ccm 73-01

GRAFIT USER NOTES

February 1973

OSU
COMPUTER CENTER
Oregon State University
Corvallis, Oregon 97331

GRAFIT USER NOTES

ccm 73-01

by

Jeff Ballance
Jo Ann Baughman
Larry Hubble

ACKNOWLEDGMENTS

We wish to express our appreciation to Lyle Ochs for his advice and helpful suggestions during the design of the system and during the writing of this manual. We also wish to thank the many users who have been so patient and helpful during the implementation of the system.

This work was supported in part by a National Science Foundation grant (GJ28453) for which we are grateful.

TABLE OF CONTENTS

NOTES ON THE GRAFIT SYSTEM

Introduction

The GRAFIT system is an interactive program for displaying data on the Tektronix terminal and/or the Calcomp plotter or on a Hewlett-Packard teletype compatible X-Y plotter and/or the Calcomp plotter. The system contains a small function translator which allows one to define functions on-line. These functions may be specified by a formula or as the solution to a system of differential equations. Commands exist for defining array storage, reading and writing data files, and for plotting functions and arrays. One of the major features of the system is the capability of defining one's own commands. This is done by simply supplying a FORTRAN compatible subroutine which carries out the desired operations.

General Comments

The GRAFIT system is stored as an overlay on the public file *GRAFIT and may be called by typing the file name as a control mode statement. Since GRAFIT may be run from a teletype or Tekterminal, the following question is asked upon entering the system:

ARE YOU AT A TEKTERMINAL?

This question must be answered with a YES or NO depending upon the device being used. After this question is answered, the program will signal that a command may be entered by printing a ">" and ringing the bell.

After the ">" has been printed and the bell rung, a command or function definition may be given. A description of each of the commands is given on the following pages.

Functions are entered in a notation similar to that of ordinary mathematics and that used by programming languages such as FORTRAN. A complete description of the language for function definitions is given in Appendix I.

The size of the area used for plotting on the various devices may be determined by the user and may differ if plotting is being done on two devices simultaneously.

The plotting area may be divided into a maximum of four plotting regions. Each of these regions may contain a set of axes and curves independent of the other plotting regions.

Shown below is the sequence of commands necessary to define and plot a function. The plot produced is also shown. (Items typed by the user are underlined throughout this document.)

```
>F(T) = A*T↑2 + SIN(W*T)
      A = 2
      W = 6
>PLOT,F
    FULL DOMAIN SPECIFICATIONS
    INITIAL VALUE = -3.0
    FINAL VALUE = 3.0
    NUMBER OF POINTS = 101
```

Typing errors may be corrected by using the backslash (\) to delete the previous character or the at sign (@) to delete the current line of input. If an input string is too long to fit on one line, it may be continued on the next line by pressing the line feed (LF) key at the end of a line. Input will be accepted from the next line after a carriage return has been outputted and the bell rung.

If plotting is to be done on the Calcomp plotter, logical unit number 10 must be equipped to the plotter and labeled before entering GRAFIT.

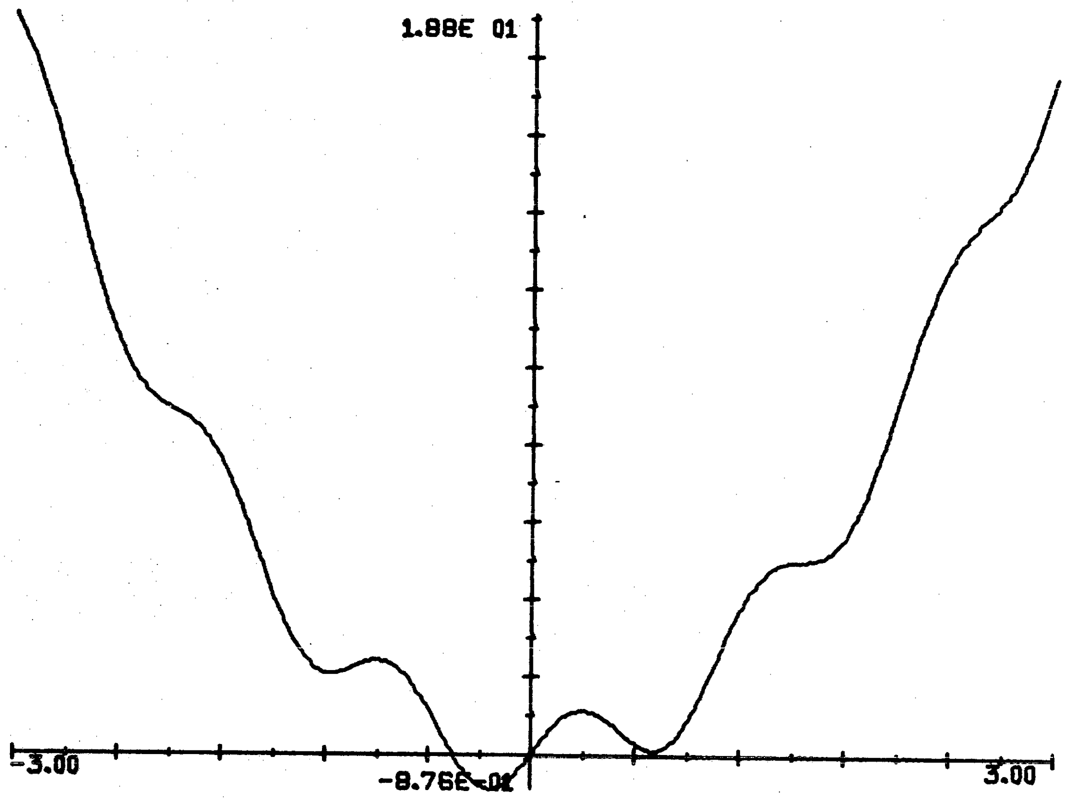


Figure 1. The plot produced by the sequence of commands given above.

DESCRIPTION OF THE COMMANDS

```
ARRAY,<array name>(<size>),...
```

The ARRAY command is used to reserve storage for singly subscripted variables. The type (real or integer) is determined by the first character of the name, i.e., names with first characters I-N are integer and names beginning with the other letters are real.

In addition to defining array storage, the command may be used to initialize an array to have functional values. The general form of the command to do this is:

```
ARRAY,<array name>(<size>) =
<function name>(<initial value>;
<final value>)
```

where the function name is the name of a previously defined function. The first argument of the call is assumed to be the initial value of the independent variable of the function. Other arguments preceding the semicolon are assumed to be fixed parameters. The semicolon and final value are optional; but if specified, will cause the entries in the array to correspond to:

$$f(t_i),\ f(t_i + \Delta t),\ldots,f(t_f)$$

where t_i is the inital value specified, t_f is the final value specified, and Δt is the increment calculated by:

$$\Delta t = \frac{t_f - t_i}{[(\text{array dimension})-1]}$$

For example, the command

```
ARRAY,A(101)=SIN(0;1)
```

would define the array A to have 101 floating-point locations and would initialize the array to contain the values of the function SIN beginning at 0, with increment 0.01.

If the final value is omitted in the function argument list, an increment of one is used.

An array may be initialized to have the same values as another array with the above form of the command. The argument list on the second array is ignored. Hence, the two arrays must be of the same dimension.

Examples:

```
ARRAY,A(101),B(26)
ARRAY,C(51) = F(0;3)
     where F is a function that has already been defined.
ARRAY,D(51) = C
     where C is the array defined in the previous example.
```

AUTOSCALE,< plotting region>,...

The AUTOSCALE command will cause the ranges of the axis to be automatically determined, i.e., the data will be scanned to pick the maximum and minimum values. The origins of the axis are determined as follows: a) if zero is in the range, the origin is set to zero; b) if zero is not in the range, the origin is set to the low value. If the range is zero, the low value will be set to -1.0, the high value will be set to 1.0, and the origin will be set to -1.0.

The plotting area may be divided by the HALF and QUARTER commands (see pages 14, 26) into plotting regions. Axis and curves drawn in each of the regions are produced independently. Thus, the plotting region parameter allows the ranges to be determined automatically in the regions specified. The plotting region mnemonics are those defined by the HALF and QUARTER commands (see pages 14, 26).

Examples of the AUTOSCALE command:

AUTOSCALE

> The ranges for the axis in all of the plotting regions will be automatically determined.

AUTOSCALE,UL,LR

> The ranges for the axis to be drawn in the upper left (UL) and lower right (LR) corners of the plotting area will be automatically determined. These plotting regions would have been defined by a QUARTER command.

COMMAND,<subroutine name>(<arg_1>,...,<arg_n>),<file or library containing the binary deck>

This command allows a user to incorporate FORTRAN compatible subroutines into the system as commands. Once defined as commands these subroutines can be executed by simply typing the subroutine name and the actual argument list.

The information that must be given in the COMMAND statement is described below:

a) The subroutine name may be that of any FORTRAN compatible subroutine and may be no more than eight characters in length.

b) The list of arguments enclosed in parenthesis following the subroutine name in the COMMAND statement must agree in type (real or integer) and number with the actual parameter list.

c) The information following the argument list in the COMMAND statement is assumed to be the name of a file containing the binary deck of the subroutine and/or a loader library specification. If the binary deck is stored on a loader library, the file name must be specified using the library specification, e.g., LIB=*ARAND where *ARAND is a loader library file. If a file name is given without a library specification, an attempt will be made to load all of the decks on the file. If this information is omitted, it will be requested.

Before giving some examples of the COMMAND statement, the following restrictions and notes should be made:

a) Subroutines used as commands <u>cannot</u> use unlabeled COMMON or the COMMON/DATA storage areas. However, subroutines may use labeled common blocks.

b) Subroutines used as commands <u>cannot</u> have more than twenty-four arguments.

Examples of the COMMAND statement are given below:

```
COMMAND,SUMSQ(A,N,SQ),*SUMSQB
COMMAND,AVERG(A,N,AV)
    ENTER THE FILE NAME:  *AVERAGE
COMMAND,SMO(A,L,WGT,M,L,B),LIB=*ARAND
```

After a subroutine has been declared as a command, it may be executed by typing the subroutine name and the actual argument list. For example, suppose the subroutine AVERG in the examples above had been made a command. Typing

```
AVERG(A,100,AV)
```

would compute the average of the 100 data points stored in the array A and return the result in the parameter AV.

DEVICE,<plotting device>

The DEVICE command is used to change the plotting device. The plotting device modifiers allowed are determined by the terminal from which the program is being run. If the input is coming from a Tekterminal, the options are:

A) CALCOMP - Plotting will be done for the Calcomp plotter. The plotting information will be written on logical unit 10, which must be equipped to the plotter or to a file which will be copied to the plotter at a later time.

B) TEK - Plotting will be done on the Tekterminal screen.

C) BOTH - Plotting will be done for both the Calcomp and Tekterminal.

If the input is coming from a teletype, the options are:

A) CALCOMP - Plotting will be done on the Calcomp plotter. The plotting information will be written on logical unit 10, which must be equipped to the plotter or to a file which will be copied to the plotter at a later time.

B) HP - This option will cause information to be output in a form suitable for a Hewlett-Packard teletype compatible X-Y plotter. If the terminal is not equipped with such a plotter, the data will be printed on the terminal.

C) BOTH - Plotting will be done for both the Calcomp and Hewlett-Packard plotters.

Notes:
1) When plotting is done on the Calcomp, the EXIT command should be used to terminate the program.
2) The ERASE command must be used before a DEVICE command in which the plotting device is changed from CALCOMP or BOTH to TEK or HP. If this is not done, a portion of the Calcomp plot will be lost.

DOMAIN,<plotting region>,< INITIAL value>,< FINAL value>,< NUMBER of points>,...

This command is used to specify the domain of the independent variable for the plot in the region specified by the user. When plotting is done in this region, the independent variable will take on the values $t_i, t_i+\Delta t, \ldots, t_f$ where

$$t_i = \text{initial value},$$
$$t_f = \text{final value, and}$$
$$\Delta t = \frac{t_f - t_0}{[(\text{number of points})-1]}$$

The plotting region parameter must be used when more than one set of axis is to be produced on the screen. If the plotting region parameter is omitted, the command is assumed to refer to the region plotted in by the first plot command following a FULL, HALF, or QUARTER command. See pages 14 and 26 for a description of the order in which the plotting regions are used.

The mnemonics used for the plotting region parameter are described in the discussion for the QUARTER and HALF commands.

If the DOMAIN command is given without any parameters, the domain specifications will be requested for all regions in which plotting is to be done. For example, if the full plotting area was to be occupied by one plot, the following sequence would occur:

```
DOMAIN
    FULL DOMAIN SPECIFICATIONS
    INITIAL VALUE = 0.0
    FINAL VALUE = 1.0
    NUMBER OF POINTS = 101
```

where the underlined quantities were entered by the user.

Further examples of the DOMAIN command are:

DOMAIN,UL,-1,1,201

This command specifies the domain of the plot in the upper left quarter of the plotting area.

DOMAIN,0,,301

Since no plotting region is specified, this command specifies the domain of the first region in the plotting area in which plotting is done. The previously specified final value will be left unchanged since it was omitted.

ERASE

The ERASE command causes the screen to be erased. Following the screen erasure, commands will be requested at the top of the screen.

ERSPLOT,<plotting region>,...,<n>,< m>

The ERSPLOT command is used to clear the plotting area and replot, for selected plotting regions, the functions specified by previous PLOT commands. The parameters n and m allow families of curves to be generated. The n parameter is the number of increments (may be non-integral) to add to each of the varying parameters before the first curve is plotted. The m parameter is the number of successive curves to generate.

For example, suppose the function F(T) is plotted on the full plotting area and contains a varying parameter "A" with a current value of 1.0 and with an increment of 0.5. The command

```
ERSPLOT,3,2
```

would cause the screen to be erased and two curves plotted corresponding to the function F(T) evaluated with A having the value 2.5 and 3.0 respectively.

Other examples of the ERSPLOT command are given below:

ERSPLOT,UL,,2
: After erasing the screen, the functions specified to be plotted in the upper left (UL) corner of the plotting area are evaluated for the current value of the varying parameters and plotted; the varying parameters are then incremented and the functions re-evaluated and plotted.

ERSPLOT
: All of the functions specified to be plotted by previously given PLOT commands are evaluated, using the current values of the varying parameters, and plotted in their respective plotting regions.

EXIT

The EXIT command may be used to terminate the program. Either the EXIT command or the ERASE command must be used before terminating plotting on the Calcomp plotter. If neither of these commands is given, a portion of the Calcomp plot may be lost.

FORMAT,<F or E><column width>.<number of digits to follow the decimal point>

This command allows the user to specify a FORTRAN F or E format for numbers written by the WRITE command (see page 33). The WRITE command produces records which contain one item from each of the elements in the list in the corresponding order. Each item will be written in the specified format. In specifying the format one should be careful to specify a field width wide enough to allow at least one blank between successive values on a line if the values are to be read by the READ command. If no FORMAT command has been given prior to a WRITE command, an E12.4 format will be used.

Examples of the FORMAT command are:

```
FORMAT,F6.2
FORMAT,E16.8
```

FULL

The FULL command resets the plotting regions so that one set of axis will fill the entire plotting area defined by the SIZE command (see page 30 for a description of the SIZE command). Successive PLOT commands will cause the screen to be erased and the plot produced, again filling the entire plotting area as currently defined by the SIZE command.

Note: Upon entering the system, the size of the plotting area is 5.5" wide and 4" high, and the plotting region is defined to fill the entire area.

HALF,<region for 1st plot>,<region for 2nd plot>

The HALF command allows the plotting area as currently defined by the SIZE command to be divided into two separate regions where plotting may be done. The plotting area may be divided vertically to give a left half (LH) and right half (RH) split or horizontally to give a top half (TH) and bottom half (BH) split. If no parameters are given on the HALF command, the horizontal division is assumed, which is equivalent to the command

```
HALF,TH,BH
```

Examples of the HALF command are:

```
HALF,LH,RH
```

The vertical division of the plotting region is made. The first set of axes, specified by the first PLOT command following the HALF command, will go on the left half (LH) of the plotting area; and the second set of axes, specified by the second PLOT command following the HALF command, will go on the right half (RH) of the plotting area.

```
HALF,TH
```

The horizontal division of the plotting region is made. Only one set of axes may appear in the plotting area at a time. Each successive PLOT command will cause the screen to be erased and a new set of axes to be drawn on the top half of the plotting area.

Notes: 1) The order of the region specifiers (TH,BH,LH,RH) determines the order in which the regions will be used by successive PLOT commands.

2) Either one or two regions may be specified--the graphs actually being produced with one or two successive PLOT commands respectively.

3) After all of the specified regions have been filled by successive PLOT commands, another PLOT command will cause the screen to be erased and the functions specified plotted in the first specified region. For example:

```
HALF,LH,RH
     The plotting area is divided vertically.
PLOT,F
     The function F will be plotted in the
     first region (LH).
PLOT,G
     The function G will be plotted in the
     second region (RH).
PLOT,H
     The function H will be plotted in the
     first region (LH) after the screen
     has been erased.
```

INCPLOT,<plotting region>,...,<n>,<m>

The INCPLOT command is used to generate families of curves in selected plotting regions from functions specified in previous PLOT commands. The parameters n and m allow families of curves to be generated. The n parameter is the number of increments (may be non-integral) to add to each of the varying parameters before the first curve is plotted. The m parameter is the number of successive curves to generate.

For example, suppose the function F(T) is plotted on the full plotting area and contains a varying parameter "A" with a current value of 1.0 and with an increment of 0.5. The command

```
INCPLOT,3,2
```

would cause the two curves to be plotted corresponding to the function F(T) evaluated with A having the value 2.5 and 3.0 respectively.

Other examples of the INCPLOT command are given below:

INCPLOT,UL,,2

The functions specified to be plotted in the upper left (UL) corner of the plotting area are evaluated for the current value of the varying parameters and plotted; the varying parameters are then incremented and the functions re-evaluated and plotted.

INCPLOT

All of the functions specified to be plotted by previously given PLOT commands are evaluated, using the current values of the varying parameters, and plotted in their respective plotting regions.

INPUT,<logical unit number or file name>

The INPUT command allows commands and other inputs to be read from a file. This command would be used if the same sequence of statements was to be used many times. The sequence of commands would be entered in the EDITOR and stored on a file and then called in using INPUT. When an end-of-file is sensed, input is again accepted from the terminal. The INPUT command can only accept information from the EDITOR that was saved with the OUT command. Do not FILE or COUT information in the EDITOR that is to be used by the INPUT command.

KILL

The KILL command may be used to clear the computer memory occupied by commands, arrays, and functions. The forms of the command are described below:

A) KILL,FUNCTIONS
All user defined functions and arrays are deleted. The storage area and names may be reused.

B) KILL,COMMANDS
All user defined commands are deleted.

C) KILL
This form of the command causes both A) and B) to be executed.

Note: Form B) or C) of the KILL command should be used if an error occurs while defining a command which places the user in control mode. When placed in control mode, one can return to GRAFIT by giving the MI control mode command.

LABEL,<plotting region>;<alphanumeric information>;...

The LABEL command allows the use of graphics input to position a message on the display or to get the coordinates of points on the graph. The particular option chosen is signaled by the character typed after the crosshairs have been turned on and positioned. The options are described below:

A) DEL - Pressing the DEL key signals that a series of straight line segments is to be drawn. The crosshairs will be turned on again and may be positioned. Pressing the DEL key again will cause a line to be drawn between the last and the current point signaled with the DEL.

B) SPACE BAR - Pressing the SPACE BAR signals that the coordinates of the point are to be printed. After the SPACE BAR has been pressed in response to graphics input, the crosshairs are turned on again so that the printed coordinates may be positioned away from the curve. After repositioning the crosshairs, pressing any key will cause a line to be drawn from the point signaled by the SPACE BAR to the current position and the coordinate pair (x,y) printed beginning at the current position.

C) Any other key - Pressing any other key will terminate the graphics input sequence. If a message is to be printed, it will be drawn starting at the current position of the crosshairs.

The plotting region must be specified if more than one is being used (the first plotting region is assumed if none is specified), to insure that the coordinates will be printed in the correct units.

If plotting is not being done on the Tekterminal, only alphanumeric information may be placed on the plotting regions. This information will be drawn beginning at the upper left-hand corner of the specified plotting region.

To illustrate the use of the LABEL command, suppose we have divided the screen using the HALF command into two plotting regions (TH and BH) and have plotted SIN(T) on the TH and COS(T) on the BH. The following sequence would produce the labeling shown in Figure 2.

```
LABEL,TH;SIN(T);BH;THIS IS A;;PLOT OF COS(T)     (CR)
```

TH sequence:	(position crosshairs)(SPACE BAR) (position crosshairs)(any other key--the coordinates are printed) (position crosshairs)(any other key--label is drawn)
BH sequence:	(position crosshairs)(DEL)(position crosshairs)(DEL--the line segment is drawn)(position crosshairs)(any other key--the first part of the message is drawn)(position cross-hairs)(any other key--the second part of the label is drawn)

Note that an alphanumeric message will not be drawn immediately following the printing of the coordinates of a point, but that in this case a new graphics input sequence is begun. This feature can be used to get the coordinates of several points with only one LABEL command. For example, giving the command:

```
LABEL;
```

will allow any number of points to be labeled with their coordinates. Upon terminating the graphics input sequence, the empty string will be drawn i.e., nothing is drawn.

Note: When plotting is being done on BOTH the Calcomp and Tekterminal, any labeling done on the Tekterminal will be done on the Calcomp plotter also.

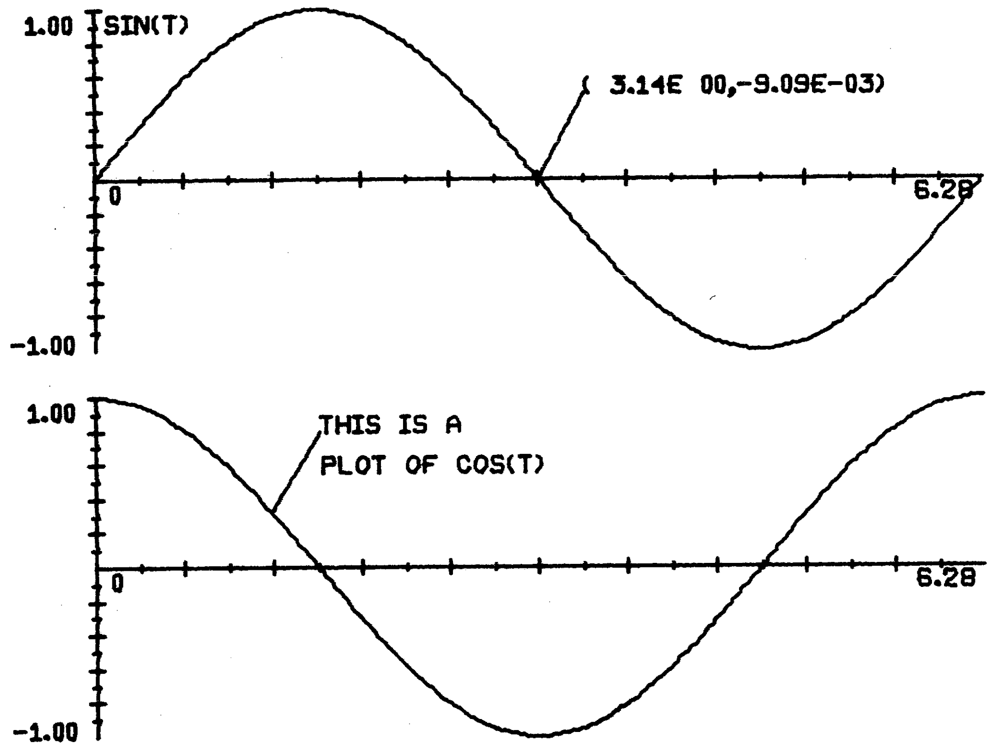


Figure 2. These plots illustrate the use of graphics input to position labels and get the coordinates of points.

ON or OFF,<plotting region>,<effect>

The ON and OFF commands allow the user to control certain plotting effects and certain calculation effects. The plotting region modifier only applies when the effect is for plotting. The plotting effect modifiers are:

AXIS	- The axis and tic mark labels will not be drawn in the specified plotting region if AXIS is OFF.
TICLABEL	- The labels on the axis's tic marks will not be drawn in the specified plotting region if TICLABEL is OFF. No labels will be drawn at the tic marks if AXIS is OFF.

The calculation effects regard the solution to differential equations. The effects and their descriptions are given below:

ITERATED - determines which predictor-corrector scheme is to be used when solutions to differential equations are calculated.

ON - Iterated Method
A fourth order Adams corrector is iterated until the normalized difference between successive corrected values is less than 10^{-10}.

OFF - Modified Predictor-Corrector Method
This method involves adding a correction to the predicted and corrected values at each step to compensate for the truncation error.

FIXED - determines whether the initial stepwidth h is to be allowed to be doubled and halved by the program or is to remain constant throughout the calculation of the solution to a system of differential equations.

ON - The stepwidth h is to remain fixed throughout the calculations.

OFF - The stepwidth h is to be halved and doubled to keep the local truncation error in the interval (10^{-5},10^{-8}).

ERROR - the value of this variable determines if the bound for the total error is to be calculated when systems of differential equations are solved.

ON - The error bound will be calculated when a system of differential equations is solved. When a system of differential equations is compiled, the quantities for the largest eigenvalue and the roundoff error used in the calculation of the error bound will be requested along with the initial conditions of the set of equations.

OFF - The error bound will not be calculated when a system of differential equations is solved and the quantities needed for the calculation of the error bound will not be requested. Default values will be assigned to these quantities.

PLOT,<function or array name>,...
PLOT,(<function or array name>,<function or array name>),...

The PLOT command will produce a plot of the specified functions and/or arrays as ordered pairs (t,f(t)) or parameterically as (f(t),g(t)). The functions will be evaluated at equally spaced points in the domain. The points plotted will be $f(t_i),f(t_i+\Delta t),\ldots,f(t_f)$ where

t_i = initial value specified in the DOMAIN command,
t_f = final value specified in the DOMAIN command, and

$$\Delta t = \frac{t_f - t_i}{[(\text{number of points})-1]}$$

Arrays are assumed to contain elements which correspond to the successive values taken on by the independent variable, i.e., $(t_i,A(1)),(t_i+\Delta t,A(2)),\ldots$ where A is an array to be plotted, and t_i and Δt are defined as above.

A parametric plot is specified by giving the ordered pairs of functions to be plotted. For example, if a plot of f(t) versus g(t) was to be produced, giving the command

PLOT,(G,F)

would cause the plot to be produced.

Another example of the PLOT command is:

PLOT,F,G

This would cause the functions or arrays F and G to be plotted as F(T) versus T and G(T) versus T on the same axis set.

In addition to specifying which functions to plot, one can also specify how the functions are to be plotted, i.e., with data marks, dashed lines, no lines connecting the data points, or any combination of these. Following a function name or ordered pair of functions with one or more of the special words

POINTS
DASHES
MARK (code for mark)

will cause the function to be plotted with the specified effects. The codes for the data mark must be one of the following:

Code	Data Mark
1	small x
2	large x
3	small +
4	large +
5	small -
6	large -
7	small '
8	large '
9	small ↑
10	large ↑
11	small ↓
12	large ↓
13	small →
14	large →
15	small ←
16	large ←
17	small ⊡
18	large ⊡
19	small ◬
20	large ◬
21	small *
22	large *
23	small ⧗
24	large ⧗
25	small ⊠
26	large ⊠
27	small ⬡
28	large ⬡

An example of the use of the effects is:

```
PLOT,F,POINTS,G,MARK(1)
```

When the plot is produced, F will be plotted with points, i.e., no lines will be drawn between the data points; and G will be plotted with a small x at each data point and with lines connecting the points.

Notes:

1) A maximum of eight functions may be specified to be plotted by one PLOT command.
2) If a DOMAIN command has not been given before the first plot command is given for a plotting area, the domain specifications will be requested.
3) Once the domain values (initial point, final point, and number of points) have been specified, they remain in force until redefined by a DOMAIN command.

```
PPLOT,<function or array name>,...
PPLOT,(<function or array name>,<function or array name>),...
```

The PPLOT command will produce polar plots of the specified functions or arrays. The description of this command is the same as for the PLOT command (see page 23) except that the ordered pairs (Θ,R(Θ)) or (TH(Θ),R(Θ)) are plotted instead of (T,F(T)) or (G(T),F(T)).

```
QUARTER,<region for 1st plot>,...
```

The QUARTER command allows the plotting area, as currently defined by the SIZE command, to be divided into four regions where plotting may be done. The parameters on the QUARTER command determine the order in which the regions will be used by successive PLOT commands. The number of parameters determines the number of axes sets that may appear in the plotting area concurrently.

If the QUARTER command is given with no arguments, the four regions are initialized for plotting so that four sets of axes may appear in the plotting area. The order in which the plotting regions will be used by successive PLOT commands in this case are:

UL - upper left corner
UR - upper right corner
LL - lower left corner
LR - lower right corner

An example of the QUARTER command is:

```
QUARTER,UL,LR
```

The plotting area will be divided into four regions. The next two PLOT commands will produce plots in the UL and LR plotting regions respectively. A third PLOT command would cause the plotting area to be cleared and the plot produced in the UL plotting region.

```
RANGE,<plotting region>,<low value dep>,<high value dep>,
<origin dep>,<low value ind>,...
```

The RANGE command is used to specify the ranges of the dependent and independent axes. If the range of the independent axis is not specified, it will be determined from the domain specifications if the ordered pairs (t,g(t)) are being plotted or by automatically scaling the first component data when ordered pairs (g(t),f(t)) are plotted. If the ranges have not been specified by a RANGE command, they will be automatically determined. The plotting area mnemonic is the same as those used on the QUARTER and HALF commands.

If no modifiers are given on the RANGE command, the pertinent information is requested for all plotting regions. A question is asked after the dependent axis range has been specified as to whether the independent axis range is to be specified. The following example is of an unmodified RANGE command with user responses underlined:

```
>RANGE
 FULL RANGE SPECIFICATION
 LOW VALUE = -1.0
 HIGH VALUE = 1.0
 ORIGIN = 0.0
 DO YOU WISH TO SPECIFY FOR BOTH AXES?  YES
 LOW VALUE = 0
 HIGH VALUE = 3.0
 ORIGIN = 0.0
```

Other examples of the RANGE command are given below:

```
RANGE,0,3,0
RANGE,UL,-1,5,0,-4,7,0
```

Note: A DOMAIN command will cause the independent variable axis range to be automatically determined. Hence, if the ranges of both axes are to be set by the RANGE command, i.e., a plot of the form (G(T),F(T)) is to be produced, the domain specifications must be given first.

READ,(<parm$_1$>,...,<parm$_n$>),<lun or file name>

READ,(<array name$_1$>(<index variable>),...,<array name$_n$>
(<index variable>),<index variable>=<initial value>,
<final value>,<increment>),<lun or file name>

The READ command is used to read parameter values or array elements from a file or from the terminal. The values are read in a free format with blanks or other special characters delimiting the numbers.

When the second form of the READ command is used, the index is set to the initial value and values read are stored in the arrays using this index until the list of array names is exhausted; then the index is incremented and the process is repeated. If the increment is not specified, it is assumed to be one. If a lun or file name is not specified, the values will be read from the terminal.

To facilitate the use of segmented files, a search-end-of-file-forward is executed on the files from which array values have been read. This is not done if parameter values have been read.

Examples of the READ command are given below:

READ,(A,B),1
: Values for the parameters A and B are read from logical unit 1.

READ,(C(I),D(I),I=1,10),2
: Values for the arrays C and D are read from logical unit 2. The values are read in the order C(1), D(1), C(2), D(2),...,C(10), D(10). After the values have been read, a search-end-of-file-forward is executed so that the next READ from lun 2 would read values from the first record following the end-of-file.

Note: An error occurs if an end-of-file or end-of-data is encountered before the list has been satisfied. The data files must not be in a compressed form as is produced by the FILE or COUT commands in the EDITOR.

REPEAT

This command allows the user to re-execute the last user supplied command given. For example, one could have specified several of the arguments as parameters in the original call. By giving these parameters new values and giving the REPEAT command, the subroutine would be called again with a new set of parameters.

RESET

The RESET command will cause all parameters and initial conditions which are varying to be reinitialized to the last values specified by the user. The last value specified may have been done using the READ command (see page 28), parameter initializing statement (see page 35), or in response to a request following a function definition (see page 37).

SIZE,<device>,<x-size in inches>,<y-size in inches>

The SIZE command allows the user to define in inches the dimension of the plotting area on the Calcomp and Tekterminal. Upon entering the system, the plotting area is defined as 5.5" x 4". The device modifier indicates whether the dimensions specified are to apply to plotting done on the Calcomp or the Tekterminal. The modifiers are:

CALC	- Calcomp plotter
TEK	- Tekterminal (or Hewlett-Packard plotter)
BOTH	- Calcomp plotter and Tekterminal (or Hewlett-Packard plotter)

The Tekterminal screen is approximately 8.18" x 6.08". On the Hewlett-Packard plotter, the user determines the actual size of his graph. This size is assumed to be 8.18 inches wide and 6.08 inches high. Any sizes specified by the SIZE command will then be proportional to the 8.18" x 6.08" assumed size.

Notes: 1) It may be necessary in some applications to have the display contain axis of a certain length. This can be done using the following formulas and table of constants:

$$\text{(x-size in inches)} = \text{(x-axis length in inches)} \cdot \text{(x-factor)}$$

$$\text{(y-size in inches)} = \text{(y-axis length in inches)} \cdot \text{(y-factor 1)} + \text{(x-axis length in inches)} \text{(y-factor 2)}$$

Display Area Division	Plotting Regions	x-factor	y-factor 1	y-factor 2
FULL	FULL	1.17021	1	0.051064
HALF	TH, BH	1.17021	2	0.24615
	LH, RH	2.82052	1	0.051064
QUARTER	UL, UR LL LR	2.82052	2	0.24615

For example, if it was desired to have the x- and y-axis four inches in length when plotting on the FULL plotting area, the size of the area would be specified as 4.68" x 4.20".

2) The sizes specified will always be supplied on the Calcomp plotter. However, the display created on the Tekterminal (or Hewlett-Packard plotter) is a constant multiple of the Calcomp plot. Therefore, if the user specifies a size on the Tekterminal which is not a multiple of the Calcomp size, the Tekterminal size is adjusted accordingly.

SYSTEM

The SYSTEM command allows the user to enter a system of differential equations. It is not required to use the SYSTEM command if a single differential equation is being entered. Appendix I contains a description of the rules for defining single as well as systems of differential equations. Following a SYSTEM command, a colon will be printed and the bell rung. This indicates that the first equation may be entered. A colon will be printed and the bell rung on each succeeding line until a carriage return is given as the first input on a line.

An example of the SYSTEM command is given below (user input is underlined):

```
>SYSTEM                        (Note:  (CR) indicates
:X'(T) = X↑2 + Y↑2 (CR)         a carriage return.)
:Y'(T) = X↑2 - Y↑2 (CR)
:(CR)                          (The initial conditions
     X = 0.0                    of the system are
     Y = 0.5                    requested.)
>
```

WRITE,(<parm$_1$>,...,<parm$_n$>),<lun or file name>

WRITE,(<array name$_1$>(<index variable>),...,<array name$_n$>
      (<index variable>),<index variable>=<initial value>,
      <final value>,<increment>),<lun or file name>

The WRITE command is used to write parameter values or array elements onto a file or the terminal. The values are written in the order specified using the current format (see page 13).

When array elements are written, all values for a particular index value are written on one record. For example, the command

```
WRITE,(A(I),B(I),I=1,101),3
```

would cause A(1) and B(1) to be written on the first record, A(2) and B(2) on the second, etc.

An end-of-file mark is written on the file after all of the specified array elements have been written. No end-of-file is written when parameter values are written.

As with the READ command, if the increment is omitted, it is assumed to be one. If a lun or file name is not specified, the values are written on the terminal. The WRITE command will create a file if the lun specified is not equipped and will create and save a file under the name specified if a file by that name does not exist.

Examples of the WRITE command are given below:

```
WRITE,(A,B),1
```

The values of the parameters A and B are written in the current format on one record on lun 1. No end-of-file mark is written on lun 1.

```
WRITE,(C(I),D(I),I=1,10),2
```

The elements of the arrays C and D are written in the current format on lun 2 in the form:

```
   Record 1:       C(1)  D(1)
   Record 2:       C(2)  D(2)
       .             .     .
       .             .     .
       .             .     .
  Record 10:      C(10) D(10)
```

An end-of-file is written on lun 2 following record 10.

Changing Parameter Values

Parameters and initial conditions may be given new values and/or increments with any of statement forms below:

```
A)  <name> = <number> BY <number>
B)  <name> = <number>
C)  <name> = BY <number>
```

The <name> may be a parameter or differential equation solution. The <number> may be any signed number. Examples of this command are:

```
1)  A = 5 BY 2
2)  Y' = -0.1,B = BY 3
```

In example 1) the parameter A is given a new value and increment. In example 2) a new initial condition is specified for Y' and the parameter B is given a new increment.

Notes: 1) If a parameter is specified which has not been used previously, it is defined and given the value specified.

2) A varying parameter can be changed to be fixed by specifying a zero increment.

APPENDIX I

The GRAFIT system contains a function translator which allows a user to specify a function in three ways:

a) by a formula, e.g.,

```
F(T) = A*T↑2 + B*T + C
```

b) as the solution of a single differential equation, e.g.,

```
Y'(T) = -Y
```

c) as the solution to a system of differential equations.

With the exceptions listed below, functions are entered in a notation similar to that of ordinary mathematics. The exceptions are:

a) Implicit multiplication is not allowed. Multiplication is indicated by a single asterisk. Thus, the product $a \cdot b$ is entered as A*B.

b) Subscripting and/or superscripting are not allowed. Exponentiation is signified by the uparrow (↑) or by two asterisks, e.g., a^2 is entered as A↑2 or A**2.

c) No distinction is made between upper- and lower-case letters.

d) The highest derivative term in a differential equation must be isolated on the left of the equal sign.

A function definition can be broken into two parts, with the equal sign as the divider. The quantity to the left of the equal sign is the function identification and gives the name of the function being defined, the order of the derivative if it is a differential equation, and the independent variables. The quantity on the right of the equal sign is a mathematical expression defining the function. An expression is composed of identifiers, constants, operators, and special symbols combined according to the rules of mathematics subject to the restrictions listed above.

An identifier may be the name of a function, array, parameter, or independent variable. An identifier name consists of an alphabetic character followed by from zero to three alphanumeric characters.

A function may be either a standard function, e.g., SIN, COS, etc. (see Table 1 for a complete list) or be one of the functions the user has already entered. In the case of a system of differential equations, references may be made to any function in the system, as well as to those previously defined.

The form of a constant is the same as that of a constant in a FORTRAN source program. A constant can be entered in scientific notation with the exponent being specified by an E followed by a signed or unsigned integer. Some examples of constants are:

1

0.2

2E-3 is equivalent to $2 \cdot 10^{-3}$

100.59

Internally, all constants are represented as floating-point numbers. Hence, there is no distinction between 1 and 1.0.

Arithmetic operators and special symbols are used to combine identifiers and constants in expressions. In addition to the arithmetic operators shown in Table 2, the differential operators ' and " have been introduced to identify the differential equation solution to be used or to specify that the numerical derivative of the function is to be used. The special symbols (,), [, and] are used to group operations in the normal way. Several symbols and words are used as separators. These are listed with their descriptions in Table 3.

Before discussing more specific features of the language, a few examples will be given to illustrate how functions specified by a formula are entered. In each of the examples, note that a parameter does not have to be defined before it is used. After compilation, parameters which have not appeared

Table 1. A list of the standard functions available to the user in defining functions.

SIN(X)	sine function
COS(X)	cosine function
SQRT(X)	square root function
ABS(X)	absolute value function
COSH(X)	hyperbolic cosine function
SINH(X)	hyperbolic sine function
TANH(X)	hyperbolic tangent function
TAN(X)	tangent function
ASIN(X)	inverse sine function
ACOS(X)	inverse cosine function
ATAN(X)	inverse tangent function
EXP(X)	exponential function--e^x
LN(X)	base e logarithm
LOG(X)	base 10 logarithm
INT(F,A,B,N)	calculates $\int_a^b f(t)dt$ using a quadrature formula with n intervals
SUM(F,K,A, B,I,)	computes $\sum_{k=a}^{b} f$ with k being incremented by i
ERR(Y)	returns the computed upper bound for the total error in the system of differential equations involving Y

Table 2. Arithmetic and relational operators.

** or ↑	exponentiation
*	multiplication
/	division
+	addition and unary plus
-	subtraction and unary minus
<	less than relation
>	greater than relation
=	equal relation
<=	less than or equal relation
>=	greater than or equal relation
#	not equal relation

Table 3. A list of the special symbols and words recognized.

; or ELSE	separates segments of a function defined by conditionals
IF	separates an expression giving a possible value for the function from the conditions under which the function will be defined by that expression

in previously-entered functions will be printed, and their values requested. A parameter may have a fixed value or be made to vary by following the fixed value with an increment. In the following example, the underlined quantities are typed by the user.

```
>Y(T) = -0.5*G*T↑2
     G = 9.8 BY 0.1
>F(T) = EXP(SIN(T)*Y(T)) - 2*COS(T/2)
>H(X,Y) = -G*X↑2 + B*F(Y)
     B = 5.27
```

Two things should be noted about the function H(X,Y) in the example. First, a function may have more than one independent variable. Second, if an independent variable has the same name as a function already defined, no ambiguity results since independent variables have been given higher priority than functions.

Many times a function is defined by several different formulas, depending upon certain conditions. To handle these functions a conditional statement has been provided, which has a form very close to that used by textbooks. For example, the function

$$f(t) = \begin{cases} \sin(t) & 0 \leq t \leq \pi/2 \\ 1 & \pi/2 < t < \pi \\ e^{t-\pi} & \pi \leq t \end{cases}$$

would be entered in the following way (the word "ELSE" can be used instead of the semicolon):

```
>F(T) = SIN(T) IF 0 <= T <= PI/2;
        1 IF PI/2 < T < PI;
        EXP(T-PI) IF PI <= T
```

The parameter PI is supplied by the program and has the value 3.14159... . If in a function using conditionals none of the conditions are satisfied, a zero value is returned.

One of the special features allowed is the option of omitting the argument list of a function appearing in an expression if it is the same as that of the function being defined. For example, if one wished to use the function Y(T) from the example in the function $u(t) = y(t) \cdot t^2$, one could enter

```
U(T) = Y*T↑2
```

instead of

```
U(T) = Y(T)*T↑2
```

This feature cannot be used if the functions have a different number of independent variables.

The normal algebraic interpretation was maintained for expressions using the unary plus and minus. For example, the expression -A↑-B is interpreted as $-(a^{-b})$. Note in the above example that a unary plus and minus may follow another operator. Other examples of the use of this feature are:

A↑-B means a^{-b}

A*-B means $a(-b)$

A+-B means $a+(-b)$

An approximation to the first or second derivative of a function may be used in an expression by following the function name with ', ", or two single primes. For example, if F(T) had been defined as in the example on page 10, entering

```
G(T) = 5*F' + T↑2
```

will cause the first derivative of F(T) to be calculated when G is evaluated. Only the first or second derivative of a function specified by a formula may be specified. It is not possible to calculate higher derivatives by successive function definitions. For example, the following construction is not allowed:

```
F(T) = T↑2
H(T) = F'(T)
G(T) = H'(T)
```

In addition to specifying a function by a formula, a function may be specified as the solution to a differential equation or a system of differential equations. These functions are entered in nearly the same way as functions specified by a formula.

When entering a differential equation, the highest derivative term is isolated on the left of the equal sign. The highest derivative term with the independent variables thus becomes the function identification. For example, to enter the differential equation

$$y' + y^2 - t^2 = 0$$

one would rewrite it as

$$y' = t^2 - y^2$$

and enter it as

```
Y'(T) = T↑2 - Y↑2
```

To enter a system of differential equations, one first gives the SYSTEM command and then enters the members of the system on succeeding lines. The system is terminated by pressing the (CR) as the first character in a line. For example, the system

$$\begin{aligned} y' - x^2 - y^2 &= 0 \\ x' - x^2 + y^2 &= 0 \end{aligned}$$

would be entered as (user responses are underlined):

```
>SYSTEM
:Y'(T) = X↑2 + Y↑2
:X'(T) = X↑2 - Y↑2
:(CR)
```

The rules for entering the expression on the right of the equal sign for these functions are the same as those discussed previously for functions specified by a formula.

After a system of differential equations has been entered and compiled, the initial conditions for the equations are requested, along with any parameters which may be undefined. For example, the above system would appear as (user responses are underlined):

```
>SYSTEM
:Y'(T) = X↑2 + Y↑2
:X'(T) = X↑2 - Y↑2
:(CR)

    Y = 0
    X = 0.5
```

In response to the requests for initial conditions in the above example, numerical quantities were entered. However, the response may be an expression as well. This feature only applies to requests for initial conditions, and does not apply to requests for parameter values. An example of this feature is (user responses are underlined):

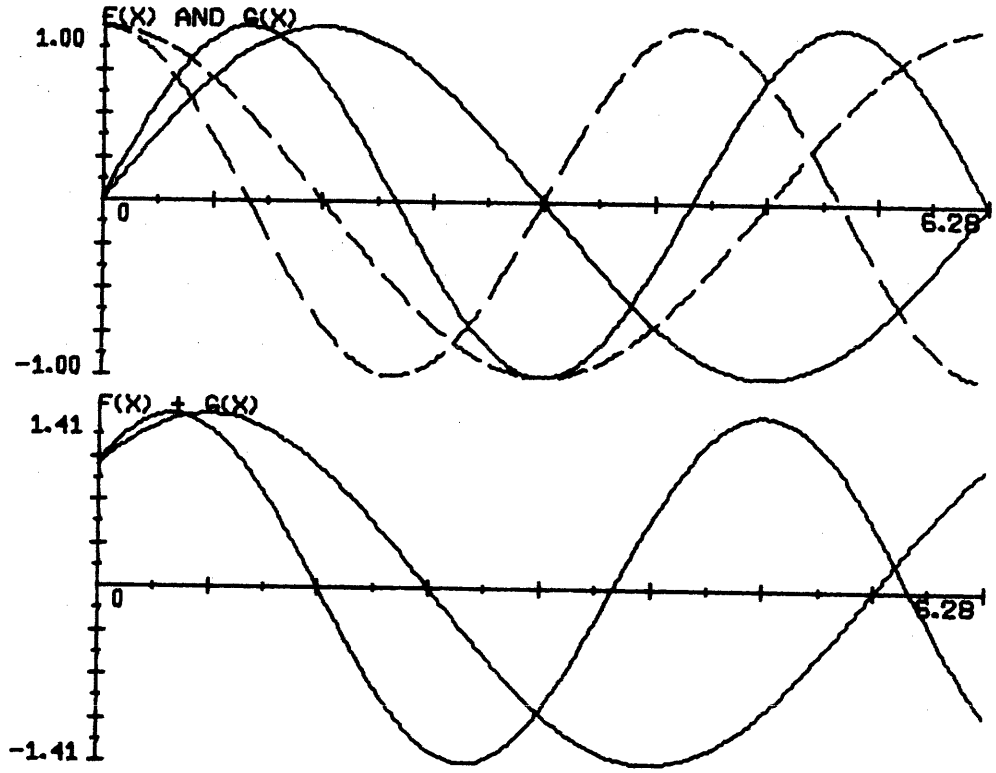
```
>SYSTEM
:X"(T) = 0
:Y"(T) = -G
:(CR)

    X = 0
    X' = VO*COS(TH)
    Y = 0
    Y' = SQRT(VO↑2 - X'↑2)
    G = 9.8
    VO = 20
    TH = .34
```

As illustrated by the example, the expression specified in response to a request for an initial condition may involve other initial conditions. These expressions are evaluated in the order they are entered; hence, one cannot have an expression involving an initial condition which will be specified later with an expression. In the above example, Y' could not have appeared in the expression for X'.

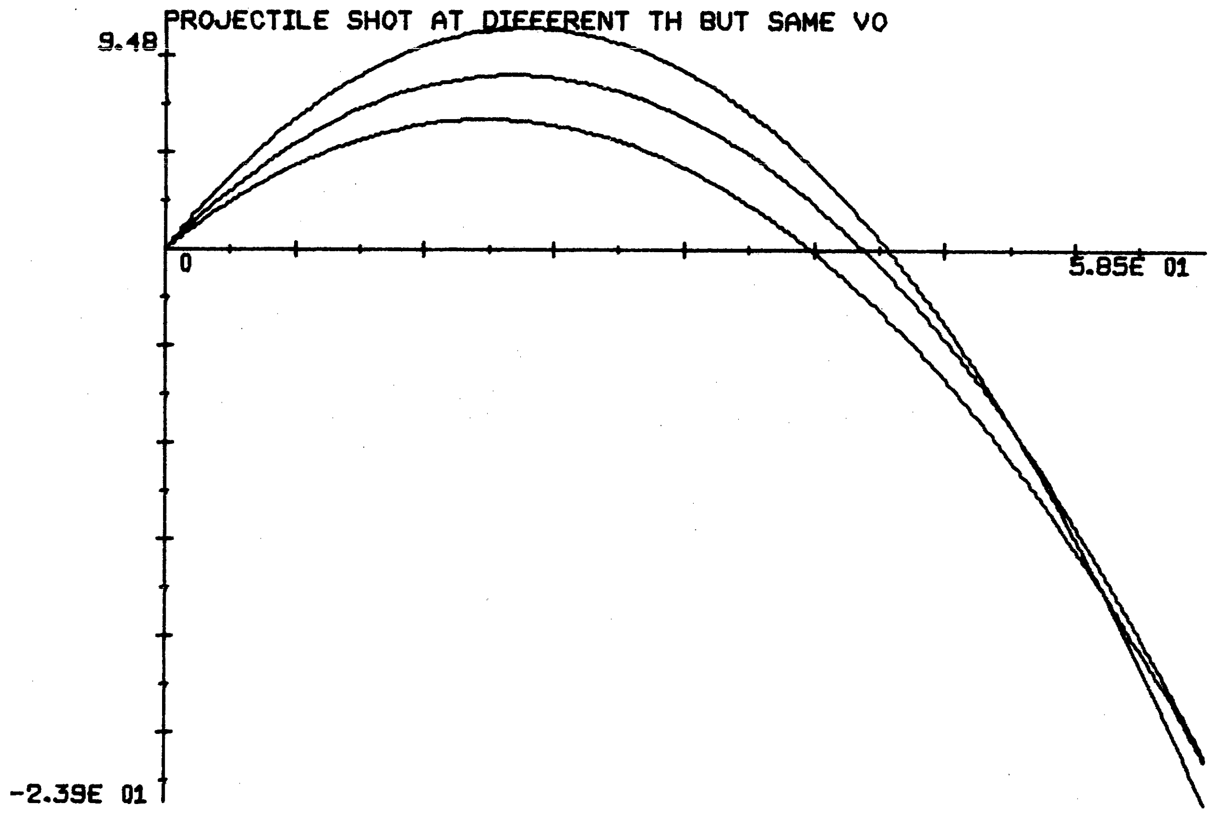
APPENDIX II

Examples

```
    ENTER COMMAND
>F(X)=SIN(K*X)
   K = 1 BY Ø.5
>G(X)=SIN(K*X+PH)
   PH = 1.57
>S(X)=F(X)+G(X)
>HALF
>PLOT,F,G,DASHES
   TH   DOMAIN SPECIFICATIONS

 INITIAL VALUE = Ø
 FINAL VALUE = 6.28
 NUMBER OF POINTS = 1Ø1
>PLOT,S
   BH   DOMAIN SPECIFICATIONS

 INITIAL VALUE = Ø
 FINAL VALUE = 6.28
 NUMBER OF POINTS = 1Ø1
>LABEL,TH;F(X) AND G(X);BH;F(X) + G(X)
>INCPLOT,1,1
>EXIT

#
```

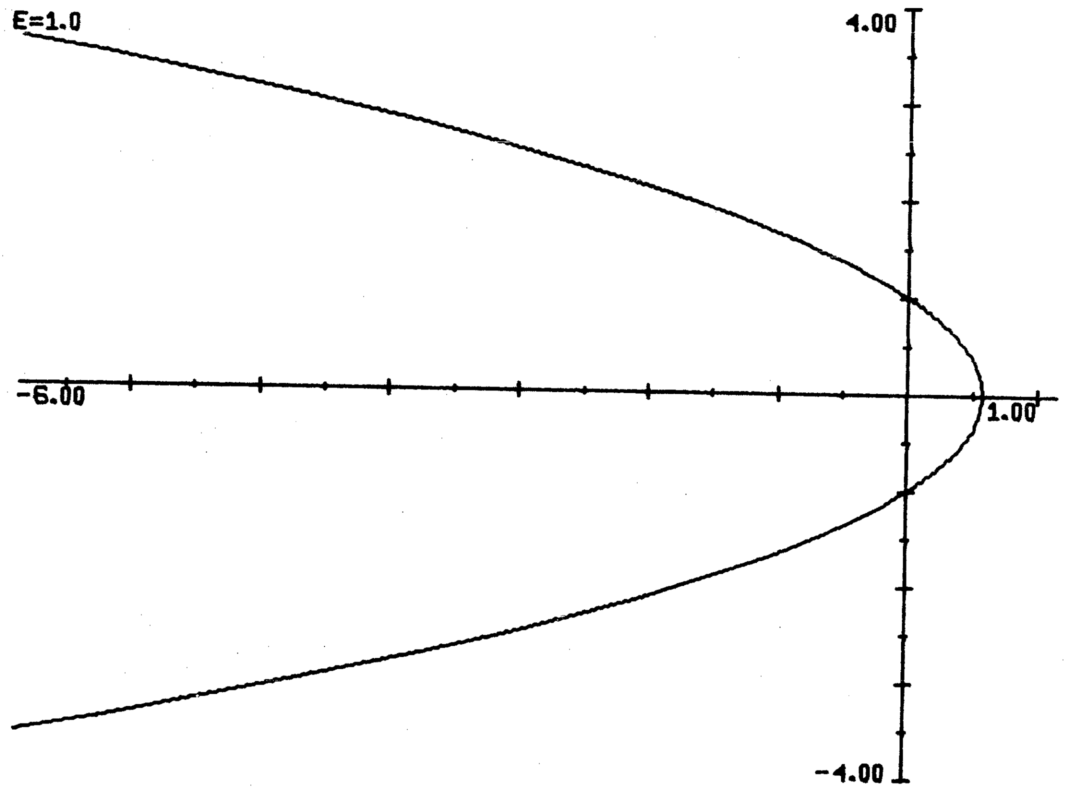


```
    ENTER COMMAND
>SYSTEM
:X"(T)=Ø
:Y"(T)=-G
:
   X = Ø
   X' = VO*COS(TH)
   Y = Ø
   Y' = SQRT(VO↑2-X'↑2)
   G = 9.8
   VO = 2Ø
   TH = .75 BY -Ø.1
>DOMAIN,Ø,4,1Ø1
>PLOT,(X,Y)
>INCPLOT,1,2
>LABEL;PROJECTILE SHOT AT DIFFERENT TH BUT SAME VO
>EXIT
#
```

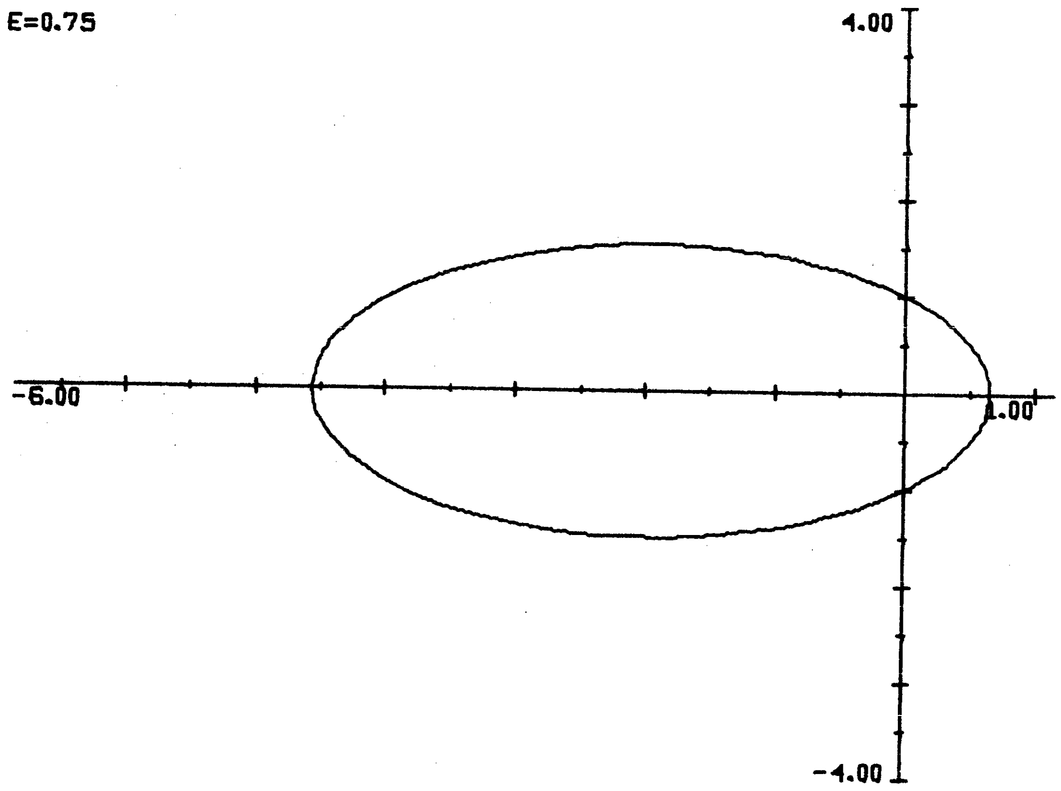


```
    ENTER COMMAND
>R(Z)=A/[1+E*COS(Z-B)]
   A = 1
   E = 1 BY -0.25
   B = 0
>DOMAIN,0,6.28,0\101
>RANGE,-4,4,0,-6,1,0
>PPLOT,R                   (See a)
>LABEL;E=1.0
>ERSPLOT,1,1               (See b)
>LABEL;E=0.75
>EXIT
#
```

a)

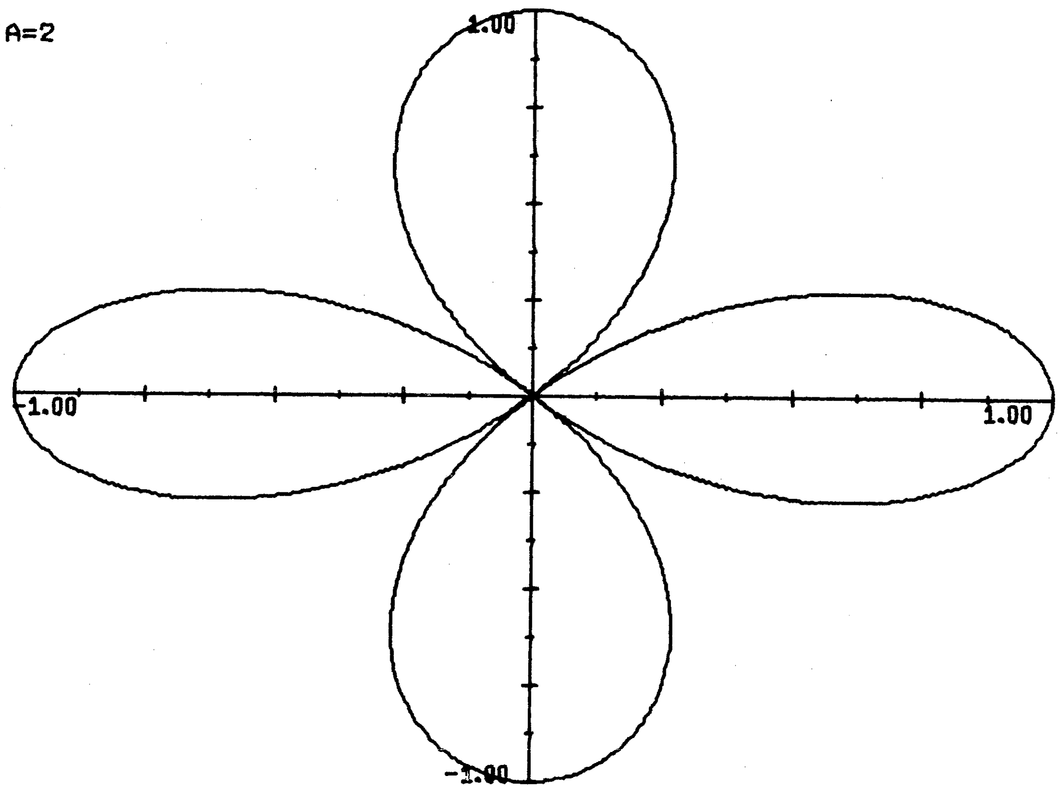


b)

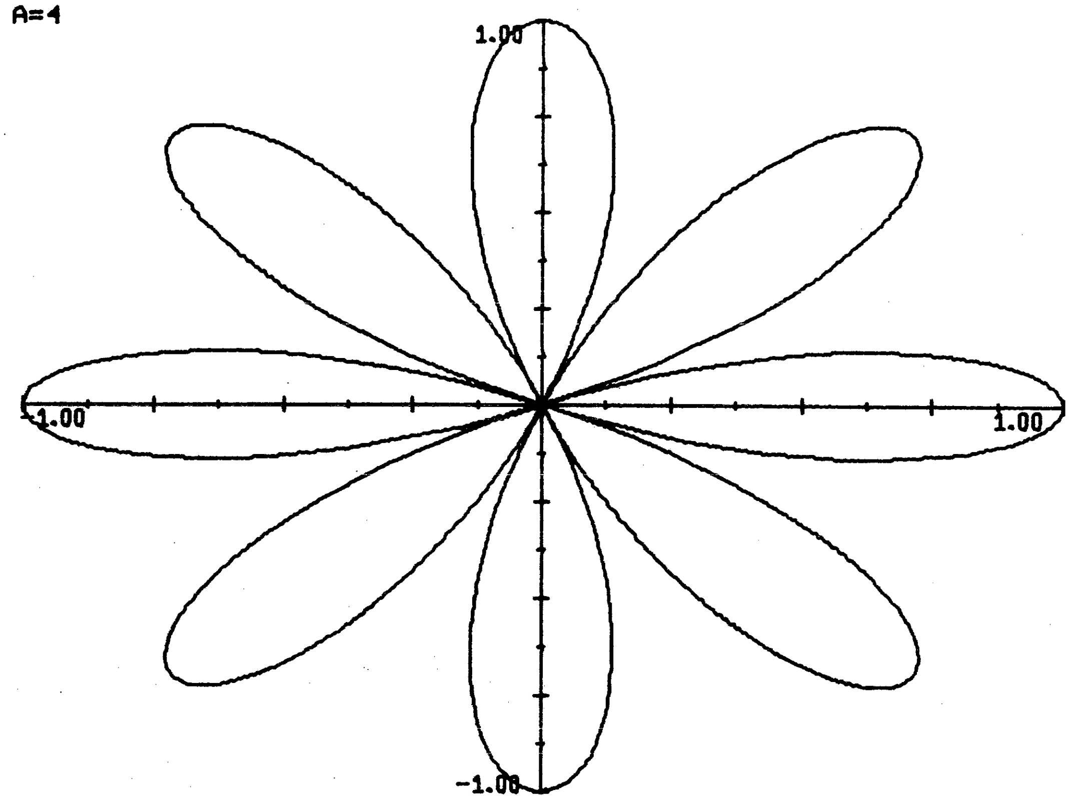


```
    ENTER COMMAND
>R(TH)=COS(A*TH)
   A = 2 BY 2
>DOMAIN,0,6.28,301
>RANGE
   FULL RANGE SPECIFICATIONS

 LOW VALUE = -1
 HIGH VALUE = 1
 ORIGIN = 0
DO YOU WISH TO SPECIFY FOR BOTH AXIS?YES
 LOW VALUE = -1
 HIGH VALUE = 1
 ORIGIN = 0
>PPLOT,R                (See a)
>LABEL;A=2
>ERSPLOT,1,1            (See b)
>LABEL;A=4
>ERSPLOT,1,1            (See c)
>LABEL;A=6
>ERSPLOT,1,1            (See d)
>LABEL;A=8
>EXIT
#
```

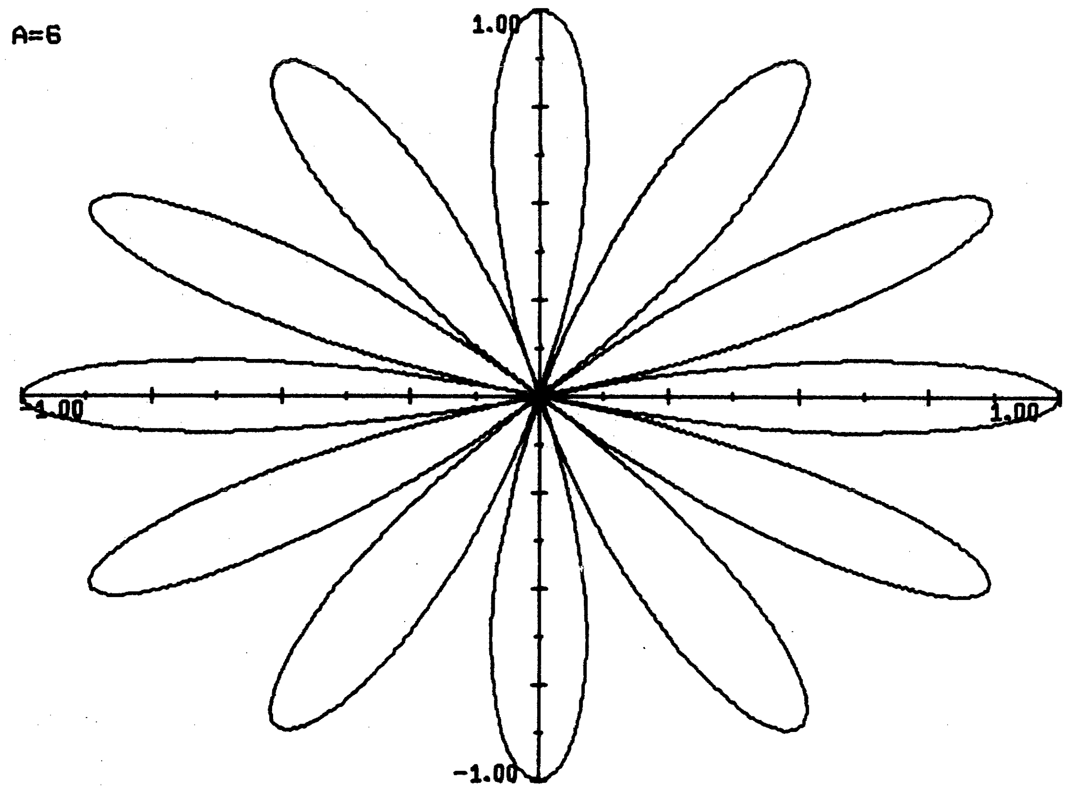
a) A=2



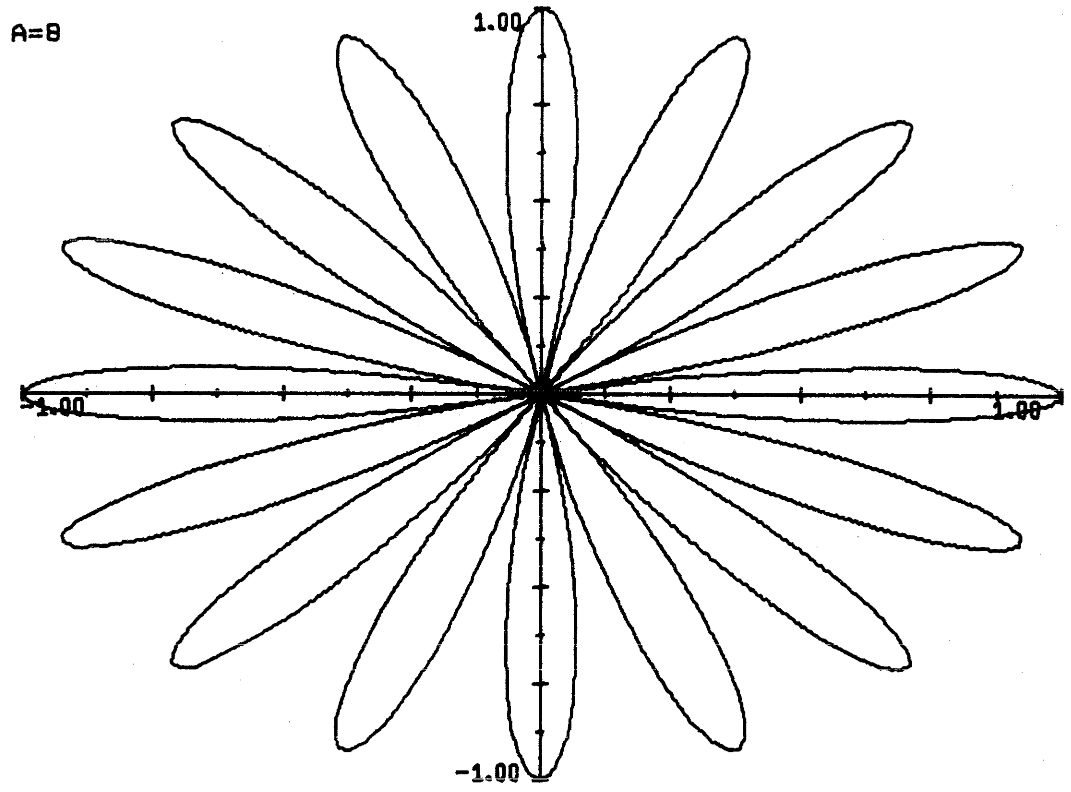
b)



c)

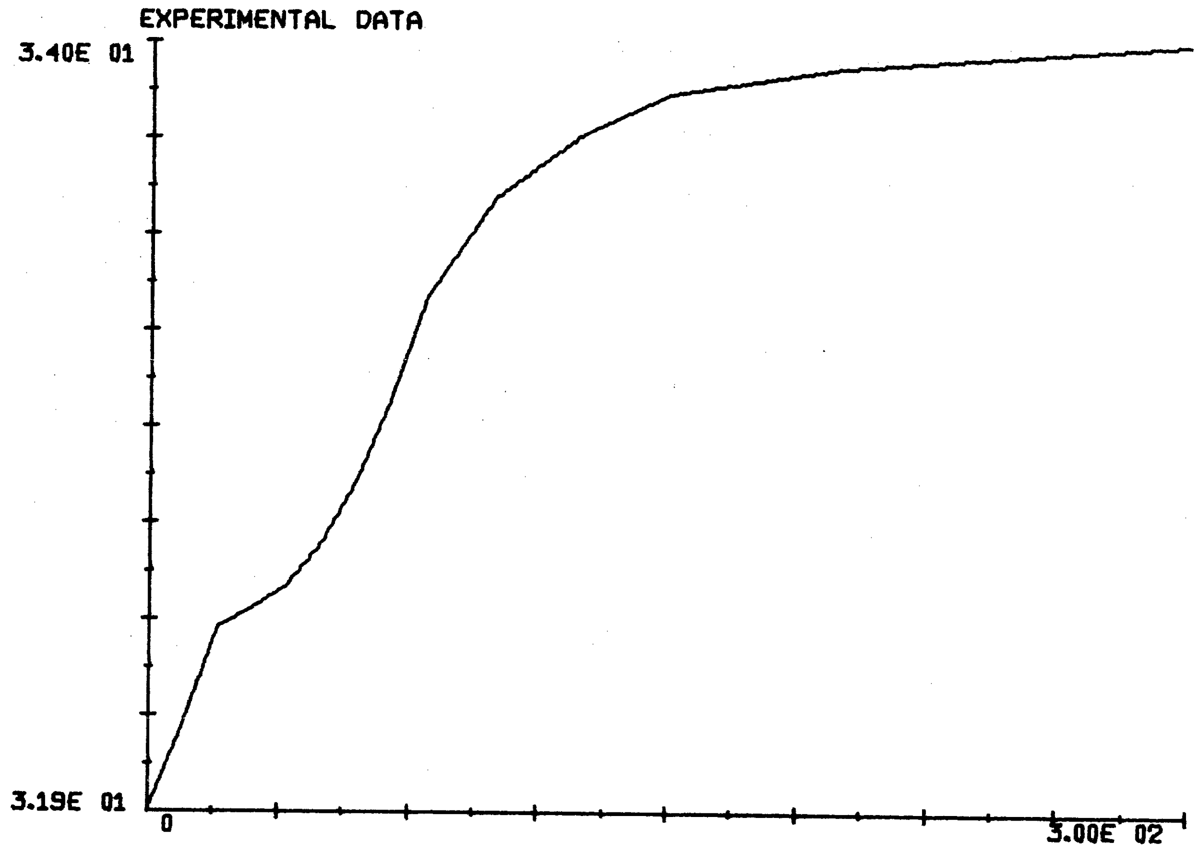


d)

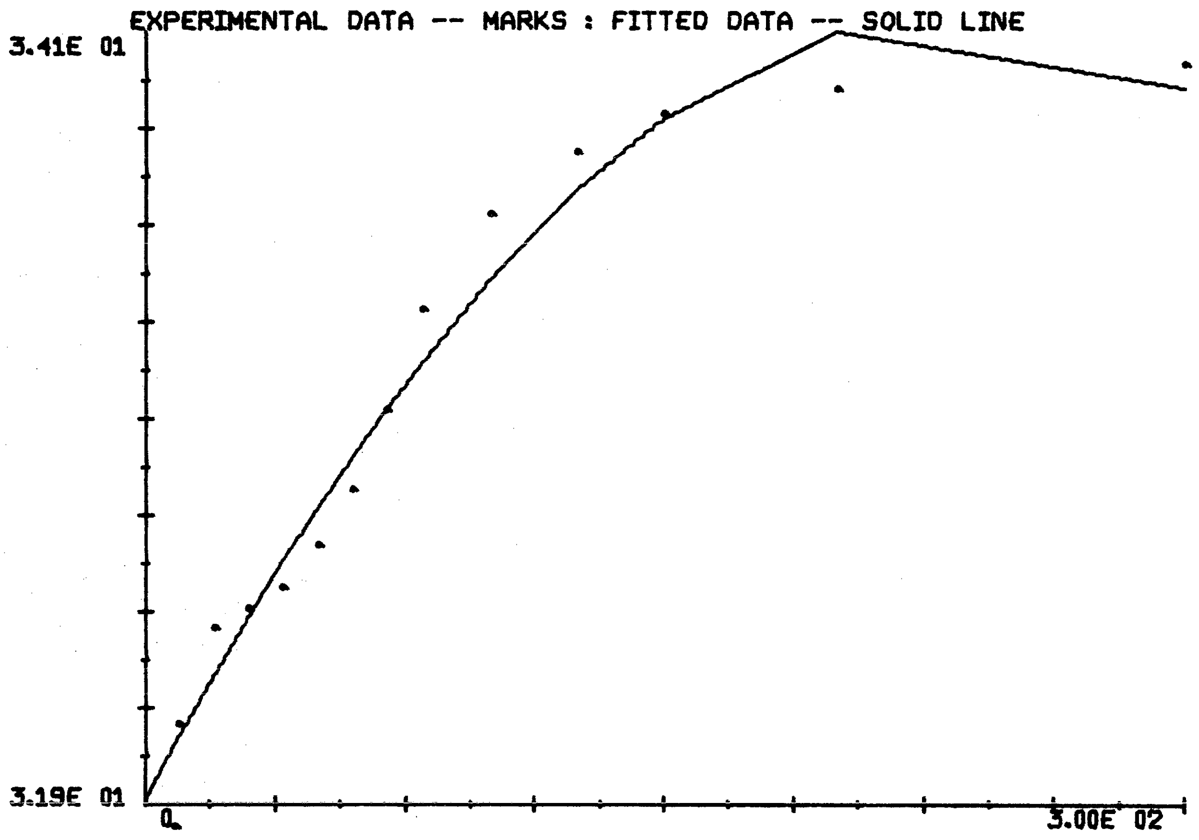


```
    ENTER COMMAND
>ARRAY,X(14),Y(14),YL(14),COEF(6)
>COMMAND,EXPFIT(X,Y,YL,N,P,M,C),*EXPFIT

>READ,(X(I),Y(I),I=1,14),*EXPDATA
>DOMAIN,1,14,14
>PLOT,(X,Y)       (See a)
>LABEL;EXPERIMENTAL DATA
>P=-1.7E-4
>EXPFIT(X,Y,YL,14,P,3,COEF)

EXPFIT ENTERED WITH NO. OF POINTS =  14
NO. OF COEF. = 3 NONLINEAR PARAMETER = -1.70000000E-04

>AUTOSCALE
>PLOT,(X,Y),POINT,MARK(19),(X,YL)      (See b)
>LABEL;EXPERIMENTAL DATA -- MARKS : FITTED DATA -- SOLID LINE
>WRITE,(COEF(I),I=1,3)
-1.3630E 03
 2.7153E 03
-1.3202E 03

>RES(I)=(Y(I)-YL(I))↑2
>AUTOSCALE
>PLOT,RES     (See c)
>LABEL;RESIDUALS SQUARED
>EXIT
#
```

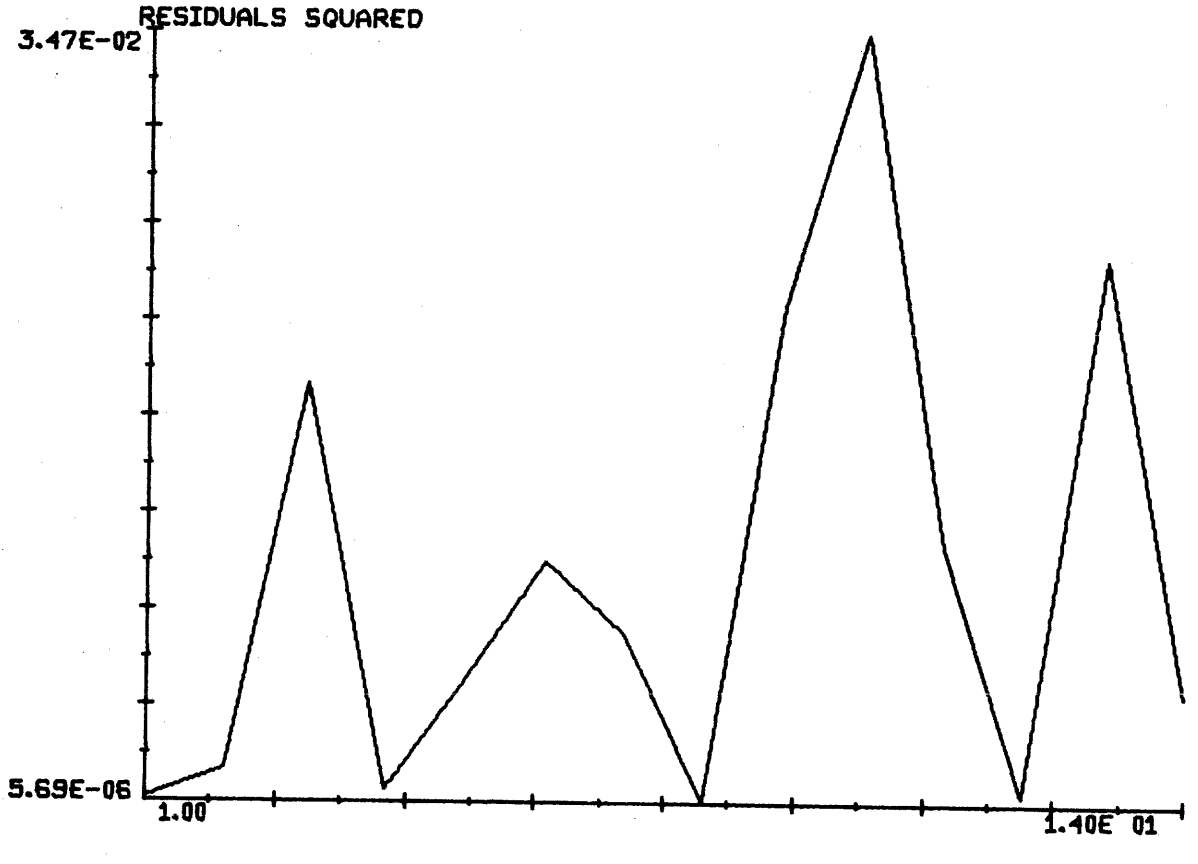
a)



b)


EXPERIMENTAL DATA -- MARKS : FITTED DATA -- SOLID LINE
3.41E 01
3.19E 01
0.
3.00E 02

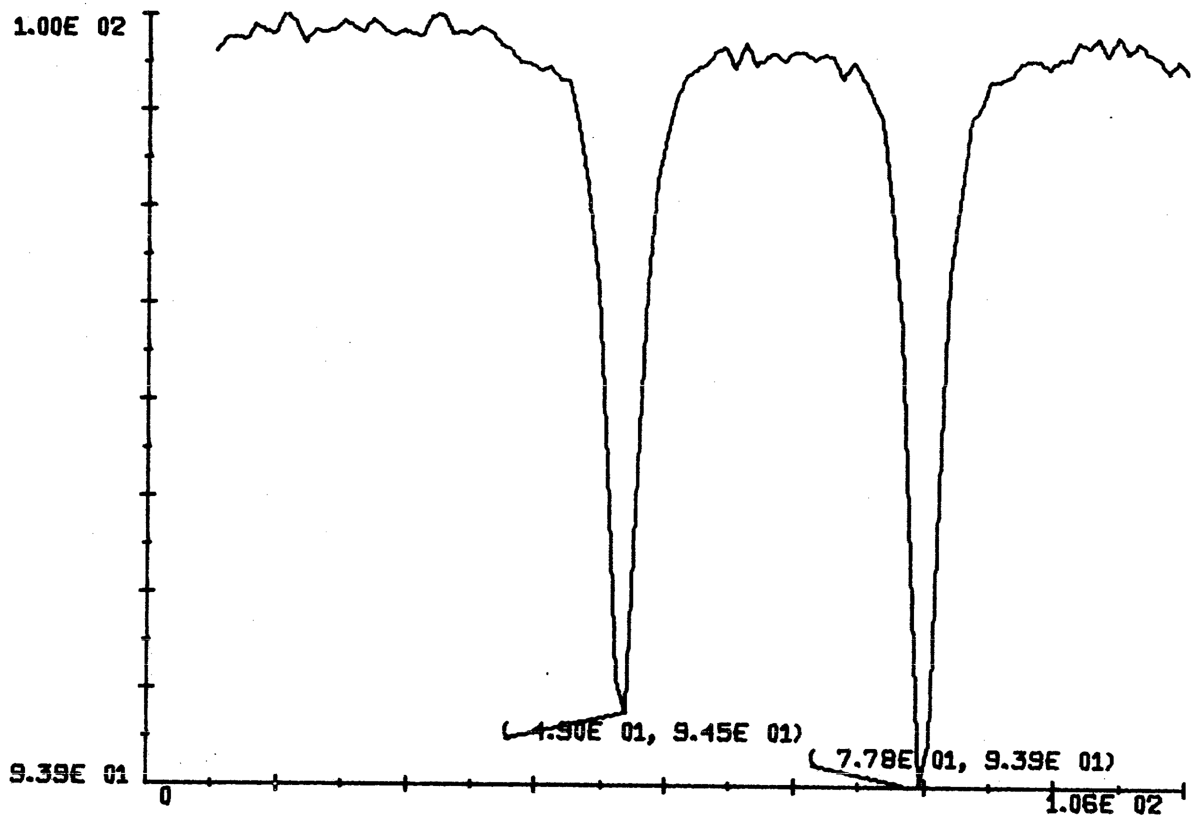
c)


RESIDUALS SQUARED
3.47E-02
5.69E-06
1.00
1.40E 01

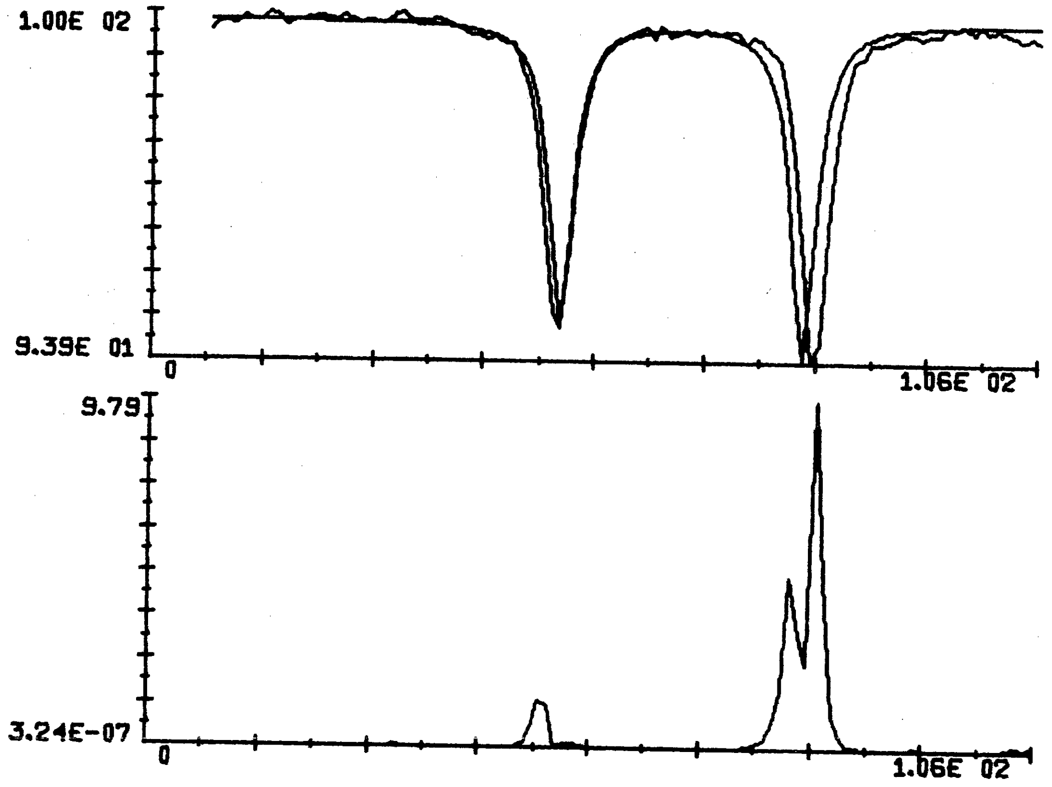
```
#*GRAFIT
    ARE YOU AT A TEKTERMINAL?  YES
 ENTER COMMAND

>ARRAY,X(100),Y(100),YL(100),B(7),P(7)
>READ,(X(I),Y(I),I=1,100),*DATAW
>DOMAIN,1,100,100
>PLOT,(X,Y)                   (See a)

>LABEL;
      (position crosshairs)(space bar)(position crosshairs)(space bar)
      (position crosshairs)(space bar)(position crosshairs)(space bar)
      (line feed)
>READ,(B(I),I=1,7)            (The parameter array is set up for the
 100                          nonlinear model.   Background first.)
 5.5  3.5  49                 (height, width, position)
 6.1  3.5  77.8                              (NLLSQ is  a nonlinear
>COMMAND,NLLSQ(X,Y,N,B,K,ITER,P),*NLLSQP     least square routine.)
>COMMAND,LOREN(X,Y,YL,N,B,K,P)               (LOREN is a routine to
>LOREN(X,Y,YL,100,B,7,P)                     evaluate the model func-
>RES(I)=(U(I)-YL(I))↑2                       tion for all values of
                                             X using the current
                                             parameter estimates.)
>HALF
>DOMAIN,BH,1,100,100
>AUTOSCALE
>PLOT,(X,Y),(X,YL)            (See b, TH)

>PLOT,(X,RES)                 (See b, BH)
>READ,(B(I),I=7,7)
 79
>REPEAT                       (Re-evaluate the model using LOREN)
>AUTOSCALE,BH
>ERSPLOT                      (See c)

>NLLSQ(X,Y,100,B,7,5,P)
>LOREN(X,Y,YL,100,B,7,P)
>AUTOSCALE,BH
>ERSPLOT                      (See d)
```

```
>WRITE,(B(F),I=1,7)
 1.0001E02
 5.8718E00
 3.6881E00
 4.8574E01
 6.4969E00
 3.6040E00
 7.9375E01
```

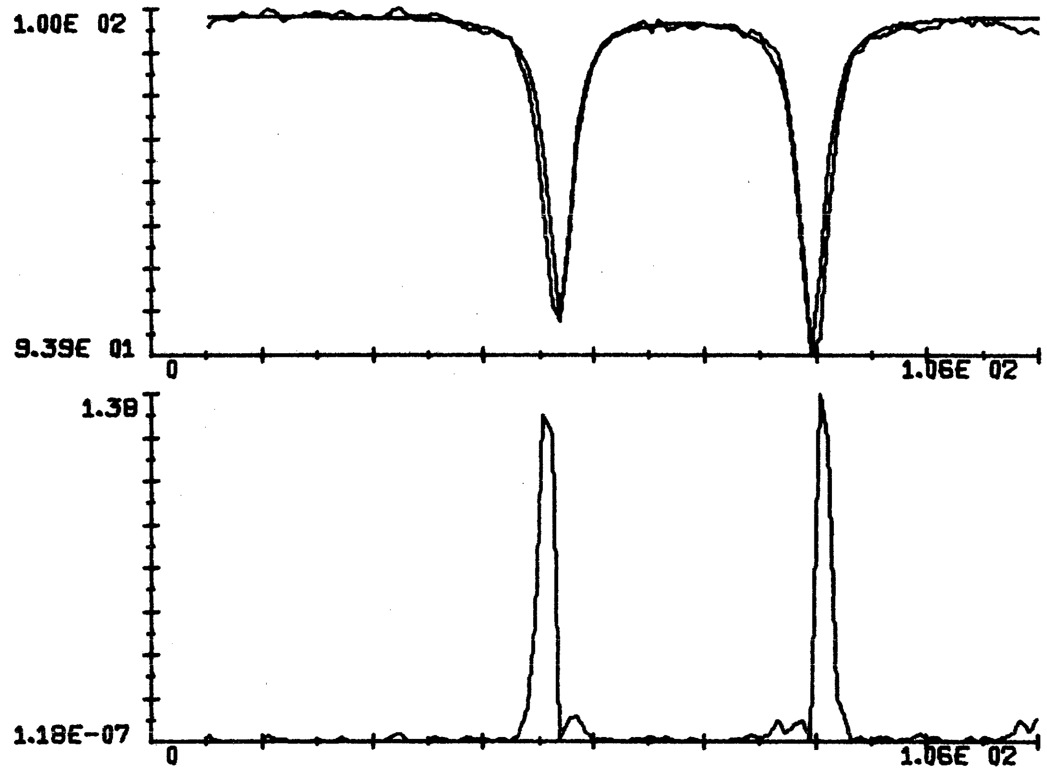
a)


1.00E 02
9.39E 01
0
1.06E 02
(4.90E 01, 9.45E 01)
(7.78E 01, 9.39E 01)

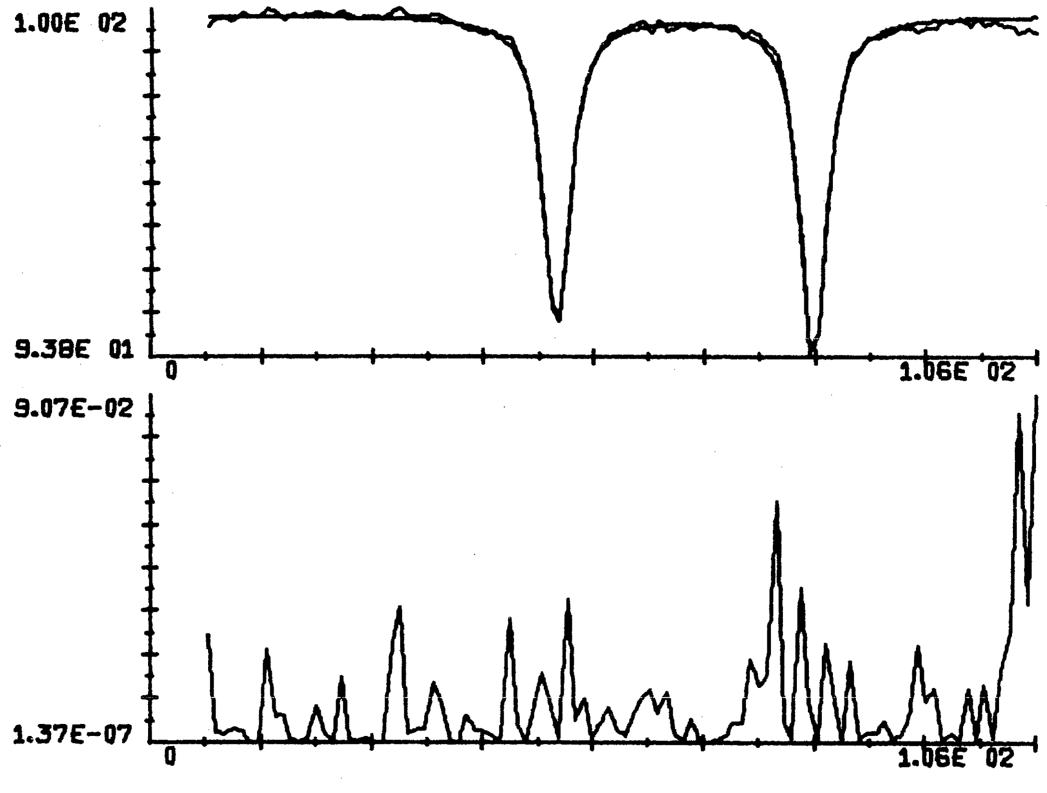
b)


1.00E 02
9.39E 01
0
1.06E 02
9.79
3.24E-07
0
1.06E 02

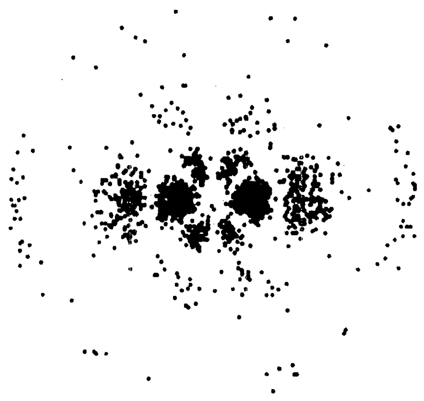
c)

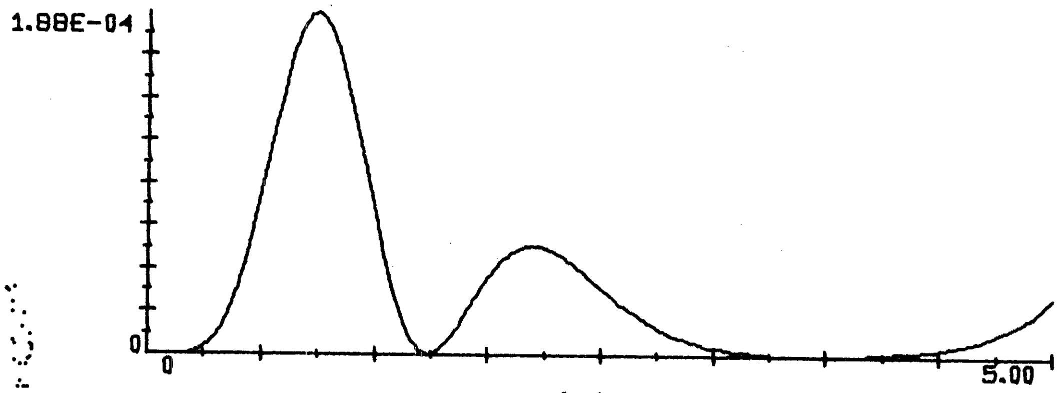

1.00E 02
9.39E 01
0
1.06E 02
1.38
1.18E-07
0
1.06E 02

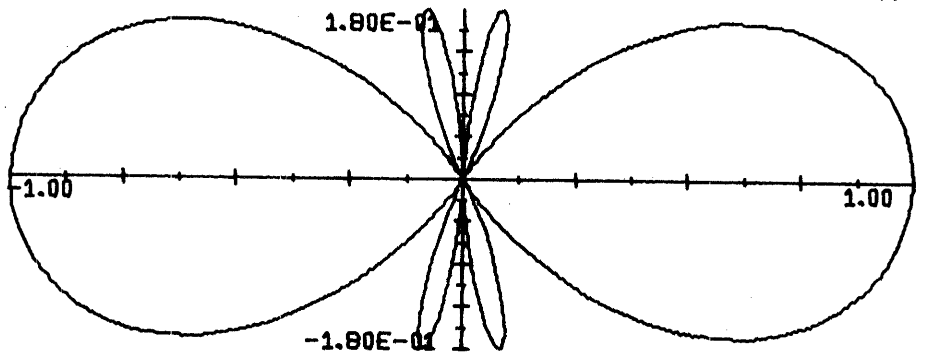
d)


1.00E 02
9.38E 01
0
1.06E 02
9.07E-02
1.37E-07
0
1.06E 02

```
#EQUIP,10 = PLOT
#LABEL,10/SAVE FOR LARRY HUBBLE
#*GRAFIT
  ARE YOU AT A TEKTERMINAL?  YES

  ENTER COMMAND
 >DEVICE,BOTH
 >P(M,TH) = 2.5*COS(TH)↑3-1.5*COS(TH)
 >L(TH) = P(M,TH)↑2
        M = 3 BY 1
 >V(R) = VO*EXP[-(R↑2/A↑2)]
        VO = -30
        A = 2
 >F(R) = C IF R<1E-3;
        M*(M+1)/(R+D)↑2
        C = 250
        D = 1.01E-2
 >U"(R) = [V+F-E]*U
        U = 0
        U' = 1E-4
        E = -2.65
 >USQ(R) = 0 IF R<1E-3;
        (U/R)↑2
 >COMMAND,POWDER2P(R,NR,TH,NTH,THIN,THFN,NPTS),*POWDER
 >ARRAY,R(101) = USQ(0;5),TH(201) = L(0;6.28)
 >HALF
 >DOMAIN,0,5,101,BH,0,6.28,201
 >PLOT,R
 >PPLOT,TH
 >POWDER2P(R,101,TH,201,0,6.28,1500)
 >         (position crosshairs to define circular region)

 >EXIT
```




1.88E-04
0
0
5.00


1.80E-01
-1.00
1.00
-1.80E-01

APPENDIX III

Special Subroutines Available

CHEBFIT

ABSTRACT:

The subroutine CHEBFIT computes the least squares fit to a set of data using Chebychev polynomials.

USAGE:

The calling sequence is:

```
     CALL CHEBFIT(X,Y,F,N,M,COEF)
```

where the parameters are:

X - An array which contains the independent variable data.

Y - An array which contains the dependent variable data.

F - An array which will contain the Chebychev polynomial values upon exiting the routine.

N - The number of data points. The arrays X, Y, and F must be dimensioned of at least length N.

M - The order of the Chebychev polynomial to use. M must not be greater than 7.

COEF - Upon exiting the routine, this array will contain the coefficients of the fitting polynomial. COEF must be dimensioned of at least length M.

NOTES:

1. A binary deck of CHEBFIT is stored on file *CHEBFIT.

EXPFIT

ABSTRACT:

The subroutine EXPFIT computes the least squares fit to a set of data using up to six exponential terms.

USAGE:

The calling sequence is:

```
CALL EXPFIT(X,Y,F,N,P,M,COEF)
```

where the calling parameters are:

X - An array which contains the independent variable data.

Y - An array which contains the dependent variable data.

F - An array which will contain the exponential function values upon exiting the routine.

N - The number of data points. The arrays X, Y, and F must be dimensioned of at least length N.

P - The nonlinear term used in each exponential term. Each exponential term is of the form:

$$c_i \cdot e^{-i \cdot p \cdot X_j}$$

M - The number of exponential terms to use in the fit. M must be in the range 1-6.

COEF - Upon exiting the routine, the array COEF will contain the coefficients of each exponential term. COEF must be dimensioned at least of length M.

NOTES:

1. A binary deck of EXPFIT is stored on file *EXPFIT.

SUBROUTINE HIST

ABSTRACT:

HIST produces a histogram of a set of data. The intervals are assumed to be equally spaced between limits specified by the user.

USAGE:

The calling sequence is:

```
CALL HIST(A,LA,HINIT,HFINAL,NINC,FREQ)
```

where the calling parameters are:

A	The array of data points.
LA	The number of data points in the array A to be used in the histogram.
HINIT	The initial interval value.
HFINAL	The final interval value.
NINC	The number of equally spaced intervals between HINIT and HFINAL.
FREQ	An array of at least length NINC. Upon return this array contains the frequence of occurance of each interval.

NOTES:

1. The binary deck of HIST is stored on the file *HISTB.
2. HIST uses plot drivers stored on file *PLOTDR.

SUBROUTINE NLLSQ(X,Y,N,B,K,ITER,P)

ABSTRACT:

NLLSQ performs a nonlinear least squares fit to a set of data using a model function supplied by the user.

METHOD:

The algorithm used was developed by D. W. Marquardt and is described completely in

> "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," J.SIAM (Vol. 11, No. 2) June, 1963, pp. 431-441.

USAGE:

The calling sequence is:

```
CALL NLLSQ(X,Y,N,B,K,ITER,P)
```

where the calling parameters are:

X - An array containing the values of the independent variable at the N observed data points.

Y - An array containing the values of the dependent variable at the N observed data points.

N - The number of data points to be used in the least squares fit.

B - An array containing the nonlinear parameter. Upon entering the routine this array must contain the initial estimates of the parameter, and upon exiting the routine the array will contain the converged parameter values.

K - The number of nonlinear parameters in the model.

ITER - The maximum number of iteration of the algorithm to be applied before stopping. If convergence is not reached before ITER iteration, an error message is printed and the calculation terminated.

P - An array of length K used by the model subroutine to store the partial derivatives of the function with respect to the parameters.

NOTES:

1. The user must supply a subroutine MODEL with the following calling sequence:

```
CALL MODEL(X,Y,B,I,K,P,RE,F)
```

   where the calling parameters are:

   X - An array containing the value of the independent variable of the observed data points.

   Y - An array containing the value of the dependent variable of the observed data points.

   B - An array containing the nonlinear parameter estimates.

   I - The index of the X array of the value of the independent variable at which the model is to be evaluated.

   K - The number of nonlinear parameters.

   P - The MODEL subroutine must evaluate the partial derivatives of the function (at the indicated value of the independent variable) with respect to the K parameters and store them in this array.

   RE - The residual defined by RE=Y(I)-F where F is the value of the modeling function at the point X(I).

   F - The value of the modeling function at the point X(I).

2. The binary decks of NLLSQ and the routines it requires are stored on the file *NLLSQB1. One needs to load this file along with the binary deck of the model subroutine being used.

SUBROUTINE POWDER1

ABSTRACT:

POWDER1 produces a randomly generated powder plot representation of a one-dimensional array of data. POWDER1 uses graphics input to determine the region in which to produce the powder plot. The region will be rectangular and is specified by positioning the crosshairs at one endpoint of the diagonal and depressing any key. When the crosshairs come on again, position them to the other end of the diagonal and depress any key.

METHOD:

The algorithm used was developed by Robert Ehrlich and is described completely in

> "Physical Simulations for an On-Line Computer-Controlled Oscilloscope," Computers in Undergraduate Science Education Conference Proceedings, August, 1970, p. 220.

USAGE:

The calling sequence is:

```
CALL POWDER1(A,NELM,NPTS)
```

where the calling parameters are:

A -	The array of data.
NELM -	The number of elements in the A array.
NPTS -	The number of points to be plotted in the powder plot (500 usually produces a pleasing display).

NOTES:

1. The binary deck of POWDER1 is saved on the file *POWDER.
2. Before POWDER1 may be used, the button marked Keyboard/ Aux must be set to both <u>keyboard and Aux</u>.
3. The maximum value of A must be greater than zero.
4. POWDER1 uses plot drivers stored on the file *PLOTDR.

SUBROUTINE POWDER2

ABSTRACT:

POWDER2 produces a two-dimensional, randomly generated powder plot of a function. The function of two variables is assumed to be separable into a product of two functions of one variable (i.e., F(X,Y) = FX(X)*FY(Y)). The powder plot is rectangular, and the region is specified using graphics input. The crosshairs are positioned at one end of the diagonal and any key is depressed. When the crosshairs come on again, position them to the other end of the diagonal and depress any key.

METHOD:

The algorithm is an adaptation of that described by Robert Ehrlich in

"Physical Simulations for an On-Line Computer-Controlled Oscilloscope," Computers in Undergraduate Science Education Conference Proceedings, August, 1970, p. 221.

USAGE:

The calling sequence is:

```
     CALL POWDER2(X,NX,Y,NY,NPTS)
```

where the calling parameters are:

X	Array containing the values of FX.
NX	Number of elements in the X array.
Y	Array containing the values of FY.
NY	Number of elements in the Y array.
NPTS	Number of points to be plotted in the powder plot (1500 usually produces a pleasing display).

NOTES:

1. The binary deck of POWDER2 is saved on the file *POWDER.
2. Before POWDER2 may be used, the button marked Keyboard/ Aux must be set to both keyboard and Aux.
3. The product of the maximum value of X and the maximum value of Y must be greater than zero.
4. POWDER2 uses plot drivers stored on the file *PLOTDR.

SUBROUTINE POWDER2P

ABSTRACT:

POWDER2P produces a two-dimensional, polar randomly generated powder plot of a function. The function of two variables is assumed to be separable into a product of two functions of one variable (i.e., $F(r,\Theta) = FR(r)*FTH(\Theta)$). The powder plot is circular and is centered in a rectangular region specified using graphics input. The crosshairs are positioned at one end of the diagonal and any key is depressed. When the crosshairs come on again, position them to the other end of the diagonal and depress any key. The powder plot will fall in the largest circle that can be centered within the rectangle specified.

METHOD:

The algorithm is an adaptation of that described by Robert Ehrlich in

> "Physical Simulations for an On-Line Computer-Controlled Oscilloscope," Computers in Undergraduate Science Education Conference Proceedings, August, 1970, p. 221.

USAGE:

The calling sequence is:

```
CALL POWDER2P(R,NR,TH,NTH,THIN,THFN,NPTS)
```

where the calling parameters are:

R -	Array containing the values of FR.
NR -	Number of elements in the R array.
TH -	Array containing the values of FTH.
NTH -	Number of elements in the TH array.
THIN -	Initial angle at which to start the powder plot.
THFN -	Final angle in the powder plot. (Normally, THIN and THFN would be set to zero and 2π

respectively. This would correspond to a circular powder plot; however, if only a wedge-shaped portion of the circular powder plot were wanted, THIN and THFN could be given the appropriate values.)

NPTS - Number of points to be plotted in the powder plot (1500 usually produces a pleasing display).

NOTES:

1. The binary deck of POWDER2P is saved on the file *POWDER.
2. Before POWDER2P may be used, the button marked Keyboard/Aux must be set to both keyboard and Aux.
3. The product of the maximum value of R and the maximum value of TH must be greater than zero.
4. POWDER2P uses plot drivers stored on the file *PLOTDR.

SUBROUTINE SURFACE

ABSTRACT:

SURFACE produces a perspective drawing of a function of two variables (i.e., a two dimensional array) with the hidden lines removed. The perspective view point may be chosen by specifying the coordinates of the view point.

METHOD:

The algorithm used to generate the perspective drawing and to remove the hidden lines is described completely in

"The Perspective Representation of Functions of Two Variables," J.ACM (Vol. 15, No. 2), April, 1968, pp. 193-204.

USAGE:

The calling sequence is:

```
CALL SURFACE(A,XL,XH,M,YL,YH,N,ZL,ZH,DX,DY,DZ)
```

where the calling parameters are:

A - A two dimensional array of M rows and N columns. The data is assumed to be stored in the following way:

$$A(1,1) = F(XL,YL), A(1,2) = F(XL,YL+\Delta Y),\ldots$$
$$A(2,1) = F(XL+\Delta X,YL), A(2,2) = F(XL+\Delta X,YL+\Delta Y),\ldots$$
$$\vdots$$

where $\Delta X = \frac{XH-XL}{M-1}$ and $\Delta Y = \frac{YH-YL}{M-1}$.

For any given columns of the matrix, X increases as the first indice increases and for any given row of the matrix, Y increases as the second indice increases.

XL - Initial value of X.

XH - Final value of X.

M - The number of rows, i.e., the number of individual X values.

YL - Initial value of Y.

YH - Final value of Y.

N - The number of columns, i.e., the number of individual Y values.

ZL - The lowest Z coordinate of any viewable point. Points which have lower Z coordinates will be hidden.

ZH - The largest Z coordinate of any viewable point. Points which have larger Z coordinates will be hidden.

DX - The X coordinate of the view point.

DY - The Y coordinate of the view point.

DZ - The Z coordinate of the view point.

NOTES:

1. The binary deck of SURFACE is stored on the file *SURF6B.
2. SURFACE uses plot drivers which are stored on the file *PLOTDR.
3. SURFACE calls several other subroutines which are in the package. Communication between the subroutines is done largely through the labeled common block LAB1.