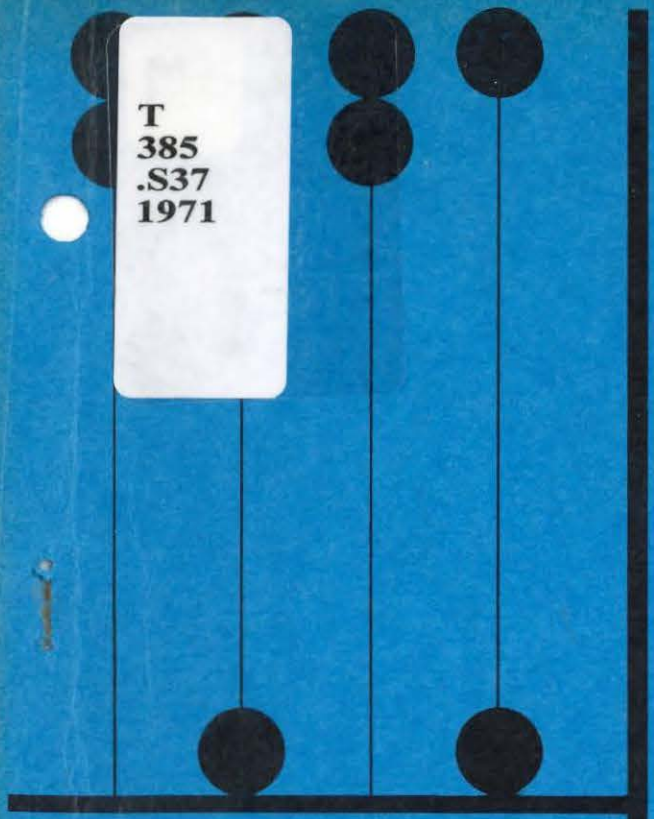
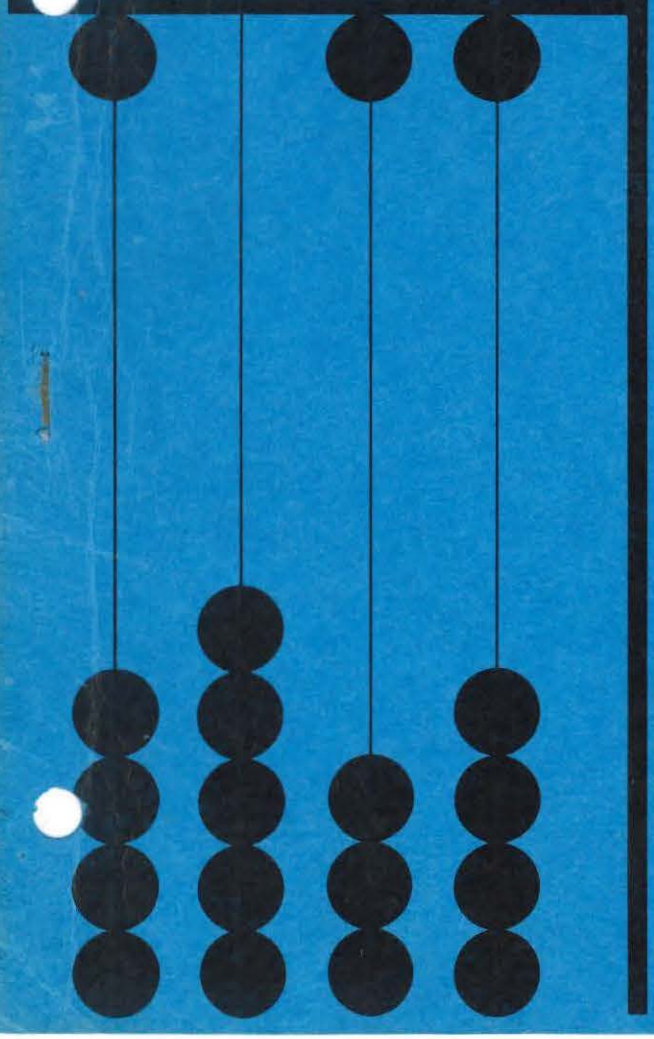


COMPUTATION CENTER
Iowa State University



May 1976
Document No. 4
Revision No. 3

The Use of SIMPLOTTER, A High Level Plotting System

D. G. Scranton
E. G. Manchester

LEGAL NOTICE

TABLE OF CONTENTS

THE USE OF SIMPLOTTER, A HIGH LEVEL PLOTTING SYSTEM

by
D. G. Scranton and E. G. Manchester

ABSTRACT

SIMPLOTTER is a high level plotting system used for producing typical graphs quickly and easily. A typical graph might be composed of several curves, numbered axes, and a small amount of labeling. The system is designed for the needs of the casual computer user and an environment of program experimentation and change. A set of data is graphed by specifying options when "calling" a problem-oriented plotting routine. Graphing options include: points, lines, interpolated or smoothed curves, superposition of additional data sets, automatic or programmer specified scaling, linear or logarithmic axes, labeling, histograms and ideograms. Automatic scaling is based on all data sets of a graph. Plots can be generated from PL/1(F), FORTRAN IV (G, H, and WATFIV) and COBOL using a common plotting concept. User program logic and plotting logic can be debugged in parallel. SIMPLOTTER is written in FORTRAN IV. SIMPLOTTER enters core storage upon completion of the user program execution. The programmer can direct plots to either a printer or an incremental plotter. Graphs drawn on the incremental plotter are suitable for publication.

CHAPTER I: INTRODUCTION TO SIMPLOTTER

The Need for a High Level Plotting System

Historically, the generation of graphs from digital computers has presented nagging problems for programmers. Often a disproportionate amount of time is spent merging plotting logic into an existing program. In some cases the merge is extremely difficult as the more common plotting routines impose a specific logic structure on the user program. Because of the interaction between the plotting routines and the user program, debugging time increases significantly. Furthermore, after plotting has been successfully merged into a program, the plotting logic is so entwined in the program logic that even a minor change becomes difficult to implement. Many managers and programmers have found that to change even the length of an axis requires a major re-writing of the program logic. Managers uniformly complain of duplication of effort after surveying completed programs. Most plotting packages presently available not only influence the logic of the user program but also dictate the computer language the programmer must use (usually FORTRAN). These plotting routines generally use a significant portion of the user region of core storage, placing further restrictions on the user's program design.

The basic problems are twofold: plotting routines are usually supplied by the manufacturers of digital plotters and are organized into the basic elements of plotting, such as a routine to

generate an axis, a routine to connect a set of points by straight lines, a routine to compute a scale factor for subsequent use by the user program, etc.. Consequently, plotting logic is scattered throughout the program, making seemingly simple modifications difficult to implement.

Secondly, scale factors must be determined and applied by the user to his internal data (usually plotting routines demand data in units of inches). Thus, the user modifies his data arrays based on the scale factor value provided by a plotting routine. Frequently this kind of interaction causes debugging problems. In addition, his arrays have been modified. If he needs to continue using them, they must be restored or a copy of the original data re-loaded. If the user wishes to plot several sets of data on the same graph (superposition), he must determine a scale factor based on all data arrays. The programmer can see that this is a jumping-off place where bookkeeping, core storage considerations, and other trivia start to influence the design of the user program--a situation to be avoided if possible.

Scope and Application Area of SIMPLOTTER

SIMPLOTTER is characterized by the following:

1. Plotting concepts are straight-forward and easy to use.
2. Programs using SIMPLOTTER are easily modified.
3. Plots can be easily directed to either a printer plotter or an incremental plotter (output suitable

for publication) without program recompilation.

4. Debugging plot logic and user program logic can be done simultaneously (in parallel).
5. The same plotting concepts can be used from PL/1(F), FORTRAN IV (G,H, and WATFIV), and COBOL.

The system is designed to generate typical graphs quickly and easily. A typical graph might be considered to be composed of several curves, numbered axes, and some labels. SIMPLOTTER minimizes the above problems for a large class of users, namely, those typical of the scientific community. The prime design criteria is to allow the casual user a simple but economical means of graphing. Programmers with "one shot" or rapidly changing programs will find SIMPLOTTER well suited to their needs. Graphs drawn on the incremental plotter are suitable for publication. A standard graph format is assumed. Axes are drawn at 0 and 90 degrees to the horizontal and are ticked and numbered (scientific notation) at every inch with numbers increasing uniformly. In cases where a logarithmic axis is opted, only an integral number of logarithmic cycles are drawn. Axis labeling areas are provided adjacent to the axes, and graph labeling is normally done in the upper right-hand corner of the graph. This format is commonly seen in journals of physics, engineering, applied mathematics, and statistics.

Although this assumed format can be altered somewhat by the user, the programming trivia increases accordingly. For extreme

deviations from the assumed format, the trivia can be nearly as great as when using the manufacturer's routines.

The FORTRAN version of SIMPLOTTER has been in use at I.S.U. since 1966 and has survived three computer exchanges (IBM 7074 to IBM 360/40 to 360/50 to 360/65). Within three months after its creation and implementation, SIMPLOTTER was generating 96 percent of all plot jobs handled by our digital plotter. This percentage has remained constant. Generally, persons using the manufacturer's package are drawing 3-dimensional and contour graphs, applications out of the scope of SIMPLOTTER. Programmers are encouraged to use the printer plotter initially. All plots at I.S.U. are directed to the printer by default. If the user chooses to use the incremental plotter, he need change only one job control card. Note that no change is made in his source program logic to direct the plots to the higher resolution incremental plotter. Although it is intended primarily for debugging, the printer plotter has resolution adequate for many applications and is a boon to turnaround time.

SIMPLOTTER Plotting Concepts

Restrictions on the user's logic structure are kept at an absolute minimum. The user sets up arrays containing the (x,y) points to be plotted (in arbitrary units, not necessarily inches). He CALLs one of SIMPLOTTER's problem-oriented routines and

specifies the options to be applied to his data. At this point the user may consider the data as having been plotted, and can modify, re-use, or destroy the data arrays as needed. The length of the x-axis, called XSIZE, is also specified in the CALL list. If XSIZE is non-zero, a new graph is begun. This type of CALL is referred to as a primary CALL. Later in the program the user may wish to superimpose a second data set onto the graph. He forms the (x,y) point arrays, specifies the options to be applied to that data, and specifies XSIZE as zero. This is referred to as a superposition CALL. To summarize:

XSIZE≠0.0	primary CALL	a new graph, labeled and with axes, is begun.
XSIZE=0.0	superposition CALL	a data set is superimposed on the previously formed graph.

A second method of specifying superposition, a special superposition routine with a shortened CALL list, is provided for convenience also. As many as 99 data set superpositions can be made on one graph, but no more than a total of 5000 points can be plotted on one graph. There is no inherent limit to the number of graphs that can be drawn.

This concept is simple enough that many users miss some important implications. To be more specific:

1. If only one data set is to be plotted on a graph, one CALL (XSIZE ≠ 0.0) is sufficient to plot the complete graph.

2. After the CALL is executed, the user need not maintain the data arrays. He can consider the data as having been plotted.

3. When superimposing (XSIZE = 0.0):

   a. Different options may be specified for each data set being superimposed (points only, smoothed curve, line, etc.)

   b. If automatic scaling is opted, scaling (user data units per inch) is determined by SIMPLOTTER on the basis of all data sets for the graph to be plotted, that is, the primary data set and all the superposition data sets. This bookkeeping convenience is probably the strongest feature of the SIMPLOTTER plotting system. Automatic scaling is an option; however, the user may specify the scaling which is applied to all data (primary and superposition) appearing on a graph.

4. None of the parameters specified in the CALL list are modified by the problem-oriented routines (including the (x,y) data arrays to be plotted).

5. Debugging user program and plotting logic is done in parallel. Because of (4) there is little interaction between the two logics. The user may well make an error in his logic which will lead to an incorrect graph. However, merely calling one of the problem-oriented plotting

routines will not cause his program to terminate abnormally. Thus, his program will continue beyond the plotting error to either normal termination or until a user program error is encountered. After his program has terminated, he will see that the generated graph is incorrect. Aided by SIMPLOTTER's Parameter Dump, he will see precisely what parameters he passed to the problem-oriented plotting routine. This allows him to correct, simultaneously, his program and plotting logic, thus, considerably reducing debugging time.

6. Debugging turnaround time is no longer when plotting than when not plotting. The printer plotter provides graphs connected to the end of the user's normal printed output; there is no waiting for the separate high resolution graph drawn by the incremental plotter.
7. Many options are easily available to the SIMPLOTTER user which he ordinarily would not take time to program himself, such as smoothing and second order Lagrangian interpolation. This avoids duplication of effort and excessive debugging time.

Overview of SIMPLOTTER Problem-Oriented Routines

The main body of this manual describes the problem-oriented routines. Most users are not expected to require assistance in using the SIMPLOTTER system. The description of the routines is mainly tutorial, and without reference to a specific programming language. It is hoped that the descriptions are brief enough that the manual can also be used for reference. The most general of the routines, GRAPH, is described first. Most users will not need to read beyond that section. Special purpose routines are described following GRAPH. To summarize the problem-oriented routines' purposes:

GRAPH is the most general of SIMPLOTTER's problem-oriented routines. Linear, log-log, or semi-log axes can be specified. Scaling is automatic or can be specified. Points can be 1) plotted with a variety of symbols, 2) connected by straight lines or an interpolated curve, or 3) represented by a smoothed curve. Axes are generated and labeling specified by the user is drawn.

DISTRI, DISTRA, and DISTRP provide all bookkeeping and calculations necessary for the construction of histograms (bar graphs representing a distribution of events) and ideograms (distribution of events each having a unique inherent error).

ORIGIN provides the user a method of altering the basic

format of a graph. The user may take the responsibility of moving the origin from graph to graph (a way of putting one graph on top of another), eliminating the drawing of an axis, etc.

LETTER allows the user to letter a graph in addition to the labeling that is provided by the other routines. In particular, this routine is useful for constructing special purpose axes and labeling particular data points.

These routines may be used in various combinations to produce a graph. For instance, a curve may be generated by GRAPH, a histogram superimposed by DISTRP, additional lettering superimposed by LETTER, etc...

The usage of each routine is described without reference to a specific programming language. Sample programs written in FORTRAN, PL/1, and COBOL, are shown in Appendix H. The user is encouraged to pick a sample program similar to his problem and work through it. Programming hints and conventions peculiar to the various languages are illustrated in sample programs.

Conventions Followed in this Manual

SIMPLOTTER's problem-oriented routines follow the conventions and defaults of the programming language being used. Thus, in all the programming languages the number and type of parameters specified in the CALL list of a problem-oriented routine must correspond to the number and type the routine expects. For example,

GRAPH expects 15 parameters; the user must supply 15 parameters in the specified order when calling GRAPH.

Naming conventions used in this manual follow those of the major programming languages. Character strings, used for labeling purposes, are explicitly specified in the parameter definition of each problem-oriented routine. Variables beginning with letters I - N are implicitly intended to be integer type, all others are of real type. "Integer" and "real" translate into each language's data attributes as follows:

Data Type	FORTRAN	PL/1	COBOL
Integer	INTEGER*4	FIXED BINARY(15)	COMPUTATIONAL
Real	REAL*4	FLOAT DECIMAL(6)	S(9) COMPUTATIONAL-1

When XSIZE = 0.0 in any CALL list, SIMPLOTTER handles the set of data in that CALL list as a superposition upon the last primary data set graphed. In this case, many of the parameters in the list are not used or referred to by the problem-oriented plotting routine (such as scaling and some labels). However, as the number, type, and order of the parameters in the CALL list are very important, all positions in the CALL list must be occupied even if not used. The user therefore must "pad" his call list so that all positions are occupied by parameters of the correct type and length. As previously mentioned, special routines are provided for superimposing data which minimize the need for 'padding'.

Using SIMPLOTTER for the First Time

Those who have not previously used SIMPLOTTER should follow the general instructions below:

1. Read the section "SIMPLOTTER Plotting Concepts" to get an overview of the plotting concepts.
2. Choose an appropriate problem-oriented routine and read its sections on "Purpose" and "Usage".
3. Work through one of the sample programs written in the programming language you choose to use. Note the associated explanations, as they point out hints, peculiarities, and the job control in the chosen programming language.
4. Use the automatic scaling feature when plotting data sets for the first time. Automatic scaling will insure that all data is plotted on the graph, which gives insight to debugging problems and optimum scaling parameters. If the scaling is not satisfactory, specify scaling directly and/or change the axis length on subsequent runs.
5. When debugging, always check the SIMPLOTTER Parameter Dump to verify that the intended parameters were passed to the problem-oriented routine. Always use the printer-plotter when debugging; it improves your turnaround time considerably. Switch to the incremental plotter only when your program is bug free.

CHAPTER II: GRAPH - THE GENERAL PURPOSE ROUTINE

GRAPH is the most general of SIMPLOTTER's problem-oriented routines. Linear, log-log, or semi-log axes can be specified. Data points can be: 1) plotted with a variety of symbols, 2) connected by straight lines or an interpolated curve, or 3) represented by a smoothed curve. Axes are generated and labeling specified by the user is drawn.

Usage of GRAPH

The user forms two arrays containing the X and Y coordinates of the points to be plotted. The names of these arrays, the number of points to be plotted, and other graphical information is passed to GRAPH through a subroutine CALL list. Included in this graphical information is the length of the X-axis. When this length is non-zero, the data set points in the arrays will be plotted on a new graph; that is, the origin is moved to a point such that the new graph will not overlay any portion of the previous one. If in the CALL list of GRAPH, the X-axis length is specified as zero, the (X,Y) data set points will be superimposed upon the graph of the previous data set. A second method of implying superposition is provided by subroutine GRAPHS, which has fewer arguments than does GRAPH. Scaling is calculated and applied on the basis of all points on a graph, primary and superposition.

CALL_List_Format

```
CALL   GRAPH   (NPTS, X, Y, ISYM, MODE, XSIZE, YSIZE,
       XSF, XMIN, YSF, YMIN, XLAB, YLAB, GLAB, DATLAB)

CALL GRAPHS (NPTS, X, Y, ISYM, MODE, DATLAB)
```

Note: Except for character strings, all parameters in the list beginning with characters I-N are intended to be integer type variables or constants. All others are intended to be of real type.

Parameter_Definition

A. Information about the data points: NPTS, X, Y.

NPTS - an integer type variable or constant that represents the number of (X,Y) data set points to be plotted.

X - the name of the real type array containing the X-coordinates of the points of this data set.

Y - the name of the real type array containing the Y-coordinates of the points of this data set.

FORTRAN NOTE: In FORTRAN G, H, and WATFIV, NPTS may be negative. The magnitude of NPTS then represents the number of points to be plotted. The sign of NPTS specifies precision as follows:

NPTS positive: X and Y contain single precision numbers.

NPTS negative: X and Y contain double precision numbers.

Both X and Y must be of the same precision.

B. Information about the plotting method: ISYM, MODE.

ISYM - the integer type variable or constant that represents one of the thirteen plot-centered symbols to be used (if any). See Appendix C for a list of available symbols.

MODE - specifies the plotting method to be used. Several terms used to explain plotting methods pertaining to curves are explained below.

Interpolated Curve: A curve is drawn connecting and passing through each of the data points sequentially, that is, points are ordered in the arrays. Since the curve passes through every point, interpolation is intended for use with well-behaved continuous points. Interpolation can result in a multi-valued curve, such as a circle.

Note: interpolation is effective only on data sets of more than 5 points.

Smoothed Curve: A smoothed curve attempts to define the tendency of the data set points by passing through a "neighborhood" of points with a small amount of scatter. Smoothing can only produce single-valued curves which

are monotone increasing in the X direction.

Note: smoothing is effective only on data sets of more than 20 points.

Example: For purposes of illustration, points are numbered to indicate their sequence in the (X,Y) arrays.

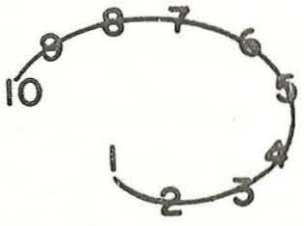
Interpolated Curve

Smoothed Curve

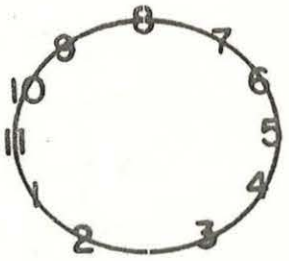
Open Curve: The interpolated curve is open if drawn such that the first and last points are not connected.

Closed Curve: A closed curve is a multi-valued curve which is drawn such that all data points are connected (including a connecting line between the first and last points).

Example:



Open Curve



Closed Curve

Interpolation and smoothing methods are described in Appendix G. The plotting method MODE must be an integer type variable or constant. The following values will determine the corresponding plotting method:

MODE = 1 points plotted with symbol ISYM and connected by an open interpolated curve.

= 2 open interpolated curve is drawn from point to point but points are not plotted with a symbol.

= 3 points plotted with symbol ISYM and connected by straight lines drawn from point to point.

= 4 straight lines are drawn from point to point but the points are not plotted with a symbol.

= 5 points plotted with symbol ISYM and connected by a closed interpolated curve.

= 6 closed interpolated curve is drawn from point to point but points are not plotted with a symbol.

= 7 points plotted with symbol ISYM.

= 11-20 original points are plotted with symbol ISYM and a smoothed curve is drawn. MODE = 11 specifies the lowest level of smoothing. Higher values of MODE provide a greater degree of smoothing.

= 21-30 points are not plotted, but a smoothed curve is drawn with smoothing increasing as MODE is increased toward 30.

NOTE: Because excessive smoothing can conceal the original structure of the data and because smoothing is a relatively expensive operation, the lowest level of smoothing (MODE = 11 or 21) is recommended as an initial try. See Appendix G for more information.

NOTE: For more information on other uses of MODE, see "Automatic Data and Label Tagging" (page 28) and "Automatic Data Point Elimination" (page 29).

C. Axes and implying superposition: XSIZE,YSIZE

XSIZE - is a real type variable or constant which defines the length of the horizontal axis in inches. The linear or log axis option is specified by the sign of XSIZE while superposition is specified by XSIZE as zero. Linear and logarithmic axes are illustrated in Figures 1 and 2 respectively.

Positive: XSIZE specifies length of linear X-axis in inches and that a new graph is to be started.

Negative: |XSIZE| specifies length of logarithmic X-axis in inches and that a new graph is to be started.

Zero: specifies superposition of this data set upon the last primary data set's graph.

YSIZE - is a real type variable or constant which specifies the length of the vertical axis in inches. The linear or log option is specified by the sign of YSIZE.

Positive: YSIZE specifies length of linear Y-axis in inches.

Negative: |YSIZE| specifies length of logarithmic Y-axis in inches.

Zero: treated like XSIZE=0, to specify superposition of this data set upon the last primary data set's graph.

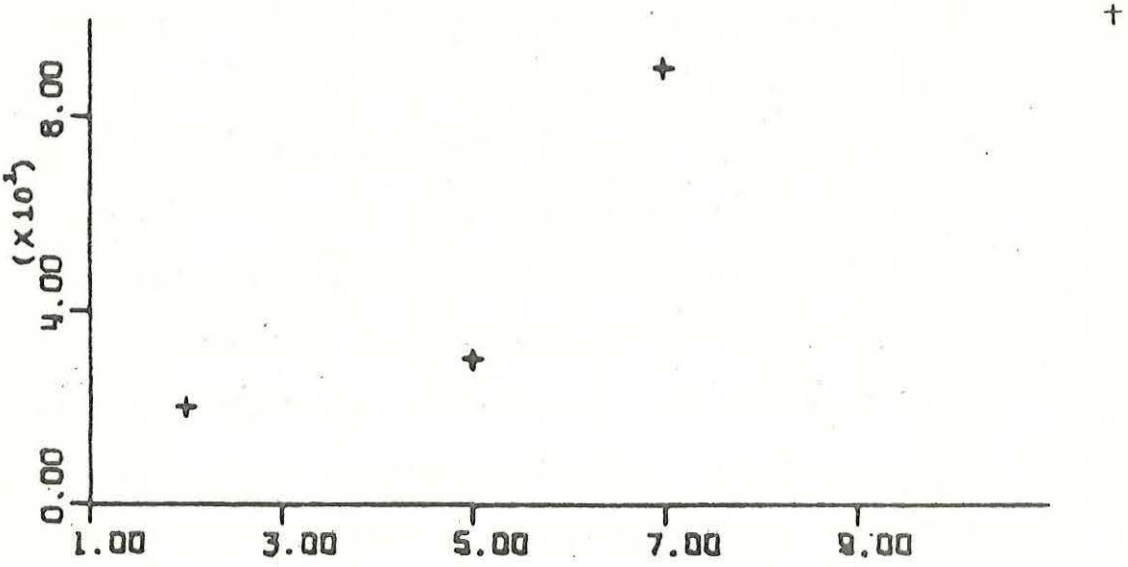


Figure 1. Generation of linear axes.

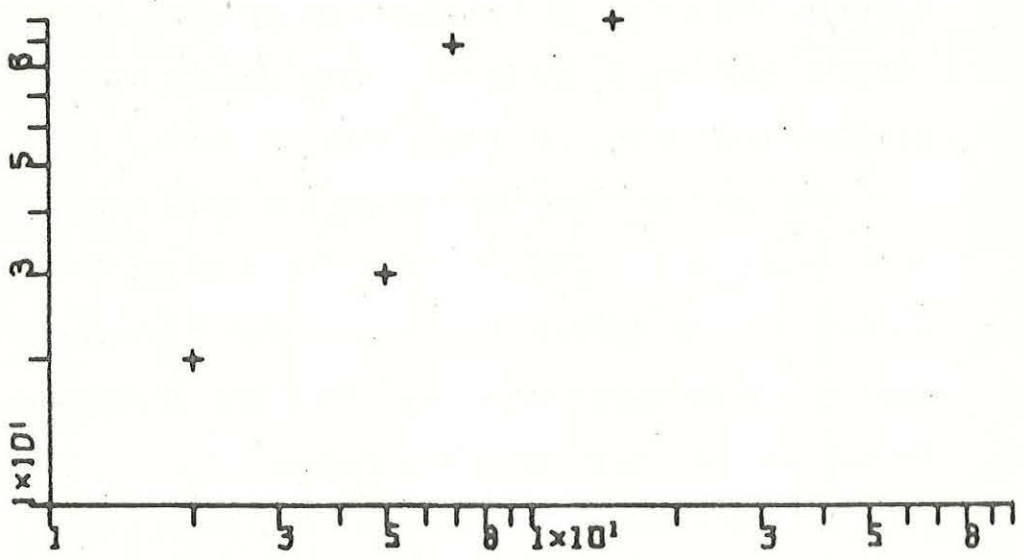


Figure 2. Generation of logarithmic axes.

D. Information about scaling: XSF, XMIN, YSF, YMIN.

The meaning of the scaling parameters (XSF, XMIN, YSF, YMIN) depends upon whether a linear or logarithmic axis is implied by the XSIZE and YSIZE parameters (see above). The horizontal or X-axis parameters are completely independent of the vertical or Y-axis parameters. Thus linear, semi-log, or log-log graphs can be drawn. For both linear and logarithmic axes, the user has the option of specifying the scaling factors or letting SIMPLOTTER determine the necessary scaling factors. If SIMPLOTTER chooses the scaling, all points in the X and Y arrays will be plotted within the specified axis lengths. If the user specifies scaling, some points may fall outside the range of the graph as defined by axis lengths and scale factors. These points will be plotted 1/2" beyond the extremes of either axis.

XSF and XMIN are explained for each case for the horizontal axis. YSF and YMIN are similar for the Y-direction. Note that SIMPLOTTER does the actual scaling internally; that is, the X and Y arrays formed by the user are not altered.

1. Producing linear axes (XSIZE positive)

The axis is ticked at every inch and a number printed in scientific notation (three

significant digits). The number is printed at each tic and depends upon the values of XSF and XMIN, as explained below.

XSF pos: SIMPLOTTER will use the value of XSF to scale data of this primary data set and all superposition data sets of this graph. XMIN will be the beginning point of the axis and will appear under the first tic. The following tics will be labeled with the values XSF+XMIN, 2*XSF+XMIN, 3*XSF+XMIN, and so on.

XSF zero: SIMPLOTTER will define the scale factor and the beginning point axis value for you. Determination of the scale factor is based upon consideration of all data sets to be plotted, primary and superposition. The scale factor and the beginning axis value are chosen so that visual interpolation between tics is easy. Specifically, 1) all data points will appear on the graph,

2) the beginning axis value is a multiple of the scale factor, 3) the smallest (X,Y) point will fall into the first inch of the graph. Only "nice" numbers, such as ± (1.0, 2.0, 4.0, 5.0, and 8.0) times 10 raised to an integer power, are chosen as scale factors.

Example: The following parameters passed to GRAPH would produce the graph shown in Figure 1.

NPTS = 4
ISYM = 3
MODE = 7

$$X = \begin{pmatrix} 2.0 \\ 5.0 \\ 7.0 \\ 15.0 \end{pmatrix} \qquad Y = \begin{pmatrix} 20.0 \\ 30.0 \\ 90.0 \\ 100.0 \end{pmatrix}$$

XSIZE = 5.0 XSF = 2.0 XMIN = 1.0
YSIZE = 2.5 YSF = 0.0 YMIN = 0.0

To summarize what these parameters indicate to SIMPLOTTER:

a. linear axes are to be drawn (XSIZE and YSIZE are positive).

b. the user specified the scaling for the x-axis (XSF = 2.0). Notice that one of the data points falls out of the range of the x-axis and is plotted on the 1/2 inch boundary beyond the end of the x-axis.

c. SIMPLOTTER is to scale the y-axis (YSF=0.0).

2. Producing logarithmic axes (XSIZE negative)

Only complete logarithmic cycles are drawn. The user must place logarithms of the coordinates (rather than the coordinates themselves) in the X array. That is, the GRAPH routine does not form the logarithms for the user. Two types of logarithmic arrays can be plotted, base 10 (common) logarithms and base e (natural) logarithms.

XSF defines the number of logarithmic cycles to be graphed. The sign of XSF specifies the type of logarithms the user has placed in the X array, while the magnitude specifies the number of cycles to be drawn. Figure 2 shows an example of the generation of logarithmic axes.

a. base 10 logarithms (XSF positive)

XSF = 0.5 means that the X array contains base 10 logarithmic values. SIMPLOTTER will determine the number of cycles needed to graph all points and the power of 10 to be printed as a label for the cycles.

XSF $\geq$ 1.0 means that the X array contains base 10 logarithmic values and the user requests that XSF cycles be drawn. XMIN is then considered to be the power of 10 to be printed as a label for the first cycle.

b. base e logarithms (XSF negative)

XSF = -0.5 means that the X array contains base e logarithmic values. SIMPLOTTER will determine the number of cycles needed to graph the points and the power of 10 to be printed as a label for the cycles.

XSF $\leq$ -1.0 means that the X array contains base e logarithmic values and the user requests |XSF| cycles. XMIN is then considered to be the power of 10 to be printed as a label of the first cycle.

Example: The following parameters passed to GRAPH would pro-

duce the graph shown in Figure 2.

NPTS=4

ISYM=3

MODE=7

$$X=\begin{pmatrix}0.693\\1.609\\1.946\\2.708\end{pmatrix} \qquad Y=\begin{pmatrix}1.301\\1.477\\1.954\\2.000\end{pmatrix}$$

XSIZE=-5.0 XSF=-0.5 XMIN=0.0

YSIZE=-2.5 YSF= 1.0 YMIN=1.0

Note that the X and Y arrays have been filled with the natural and common logarithms, respectively, of the X and Y arrays of the previous example. To summarize what these parameters indicate to SIMPLOTTER:

1. X-axis specification:
   a. logarithmic axis is to be drawn (XSIZE negative)
   b. X array indicated to contain natural logarithms (XSF negative)
   c. SIMPLOTTER is to determine the number of cycles drawn (XSF = -0.5)
2. Y-axis specifications:
   a. logarithmic axis to be drawn (YSIZE negative)
   b. Y array stated to contain common logarithms (YSF positive)
   c. the number of cycles and the power of ten of the initial cycle is specified by the user. (YSF=1.0, YMIN=1.0, means that one cycle is to be drawn, and 10^1 is to be printed as the label of the first cycle).

E. Information about labeling: XLAB,YLAB,GLAB,DATLAB

Axis labeling areas are located along each axis and a graph identification area is located in the upper right corner of a graph. If specified, each data set can have an identifying label also which is listed under the graph label. Graphs in Appendix H illustrate the areas where labels are drawn. When no labels are wanted, the label parameters must be specified as blank character strings. Figures 1 and 2 were produced with blank labels, for instance.

In most languages label string parameters may be specified in the CALL list by either variable names or by literal character strings. The label appearing under the X-axis, XLAB, will be explained in detail; the others are similar in form.

1. Labels as variable names

XLAB - the name of the character string of up to 20 characters which will be used to label the horizontal axis. The user can create the string in a variety of ways, several of which are shown in Appendix H. Note: semi-colons in labels should be used carefully, as semi-colons are interpreted as termination symbols for labels.

2. Labels as literals in the CALL list

XLAB may appear in the CALL list as a literal character string constant of up to 20 characters enclosed between quotes. If the literal is less than 20 characters long, the last character must be a semi-colon. If there are more than 20 characters in the label, only the first 20 will be drawn.

Note to COBOL users: most COBOL compilers do not support literals in CALL lists.

Example: The following CALL list shows the use of both variable names and character literals to specify labels:

```
CALL GRAPH (NPTS, X, Y, ISYM, MODE, XSIZE, YSIZE,
            XSF, XMIN, YSF, YMIN, XLAB, YLAB,
            'EXPERIMENTAL DATA #1','GRAPH ONE;')
```

The different label positions in the CALL list are:

XLAB - label for the horizontal axis.

YLAB - label for the vertical axis.

GLAB - label for graph identification. GLAB occupies one line and is positioned in the upper right hand corner with the first character beginning at (XSIZE-2.2, YSIZE-0.3) inches.

DATLAB - label to identify the data sets. It occupies one line per data set and is positioned immediately under GLAB or a previ-

ous DATLAB. DATLAB can be used to tag superposition data sets (see notes on "Data and Label Tagging" below).

Note that these standard labeling options (and other options as well) can be changed somewhat by the use of routine ORIGIN. The user is cautioned however, that graphing trivia becomes more abundant when the standard graph format is altered.

Automatic Data and Label Tagging

Often it is useful to associate a label with a set of superposition data points as well as with a primary data set. The DATLAB variable or literal of the primary data set CALL is always drawn regardless of the value of MODE. For superposition CALLs, DATLAB is drawn only when specified by a special value of MODE.

By specifying MODE between 100 and 130 in a superposition CALL containing a legitimate variable or literal in the DATLAB position of the CALL list, the following action will be taken by SIMPLOTTER:

1. the superposition data set indicated will be plotted according to the plotting method indicated by the last two digits of MODE.
2. the character string variable or literal in the DATLAB position of the CALL list will be printed immediately under the labels of previous data sets.

3. the symbol ISYM specified to plot the points of this data set will be printed just to the right of the data set label.

Automatic Data Point Elimination

Occasionally data sets contain points which the user would like to throw out at plotting time. This situation could arise, for instance, if data were being recorded by a faulty machine and the fifth piece of data of each record were known to be in error. In a case such as this the user would like to omit the plotting of the fifth piece of data of each record.

The user can omit erroneous data by giving MODE a negative value and setting either coordinate of the unwanted data equal to zero. If MODE is negative, all data points whose X or Y coordinates are zero, will be ignored by SIMPLOTTER at plotting time. All other points will be plotted as specified by the absolute value of the last two digits of MODE, as outlined in the Parameter Definition of MODE.

CHAPTER III: SPECIAL PURPOSE ROUTINES

DISTRI, DISTRA, DISTRP - Plotting distributions

These routines form and plot histograms and ideograms. A brief description of histograms and ideograms follows:

Histograms (bar graphs): used to show distributions of discrete events. Discrete events are treated as if they have no inherent error. The abscissa (usually the x-axis) is divided into a number of intervals (bars). When an event falls within the boundaries of a particular bar, the height of that bar is increased by a certain specified amount. No other bars are affected. A histogram might be used to show graphically the number of babies born each month (heights of 12 bars) with each month represented sequentially by one bar.

Ideograms: used to show distributions of events, each event having an inherent error associated. Such a situation commonly arises when difficult measurements, or other error inducing methods, are involved in obtaining data. The exact coordinate of an event is not known precisely in this case. However, the event's coordinate can usually be narrowed to some interval surrounding a most probable value.

This type of data can be stated in the form:

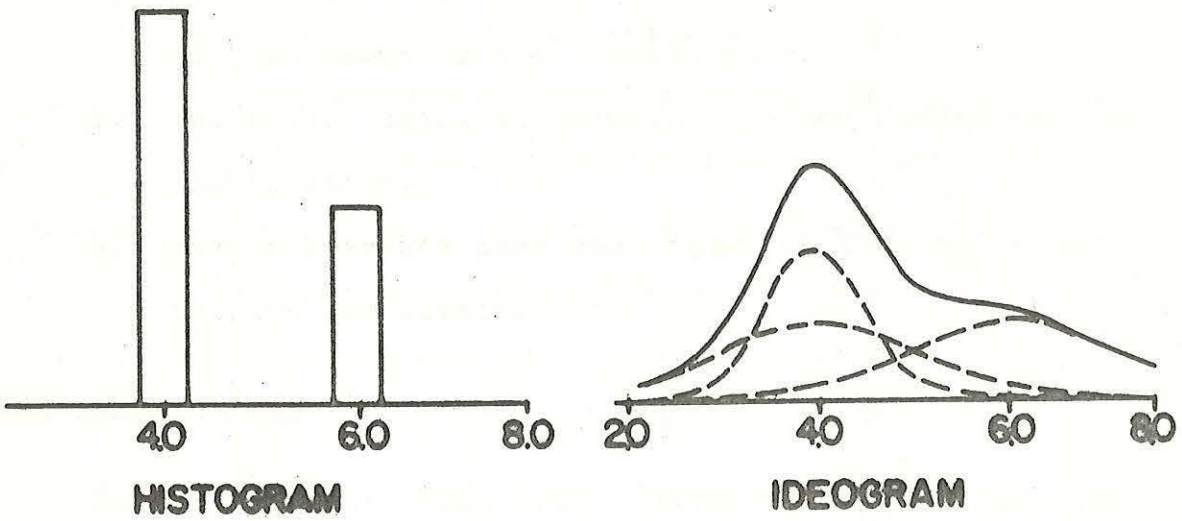
$$A \pm \sigma$$

where A is the most probable coordinate of the event, and σ is the associated error (usually in units of standard deviation). SIMPLOTTER's ideogram method treats each event as a normal probability (Gaussian) curve of unit area, centered at A, and of standard deviation σ. The resultant distribution is the sum of all the individual event normal curves.

Ideograms differ from histograms in that they take errors into consideration. Histograms lump all events equally into one interval regardless of error. Ideograms distribute events into all intervals according to the error. The following example illustrates this point:

Example: Three events will be plotted on both a histogram and an ideogram. The solid line represents the resultant distributions, the dashed lines represent the individual normal curves.

$$A_1 = 4.0 \qquad \sigma_1 = 1.0$$
$$A_2 = 4.0 \qquad \sigma_2 = 0.5$$
$$A_3 = 6.0 \qquad \sigma_3 = 1.0$$



The following point should be noted concerning ideograms:

Well-defined events (small σ) contribute more to the structure of the resultant distribution than do ill-defined events. A well-defined normal curve has a high, narrow peak at its most probable coordinate A. Thus it contributes heavily to the resultant curve at A but contributes relatively little elsewhere. An ill-defined event (large σ) is spread out having a low, broad rise at A, so it contributes slightly to all portions of the distribution and is commonly called "background".

Usage

There are three types of CALLs. The initialization CALL defines the number of intervals (bars) and their width. The building CALLs add events to the histogram or ideogram being constructed. One building CALL is needed for each event to be added. The finalization CALL is made to notify SIMPLOTTER that the building process is complete; that is, the histogram or ideogram is ready to be plotted. The X-axis length, XSIZE, is specified in the finalization CALL. If XSIZE is non-zero, the histogram or ideogram is begun on a new graph. If XSIZE is zero, or if the superposition routine is CALLed, the histogram or ideogram is superimposed on previous data sets. DISTRA does all the bookkeeping necessary to construct the histogram or ideogram. It needs two arrays to remember the information however, between CALLs. This form of implementation was chosen to allow the user to construct more than one distribution at a time. The user must not tamper with these work arrays between building CALLs.

Distributions, histograms or ideograms, often are created from large volumes of data stored on magnetic tape or disk. Notice that the use of DISTRA does not require all events to be in storage at the same time. The normal use of these routines to form a distribution is:

1. define the range of the distribution by calling DISTRI.
2. read an event from tape (or disk) and CALL DISTRA to add the event to the distribution.
3. repeat 2. until all events have been added to the distribution.
4. when all events have been read, CALL DISTRP or DISTRS to plot the distribution.

CALL_List_Format

Initialization:	CALL DISTRI (ABCISA, HTS, INTVLS, AMIN, AMAX)
Building:	CALL DISTRA (ABCISA, HTS, A, SIGMA, WEIGHT)
Finalization:	CALL DISTRP (THETA, ABCISA, HTS, ISYM, MODE, XSIZE, YSIZE, XSF, XMIN, YSF, YMIN, XLAB, YLAB, GLAB, DATLAB)
	CALL DISTRS (THETA, ABCISA, HTS, ISYM, MODE, DATLAB)

Note: Except for characters strings, all parameters beginning with the characters I - N are intended to be integer type variables or constants. All others are intended to be real type variables or constants.

Parameter_Definitions

ABCISA AND HTS - work arrays used to store the distribution between CALLs to DISTRI, DISTRA, DISTRP. These real type arrays must each be declared

(dimensioned) in the user program for (INTVLS + 2) words. A pair of these arrays must exist for each distribution being simultaneously constructed.

INTVLS - an integer variable or constant specifying the number of intervals of evaluation to be constructed.

AMIN - the beginning coordinate of the 1st interval.

AMAX - the coordinate of the end of the last interval.

A - the coordinate (abcissa) of the most probable value of the event.

SIGMA - histogram: SIGMA = 0.0 is necessary to indicate that a histogram is being constructed and the event, A, is to be treated as if it has no inherent error.

- ideogram: SIGMA > 0.0 indicates that an ideogram is being constructed and SIGMA is the standard deviation (error) associated with the event.

WEIGHT - histogram: the amount added to the appropriate 'bar'. WEIGHT = 1.0 is the usual case.

- ideogram: the area (in user units) of the normal curve representing the event. Usually WEIGHT = 1.0. The equation of the normal curve used is:

$$Y = NORM * e^{-\frac{1}{2}\left(\frac{A-X}{SIGMA}\right)^2} \qquad \text{where } NORM = \frac{WEIGHT}{\sqrt{2\pi} * SIGMA}$$

where Y is the height contribution at the evaluation point X.

THETA - the angle, in degrees, relative to the horizontal at which the abscissa is to be drawn. Allowable angles are 0.0 and 90.0 degrees.

ISYM, MODE, XSIZE, YSIZE, XSF, XMIN, YSF, YMIN, XLAB, YLAB, GLAB, and DATLAB are as defined in routine GRAPH.

Note:

Knowledge of the structure and contents of the work arrays, ABCISA and HTS, is not necessary for normal applications of distributions. However, for special cases, the programmer may wish further insight into the structure of the arrays.

The length of each array (in words) is stored in the first word of ABCISA. The remaining elements of ABCISA contain the evaluation points of the distribution. For a histogram, the points of evaluation are the maximum abscissas of each "bar" to be formed. The points of evaluation must be

stored in increasing order. DISTRI initializes HTS to zero and ABCISA to the following values:

```
ABCISA(1) = INTVLS + 2.0
ABCISA(2) = AMIN
ABCISA(3) = AMIN + (AMAX-AMIN)/INTVLS
ABCISA(4) = AMIN + 2 (AMAX-AMIN)/INTVLS
    .
    .
ABCISA(INTVLS+2) = AMAX
```

The first time DISTRA is CALLed a positive or negative '1.0' is stored in HTS(1) indicating the construction of a histogram or ideogram respectively. After each CALL DISTRA, the HTS array is updated to contain the following information:

histogram: HTS(1) contains the number of times DISTRA has been CALLed. For I>1, HTS(I) contains the height of the bar whose greatest abscissa is ABCISA(I)

ideogram: HTS(1) contains the negative of the number of times DISTRA has been CALLed. For I>1, HTS(I) contains the subtotal of all Gaussians evaluated at ABCISA(I)

DISTRP and DISTRS use the information collected in ABCISA and HTS to draw the distributions but do not alter the arrays.

ORIGIN - Modifying the Standard Graph Format

ORIGIN allows the user to change several of the SIMPLOTTER defaults concerning graph formatting. Axes can be omitted, the graph and data set label area can be repositioned, the size of the plotting symbols can be modified, and the automatic origin positioning between graphs can be cancelled. The user is advised, however, that program writing time, debugging time, and frustration time may increase if these defaults are altered. The defaults were chosen to minimize user interaction with plotting trivia.

Usage

Modifications to the graph format are introduced into SIMPLOTTER by specifying a modification number when referencing ORIGIN. Modifications stay in effect until changed by another CALL ORIGIN. Modifications apply to entire graphs, and must be specified before the graph to which they are to apply is begun. A superposition CALL immediately following a CALL ORIGIN is illegal.

One modification can be made per CALL; multiple CALLs are required for more than one modification. However, one CALL ORIGIN can restore the SIMPLOTTER default format.

CALL List Format

CALL ORIGIN (AA, BB, LATCH)

Parameter and Option Definition

The value of LATCH specifies the modification to be introduced into SIMPLOTTER. AA and BB have different meanings for each value of LATCH.

1. Controlling the Origin and Omitting Axes (LATCH = 1, 2, 3, 4)

The user may assume the responsibility of controlling the position of the origin at all times. This is a means of stacking graphs on top of one another, a means of effectively plotting more than 5000 points on one graph. The origin is a reference point used by the plotter to construct a graph. Initially, the position of the origin is (0.0, 0.0). Normally SIMPLOTTER moves the origin when a new graph is to be plotted. When the user assumes control of the origin, he must move the origin when starting a new graph. If the origin is not moved before beginning a new graph, the new graph will be drawn on top of the previous graph.

AA - the number of inches the X-origin is to be moved from the previous position of the origin

BB - the number of inches the Y-origin is to be moved from the previous position of the origin

LATCH = 1 Draw both axes, XLAB and YLAB.

2 Omit the X-axis and XLAB.

3 Omit the Y-axis and YLAB.

4 Omit X-axis, Y-axis, XLAB, and YLAB.

2. Changing the height of plotting symbols (LATCH=5)

To alter the height of the plotting symbol referenced by ISYM:

AA - the new height (inches) of the plotting symbols

BB - 0.0 (not used)

LATCH = 5

3. Repositioning Graph Label Area (LATCH=6,7)

Normally the graph labels are positioned in the upper right hand corner of each graph. The graph label, GLAB, is drawn at coordinates (XSIZE-2.2, YSIZE-0.3). All of the superposition data set labels, DATLABs, are downward listed under the graph label. The position of this block of labels can be changed by the user, by calling ORIGIN with LATCH = 6.

AA - the X coordinate (in inches) of the

new starting position of the graph label, GLAB.

BB - the Y coordinate (in inches) of the new starting position of the graph label, GLAB.

LATCH = 6

A call to ORIGIN with LATCH = 7 restores the label block to its normal position.

4. Restoring the standard graph format (LATCH=0,8)

AA - the number of inches the X origin is to be moved for the next graph. If the user has had control of the origin, AA should be large enough to move the origin beyond the previous graph. After this, SIMPLOTTER will move the origin between graphs.

BB - 0.0 (not used)

LATCH = 0 or 8

If LATCH = 0, the SIMPLOTTER parameter dump will not be printed.

If LATCH = 8, the parameter dump will be printed.

LETTER - Additional Labeling and Data Tagging

LETTER allows additional lettering to be drawn on a graph. A string of up to 80 characters can be drawn at various positions and angles. LETTER provides a convenient means of producing annotations in a fixed position of a graph. For example, a block of lettering is often used as a key and placed in one of the corners of the graph.

Usage

The user forms a character string and passes it to LETTER or LETTRS for drawing. LETTER provides the capability of starting a new graph, and LETTRS provides a convenient means of superimposing strings.

CALL List Format

```
CALL  LETTER  (X0, Y0, HEIGHT, STRING, THETA, NCHAR,
               XSIZE, YSIZE, XSF, XMIN, YSF, YMIN,
               XLAB, YLAB, GLAB, DATLAB)

CALL LETTRS (X0, Y0, HEIGHT, STRING, THETA, NCHAR)
```

Parameter Definition

X0 and Y0 - the coordinates in inches of the lower left hand corner of the first character to be drawn.

HEIGHT - the height of the characters (in inches) to be drawn.

STRING - the character string to be drawn. The string

can contain up to 80 characters. STRING can be the name of a character string, or the actual character string enclosed between quotes. In the latter case, the string should be terminated by a semi-colon.

THETA - the angle in degrees relative to the horizontal at which STRING is to be drawn.

NCHAR - the maximum length of STRING (number of characters). If a semi-colon is encountered before NCHAR characters have been plotted, the semi-colon is taken to indicate the end of the string. That is, the semi-colon and remaining characters are not drawn. If there is no semi-colon in STRING, the first NCHAR characters of STRING are drawn. If NCHAR is greater than 80, only the first 80 characters are drawn.

XSIZE, YSIZE, XSF, XMIN, YSF, YMIN, XLAB, YLAB, GLAB, and DATLAB are defined in routine GRAPH.

Example: A string named STR is to be plotted vertically beginning at the point (1.0, 2.0) inches with character height of 0.1 inch. At the time LETTER is called STR contains the characters HI-THERE, an 8 character string. The statement to plot STR is:

```
CALL LETTRS (1.0, 2.0, 0.1, STR, 90.0, 8)
```

ACKNOWLEDGEMENTS

SIMPLOTTER has been operational at I.S.U. since 1966 and has been maintained and modified slightly by several persons. The authors wish to thank David Erbeck and JoAnn Eischied for their early work with the smoothing and sorting routines used in SIMPLOTTER. Steven Hollatz pioneered the printer plotter and demonstrated it's effectiveness. Thanks to F. S. Carlsen for helping with many aspects of the system, and to George Covert, whose presence always serves as a stabilizing force.

APPENDIX A: DEBUGGING AND THE SIMPLOTTER PARAMETER DUMP

SIMPLOTTER is designed such that the user can debug his program logic in parallel with his plotting logic. This feature means that the time required to get a program debugged is usually no longer when plotting than when not plotting. For instance, when a mistake exists in the parameter list of a SIMPLOTTER problem-oriented routine, the program usually does not terminate at that point; the program usually continues on to a terminal error in the user program logic. After looking at the user output, the SIMPLOTTER Parameter Dump, and the graphical output, an attempt can be made to correct both program logic and plotting logic errors before re-submitting for the next debugging trial.

Debugging the plot logic errors is made considerably easier by using the SIMPLOTTER parameter dump which is printed before each graph is generated. The parameter dump lists all parameters given to the problem-oriented routines. As it is impractical to list completely all the (X,Y) coordinates of each data set, only the first four (X,Y) data points of the first three data sets are shown. When the user notices that one of the parameters given to the problem-oriented routine does not match what he intended, he has a handle on his programming error.

The printer-plotter feature of SIMPLOTTER should be used until a program is thought to be debugged. As the printer-plots immediately follow the user's printed output, there is no time wasted waiting for the graphs to be plotted on the incremental plotter. The resolution of the printer is nearly always adequate for debugging purposes, and quite often is good enough for production use. In any case, after a program has been debugged, no change in program logic is needed to produce the plots on the higher resolution incremental plotter. Only one job control card change is needed.

The problem-oriented routines are available in PL/1(F), FORTRAN IV (G, H, and WATFIV) and COBOL. The PL/1(F) and COBOL user should normally debug his program using the printer-plotter. The FORTRAN user should, as usual, debug with WATFIV and the printer-plotter. For production work FORTRAN programs should be run using FORTRAN G or H with either the printer or incremental plotter.

APPENDIX B: THE PRINTER PLOTTER

The printer plotter facility of SIMPLOTTER was designed as a debugging tool. For many purposes the printer plotter is used for production jobs also.

Advantages of the printer plotter:

1) Printed graphs are attached to the normal printed output.
2) Turnaround time is reduced because no time is spent waiting for the incremental plotter to draw the graph. Typical wait time for the incremental plotter at I.S.U. is several hours.

Peculiarities of printer plotter:

1) Only the first 9 by 12 inches of a graph are plotted. For example, the user may specify an x-axis of 15 inches. That graph is completely calculated on the basis of a 15 inch axis; that is, the scale factor is chosen and points scaled, as if a 15 inch axis were going to be drawn. However, only that portion of the graph which can be plotted on one page is printed. Thus, axes longer than a printed page can accommodate are truncated at plot time.
2) The order of CALLs to problem-oriented routines is important if two data sets plot points in the same area of a graph. The rule adopted is:

a) all axis generation and labeling are formed before any data is plotted. Thus data, when plotted, can overlay and destroy labels.

b) data sets are plotted in the order that their problem-oriented routines were called.

Note: character strings plotted by CALLing LETTER are plotted before any data points are plotted. For example, if a curve extends into the upper right hand corner of the graph, it may replace a portion of the previously generated graph label. A second data set passing near the first one may replace part of the first data set.

3) The printer plotter puts as many complete graphs on one page as possible. All graphs produced while the user has control of the origin will be on one page.

4) There are some discrepencies between the symbols plotted by the printer, and those (ISYM) of the incremental plotter. Not all the plotter symbols have equivalents on the printer, although some do. Substitutions are made when necessary and are shown in Appendix C.

5) Printer plots are designed to be printed at 6 lines per inch. A plot printed at 8 lines per inch will be compressed in the y-direction.

APPENDIX C: PLOTTING SYMBOLS AVAILABLE

NOTES:

1) The plot centered symbols are used to plot (X,Y) data points, and are referenced by ISYM (see GRAPH). Note that the printer plotter must make some substitutions as not all symbols are available on printers.

2) The special characters, used for labeling, can be drawn by the incremental plotter only; the printer plotter ignores them. They usually are multi-punched onto cards and read into the computer as character strings.

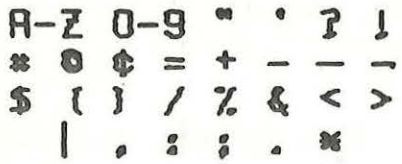
3) Special control characters are provided to allow superscripting, subscripting, etc. Although these characters are not drawn, they must be counted as part of the length of a character string. These control characters are effective only with the incremental plotter; the printer plotter will treat them as blanks.

CENTERED DATA POINT SYMBOLS

Prtr Sym	Incr Sym	Sym Ref#	Prtr Sym	Incr Sym	Sym Ref#
		0	7		7
O		1	Z		8
2		2	Y		9
+		3	A		10
X		4	*		11
5		5	C		12
6		6	|	|	13

LABELING CHARACTERS AND CONTROL PUNCHES

Standard Keypunch Characters



Non-Standard Keypunch Characters

Char	Multi-punch	Char	Multi-punch
	-984		-985
	-986		-987
	-0981		091
	092		093
	094		095
	096		097
	098		0981
	0982		0983
	0984		0985
	&-0981		91
	92		93
	94		95
	96		97
	98		981
	982		983
	984		985
	986		987
	&986		&987
	&-981		-92
	-93		-94
	-96		-97
	-98		-982
	-983		&-

Labeling Character Control

Function	Multi-punch
begin superscripting	0986
end superscripting	0987
begin subscripting	0987
end subscripting	0986
backspace	-91
null	-981
carriage return	-95

Figure 3. Plotting symbols and characters available.

APPENDIX D: THE INCREMENTAL PLOTTER

High quality plotting is done at the ISU Computation Center by a CalComp Digital Incremental Plotter. It features a liquid ink pen which moves in increments of 1/100 inch. The incremental plotter is intended for the production of finished, publishable plots.

The maximum size of a plot on the incremental plotter is 11 inches by 120 feet. There is usually a delay of several hours between the computation and the drawing of a graph.

Several types of graph paper are available for plots. Paper types can be specified to the plotter operator by using the FORM parameter or the PARM.PLOT parameter on a //SMPLTTR EXEC card. For example:

```
//SMPLTTR EXEC PLOT,PLOTTER=INCRMNTL,FORM=F
```

(This specifies type 02 paper.)

The FORM field is optional. If it is omitted, FORM=F, indicating type 02 paper, will be assumed.

The most commonly used types of paper are:

Type	Description of Paper	FORM Designation
00	plain white	W
02	20 divisions/inch	F

When the FORM parameter (F or W) is used, plots are recorded in disk data sets and are later transferred to a plot tape, which is then manually mounted on the plotter for off-line plotting. When special pen points, or paper types other than 00 or 02 are used, plots are recorded directly onto magnetic tape during execution of the job. To provide for direct taping, the user must insert the PLOT.PLOTTAPE DD card in his deck after the //SMPLTTR EXEC card. The form of the PLOT.PLOTTAPE DD card is as follows:

```
//PLOT.PLOTTAPE  DD  DSNAME=SPLOTCC,
    DISP=(NEW,KEEP),UNIT=(TAPE7,,DEFER),
    VOLUME=(PRIVATE,SER=TPPLOT)
```

In addition, the PARM field should be used on the EXEC card instead of the FORM parameter. This will cause a message to be printed to the plotter operator. Examples of this usage follow:

```
//SMPLTTR EXEC PLOT,PLOTTER=INCRMNTL,
    PARM.PLOT='SEE SUBMITTAL SHEET'
//SMPLTTR EXEC PLOT,PLOTTER=INCRMNTL,
    PARM.PLOT='PLOT ON TYPE 09 PAPER'
```

APPENDIX E: RESTRICTIONS OF SIMPLOTTER

At Iowa State, the graphing subroutines are currently implemented in PL/1(F), FORTRAN IV (G, H, and WATFIV) and COBOL.

The following absolute restrictions are part of the basic design of SIMPLOTTER and cannot be altered. Very seldom will these restrictions affect the design of a user's program.

1. The total number of data set superpositions on any one graph must not exceed 99.
2. The total number of data points (primary and all superposition data sets) must not exceed 5000 on any one graph. However, there is no inherent limit to the number of graphs drawn.
3. When the user's program uses overlay concepts, subroutine SMINIT must be placed in the root section. SMINIT is the communications routine between the various problem-oriented plotting routines.
4. Axes can be drawn only at 0.0 and 90.0 degrees to the horizontal.
5. The file name (or DDNAME), FT14F001, is used by the SIMPLOTTER routines and must not be used by the user program. This file is used to store the SIMPLOTTER Data Set (SDS) which is subsequently plotted when the //SMPLTTR EXEC PLOT job control card is encountered.

The following restrictions are also implemented within SIMPLOTTER in order to insure reasonable graphs.

1. XSIZE and YSIZE should be greater than 2.0 inches. Otherwise, graphs are not very meaningful as only one labeled tic will appear.
2. YSIZE must be less than the height of plotting paper being used. This maximum vertical axis size depends upon the type of plotter at your installation. At I.S.U. this limits YSIZE to 10.0. If YSIZE is greater than the maximum size, SIMPLOTTER will plot as much of the axis as possible.
3. The axis label (XLAB or YLAB) requires about 3.8 inches. If an axis is less than 3 inches, part of the label may extend outside the graph area. In the case of YLAB, this could force the plotter pen off the paper. To avoid this for short axes, the label could start with blanks and the data could be scaled so that SIMPLOTTER need not print a scale factor. (See Figure 22, page 100 for an example of a label printed with a scale factor.)
4. Interpolation (MODE = 1, 2, 5, or 6) requires at least four points. If the number of points is less than four, the curve will not be drawn.
5. Smoothing (MODE = 11 to 30) requires at least six points with distinct x coordinates. (For smoothing, SIMPLOTTER lumps points with x coordinates very close together into

a single point whose y coordinate is the average y.) After smoothing, the curve is plotted using interpolation. If the number of points to smooth is less than six, no smoothing is done. If less than four, the curve will not be drawn.

APPENDIX F; SIMPLOTTER PLOTTING SYSTEM JOB CONTROL

To use SIMPLOTTER, simply add the SIMPLOTTER job control cards to the end of your "normal job." In the following examples, the "normal job" is considered to be the portion of the job prior to the SIMPLOTTER job control cards. Therefore, the "normal job," excluding the CALLs to SIMPLOTTER's problem-oriented routines, is a typical user job which runs without plotting.

Note: There are three SIMPLOTTER job control cards, as shown on the following two pages.

Example 1. Figure 4 illustrates the construction of a typical WATFIV printer plotter job.

Example 2. FORTRAN G and H, PL/1(F), and COBOL jobs using the printer plotter are typified by the job construction of Figure 5. The "normal job" shown uses the procedure FORTG. "Normal jobs" using other procedures can be substituted. Some commonly used procedures named on the //STEP1 EXEC card of "normal jobs" are:

FORTGCG FORTH PL1F COBUCLG

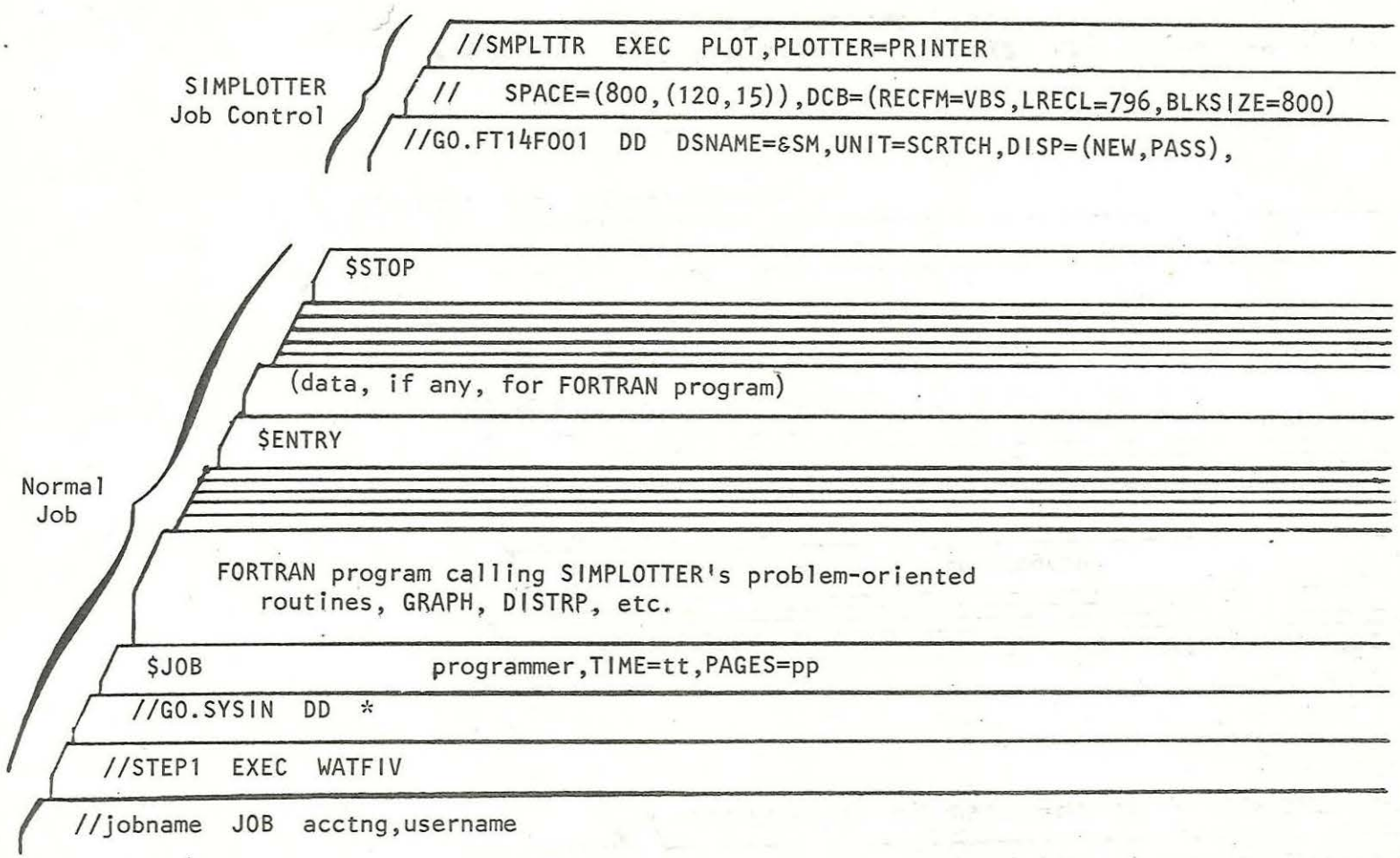


Figure 4. WATFIV job construction using the printer plotter.

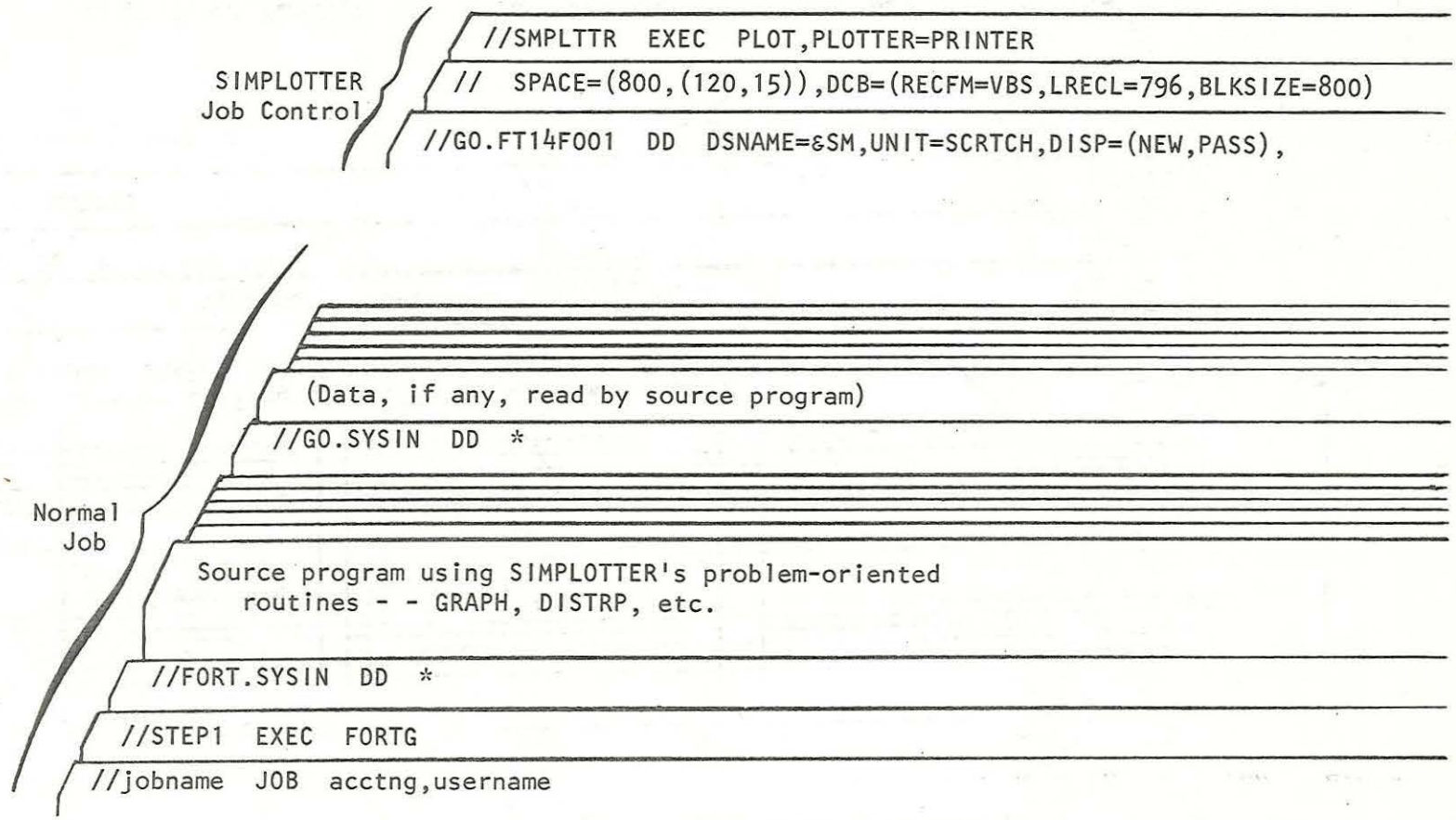


Figure 5. FORTRAN G and H, PL/1(F) and COBOL job construction using the printer plotter.

Example 3. To convert the jobs of Examples 1 and 2 to the incremental plotter, modify only the //SMPLTTR card to read,

```
//SMPLTTR EXEC PLOT,PLOTTER=INCRMNTL
```

General Information Concerning Job Control With SIMPLOTTER

SIMPLOTTER is currently implemented on an IBM 360/65 computer at I.S.U. The job control in the preceding examples reflects OS/360 conventions and the following SIMPLOTTER concepts:

1. The problem-oriented routines are supplied automatically to the "normal job." The FORTRAN, PL/1, and COBOL user does not have to supply an object deck or a job control card to represent these routines; they are supplied in the same manner as commonly used functions; SQRT, EXP, etc..
2. The //FT14F001 job control cards are mandatory when calling any of the problem-oriented routines. If not supplied, the user program will terminate the first time a problem-oriented routine is called.
3. The //SMPLTTR job control card directs graphs to either the printer or the incremental plotter. If this card is omitted, no plots will be drawn. However, the "normal job" will run as if no reference has been made to the plotting routines. That is, the job will not terminate due to the omission of this card. The general form of the //SMPLTTR card is shown in Figure 6.

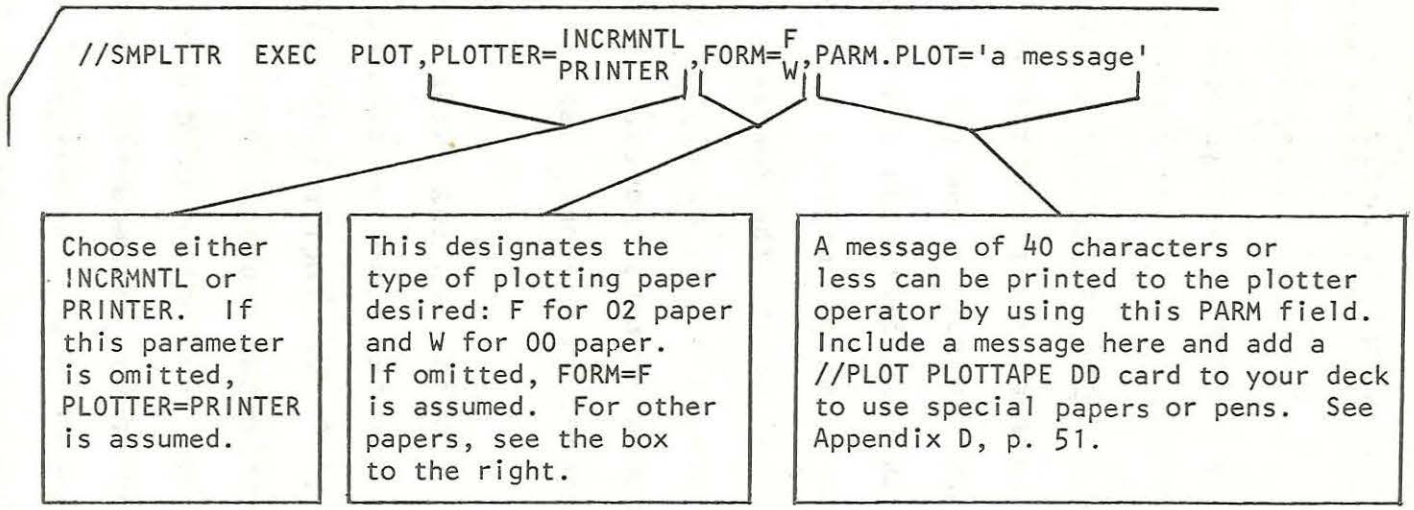


Figure 6. The general form of the //SMPLTTR job control card.

APPENDIX G: INTERPOLATION, SMOOTHING, and SORTING TECHNIQUES

I. Interpolation

The user may specify that a curve is to be drawn connecting each of his data points by setting MODE = 1, 2, 5, or 6 when CALLing GRAPH. SIMPLOTTER uses a second degree Langrangian interpolation polynomial in constructing the curve. For a set of data points (u,v) the interpolation polynomial of degree m may be concisely stated as:

$$(1)\quad I(u) = \prod_{m+1}(u) \sum_{j=o}^{m} \frac{v_j}{(u-u_j)\prod'_{m+1}(u_j)}$$

For the case of second degree (m = 2) polynomial connecting the points (u_0, v_0), (u_1, v_1), and (u_2, v_2), the computational form becomes:

$$(2)\quad I(u) = v_o \frac{(u-u_1)(u-u_2)}{(u_o-u_1)(u_o-u_2)} + v_1 \frac{(u-u_o)(u-u_2)}{(u_1-u_o)(u_1-u_2)} + v_2 \frac{(u-u_o)(u-u_1)}{(u_2-u_o)(u_2-u_1)}$$

SIMPLOTTER uses the following algorithm when interpolating along a set of data:

Step 1: To begin the interpolation in the segment x_i through x_{i+1}, we construct a work data set including the immediate

neighboring points (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}), and (x_{i+2}, y_{i+2}). The work set is rotated, if necessary, such that the x-coordinates are monotone increasing. The rotated set will be denoted by $(x_i, y_i)^r$, $(x_{i+1}, y_{i+1})^r$, etc..

Step 2: Two interpolation polynomials are evaluated using equation (2).

$I_1(x)$ is evaluated substituting

$$(u_o, v_o) \leftarrow (x_{i-1}, y_{i-1})^r$$

$$(u_1, v_1) \leftarrow (x_i, y_i)^r$$

$$(u_2, v_2) \leftarrow (x_{i+1}, y_{i+1})^r$$

$I_2(x)$ is evaluated substituting

$$(u_o, v_o) \leftarrow (x_i, y_i)^r$$

$$(u_1, v_1) \leftarrow (x_{i+1}, y_{i+1})^r$$

$$(u_2, v_2) \leftarrow (x_{i+2}, y_{i+2})^r$$

That is, I_1 uses the data point to the left of the segment, and I_2 uses the data point to the right. The interpolated value chosen is the one nearest an imaginary straight line connecting (x_i, y_i) and (x_{i+1}, y_{i+1}). This method is used to reduce 'bubbles' which otherwise may occur near sharp corners of ill-behaved data.

Step 3. The chosen interpolated value is rotated back to the original coordinate system and a straight line drawn to it from

the previous interpolated point.

Step 4. The point of evaluation, x, is incremented by 0.1 inch. If the new x lies within the segment being interpolated, the process is repeated from Step 2. Otherwise the next segment is begun ($i \leftarrow i + 1$) and the process returns to Step 1.

The user should note the following when interpolating:

1. Curves constructed go from point to point as the points are ordered in the (x,y) arrays. Thus multivalued curves can be constructed, such as a circle.
2. The method gives best results when more than 10 points are given and the angle between points is less than 90 degrees.

II. Smoothing

The user may specify that a 'smoothed' curve be drawn through his data points by specifying MODE from 11 through 29 in a CALL GRAPH statement. By a smoothed or graduated curve, we mean a curve passing through the neighborhood of the data points and not necessarily through the points themselves.

Let the set of user data points be denoted by $(x,y)^0$, and let us consider one of these points $(x_i,y_i)^0$. By least squares we fit a polynomial of degree m to p+1 consecutive points of $(x,y)^0$, where p is an even number and is greater than m. We choose the p points such that p/2 points lie on each side of the point under consideration, $(x_i,y_i)^0$. We evaluate the polynomial at x_i to obtain a generally 'smoother' point, $(x_i,y_i)^1$ than the

original point $(x_i, y_i)^0$. The process is repeated for each point of $(x,y)^0$ to produce a smoother set of data $(x,y)^1$. If MODE is specified to be 11 or 21, an interpolated curve is drawn through the set of points $(x,y)^1$.

A still smoother set of points is generated by applying the same process to $(x,y)^1$ to obtain $(x,y)^2$. MODE = 12 or 22 draws an interpolated curve through the set $(x,y)^2$. Higher levels of smoothing are obtained similarly. However, using SIMPLOTTER, no higher level of smoothing than $(x,y)^9$ can be specified.

The user should note the following when smoothing with this technique:

1. Smoothing is most effective on data sets of more than 20 points.
2. Data points should be arranged in the (x,y) arrays to be monotonely increasing in x-coordinates.
3. First attempts at smoothing a data set should use MODE = 11 or 21, that is, the lowest level of smoothing. Too much smoothing can result in the loss of the trend of the original data points. High levels of smoothing are expensive also, as all lower levels must be calculated in the computation of the high level set.

* * *

And now a little rhetoric for those who turn on for numerical analysis. Formation and evaluation of a polynomial of degree 3 for every point in a given data set, repeated for each level of smoothing, could be prohibitive. Milne[1] outlines a computationally efficient method for data points equally spaced in x-coordinates.

Suppose there is a function $y=f(x)$ satisfied by each of the given data points. Let the values of y be denoted by $(y_0, y_1, \ldots, y_n)$ and the respective x values by $(0,1,\ldots,n)$. The approximation polynomial derived by Milne is:

$$(3)\quad Q_m(x) = \sum_{k=o}^{m} \left[\frac{\sum_{x=o}^{n} P_{k,n}(x) f(x)}{\sum_{x=o}^{n} P^2_{k,n}(x)} \right] P_{k,n}(x)$$

where

m is the degree of the polynomial

n+1 is the number of consecutive points used to evaluate the polynomial

$p_{m,n}(x)$ is one of a set of orthogonal polynomials

such that

$$P_{m,n}(x) = \sum_{k=o}^{m} (-1)^k \binom{m}{k}\binom{m+k}{k}\frac{x^k}{n^k}$$

This forbidding looking polynomial simplifies to a compact computational form when evaluated at one of the given data points, as is shown by the following illustration.

Let us take the degree m = 3 and five points, so that n = 4. The midpoint is then x = 2, the point at which the approximating polynomial will be evaluated. Let $(y_0, y_1, y_2, y_3, y)_4$ be the five consecutive data values of f(x) and let y_2' be the value obtained. Then putting m = 3, n = 4, x = 2 in equation (3), we have

$$(4) \quad y_2' = \sum_{k=o}^{3} \left[\frac{\sum_{j=o}^{4} P_{k,4}(j) y_j}{\sum_{j=o}^{4} P_{k,4}^2(j)} \right] P_{k,4}(2)$$

Interchanging the summations with respect to j and k, we can write

$$(5) \quad y_2' = \sum_{j=o}^{4} y_j \left[\frac{\sum_{k=o}^{3} P_{k,4}(j) P_{k,4}(2)}{\sum_{k=o}^{4} P_{k,4}^2(j)} \right]$$

Now the quantity in square brackets is independent of the given data; that quantity can be computed once and for all. Milne has

calculated those values and, when plugged into equation (5), yield

$$(6)\quad y_2' = \frac{1}{35}\left[-3y_o + 12y_1 + 17y_2 + 12y_3 - 3y_4\right]$$

This type of equation is ideal for a computer and makes the method previously described feasible.

SIMPLOTTER uses a degree 3 polynomial calculated from 13 consecutive points whenever possible. Near the beginning and end of the given data set, the number of consecutive points available drops below 13 and other equations are used. All equations used are of the same form as equation (6) and are evaluated at the midpoint. The following chart lists the polynomial coefficients used (note the n + 1 coefficients are symmetrical about their midpoint):

m	n+1	x		coefficients
3	13	6	$\frac{1}{143}$	[- 11, 0, 9, 16, 21, 24, 25, ...]
3	7	3	$\frac{1}{21}$	[- 2, 3, 6, 7, . . .]
3	5	2	$\frac{1}{35}$	[- 3, 12, 17, ...]
1	3	1	$\frac{1}{2}$	[1, 1, 1]
-	1	0	1	(i.e., $y_o' = y_o$)

As stated previously, the approximating polynomials are derived under the assumption of equally spaced points. When a data set is encountered which does not satisfy this condition, SIMPLOTTER 'weights' the points in an attempt to approximate the condition. Years of observation of smoothed curves indicates that this technique produces visually acceptable curves.

III. Sorting

Note: The following sorting technique is not implemented in SIMPLOTTER at present. However, this section may be of instructional value to users wishing to perform sorts on their data sets for efficient plotting.

Plotting symbols at the (x,y) points is a commonly used option when CALLing GRAPH, that is, specifying MODE=7. The incremental plotter draws the points in the sequence that they appear in the (x,y) arrays. Time is wasted travelling from point to point when drawing unordered arrays. As it makes no difference in the final appearance of the graph, SIMPLOTTER reorders or sorts all MODE=7 data sets of more than 200 points. Note, however, that the sorting is done at plotting time, that is, the user's (x,y) data arrays are not changed or reordered in any way when CALLing GRAPH.

The sorting technique used is designed to eliminate unnecessary travelling time between points. A large amount of plotter time can be saved if large data sets are sorted by SIMPLOTTER before plotting. Rough estimates of plotting

times, based on random point positions, are:

$$\text{(sorted time)} = \frac{n(30 + 48w)}{15000} \text{ minutes}$$

$$\text{(unsorted time)} = \frac{n(30 + 33\sqrt{XSIZE^2 + YSIZE^2})}{15000} \text{ minutes}$$

where $w = \text{Max}[0.5, \sqrt{\frac{3*XSIZE*YSIZE}{n}}]$

n = number of points to plot

XSIZE = length (inches) of the x-axis

YSIZE = length (inches) of the y-axis

For example, a 7 1/2 by 10 inch, 500 point graph requires about 14 minutes to plot if the points are randomly distributed. SIMPLOTTER sorts the points into good plotting order in 0.5 seconds (IBM 360/65) and plotting time is cut to 2 minutes. At current I.S.U. charge rates, the user saves about $1.00 and obtains better plotter turnaround time. For 2,500 random points the sort requires 3.0 seconds; a net savings of 61 plotter minutes and about $5.00 is obtained. The dashed lines in Figures 7 and 8 illustrate paths travelled by the pen while drawing unsorted and sorted points.

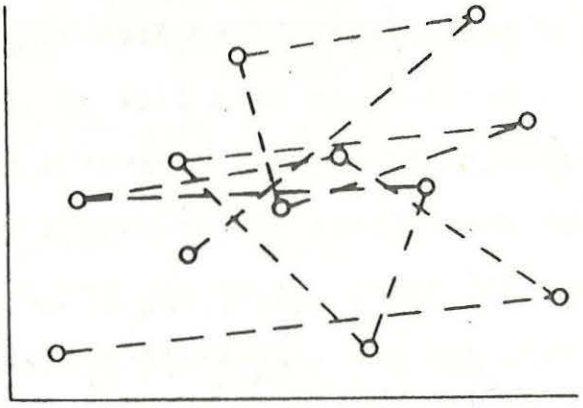


Figure 7. Path through original points.

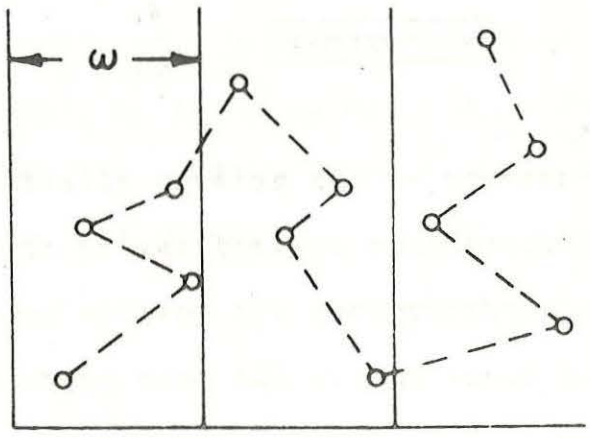


Figure 8. Path through sorted points.

However, a disadvantage of SIMPLOTTER's sorting technique is that resolution drops to 1/32 inch. For those who find this objectionable (the eye can hardly detect it) an 'unsorted' MODE is provided. MODE=0 is equivalent to MODE=7 except that plotting time is increased and point resolution is approximately 1/50 inch; that is, no sort is performed.

Let us first obtain an approximation of the number of plotter steps required to plot the original points. Plotter steps can be easily converted to time; the I.S.U. plotter moves at a rate of 15,000 steps per minute. Assume that the points are in uniform, random order. Drawing the average symbol requires 30 steps, that is 3 steps to lower the pen, 17 steps to draw the average symbol, and 10 steps to raise the pen. The number of steps to move the pen 1 inch is 100 steps and the expected distance to connect any two points, uniformly distributed over a range XSIZE, is XSIZE/3. If S is the number of steps to draw n random points,

$$n[30 + 100*\max(\frac{XSIZE}{3},\frac{YSIZE}{3})] < S_o < n[30 + 100\sqrt{(\frac{XSIZE}{3})^2 + (\frac{YSIZE}{3})^2}]$$

The sorting method first sorts the $(x,y)^0$ points into ascending x-order, defined as the set $(x,y)^1$ having $x^1_i \leq x^1_{i+1}$. The x-coordinate range is divided then into a number of intervals or strips, which we will conceptually number 1,2,3,... from left to right. The points whose x-coordinates lie in strip 1 are sorted into ascending y-order, the points in strip 2 into descending y-order, strip 3 ascending, etc. The path followed in drawing the

points is shown in Figure 8.

The width of each strip is chosen such that we get much of the vertical movement 'free'. The incremental plotter is capable of incrementing both the x and y motors simultaneously. So if we arrange the strip width such that the expected x move is greater than the mean y move, we will make most of the y moves during the plotter steps of the x moves. Let the strip width be denoted by w. Then the average x move to the next random point is w/3. If there are n random points per graph, there will be an average of m points per strip, where m=n*w/XSIZE, and the mean y distance between the points in a strip is YSIZE/m . To minimize the cost of vertical movements,

$$\frac{w}{3} \geq \frac{XSIZE*YXIZE}{n*w}$$

The liquid ink pen clots when w is less than 1/2 inch, so,

$$w = \max(0.5, \sqrt{\frac{3*XSIZE*YXIZE}{n}-})$$

If this condition actually gave us all y- movement free of charge, the number of steps to travel between points would equal the number of steps to travel between the x-coordinates of the points in a strip. The total steps over all strips would be:

$$S \doteq n\ [30 + 100*\frac{w}{3}\ ] + 100*XSIZE$$

Experimentally it is found that our chosen strip width effectively

gives us about 70% of the y movement 'free'. The expression for estimating plotting steps required is then:

$$S_{smp1ttr} = n\ [30 + 48w] + 100*XSIZE$$

The sorting method applied to each strip is of the Bucharest type and is discussed by Iverson[2]. The execution time is proportional to $C*n*\log_2 n$, where the constant C depends on the machine and compiler used, and on the order (disorder) of the input points. Briefly described, the method considers an input list to be made of a number of shorter sublists already in order. Half the sublists are stored in forward order beginning at P_1, the other half in backward order beginning at P_n. The first pass through the list merges pairs of these sublists from each end of the input list, and stores them in the output list. The output list is stored in P_{n+1} through P_{2n}, and like the input list, is made of sublists stored forward from P_{n+1} and backward from P_{2n}. On successive passes through the list, the input and output areas are switched. Thus, each pass halves the number of sublists. When only one sublist remains, the list is in order.

REFERENCES

1. W. E. Milne, Numerical Analysis, Princeton University Press, New jersey (1949), pp 83-84, 275-280.

2. K. E. Iverson, A Programming Language, John Wiley and Sons, New York (1962), pp 206-211.

APPENDIX H. SAMPLE PROGRAMS

The following programs illustrate the use of SIMPLOTTER plotting concepts in each of the languages. Sample jobs include:

1. a problem definition
2. method of solution
3. comments and programming hints
4. a complete listing of the job deck
5. the SIMPLOTTER Parameter Dump generated by the sample job
6. the printer plotter graphs produced
7. the incremental plotter graphs which were obtained by resubmitting the sample job

Sample Job#1 (PL1, WATFIV, and FORTRAN G)

Problem Definition:

A set of measured data points are to be visually compared with a predicted curve. Display the information on two separate graphs, one with linear axes and the other with semi-logarithmic axes.

Method:

Axis labels and the measured data points are read from cards and 10 points computed along the predicted curve. The measured data points are plotted specifying linear axes (XSIZE and YSIZE positive), automatic scaling in the Y direction (YSF=0.0), and user specified scaling in the X direction (XSF=0.4). An interpolated curve (MODE=2) through the 10 predicted points is superimposed on the measured data points.

Prior to starting the second graph, natural logarithms are stored in the plot arrays. On the second graph the predicted curve is plotted first, specifying that the Y direction is logarithmic (YSIZE negative) and Y scaling is automatic (YSF=-0.5). The measured data points are superimposed, specifying that the data label should be located in the graph label area (MODE>100).

Comments:

1. On the first printer plotter graph only 7 of the 8 measured data points appear. One of the points was overlayed by the predicted curve which was superimposed. On the second graph all 8 data points appear as the data was superimposed on the predicted curve. The order of superposition is sometimes important when using the printer plotter, but is not a factor when using the incremental plotter.
2. Two data points have X coordinates which fall outside the X direction range specified by the user. They are plotted 1/2 inch beyond the end of the X axis, as a debugging aid. For initial views of data, automatic scaling is recommended, as scaling is determined such that all data falls within the specified graphical area.

PL/1 Notes:

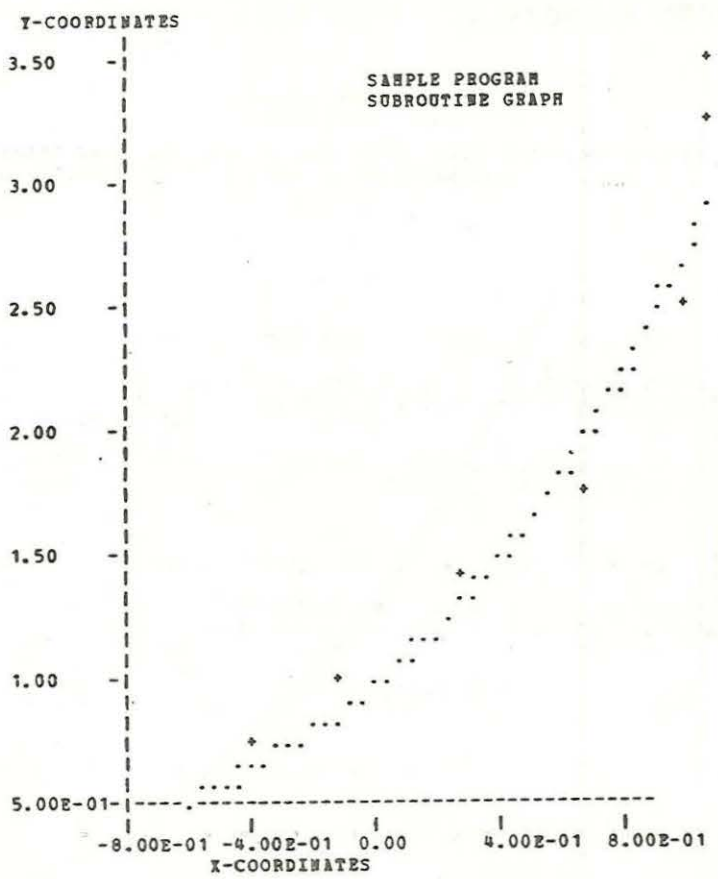
A DECLARE ENTRY GRAPH statement is used to insure that all arguments will be of proper base, scale, and precision, when received by subroutine GRAPH.

```
//C376#PLG JOB A0900,GIB
//S1 EXEC PL1F
//PL1L.SYSIN DD *
 SAMPLE_GRAPH:            PROCEDURE OPTIONS(MAIN);

 DCL (DATA_X(8), DATA_Y(8), PRED_X(10), PRED_Y(10)) FLOAT(6);
 DCL (X_LABL, Y_LABL) CHARACTER(20);
 DCL GRAPH ENTRY (FIXED BIN,(*)FLOAT,(*)FLOAT,FIXED BIN,FIXED BIN,
                  FLOAT,FLOAT,FLOAT,FLOAT,FLOAT,FLOAT,
                  CHAR(20),CHAR(20),CHAR(20),CHAR(20)),

    GRAPHS ENTRY (FIXED BIN,(*)FLOAT,(*)FLOAT,FIXED BIN,FIXED BIN,
                                                CHAR(20));

 GET EDIT(X_LABL, Y_LABL)(2 A(20));
 GET SKIP EDIT ((DATA_X(I),DATA_Y(I)  DO I=1 TO 8))(16 F(4,0));

 DO I=1 TO 10; PRED_X(I)=(I-4)/5.0; PRED_Y(I)=EXP(PRED_X(I)); END;

 CALL GRAPH (8, DATA_X,DATA_Y, 3, 7, 4.25, 6.25, 0.4, -0.8, 0.0, 0.0,
               X_LABL, Y_LABL, 'SAMPLE PROGRAM', 'SUBROUTINE GRAPH');
 CALL GRAPHS(10,PRED_X,PRED_Y, 0, 2, ' ');

 DO I=1 TO 8; DATA_Y(I)=LOG(DATA_Y(I)); END;
 DO I=1 TO 10; PRED_Y(I)=LOG(PRED_Y(I)); END;

 CALL GRAPH(10, PRED_X,PRED_Y, 0, 2, 4.25, -6.25, 0.4, -0.8, -0.5, 0.0,
                X_LABL, Y_LABL, 'SAMPLE PROGRAM', 'PREDICTED CURVE');
 CALL GRAPHS(8, DATA_X,DATA_Y, 3, 107, 'MEASURED DATA');

 END SAMPLE_GRAPH;
//GO.SYSIN DD *
  X-COORDINATES          Y-COORDINATES
-.4 .8  .7  1.75 -.1 1.0 0.5 1.6 1.2 3.3 .3  1.4 1.0 2.5 2.0 3.5
//GO.FT14F001 DD DSNAME=&SM,UNIT=SCRTCH,DISP=(NEW,PASS),
// SPACE=(800,(120,15)),DCB=(RECFM=VBS,LRECL=796,BLKSIZE=800)
//SMPLTTR EXEC PLOT,PLOTTER=PRINTER
//
```

Figure 9. Listing of PL/1 sample job#1 input deck.

```
SIMPLOTTER PARAMETER DUMP....PRINTER PLOTTER VERSION....04/13/70   (SCRANTON & MANCHESTER, COMPUTATION CENTER, I.S.U., AMES, IOWA)
                                    1. A SUMMARY OF PARAMETERS GIVEN SIMPLOTTER VIA CALL LISTS
                                    2. FIRST FOUR (X,Y) POINTS OF UP TO THREE DATA SETS
                                    3. SCALE FACTORS ACTUALLY USED BY SIMPLOTTER
                 EACH GRAPH IS REPRESENTED BY ONE BLOCK OF DATA

**********EXPANDED SIMPLOTTER V12/2/70-A;       PL/1          -SIMPLOTTER

     XSIZE       YSIZE   XSF       XMIN       YSF       YMIN        X-AXIS LABEL        Y-AXIS LABEL        GRAPH LABEL
     4.250       6.250  0.400E 00 -0.800E 00  0.0        0.0          X-COORDINATES       Y-COORDINATES      SAMPLE PROGRAM
DATA NO OF  SYM OPT
SET  POINTS BOL ION         (X1,Y1)                (X2,Y2)                (X3,Y3)                (X4,Y4)          DATA SET LABEL
  1     8     3   7  -0.400E 00, 0.800E 00  0.700E 00, 0.175E 01 -0.100E 00, 0.100E 01  0.500E 00, 0.160E 01  SUBROUTINE GRAPH
  2    10     0   2  -0.600E 00, 0.549E 00 -0.400E 00, 0.670E 00 -0.200E 00, 0.819E 00  0.0      , 0.100E 01

CALC XSF.   0.400E 00   CALC XMIN.  -0.800E 00   CALC YSF.   0.500E 00   CALC YMIN.   0.500E 00  TOTAL NO. OF DATA SETS REQ:  2


     XSIZE       YSIZE   XSF       XMIN       YSF       YMIN        X-AXIS LABEL        Y-AXIS LABEL        GRAPH LABEL
     8.250      -6.250  0.400E 00 -0.800E 00 -0.500E 00  0.0          X-COORDINATES       Y-COORDINATES      SAMPLE PROGRAM
DATA NO OF  SYM OPT
SET  POINTS BOL ION         (X1,Y1)                (X2,Y2)                (X3,Y3)                (X4,Y4)          DATA SET LABEL
  1    10     0   2  -0.600E 00,-0.600E 00 -0.400E 00,-0.400E 00 -0.200E 00,-0.200E 00  0.0      , 0.0          PREDICTED CURVE
  2     8     3 107  -0.400E 00,-0.223E 00  0.700E 00, 0.560E 00 -0.100E 00, 0.0        0.500E 00, 0.470E 00  MEASURED DATA

CALC XSF.   0.400E 00   CALC XMIN.  -0.800E 00   CALC YSF.   0.200E 01   CALC YMIN.  -0.100E 01  TOTAL NO. OF DATA SETS REQ:  2



*****PRINTER-PLOTTER DIAGNOSTIC MESSAGES*****
***NOTE: ESTIMATED TIME TO PLOT THIS GRAPH ON THE INCREMENTAL PLOTTER IS      3   MINUTES
```

Figure 10. SIMPLOTTER Parameter Dump produced by sample job#1.



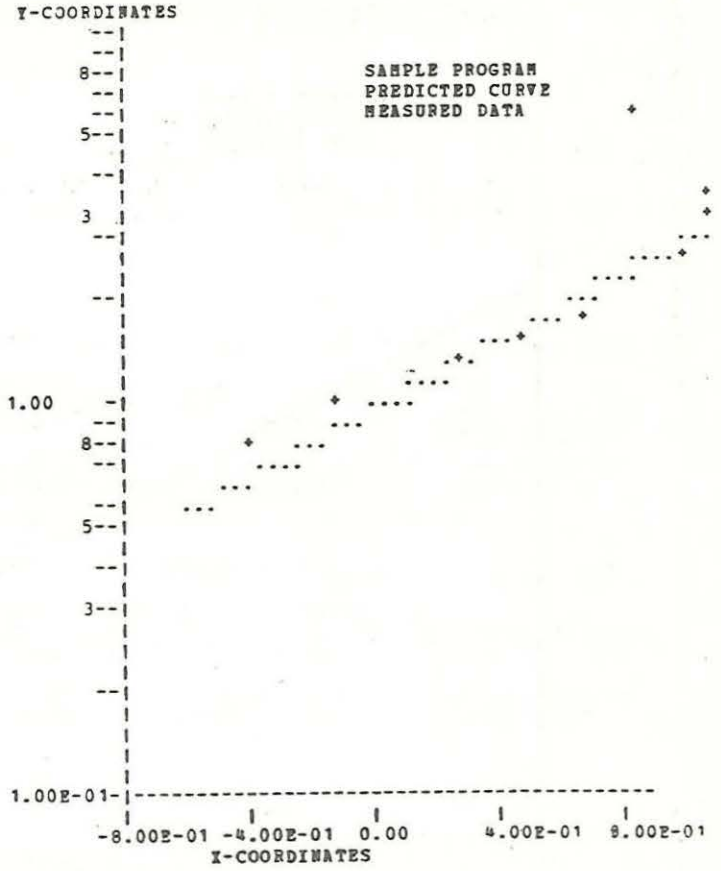


Figure 11. Printer plotter graph of sample job#1.

```
***** PRINTER PLOTTER ENDS *****
ALL PLOTS GENERATED IN THIS PROGRAM WOULD TAKE      3 MINUTES (ESTIMATE) TO PLOT ON THE INCREMENTAL PLOTTER.
YOU MAY USE THE INCREMENTAL PLOTTER BY CHANGING ONLY YOUR  EXEC PLOT  CARD AT THE BACK OF YOUR DECK TO:
         //STEP2  EXEC  PLOT,PLOTTER=INCRMNTL
```


Figure 12. Printer plotter termination message from sample job#1

Figure 13. Incremental plotter graph of sample job#1.

FORTRAN G Notes:

The labels passed in CALL lists must be terminated by a semi-colon if the label is less than 20 characters long.

The output of FORTRAN G sample job#1 is the same as the output of the PL/1 sample job#1. The parameter dump, printer plotter graphs, and incremental plotter graphs, are shown in Figures 10, 11, 12, and 13, respectively.

```
//C376#F4G JOB A0900,GIB
//S1 EXEC FORTG
//FORT.SYSIN DD    *
C
      REAL DATAX(8), DATAY(8), PREDX(10), PREDY(10), XL(5), YL(5)
C
      READ (5,10) XL,YL
      READ (5,20) (DATAX(I), DATAY(I), I=1,8)
C
      DO 100  I=1,10
      PREDX(I)=(I-4)/5.0
100   PREDY(I) = EXP(PREDX(I))
C
      CALL GRAPH (8, DATAX,DATAY, 3, 7, 4.25, 6.5, 0.4, -0.8, 0.0, 0.0,
     &             XL, YL, 'SAMPLE PROGRAM;', 'SUBROUTINE GRAPH;')
      CALL GRAPHS (10, PREDX,PREDY, 0, 2, ';')
C
      DO 200 I=1,8
200   DATAY(I) = ALOG(DATAY(I))
      DO 300  I=1,10
300   PREDY(I) = ALOG(PREDY(I))
C
      CALL GRAPH (10, PREDX,PREDY, 0,2, 4.25, -6.5, 0.4, -0.8, -0.5, 0.0,
     &            XL, YL, 'SAMPLE PROGRAM;', 'PREDICTED CURVE;')
      CALL GRAPHS (8, DATAX,DATAY, 3, 107, 'MEASURED DATA;')
C
   10 FORMAT (10A4)
   20 FORMAT (16F4.0)
      STOP
      END
//GO.SYSIN DD *
  X-COORDINATES          Y-COORDINATES
-.4 .8  .7  1.75 -.1 1.0 0.5 1.6 1.2 3.3 .3  1.4 1.0 2.5 2.0 3.5
//GO.FT14F001 DD DSNAME=&SM,UNIT=SCRTCH,DISP=(NEW,PASS),
// SPACE=(800,(120,15)),DCB=(RECFM=VBS,LRECL=796,BLKSIZE=800)
//SMPLTTR EXEC PLOT,PLOTTER=PRINTER
//
```

Figure 14. Listing of FORTRAN G sample job#1 input deck.

WATFIV Notes:

The labels passed in CALL lists must be terminated by a semi-colon if the label is less than 20 characters in length.

Notice that the FORTRAN source and data cards are the same as in the FORTRAN G sample program. Following good economical debugging procedure, programs can be debugged in WATFIV and switched to FORTRAN G AND H for production running with only a change of control cards.

The output of WATFIV sample job#1 is the same as the output of the PL/1 sample job#1. The parameter dump, printer plotter graphs, and incremental plotter graphs, are shown in Figures 10, 11, 12, and 13, respectively.

Note to non-I.S.U. users:

Some releases of WATFIV will not allow variable length labels to be passed through a CALL list. In such cases all label parameters passed to GRAPH must be 20 characters in length. The labels named XL and YL in the sample program are 20 characters in length, as is the literal '123456789ABCDEFGHIJK'. Label parameters 20 characters in length can be passed to GRAPH sucessfully in all versions of WATFIV and FORTRAN G and H.

```
//C376#W5G JOB A0900,GIB
//S1 EXEC WATFIV
//GO.SYSIN DD   *
$JOB                    GIB,TIME=5,PAGES=50
C
      REAL DATAX(8), DATAY(8), PREDX(10), PREDY(10), XL(5), YL(5)
C
      READ (5,10) XL,YL
      READ (5,20) (DATAX(I), DATAY(I), I=1,8)
C
      DO 100  I=1,10
      PREDX(I)=(I-4)/5.0
100   PREDY(I)  = EXP(PREDX(I))
C
      CALL GRAPH (8, DATAX,DATAY, 3, 7, 4.25, 6.5, 0.4, -0.8, 0.0, 0.0,
     &             XL, YL, 'SAMPLE PROGRAM;', 'SUBROUTINE GRAPH;')
      CALL GRAPHS (10, PREDX,PREDY, 0, 2, ';')
C
      DO 200 I=1,8
200   DATAY(I) = ALOG(DATAY(I))
      DO 300  I=1,10

300   PREDY(I) = ALOG(PREDY(I))
C
      CALL GRAPH (10, PREDX,PREDY, 0,2, 4.25, -6.5, 0.4, -0.8, -0.5, 0.0,
     &             XL, YL, 'SAMPLE PROGRAM;', 'PREDICTED CURVE;')
      CALL GRAPHS (8, DATAX,DATAY, 3, 107, 'MEASURED DATA;')
C
   10 FORMAT (10A4)
   20 FORMAT (16F4.0)
      STOP
      END
$ENTRY
  X-COORDINATES          Y-COORDINATES
-.4 .8  .7  1.75 -.1 1.0 0.5 1.6 1.2 3.3 .3  1.4 1.0 2.5 2.0 3.5
$STOP
//GO.FT14F001 DD DSNAME=&SM,UNIT=SCRTCH,DISP=(NEW,PASS),
// SPACE=(800,(120,15)),DCB=(RECFM=VBS,LRECL=796,BLKSIZE=800)
//SMPLTTR EXEC PLOT,PLOTTER=PRINTER
//
```

Figure 15. Listing of WATFIV sample job#1 input deck.

Sample Job#2 (COBOL)

Problem Definition:

Four sets of data representing a breakdown of unit record jobs by four catagories covering a 10 month period are to be plotted. The breakdown is to be by: keypunch, reproducer, card sorter, and card interpreter. A second graph is to be plotted displaying only the keypunch data.

Method:

The data is punched on 12 cards. The first contains the X axis label, Y axis label, graph label, and the data label for the keypunch data. The second data card contains the data labels for the reproducer, card sorter, and card inerpreter, respectively. The following 10 cards contain data for the four types of data. The X coordinate data (0,1,...9) is generated by the program.

GRAPH is called to establish a primary graph for the keypunch data (XSIZE positive). GRAPHS is called 3 times to superimpose the other sets of data. The user specifies scaling in the Y direction since it is known that a scale factor of 25.0 fits the data and 25.0 is not a "nice scaling number" in SIMPLOTTER's opinion. Data labels are "tagged" (MODE>100) by their plotting symbol.

A second CALL of graph is made to form a new graph containing the keypunch data alone.

Comments:

1. The first point of 'reproducer' and 'keypunch' data both occupy the same point on the printer plotter graph. The most recent superposition overlays the previous symbol at the contested print position. The order of superposition is not a factor when using the incremental plotter.
2. The third data point of 'card interpreter' falls outside the bounds of the user defined Y range and is plotted 1/2 inch above the Y axis as a debugging aid.
3. American National Standard COBOL does not support either character or arithmetic literals in CALL lists.

```
//C376#CBG JOB A0900,GIB
//STEP33 EXEC COBUCLG
//COB.SYSIN  DD  *
010010 IDENTIFICATION DIVISION.
010020 PROGRAM-ID. MAINPROG.
010070 ENVIRONMENT DIVISION.
010080 CONFIGURATION SECTION.
010090 SOURCE-COMPUTER. IBM-360-I65.
010100 OBJECT-COMPUTER. IBM-360-I65.
010110 INPUT-OUTPUT SECTION.
010120 FILE-CONTROL.
010130     SELECT CARD-FILE ASSIGN TO UT-2314-S-CARDIN.
010140 DATA DIVISION.
010150 FILE SECTION.
010160 FD  CARD-FILE
010170     RECORDING MODE F, BLOCK CONTAINS 0 RECORDS,
010180     LABEL RECORD IS STANDARD, DATA RECORD IS CARD-REC.
010190 01  CARD-REC     PICTURE X(80).
010200 WORKING-STORAGE SECTION.
010210 01  NPTS   PICTURE S9(8),          COMPUTATIONAL.
010220 01  KS     PICTURE S9(8), COMPUTATIONAL.
010230 01  MOD, PICTURE S9(8), COMPUTATIONAL.
010240 01  XSIZE, VALUE 4.5E+0, COMPUTATIONAL-1.
010250 01  YSIZE, VALUE 8.5E+0, COMPUTATIONAL-1.
020010 01  XSF    VALUE .0E+1,  COMPUTATIONAL-1.
020020 01  XMIN   VALUE .0E+1,  COMPUTATIONAL-1.
020030 01  YSF    VALUE 2.5E+1,  COMPUTATIONAL-1.
020040 01  YMIN   VALUE .0E+1,  COMPUTATIONAL-1.
020050 01  TX.
020060     02  ARAY COMPUTATIONAL-1.
020070         03  XT OCCURS 10  TIMES.
020080 01  TY.
020090     02  ARAY2 COMPUTATIONAL-1.
020100         03  YT OCCURS 10  TIMES.
020110 01  UY.
020120     02  YU, OCCURS 10 TIMES, COMPUTATIONAL-1.
020130 01  VY.
020140     02  YV OCCURS 10 TIMES, COMPUTATIONAL-1.
020150 01  WY.
020160     02  YW OCCURS 10 TIMES, COMPUTATIONAL-1.
020170 01  I    PICTURE S9(4)  VALUE ZERO, COMPUTATIONAL.
```

Figure 16. Listing of COBOL sample job#2 input deck.

```
020180 01   PNTS-VALUE.
020190      02  Y1, PICTURE 999.
020200      02  Y2, PICTURE 999.
020210      02  Y3, PICTURE 999.
020220      02  Y4, PICTURE 999.
020230      02  FILLER, PICTURE X(68).
020240 01   LABL-VALUE.
020250      02  XLAB       PICTURE X(20).
030010      02  YLAB       PICTURE X(20).
030020      02  GLAB       PICTURE X(20).
030030      02  DATLAB     PICTURE X(20).
030040 01   LABEL2.
030050      02  SUPER1, PICTURE X(20).
030060      02  SUPER2, PICTURE X(20).
030070      02  SUPER3, PICTURE X(20).
030080      02  FILLER, PICTURE X(20).
030090 PROCEDURE DIVISION.
030100 READ-1.
030110      OPEN INPUT CARD-FILE.
030120      READ CARD-FILE INTO LABL-VALUE; AT END, GO TO CALL-G.
030130      READ CARD-FILE INTO LABEL2; AT END, GO TO CALL-G.
030140 LOOP-1.
030150      READ CARD-FILE INTO PNTS-VALUE; AT END, GO TO CALL-G.
030160      ADD 1 TO I.                      MOVE Y1 TO YT (I).
030170      MOVE Y2 TO YU (I).               MOVE Y3 TO YV (I).
030180      MOVE Y4 TO YW (I).               COMPUTE XT (I) = I - 1.
030190      GO TO LOOP-1.
030200 CALL-G.
030210      COMPUTE NPTS = I.                COMPUTE MOD = 103.
030220      COMPUTE KS = 3.
030230      CALL 'GRAPH'  USING NPTS TX TY KS MOD XSIZE YSIZE XSF
030240          XMIN YSF YMIN XLAB YLAB GLAB DATLAB.
030250      COMPUTE KS = 4.
040010      CALL 'GRAPHS' USING NPTS TX UY KS MOD SUPER1.
040020      COMPUTE KS = 10.
040030      CALL 'GRAPHS' USING NPTS TX VY KS MOD SUPER2.
040040      COMPUTE KS = 11.
040050      CALL 'GRAPHS' USING NPTS TX WY KS MOD SUPER3.
040060      CALL 'GRAPH'  USING NPTS TX TY KS MOD XSIZE YSIZE XSF
040070          XMIN YSF YMIN XLAB YLAB GLAB DATLAB.
040080      CLOSE CARD-FILE.
040090      STOP RUN.
```

Figure 16. continued.

```
//GO.SYSOUT DD SYSOUT=A,DCB=(RECFM=FA,LRECL=121,BLKSIZE=121,BUFNO=1)
//GO.CARDIN  DD  *
 SAMPLE COBOL PLOT      Y-COORDINATES     SPECIAL SERVICES     KEYPUNCH;
REPRODUCER           CARD SORTER          CARD INTERPRETER
100100004152
070101024185
060092012226
125079011158
098072008161
073060002117
061044003104
098084005146
053106012165
107150012164
//GO.FT14F001 DD DSNAME=&SM,UNIT=SCRTCH,DISP=(NEW,PASS),
// SPACE=(800,(120,15)),DCB=(RECFM=VBS,LRECL=796,BLKSIZE=800)
//SMPLTTR EXEC PLOT,PLOTTER=PRINTER
//
```

Figure 16. continued.

```
SIMPLOTTER PARAMETER DUMP....PRINTER PLOTTER VERSION....04/13/70   (SCRANTON & MANCHESTER, COMPUTATION CENTER, I.S.U., AMES, IOWA)
                                    1. A SUMMARY OF PARAMETERS GIVEN SIMPLOTTER VIA CALL LISTS
                                    2. FIRST FOUR (X,Y) POINTS OF UP TO THREE DATA SETS
                                    3. SCALE FACTORS ACTUALLY USED BY SIMPLOTTER
                 EACH GRAPH IS REPRESENTED BY ONE BLOCK OF DATA

**********EXPANDED SIMPLOTTER V12/2/70-A;            COBOL-SIMPLOTTER

     XSIZE      YSIZE   XSF       XMIN       YSF       YMIN        X-AXIS LABEL          Y-AXIS LABEL          GRAPH LABEL
     4.500      8.500  0.0        0.0       0.250E 02  0.0          SAMPLE COBOL PLOT     Y-COORDINATES     SPECIAL SERVICES
DATA NO OF  SYM OPT
SET  POINTS BOL ION          (X1,Y1)              (X2,Y2)              (X3,Y3)              (X4,Y4)          DATA SET LABEL
  1    10     3 103   0.0      , 0.100E 03  0.100E 01, 0.700E 02  0.200E 01, 0.600E 02  0.300E 01, 0.125E 03  KEYPUNCH
  2    10     4 103   0.0      , 0.100E 03  0.100E 01, 0.101E 03  0.200E 01, 0.920E 02  0.300E 01, 0.790E 02  REPRODUCER
  3    10    10 103   0.0      , 0.400E 01  0.100E 01, 0.240E 02  0.200E 01, 0.120E 02  0.300E 01, 0.110E 02  CARD SORTER

CALC XSF.   0.200E 01   CALC XMIN.   0.0        CALC YSF.   0.250E 02   CALC YMIN.   0.0        TOTAL NO. OF DATA SETS REQ:  4


     XSIZE      YSIZE   XSF       XMIN       YSF       YMIN        X-AXIS LABEL          Y-AXIS LABEL          GRAPH LABEL
     4.500      8.500  0.0        0.0       0.250E 02  0.0          SAMPLE COBOL PLOT     Y-COORDINATES     SPECIAL SERVICES
DATA NO OF  SYM OPT
SET  POINTS BOL ION          (X1,Y1)              (X2,Y2)              (X3,Y3)              (X4,Y4)          DATA SET LABEL
  1    10    11 103   0.0      , 0.100E 03  0.100E 01, 0.700E 02  0.200E 01, 0.600E 02  0.300E 01, 0.125E 03  KEYPUNCH

CALC XSF.   0.200E 01   CALC XMIN.   0.0        CALC YSF.   0.250E 02   CALC YMIN.   0.0        TOTAL NO. OF DATA SETS REQ:  1


*****PRINTER-PLOTTER DIAGNOSTIC MESSAGES*****
***NOTE: ESTIMATED TIME TO PLOT THIS GRAPH ON THE INCREMENTAL PLOTTER IS      4   MINUTES
```

Figure 17. Simplotter Parameter Dump produced by sample job#2.

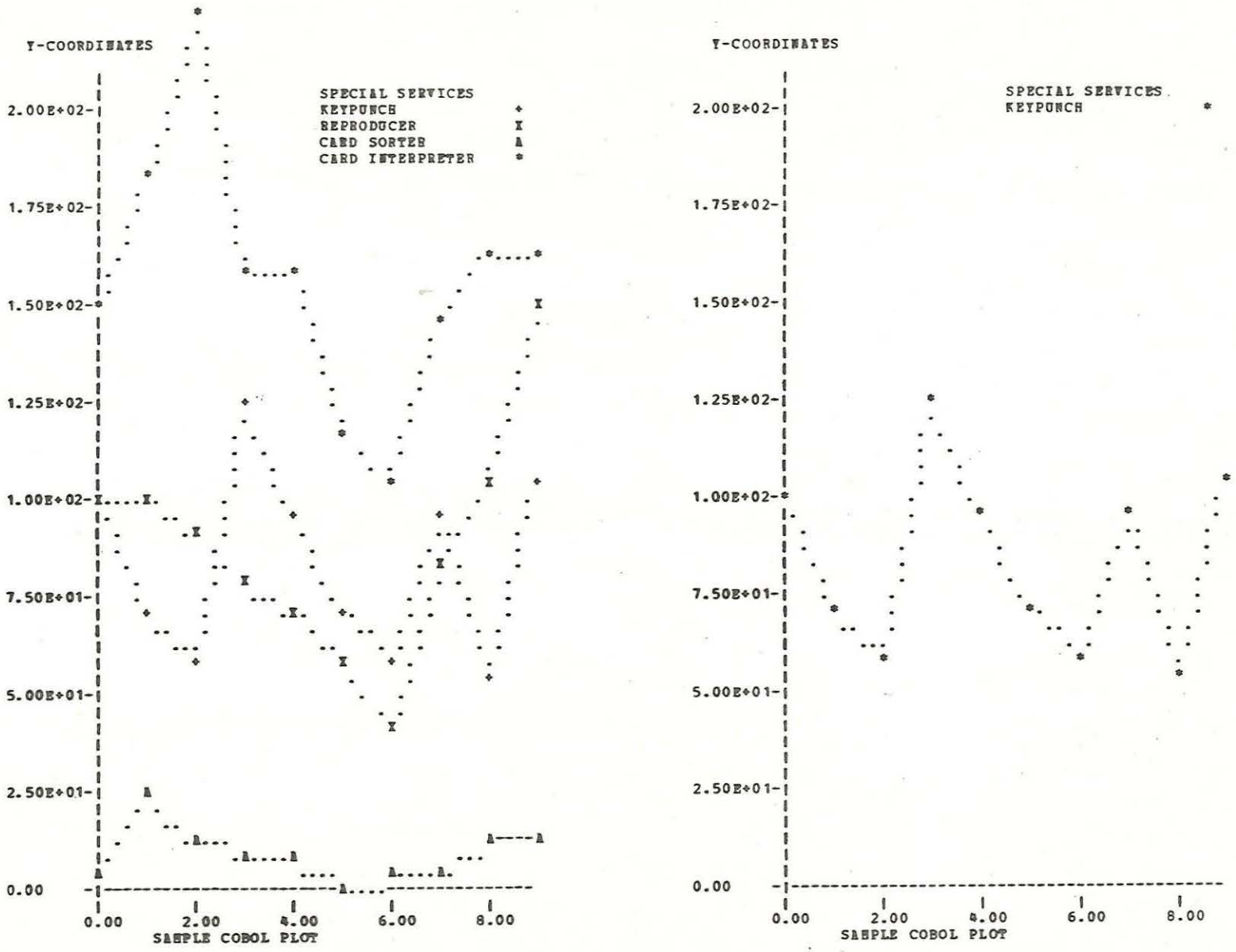


Figure 18. Printer plotter graph of sample job#2.

```
***** PRINTER PLOTTER ENDS *****
ALL PLOTS GENERATED IN THIS PROGRAM WOULD TAKE      4 MINUTES (ESTIMATE) TO PLOT ON THE INCREMENTAL PLOTTER.
YOU MAY USE THE INCREMENTAL PLOTTER BY CHANGING ONLY YOUR  EXEC PLOT  CARD AT THE BACK OF YOUR DECK TO:
          //STEP2  EXEC  PLOT,PLOTTER=INCRMNTL
```


Figure 19. Printer plotter termination message from sample job#2.

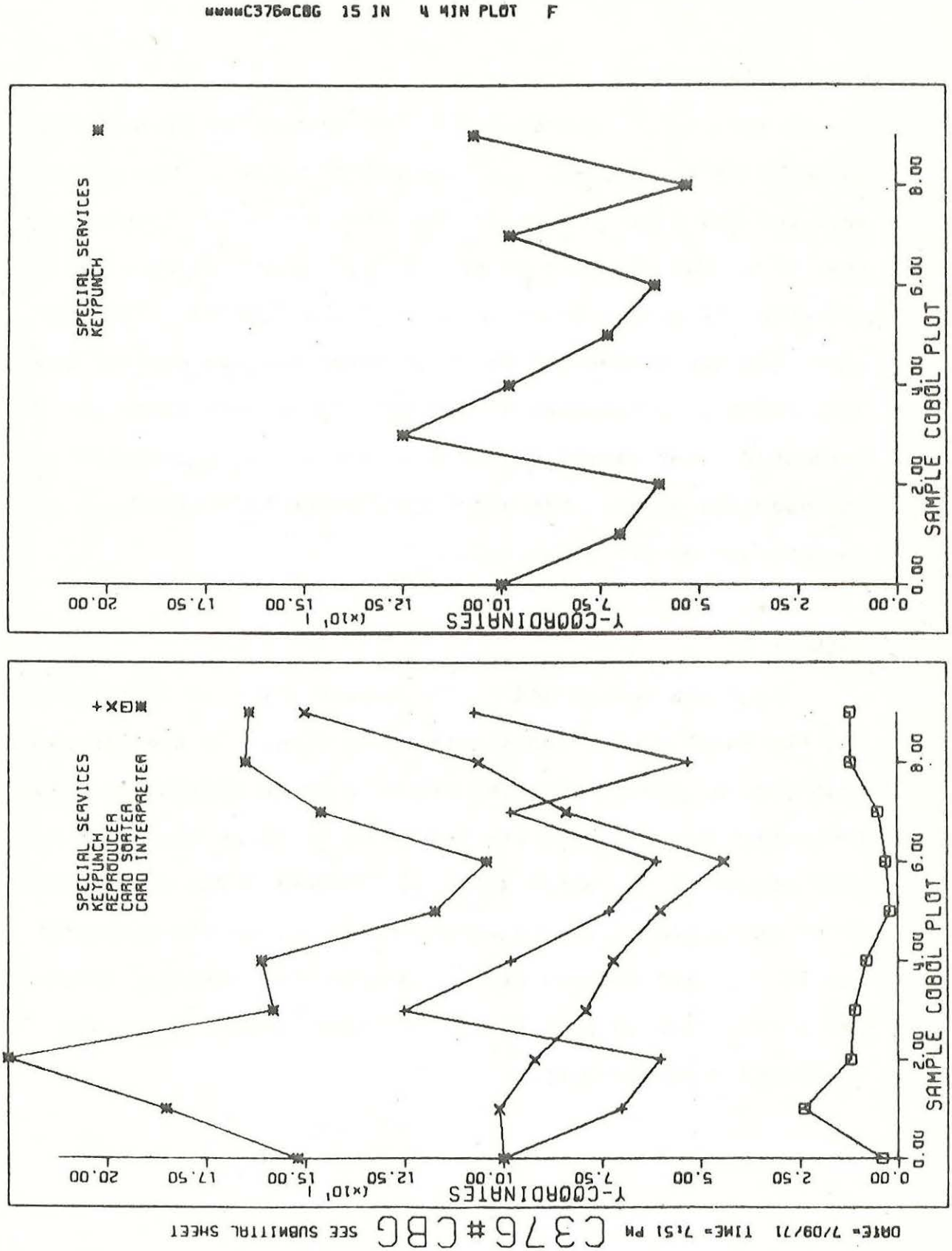


Figure 20. Incremental plotter graph of sample job#2.

Sample Job#3 (PL/1, WATFIV, FORTRAN G AND H, and COBOL)

Problem Definition:

A set of 256 consecutive Y coordinates has been punched on cards by an automatic data recording device. The corresponding 256 X cordinates of the data are to be program generated such that adjacent points will be separated by a constant distance. A second graph is to be drawn directly above the first showing a smoothed curve representing the path of the data points. As several of the data points are known to be erroneous, they should not be included in the calculation of the smoothed curve. The erroneous values have zero Y coordinates on the input cards.

Method:

The graph labels and Y coordinates are read from cards and the associated X coordinates generated. As the points are numerous, a reduced plotting symbol size is specified (ORIGIN, LATCH=5). Graph labels are specified to be in the upper left hand corner of the graph (ORIGIN, LATCH=6). The second graph is to be located directly above the first, so the automatic origin movement between graphs is cancelled (ORIGIN, LATCH=1). GRAPH is CALLed to plot the points with automatic scaling specified (XSF=YSF=0.0).

The origin for the second graph is moved vertically upward 4.5 inches and the X axis is marked for omission (ORIGIN, LATCH=2). GRAPH is CALLed specifying a 5th level smoothed curve (|MODE|=25) calculated on the basis of all non-zero data points (MODE negative). LETTER is used to draw a message on the second graph.

Comments:

As the usage of the SIMPLOTTER routines is similar in all the languages, only the WATFIV listing and its incremental plot are included. Sample job#1 and job#2 illustrate the particulars for each of the languages.

```
//C376AW50 JOB A0900,GIB
//S1 EXEC WATFIV
//GO.SYSIN DD *
$JOB                    GIB,TIME=10,PAGES=50
      REAL*4      X(256),Y(256),XL(5),YL(5),GL(5),DL(5), MESSG(20)
      DO 30 I=1,256
   30 X(I)=I
      READ (5,100)  XL,YL,GL,DL
      READ (5,200) Y
      CALL ORIGIN (0.05, 0.0, 5)
      CALL ORIGIN (0.2, 3.6, 6)
      CALL ORIGIN (0.0, 0.0, 1)
      CALL GRAPH(256, X,Y, 13, 7, 6.5, 4.0, 0.0, 0.0, 0.0, 0.0,
     &                                                  XL, YL, GL, DL)
      CALL ORIGIN (0.0,4.5,2)
      READ (5,100) XL,YL,GL,DL
      CALL GRAPH (256, X,Y, 0, -25, 6.5, 4.0, 0.0, 0.0, 0.0, 0.0,
     &                                                  XL, YL, GL, DL)
      READ (5,100) MESSG
      CALL LETTRS (3.0,  1.0, 0.15, MESSG,  0.0, 80)
      STOP
  100 FORMAT (20A4)
  200 FORMAT (16F5.0)
      END
$ENTRY
CHANNELS              NO. OF COUNTS        SAMPLE JOB#3         NO POINT ELIMINATION
0000000000000000000000000000000000000000000010000000000000010000000000000000001
0000000001000000000000002000000000000000000010000100000000000000000010000100000
0000000001000060002300004000000000100000000020001100000002000024100280002790035 7
0040300412004510043900432004300041300428000000046800470004640042500446003800042 7
0040600355003580032200340003470029000285002600025500000002550030300241002740025 4
0029800269002890026600323003560040300482005120063000785009540110001484017030202 8
0226602690028010298603089029590288402796023940212501874014830130500938007540056 6
0000000322002580018300115000760006800048000400002200030000250002200025000190001 5
0003700000000000000000000000000000000000000000000000000000000C000000000000000000
0000000000000000000C000000000000000000000000000000000000000000000000000000000100000
0000000000000020001800004000000000100001000010000500092001620017800196002320022 1
0000000322003000034300287003430033000312003400031300310003400032100309003050028 7
0027300253002610022500224002130021100215002160019300181001750020400193001730018 8
0021300213002080024000240002630027900324003990045500604007200091601038013490150 9
0178501982021490215200000022840212601893017960152201289010700086200661004890037 1
0026000162001390011600076000550004000027000190002400017000210001800017000150001 5
CHANNELS              NO. OF COUNTS        SAMPLE JOB#3         ZERO PTS. ELIMINATED
      IS NOT THOU ART PLOTTER FUN?;  $$$
      $STOP
      //GO.FT14F001 DD DSNAME=&SM,UNIT=SCRTCH,DISP=(NEW,PASS),
      // SPACE=(800,(120,15)),DCB=(RECFM=VBS,LRECL=796,BLKSIZE=800)
      //SMPLTTR   EXEC PLOT,PLOTTER=INCRMNTL,FORM=F
```

Figure 21. Listing of WATFIV sample job#3 input deck.

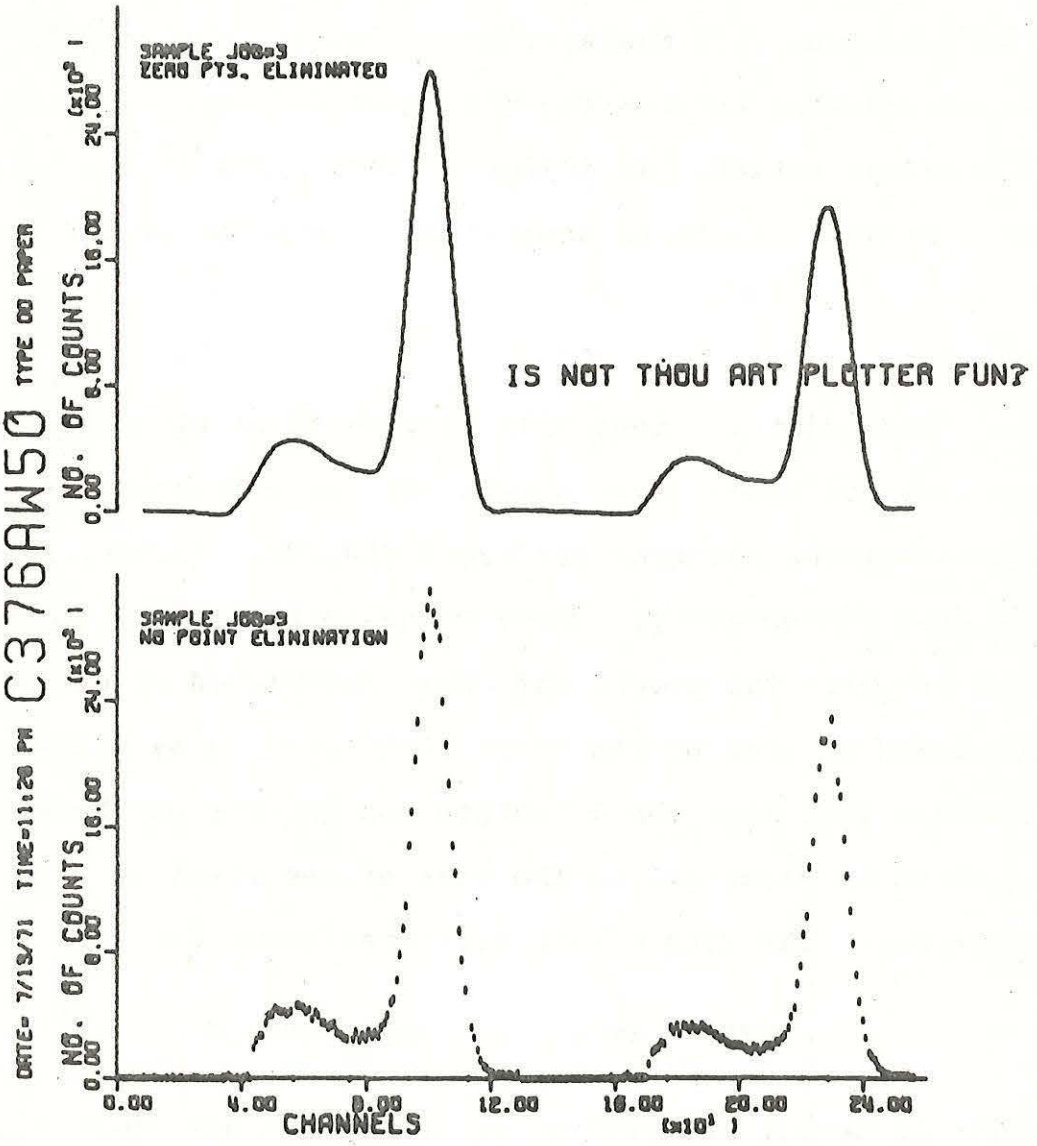


Figure 22. Incremental plotter graph of sample job#3.

Sample Job#4 (PL/1, WATFIV, FORTRAN G&H, and COBOL)

Problem Definition:

A set of measured data is to be plotted as a distribution. The first graph is to show the data in histogram and ideogram form, with the abscissa along the horizontal. The second graph is to show the histogram with the abscissa in the vertical direction. An ideogram, normalized to the area of the histogram, is to be superimposed upon the second graph.

Method:

Three distributions will be created simultaneously. The abscissa and height work arrays for the histogram, ideogram, and normalized ideogram are named (AH,HH), (AI,HI), and (AN,HN), respectively. Three CALLs to DISTRI initialize the work arrays. The events and their associated errors are read and added to each of the three distributions by CALLing DISTRA. Note that the weighting factor, WT, of the normalized ideogram is specified as the area of one event on the histogram. The area of one histogram event is:

```
AREAH = WT*(AMAX - AMIN)/INTVLS
```

which is merely the product of the height and 'bar' width of one histogram event.

Comments:

Notice that the un-normalized ideogram is much smaller than the corresponding histogram on the first graph. Histograms are constructed on the basis of each event having a specified height; ideograms are constructed on the basis of each event having a specified area. The weighting factor, WT, provides a convenient means of normalizing histograms and ideograms to each other as is illustrated by the sample job.

As the usage of the SIMPLOTTER routines is similar in all the languages, only the WATFIV listing and its incremental plot are included. Sample job#1 and job#2 illustrate the particulars for each of the languages.

```
//C376AW50 JOB AC900,GIB
//S1 EXEC WATFIV
//GO.SYSIN DD *
$JOB                    GIB,TIME=10,PAGES=50
      REAL AH(30),HH(30),AI(102),HI(102),AN(102),HN(102)
      REAL XL(5),YL(5),GL(5),DL(5)
      CALL DISTRI (AH,HH,  20, 100.0, 500.0)
      CALL DISTRI (AI,HI, 100, 100.0, 500.0)
      CALL DISTRI (AN,HN, 100, 100.0, 500.0)
      DO 100 I=1,10
        READ (5,901)  A,SIGMA
        CALL DISTRA (AH,HH, 1, 0.0,   1.0)
        CALL DISTRA (AI,HI, A, SIGMA, 1.0)
        CALL DISTRA (AN,HN, A, SIGMA, 20.0)
  100 CONTINUE
      READ (5,902) XL,YL,GL,DL
      CALL DISTRP (0.0,AH,HH, 0, 4, 4.0, 6.0, 0.0, 0.0, 0.0, 0.0,
     &                                    XL,YL,GL,DL)
      READ (5,902) DL
      CALL DISTRS (0.0, AI, HI, 0, 102, DL)
      READ (5,902) XL,YL,GL,DL
      CALL DISTRP (90.0,AH,HH, 0, 4, 6.0, 4.0, 0.0, 0.0, 0.0, 0.0,
     &                                    XL,YL,GL,DL)
      READ (5,902) DL
     CALL DISTRS (90.0,AN,HN, 0, 102, DL)
      STOP
  901 FORMAT ( 2F10.0)
  902 FORMAT (20A4)
      END
$ENTRY
325.0      13.0
125.0      100.0
303.0      15.0
575.0      70.0
340.0      17.0
375.0      15.0
270.0      15.0
653.0      90.0
300.0      13.0
398.0      25.0
 ABSCISSA;          ;                    SAMPLE JOB#4;       HISTOGRAM;
AND IDEOGRAM;
;                   ABSCISSA;            SAMPLE JOB#4;       HISTOGRAM AN
NORM. IDEOGRAM;
$STOP
//GO.FT14F001 DD DSNAME=&SM,UNIT=SCRTCH,DISP=(NEW,PASS),
// SPACE=(800,(120,15)),DCB=(RECFM=VBS,LRECL=796,BLKSIZE=800)
//SIMPLTTR EXEC PLOT,PLOTTER=INCRMNTL,FORM=F
```

Figure 23. Listing of WATFIV sample job#4 input deck.

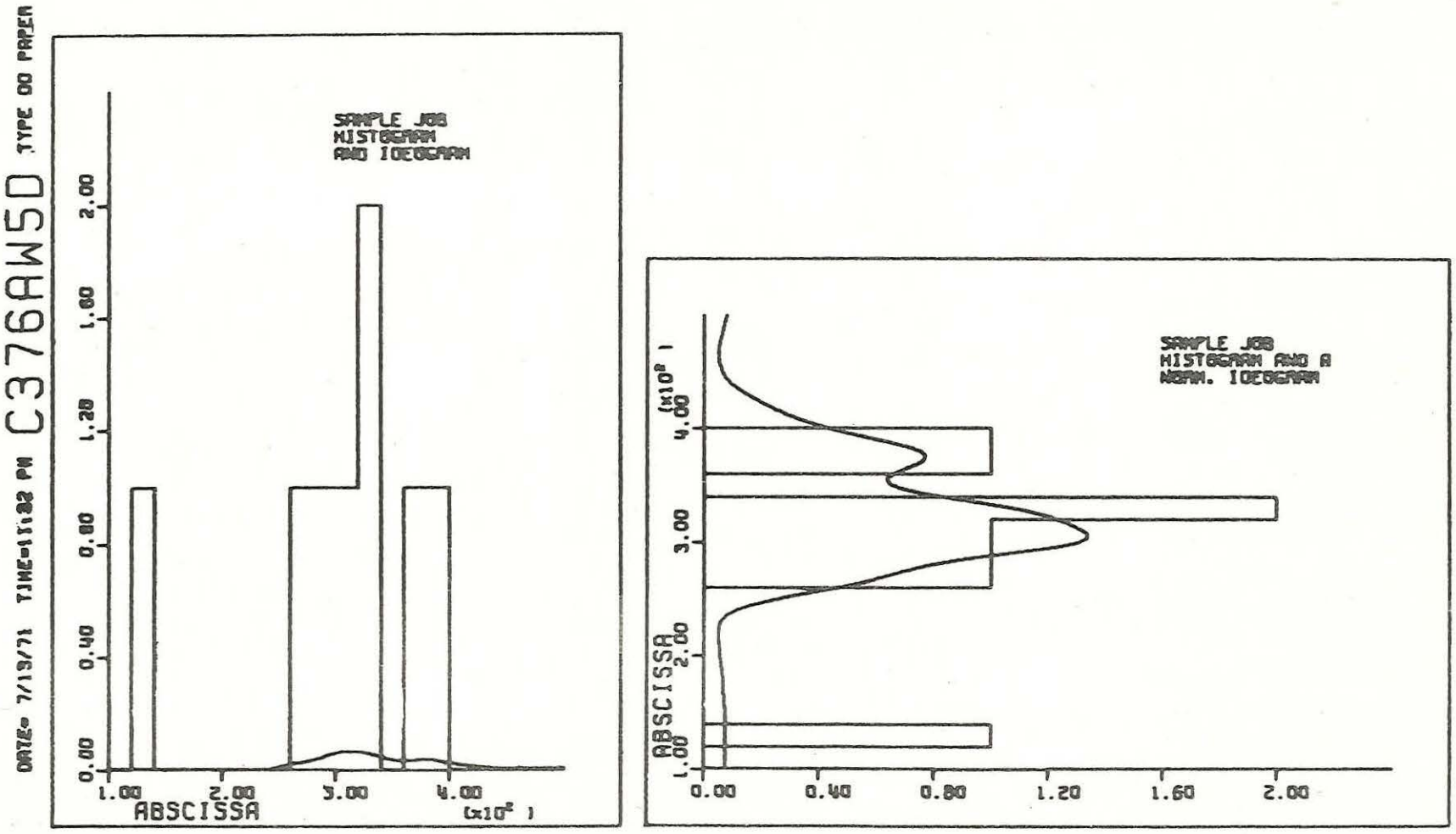


Figure 24. Incremental plotter graph of sample job#4.