 5 \ .$$

Assume we have already read the data into three arrays X, Y, and W. Let A be a matrix with A(O, K) denoting its kth column. The OPS-3 code to solve the problem will then be:

```
SET A(0, 1) = W
SET A(0, 2) = W*X*X
SET A(0, 3) = W*X.P.4
SET A(0, 4) = W*X.P.6
SET Z       = W*Y
LINFIS Z A 4 5 2
```

The last line of the code is an invocation of the linear least squares curve-fitting operator in its short form with all parameters given immediately. The guided form of the call would be

```
LINFIT Z A
```

and the OPS-3 system would then come back and ask for values of the other parameters which are:

NV = 4 = number of variables

NO = 5 = number of observations

K = 2 = code to indicate option which suppresses the constant term of the linear least squares fitting function

The printed output from the LINFIT operator includes the coefficients of fit and a measure of the goodness of the fit.

E. JOSS Solution

JOSS does not have complex operators included in the language so the code to solve a problem such as the solution of a set of simultaneous linear equations must be provided by the user. However, programs may be stored for later use so such frequently encountered problems may be solved by calling out a previously coded set of JOSS instructions. Since we do not wish to take the space here to show the solution to the non-linear constrained least squares problem, we will consider only the linear problem and assume that a program to solve a set of simultaneous linear equations is already stored. All we need provide is the coefficients of the weighted least squares normal equations. This would be done on a JOSS console as follows (U - denotes inputs of the JOSS user. J — denotes outputs from JOSS):

(First we read in the data)

```
U     4.1    DEMAND X(I).
      4.2    DEMAND Y(I).
      4.3    DEMAND W(I).
      DO PART 4 FOR I=1(1)5.
J/U   X(1) = 0.713
      Y(1) = 2.7
      W(1) = 5.0
        .
        .
        .
      X(5) = 0.994
      Y(5) = 22.4
      W(5) = 0.625
```

(Now we calculate the coefficients of the normal equations)

```
U    5.1    SET Z(K) = Z(K) + (W(I)*2)·Y(I)·X(I)*(2(K-1))
       6.1  SET Z(K) = 0
       6.2  DØ PART 5 FOR I=1(1)5
            DØ PART 6 FOR K=1(1)4
       7.1  SET A(I,J) = A(I,J) + (W(K)*2)·X(K)* (2(I+J-2))
       8.1  SET A(I,J) = 0
            DØ PART 7 FOR K=1(1)5
       9.1  DØ PART 8 FOR J=1(1)4
            DØ PART 9 FOR I=1(1)4
            TYPE Z
J               Z(1) = ....
                Z(2) = ....
                Z(3) = ....
                Z(4) = ....
```

Here JOSS will type out the computed values of the right hand sides of the normal equations. The coefficients stored in the matrix A could also be printed, if desired; then the precoded program to solve simultaneous equations could be called on to provide the desired solution to the weighted least squares problem.

F. Lincoln Reckoner Solution

As stated by Stowe et al. [1966], the Reckoner "is primarily a facility for making use of routines, not for writing them." There are routines or operations available in the Reckoner public library to perform many operations including solving a matrix equation, AX = B, but again there is no provision for

for automatically solving the non-linear least squares problem with constraints. Therefore let us only consider the weighted least squares problem as we have done for several other systems.

Assume the data has been entered into the arrays X, Y, and W. We could then form arrays as follows:

$X1 = W$

$X2 = W \cdot X^2$

$X3 = W \cdot X^4$

$X4 = W \cdot X^6$

$Z = W \cdot Y$

A two-dimensional array A would then be formed with X1, X2, X3, X4 as its four columns. There is an operation available in the Reckoner library for replacing a column of a given matrix by a given nxl array which could be used to form A from X1, X2, X3, X4. Four lines of Reckoner code would then solve the linear, weighted least squares problem

$$A^T A\beta = A^T Z.$$

The Reckoner code is as follows:

```
TRANSPOSE A APRIME
MATMUL APRIME A L
MATMUL APRIME Z R
SOLVE L R BETA .
```

The coefficients of the least squares solution are now stored in the array BETA.

There is graphical output available with the Lincoln Reckoner and a routine available in the Reckoner public library to facilitate plotting on the CRT. A plot of the data points can be obtained directly from the data arrays X and Y. To obtain a plot of the least squares curve determined by the

coefficients stored in BETA some further calculations are necessary. Two arrays would need to be generated; XPLOT with some equally spaced values of X over the range of interest and the corresponding values of the fitting function in YPLOT. To compute each value stored in the array YPLOT some code such as the following would need to be executed.

```
MULT    XPLOT(I)    XPLOT/I)    XSQ
MULT    BETA(4)     XSQ         T
ADD     T           BETA(3)     T
MULT    T           XSQ         T
ADD     T           BETA(2)     T
MULT    T           XSQ         T
ADD     T           BETA(1)     YPLOT(I)
```

After XPLOT and YPLOT are appropriately defined, the routine to generate a plot on the CRT can be invoked.

G. TOC Solution

The TOC system is derived from the Culler-Fried system; however it has been changed to allow a more direct entry of formulas so that coding a solution to the weighted, constrained least squares problem under consideration should be somewhat more easily accomplished than with the Culler-Fried system. Thus instead of the code shown in paragraph III. A to evaluate the polynomial

$$p(X) = A + BX^2 + CX^4 + DX^6$$

we would have in the TOC system the code:

```
P(X, A, B, C, D) = A + B*X**2+C*X**4+D*X**6
E   P(X, A, B, C, D)
STORE   POLY    .
```

Any further evaluations of the polynomial for different values of X, A, B, C, or D can be accomplished simply by writing

E P(X, A, B, C, D) .

As with the Culler-Fried and JOSS systems there are no higher level operations provided for the solution of equations, or minimization, for example. Therefore, all phases of the least squares problem have to be coded. User-defined console programs can be stored however and reused again and again. Console programs could be coded and stored to perform the computations necessary to the solution of the least squares problem. We will not go into this code at this point.

H. MAP Solution

One of the several complex procedures included in the MAP system is a procedure for least squares analysis. There is no direct method of solving the constrained, weighted least squares problem but the straightforward linear least squares problem is handled quite easily.

First the four functions,

$F1(X) = 1$

$F2(X) = X^2$

$F3(X) = X^4$

$F4(X) = X^6$

should be defined and the data read into arrays XDATA and YDATA. The arrays YDATA(X), F1(X), F2(X), F3(X) and F4(X) must all be defined for the same values of X, namely those found in XDATA. Once the functions are all appropriately defined, the interaction can take place as follows (user type-in will

be underlined):

LEAST SQUARE

I CAN FIT EQUATIONS OF THE FORM

V(Y)=XA*FA(Y) + XB*FB(Y)+XC*FC(Y)+XD*FD(Y)+XE*FE(Y) WITH A MAXIMUM OF 5 UNKNOWNS, XA, XB, ETC., AND 100 DATA POINT.

WHAT IS THE NAME OF THE VARIABLE COMPARABLE TO V(Y).

YDATA(X)

HOW MANY FUNCTIONS, FA(Y), FB(Y), ETC., WILL BE REQUIRED TO FIT THE DATA. 4

PLEASE PRINT ON THE NEXT LINE THE NAMES OF THE 4 FUNCTIONS REQUIRED.

F1(X) F2(X) F3(X) F4(X)

Following this interaction MAP will print the coefficients of the normal equations and the results of their solution. Then the user is presented with options to have the fitted curve and the residuals printed.

If a graphical output terminal is available the MAP plot command can be used to display the resulting fitted curve. A point plot of YDATA(X) and a line plot of YFIT(X) could be obtained by the following interaction (we assume the least squares fit has been calculated and stored in YFIT(X)):

PLOT

PLOT WILL CREATE A GRAPH OF THE DESIRED FUNCTION(S). WHAT FUNCTIONS WOULD YOU LIKE TO PLOT. YDATA(X) YFIT(X).

SHOULD THE PLOT BE LINEAR, LOG-LOG, LINEAR-LOG, OR LOG-LINEAR.

LINEAR

DO YOU WANT A POINT OR LINE PLOT OF YDATA(X). POINT

DO YOU WANT A POINT OR LINE PLOT OF YFIT(X). LINE

IF YOU DO NOT WANT ALL OF THE POINTS OF THE FUNCTION(S) PLOTTED, TYPE THE RANGE AND/OR INTERVAL IN X TO BE USED. OTHERWISE JUST GIVE A CARRIAGE RETURN.

I. AMTRAN Solution

The AMTRAN system is based on the Culler-Fried system with a large keyboard providing many function buttons. AMTRAN provides the facility to store user programs (operations) "under" buttons for later use. There is a button which invokes a "locate minimum" operator which could be used in solving the weighted, constrained least squares problem we have posed. If the operation locates a minimum with respect to a single variable it could be used to construct a Gauss-Seidel type of minimization of the sum-of-squares,

$$S = \begin{cases} \sum_{i=1}^{5} \left[W_i^2 \; y_i - a_0 - a_1 X_i^2 - a_2 X_i^4 - a_3 X_i^6 \right]^2 & \text{if } a_0 > 0 \\ 10^{30} \text{ if } a_0 \leq 0 \; . \end{cases}$$

If the "locate minimum" operates on a function of several variables it could be used directly on the function S given above.

A scope is available for output from the AMTRAN system. An instruction such as

DISPLAY SCOPE Y

will cause display of the function described by the values in the array Y to be displayed on the scope. Such instructions could be used to display the fitted curve.

IV. HOW WOULD PEG SOLVE THE SAME PROBLEM?

In Smith [1969] the workings of PEG are explained in more detail. Here we will just outline the steps necessary to solve the weighted, constrained least squares problem.

There are one or two preparatory steps to be carried out before going on-line with PEG. First a FORTRAN function must be coded to evaluate the function

$$p(x) = a_0 + a_1x^2 + a_2x^4 + a_3x^6 \quad .$$

The constraint on a_0 is handled in the code for evaluating $p(x)$ by the following method. Each evaluation is preceded by a test on a_0. If $a_0 < 0$ then $p(x)$ is set to 10^{30} (a large number less than the square root of the largest number that can be held in the machine). If $a_0 \geq 0$ $p(x)$ is evaluated normally. By returning a large value for $p(x)$ if $a_0 < 0$, the sum of squares of residuals which is being computed becomes vary large. Consequently negative values of a_0 are avoided by the minimization routine as it minimizes the sum of squares of residuals. This coding of "user functions" is detailed in Smith [1969]. A second step, prepunching the data on cards is optional since there are only five data points and we could easily enter then through the keyboard when we get on-line. Let us assume for this problem that we will enter the data on-line.

When the code for $p(x)$ (constrained) is ready, we include the cards with the running deck for PEG and submit the total deck to be run. When our program begins execution an introductory picture appears on the IBM 2250 display unit. At this point we can sit down at the console and by selecting options with the light-pen and entering numbers from the keyboard we can obtain the desired least squares fit. Rather than show pictures of the scope images that appear on the screen as we

step through this problem, we will give a simple word description of each picture and indicate what user action is necessary at each step. In the following list "Lightpen (continue)" means to touch the "* CONT" option with the lightpen, this will cause the program to go on to the next step. The other lightpen actions require choosing an item from a list by touching the selected item with the lightpen. In Smith [1969], typical pictures of the actual scope images are shown for the various steps involved in the data-fitting process.

Pictures Seen During Solution of Least Squares Problem

	Picture	Action Required
(1)	Introduction	Lightpen (continue)
(2)	Choose function	Lightpen (choose user function) Lightpen (continue)
(3)	Choose data mode	Lightpen (choose keyboard) Lightpen (continue)
(4)	Enter data from keyboard	Lightpen (choose weights) Keyboard (enter data points with weights) Lightpen (continue)
(5)	Display titles	Lightpen (continue)
(6)	Make corrections	Lightpen (continue)
(7)	Perform transformation on data	Lightpen (continue)
(8)	Delete data points	Lightpen (continue)
(9)	Enter number of parameters and first guess for parameters	Keyboard (enter 4--number of parameters) Lightpen (choose automatic parameter adjustment) Keyboard (enter first guess--0.0,0.0,0.0,0.0) Lightpen (continue)
(10)	Choose minimization method	Lightpen (choose "direct search") Lightpen (continue)

(11)	Option to enter parameters to "direct search"	Lightpen (use preset values and go on)
(12)	Iteration 0 displays data with fit superimposed	Lightpen (continue)
(13)	Iteration 1 • • •	Lightpen (continue)
(14)	Iteration N	Lightpen (stop iteration if convergence seems to have obtained--program will also automatically terminate iteration)
(15)	Comparison of starting fit and converged fit	Lightpen (use best values and continue)
(16)	Option to examine fit with iteractive minimizer	Lightpen (terminate iteration)
(17)	Choose display mode	Lightpen (choose display data and fit with fit display Lightpen (continue)
(18)	Display of data with fit superimposed and table of fit	Lightpen (back-up to choose another display mode)
(19)	Choose display mode	Lightpen (choose display data and fit with coefficients displayed) Lightpen (continue)
(20)	Display of data with fit superimposed and table parameter values	Lightpen (back-up to choose another display mode)
(21)	Choose display mode	Lightpen (choose display extrapolated fit)
(22)	Display data on fit extrapolated	Lightpen (continue)
(23)	Branching display	Lightpen (choose terminate computer processing) Lightpen (continue)

END

Note that we are led through the data-fitting process by the computer program. At several points we are presented with the opportunity to perform some optional task, for example deletion of data points. In such cases if we wish to bypass that step we simply use the lightpen to continue to the next step. Where there are default options built into the program we may always "continue" by lightpen action. On the other hand, if, for example, the program requests keyboard entry of the number of parameters involved, and there is no default option, the lightpen will have no effect until the required keyboard entry has been accomplished.

As we step through the iterations shown in displays (12) through (14) above, we can watch the improvement in the fit of successive iterations. If the fit does not improve, it is possible to terminate iterations and try different starting values. The display (22) above allows us to examine the extrapolated fit. This is of special interest in this problem since the constraint on the first parameter is supposed to keep the approximating function from ever being negative. A visual examination of the extrapolated fit can provide verification of the success or failure of the constraint.

V. SUMMARY

Table 2 summarizes some of the characteristics of the general purpose systems and of the special purpose systems. It is apparent that Culler and Fried were the first to develop an on-line interactive system for solving mathematical problems. Their work began in 1961. JOSS, MAP, and OPS appear to have all become operational in 1964. AMTRAN and the Lincoln Reckoner are chronologically more recent as they both came into use early in 1966. Since 1966 we have NAPSS and POSE as the most recent systems. NAPSS is still under development.

TABLE 2

SUMMARY OF ON LINE SYSTEMS

System	Dates	Home Institution	Machine (operating system)	Console Facilities
General purpose systems				
Culler-Fried	1st developed 1961	TRW Canoga Park, Calif.	RW-400	two 48 button keyboards and a 5" storage scope
	since late 1964	TRW Systems Redondo Beach, Calif.	BR-340 (4 users)	
		University of Calif. at Santa Barbara	RW-400	"
	In 1968	University of Calif. at Santa Barbara	IBM 360/65 (many users)	"
TOC	began development in 1966	Aiken Computation Lab., Harvard University	(CTSS)	65 operator keys, plus a full typewriter keyboard and a storage oscilloscope
AMTRAN	available since early 1966	NASA Marshal Space Flight Center, Huntsville, Alabama	1620	224 button keyboard, typewriter, two 5" storage scopes
Lincoln Reckoner	in routine use since spring 1966	Lincoln Laboratories of M.I.T.	TX-2 (APEX time sharing system)	a two-keyboard typewriter, a refreshed CRT, an on-line xerox printer, paper tape input available
MAP	in use since mid.-1964	Massachusetts Institute of Technology	(CTSS)	typewriter or teletype storage scope or refreshed scope if available
NAPSS	under development since 1966	Purdue University, Lafayette, Indiana	proposed (a time sharing system)	proposed typewriter scope for display
POSE	began work early in 1967	Aerospace Corporation, San Bernardino, Calif.	1800 (presently Batch) proposed (time sharing)	proposed typewriter and scope

OPS-3	Predecessor begun in Spring 1964	Massachusetts Institute of Technology	(CTSS)	typewriter or teletype
JOSS	daily use since 1964	RAND Corporation, Santa Monica, Calif.	JOHNNIAC (dedicated to a few consoles) PDP 6 (since Feb. 1966) (time shared)	typewriters "
Special Purpose Systems				
STATPAC (statistically oriented)	in use in 1965	Decision Sciences Lab., Hanscom Field, Bedford Massachusetts		CRT with light pen, paper type input
Marchuk and Yershov (differential equations)	being planned in 1965	Russia		
Gear (differential equation)	in use in 1966	University of Illinois, Urbana, Illinois	ILLIAC-II	teletypes (33 and 35)
Dixon (statistical)	some parts in use in 1968	UCLA, Health Sciences Computing Facility, Los Angeles, Calif.	IBM 360/75 (high priority partition)	IBM 2250 scope with light pen and special function keyboard
PEG (data-fitting)	available for use Fall 1968	SLAC (Stanford Linear Accelerator Center)	IBM 360/75 (high priority partition) After October 1968 on IBM 360/91	IBM 2250 scope with light pen

The special purpose systems STATPAC and Gear's system became operational in 1965 and 1966 respectively. Dixon's on-line statistical programs are partially completed and the PEG system is now operational.

A simple comparison of the languages of the general purpose systems is given in Table 3. The comparison is for a very simple problem, nevertheless it gives an indication of the type of commands (instructions) that are used with each system.

Some of the special purpose systems have languages associated with them. For example, Gear's differential equation system has a language which is used to enter the systems of equations to be solved and control the processing. On the other hand Dixon's statistical programs and the PEG system use lightpen selection of options rather than having user specification by some kind of command language.

In Table 4 we present a parallel between various coding systems used in batch processing and some of the interactive systems discussed in this survey. The purpose of this comparison is to show that just as there are various levels of programming used in batch processing, there are various levels of interaction (levels of console programming) used in on-line problem solving. The reason we have various levels of programming is not just due to the historical fact that machines were invented first and the languages have evolved from the basic machine instructions, but due to a need for communication with the machine at various levels.

We find that some problems are more efficiently solved by using a higher level language. For example, a one-shot problem is best coded in FORTRAN (for example) as opposed to assembler language because of the savings in programming time where as a code that is to be run several hundred times is

TABLE 3

A SIMPLE LANGUAGE COMPARISON

Coding the following problem:

assume: A and B are loaded

compute: $R \leftarrow \sqrt{A^2 + B^2}$

System	Code for $R \leftarrow \sqrt{A^2 + B^2}$
Culler-Fried	(each item as a button push) LOAD A MULT A STORE T LOAD B MULT B ADD T SQRT STORE R
TOC	LET X(A, B) = SQRT (A ** 2 + B * * 2) E X(A, B) STORE R
AMTRAN	Similar to Culler-Fried for pushbutton version
Lincoln Reckoner	MULT A A T MULT B B R ADD T R R SQRT R R
MAP	R = SQRTF (A * A + B * B)
NAPSS	R = √(A↑2 + B↑2)
POSE	R = SQRT (A * A + B * B)
OPS-3	SET R = SQRT (A * A + B * B)
JOSS	SET R = SQRT (A * 2 + B * 2)

TABLE 4

Compare Levels of Programming Languages

Batch Processing	On-line Interactive Processing
(1) Assembler Language	(a) "Interactive machine language programming" see Lampson (1965) (b) Many of the operations of the Culler-Fried system are similar to assembler Language instructions. (c) AMTRAN (similar to Culler-Fried)
(2) Procedural Language (e.g. FORTRAN ALGOL)	(a) JOSS (b) on-line interpreters such as QUICK-TRAN These languages provide on-line capability similar to FORTRAN and ALGOL (algebraic statements)
(3) Using a comprehensive library of subroutines (procedures). (e.g. coding input and output drivers for existing subroutines)	(a) MAP (b) NAPSS These systems have operators that are backed by comprehensive algorithms. The user languages are supposed to be quite "natural"
(4) Using "Canned" programs. (a user need only prepare data according to a pre-specified order and format)	(a) Dixon's system at UCLA (b) PEG These systems allow user interaction in a single problem area with no "language" involved.

better coded in assembler language because the object code can be made more efficient thereby saving considerable machine time. As another example, consider the social scientist who wishes to perform a standard statistical analysis on some data but he knows no FORTRAN: not even enough to code a driver for an existing library subroutine (procedure) which would perform the desired analysis. For this user, the "canned" program which includes input and output and only requires a user to prepare data in some standard fashion, is the appropriate language level. Results can be obtained with relatively little effort by the person who is interested in them without the need for an intermediary (a programmer) or the need for the social scientist to learn a programming language.

To conclude this survey, let us list with brief descriptions some references which are applicable in various areas of interactive graphical systems. The areas covered are:

(1) Aids to the implementation of interactive graphical systems.

(2) Interactive console developments.

(3) Extra added attractions-features to add to a "total" system.

(4) Future systems.

(5) On-line systems without mathematical capabilities.

(1) Aids to the implementation of interactive graphical systems

Feingold [1967] — Describes a language, PLANIT (Programming LANguage for Interaction and Teaching), developed at Systems Development Corporation, Santa Monica, California. PLANIT is written in the JOVIAL language and used on the IBM AN/FSQ-32 V computer.

Pankhurst [1968] Describes GULP (General Utility Language Processor), a compiler-complier for verbal and graphic languages. Written at the University of Cambridge, England, to fit in a small computer memory, a PDP-7.

Thomas [1967] Describes a system designed to provide a user-computer interface employing a model to represent drawing information in the computer. The system, GRASP (a GRAphic Service Program), was written at IBM, Kingston, New York, for system 360 computers with 2250 display units.

Hurwitz et al. [1967] Describes GRAF (GRaphic Additions to Fortran), a language which is Fortran plus some added statements to facilitate graphics programming. GRAF was written at IBM Los Angeles Scientific Center and the Health Sciences Computing Facilities, UCLA, to extend OS/360 Fortran IV (E level) and to operate with the IBM 2250 I. The programs described by Dixon [1967] were implemented using GRAF.

Newman [1968] Describes a system for developing graphical problem-oriented languages. The work was done at Harvard University and developed on a PDP-7 computer with a DEC 340 display.

Roberts [1966] Describes a Graphical Service System (GSS) with variable syntax being developed at Lincoln Laboratory, MIT.

Kulsrud [1968] — Describes a general purpose graphic language to handle generation and display as well as analysis of pictures. The language was developed at Yale University, New Haven, Connecticut.

(2) Interactive console developments

Christensen and Pinson [1967]

Kopel [1968]

Ninke [1968]

Barlett et al. [1968]

Ninke [1965] — These all describe some aspects of a comprehensive hardware setup at Bell Telephone Laboratories.

Lewin [1967] — Gives an introduction to the hardware and software aspects of the various types of computer graphic terminals available in 1967.

Rippy and Humphries [1965] — Describes a machine developed at NBS, Washington, D.C., as a research tool for the investigation of man-machine communication techniques involving CRT displays.

Lewin [1965] — Describes a tablet-type graphic input device designed to minimize the amount of associated circuitry, developed at the RCA Laboratories, Princeton, N.J.

Haring [1965] — Describes the "beam pen", a novel high-speed, input/output device for CRT display systems, developed at the Electronic Systems Laboratory, MIT.

Davis and Elliot [1964] Describe the RAND tablet, a device for man-machine graphical communication developed at the RAND Corporation, Santa Monica, Calif.

Stotz [1963] Discusses man-machine console facilities needed for computer-aided design and the display system being developed at MIT.

Gallenson [1967] Describes a graphic tablet display designed at System Development Corporation, Santa Monica, for use under time-sharing.

Machover [1967] A review of graphic CRT terminals commercially available with an attempt to clarify some commonly used display terms.

Mahan [1968] Gives a state-of-the-art survey of the data display field with a hardware orientation.

(3) Extra added attractions-features to add to a "total" system.

Mermelstein and Eyden [1964] Describe a system for the automatic on-line recognition of handwritten words developed at MIT.

Lee [1968]

Allen [1968] Describe some research into the problem of machine-to-man communication by speech done at MIT.

Feldman [1968] Describes some research into the problem of computer input of forms done at the Walter Reed Army Institute of Research, Washington, D.C.

Teixeria and Sallen [1968] Describe the Sylvania data tablet: a new approach to graphic data input. This device was developed at Sylvania Electronic Systems, Waltham, Mass.

Smura [1968] — Discusses a new approach to hardware and software for graphical data processing that could replace, in some cases, present-day peripheral devices.

Walter [1965] — Discusses the use of color in an on-line graphical environment.

Tobey [1966]

Sammet and Bond [1964] — Give an introduction to FORMAC, a programming system designed to permit the manipulation of mathematical expressions.

Hearn [1967] — Describes REDUCE, a user oriented interactive system for algebraic simplification developed at Stanford University.

Klerer and May [1964], [1965a], [1965b]

Klerer and Grossman [1967] — Describe a system using two-dimensional input-output by typewriter terminals developed at Columbia University, Hudson Laboratories.

(4) Future systems

Licklider [1965] — Describes man-computer interaction and its promise. A section titled "My partner – the machine" describes an interactive system he envisions.

Sutherland [1965] — Describes the attributes of "the ultimate display" using many already developed characteristics.

(5) On-line systems without mathematical capabilities (these are included for the sake of completeness)

Lewis [1968]	Describes SHAPESHIFTER, an interactive program for experimenting with complex-plane transformations developed at the National Institutes of Health, Bethesda, Maryland.
Lampson [1965]	Describes a system allowing interactive machine language programming developed at the University of California, Berkeley.
Sutherland [1963]	Describes SKETCHPAD, a man-machine graphical communications system developed at MIT.
Johnson [1963]	Describes SKETCHPAD-III, a computer program for drawing in three dimensions developed at MIT.
Colin [1966]	Describes a simple program for use in the conversational mode, written at the University of Lancaster, England, to gain experience in man-machine communication.
Dunn and Morrissey [1964]	Describe an experimental system for remote computing using conversation source-language debugging techniques developed at the IBM Development Laboratory, New York, N. Y.
Abraham et al. [1968]	Describe an on-line multiprocessing interactive computer system for neurophysiological investigations developed at the UCLA Brain Research Institute.

REFERENCES

Abraham, F. D., Betyar, L., and Johnston, R. [1968]. An on-line multiprocessing interactive computer system for neurophysiological investigations. 1968 Spring Joint Computer Conference, Thompson Books, Washington, D.C., 345-352.

Allen, J. [1968]. Machine-to-man communication by speech, part II: Synthesis of prosodic features of speech by rule. 1968 Spring Joint Computer Conference, Thompson Books, Washington, D.C., 339-344.

Allen, T. R. and Foote, J. E. [1964]. Input/output software capability for a man-machine communication and image processing system. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 387-396.

Anderson G. B., Bertran, K. R., Conn, R. W., Malmquist, K. O., Millstein, R. E., and Tokubo, S. [1968]. Design of a time-sharing system allowing interactive graphics. Proceedings — 1968 ACM National Conference, Brandon/Systems Press, Princeton, N. J., 1-6.

Ball, G. H. and Hall, D. J. [1967a]. PROMENADE, an on-line pattern recognition system. SRI Technical Report No. RADC-TR-67-310 (September).

Ball, G. H. and Hall, D. J. [1967b]. A clustering technique for summarizing multivariate data. Behavioral Sciences, 12, No. 2, (March), 153-155.

Barlett, W. S., Busch, K. J. Flynn, M. L., and Salmon, R. L. [1968]. SIGHT, a satellite interactive graphic terminal. Proceedings — 1968 ACM National Conference, Brandon/Systems Press, Princeton, N. J., 499-509.

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.

Bryan, G. E., 1967. JOSS: 20,000 hours at the console — a statistical summary. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 769-777.

Cameron, S. H., Ewing, D., and Liverright, M. [1967]. DIALØG: a conversational programming system with a graphical orientation. Communications of the ACM, Vol. 10, No. 6, (June), 349-357.

Chasen, S. H. [1965]. The introduction of man-computer graphics into the aerospace industry. 1965 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 883-892.

Christensen, C. and Pinson, E. N. [1967]. Multi-function graphics for a large computer system. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 697-711.

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.

Cole, P. M., Dorn, P. H., and Lewis, C. R. [1964]. Operational software in a disk oriented system. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 351-362.

Colin, A. J. T. [1966]. A simple program for use in the "conversational mode". The Computer Journal, Vol. 9, 238-241.

Conn, R. W. and vonHoldt, R. E. [1965]. An online display for the study of approximating functions. Communications 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.

Culler, G. J. and Huff, R. W. [1962]. Solution of nonlinear integral equations using on-line computer control. 1962 Spring Joint Computer Conference, National Press, Palo Alto, 129-138.

Davis, M. R. and Ellis, T. O. [1964]. The RAND Tablet: a man-machine graphical communication device. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 325-331.

deMaine, P.A.D. [1965]. The self-judgment method of curve fitting. Communications of the ACM, Vol. 8, No. 8, (August), 518-526.

Dixon, W. J. (Ed.) [1964]. BMD, Biomedical Computer Programs. U. of California, Los Angeles.

Dixon, W. J. [1967]. Use of displays with packaged statistical programs. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 481-484.

Dunn, T. M. and Morrissey, J. H. [1964]. Remote computing an experimental system. Part 1: external specifications. 1964 Spring Joint Computer Conference, Spartan Books, Washington, D.C., 413-423.

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.

Eusebio, J. W. and Ball, G. H., [1968]. ISODATA-LINES — A program for describing multivariate data by piecewise-linear curves. Proceedings of International Conference on Systems Science and Cybernetics, University of Hawaii, Honolulu, Hawaii, (January), 560-563.

Feingold, S. L. [1967]. PLANIT — A flexible language designed for computer-human interaction. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 545-552.

Feldman, A. [1968]. Computer input of forms. 1968 Spring Joint Computer Conference, Thompson Books, Washington, D.C., 323-331.

Fried, B. [1967]. On the user's point of view. Presented at the ACM Symposium for Experimental Applied Mathematics, August 26-28, 1967 (Proceedings in Press).

Gallenson, L. [1967]. A graphic tablet display console for use under time-sharing. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 689-695.

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.

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.

Hargreaves, B., Joyce, J. D., and Cole, G. L. [1964]. Image processing hardware for a man-machine graphical communication system. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 363-386.

Haring, D. R. [1965]. The beam pen: A novel high speed, input/output device for cathode-ray-tube display systems. 1965 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 847-855.

Hearn, A. C. [1967]. REDUCE — a user oriented interactive system for algebraic simplification. Presented at the ACM Symposium for Experimental Applied Mathematics, August 26-28, 1967 (Proceedings in Press).

Hurwitz, A., Citron, J. P., and Yeaton, J. B. [1967]. GRAF: Graphic additions to FORTRAN. 1967 Spring Joint Computer Conference, Thompson Book Company, Thompson Books, Washington, D.C., 553-557.

Jacks, E. L. [1964]. A laboratory for the study of graphical man-machine communication. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 343-350.

Johnson, T. E. [1963]. Sketchpad III: A computer program for drawing in three dimensions. 1963 Spring Joint Computer Conference, Spartan Books, Washington, D.C., 347-353.

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).

Karplus, W. J. (Ed.) [1967]. On-Line Computing, McGraw-Hill Book Co., New York.

Klerer, M. and Grossman, F. [1967]. Further advances in two-dimensional input-output by typewriter terminals. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 675-687.

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.

Klerer, M. and May, J. [1965a]. A user oriented programming language. The Computer Journal, Vol. 8, No. 3 (July), 103-109.

Klerer, M. and May, J. [1965b]. Two-dimensional programming. 1965 Fall Joint Computer Conference, Spartan Books, Washington D.C., 63-75.

Kopel, P. S. [1968]. Interactive Computer Graphics. Bell Laboratories RECORD, Vol. 46, No. 6 (June), 189-196.

Krull, F. N. and Foote, J. E. [1964]. A line scanning system controlled from an on-line console. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 397-410.

Kulsrud, H. E. [1968]. A general purpose graphic language. Communications of the ACM, Vol. 11, No. 4 (April), 247-254.

Lampson, B. W. [1965]. Interactive machine-language programming. 1965 Fall Joint Computer Conference, Part II, Spartan Books, Washington, D.C., 141-149.

Lee, F. F. [1968]. Machine-to-man communication by speech, Part I: generation of segmented phonemes from text. 1968 Spring Joint Computer Conference, Thompson Books, Washington, D.C., 333-338.

Lewin, M. H. [1965]. A magnetic device for computer graphic input. 1965 Fall Joint Computer Conference, Spartan Books, Washington, D.C., 831-838.

Lewin, M. H. [1967]. An introduction to computer graphic terminals. Proceedings of the IEEE, Vol. 55, No. 9 (September), 1544-1552.

Lewis, H. R. [1968]. SHAPESHIFTER: An interactive program for experimenting with complex-plane transformations. Proceedings — 1968 ACM National Conference, Brandon/Systems Press, Princeton, N. J., 717-724.

Licklider, J.C.R. [1965]. Man-computer partnership. Int. Sci. and Tech. (May), 18-26.

Licklider, J.C.R. and Clark, W. E. [1962]. On-line man-computer communication. 1962 Spring Joint Computer Conference, National Press, Palo Alto, 113-128.

Machover, C. [1967]. Graphic CRT terminals - characteristics of commercially availalbe equipment. 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C. 149-159.

Mahan, R. E. [1968]. A state-of-the-art survey of the data display field. AEC Research and Development Report No. BNWL-725, UC-2, Battelle Northwest Laboratory, Richland, Washington.

Mann, R. W. [1965]. Computer-aided design. Proceedings of the IFIP Congress 1965, Part II, 476.

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

Mermelstein, P. and Eyden, M. [1964]. A system for automatic recognition of handwritten words. 1964 Fall Joint Computer Conference, Spartan Books, Washington, D. C. 333-342.

Moore, D. W. G. and Erickson, M. J. [1966]. The display as a research tool. Preceedings of Australia Computer Conference. Canberra (May), 16/1/1 - 16/1/4.

Moore, D. W. G., Jarvis, C. L. and Nicholls, I. G. [1966]. User efficiency in a time shared environments. Proceedings of Australia Computer Conference. Canberra (May), 11/3/1 - 11/3/4.

Newman, W. M. [1966]. An experimental program for architectural design. The Computer Journal, Vol. 9, No. 1, 21-26.

Newman, W. M. [1968]. A system for interactive graphical programming. 1968 Spring Joint Computer Conference, Thompson Books, Washington, D. C., 47-54.

Ninke, W. H. [1965]. GRAPHIC 1 - a remote graphical display console system. 1965 Fall Joint Computer Conference, Spartan Books, Washington, D. C., 839-846.

Ninke, W. H. [1968]. The growth of computer graphics at Bell Laboratories. Bell Laboratories RECORD, Vol. 46, No. 6 (June), 180-188.

Pankhurst, R. J. [1968]. GULP—a compiler-compiler for verbal and graphic languages. Proceedings—1968 ACM National Conference, Brandon/Systems Press, Princeton, N. J., 405-421.

Pyle, I. C. [1965]. Data input by question and answer. Communications of the ACM, vol. 8, No. 4 (April), 223-226.

Reinfelds, J., Flenker, L. A., and Seitz, R. N. [1966]. AMTRAN, a remote-terminal, conversational-mode computer system. Proceedings 1966 ACM National Conference, Thompson Book Company, Washington, D. C., 469-477.

Rice, J. R. [1967]. On the construction of polyalgorithms for automatic numerical analysis. Report No. CSD-TR-10, Computer Sciences Department, Purdue University (June).

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

Rippy, D. E. and Humphries, D. E. [1965]. MAGIC — a machine for automatic graphics interface to a computer. 1965 Fall Joint Computer Conference, Spartan Books, Washington, D. C., 819-830.

Roberts, L. G. [1966]. A graphical service system with variable syntax. Communications of the ACM, Vol. 9, No. 3 (March), 173-175.

Roman, R. V. and Symes, L. R. [1967a]. Syntactic and semantic description of the numerical analysis programming language (NAPSS). Report No. CSD-TR-11, Computer Sciences Department, Purdue University (May).

Roman, R. V. and Symes, L. R. [1967b]. Implementation considerations in a numerical analysis problem solving system. Presented at the ACM Symposium for Experimental Applied Mathematics, August 26-28, 1967 (Proceedings in Press).

Roos, D. [1965]. An integrated computer system for engineering problem solving. 1965 Fall Joint Computer Conference, Part II, Spartan Books, Washington, D. C., 151-159.

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.

Sammet, J. E. and Bond, E. R. [1964]. Introudction to FORMAC. IEEE Trans. on Electronic Computers, (August), 386-394.

Sashkin, L., Schlesinger, S., and Reed, K. [1967]. Two analyist-oriented computer languages: EASL and POSE. Report No. ATR-68 (S8111)-2, Aerospace Corporation, San Bernardino, California (November).

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.

Schwartz, J. I., Coffman, E. G., and Weissman, C. [1964]. A general-purpose time-sharing system. 1964 Spring Joint Computer Conference, Spartan Books, Washington, D. C., 397-411.

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.

Shaw, J. C. [1965]. JOSS: Conversations with the Johnniac open-shop system. Proceedings of the IFIP Congress 65, Part II. 544-545.

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. [1969]. The use of man-machine interaction in data-fitting problems. Ph.D.dissertation, Computer Science Department, Stanford University.

Smura, E. J. [1968]. Graphical data processing. 1968 Spring Joint Computer Conference, Thomspon Books, Washington, D. C., 111-118

Stotz, R. [1963]. Man-machine console facilities for computer-aided design. 1963 Spring Joint Computer Conference, Spartan Books, Washington, D. C., 323-328.

Stowe, A. N., Wiesen, R. A., Yntema, 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.

Sutherland, I. E. [1963]. Sketchpad: A man-machine graphical communication system. 1963 Spring Joint Computer Conference, Spartan Books, Washington, D. C., 329-346.

Sutherland, I. E., [1965]. The ultimate display. Proceedings of the IFIP Congress 1965, Part II.

Symes, L. R. [1968]. Private communication.

Symes, L. R. and Roman, R. V. [1967]. Structure of a language for a numerical analysis problem solving system. Report No. CSD-TR-12, Computer Sciences Department, Purdue University.

Teixeira, J. F. and Sallen, R. P. [1968]. The Sylvania data tablet: A new approach to graphic data input. 1968 Spring Joint Computer Conference, Thompson Books, Washington, D.C., 315-321.

Thomas, E. M. [1967]. GRASP — a graphic service program. Proceedings — 1967 ACM National Conference, Thompson Book Company, Washington, D.C., 395-402.

Tobey, R. G. [1966]. Eliminating monotonous mathematics with FORMAC. Communications of the ACM, Vol. 9, No. 10 (October), 742-751.

Uncapher, K. [1965]. The man-machine interface. 1965 Fall Joint Computer Conference, Part II, Spartan Books, Washington, D.C., 88-91.

Walter, C. M. [1965]. Color — a new dimension in man-machine graphics. Proceedings of the IFIP Congress 1965, Part II.

Ward, J. E. [1965]. Display systems research, Project MAC Report: Progress to July 1964, MIT.

Whiteman, I. R. [1966]. New computer languages. Int. Sci. and Tech. (April), 62-68.

Wiesen, R. A., Yntema, D. B., Forgie, J. W., and Stowe, A. N. [1967]. Coherent programming in the Lincoln Reckoner. Presented at the ACM Symposium for Experimental Applied Mathematics, August 26-28, 1967 (Proceedings in Press).

Winiecki, K., Editor [1966]. Culler on-line system users manual. Harvard University Computation Laboratory, Cambridge, Massachusetts.

Wood, L. H., Reinfelds, J., Seitz, R. N., and Clem, P. L., Jr. [1966]. The AMTRAN system. DATAMATIØN (October), 22-27.

Yershov, A. P. [1965]. One view of man-machine interaction. Communications of the ACM, Vol. 12, No. 3 (July), 315-325.