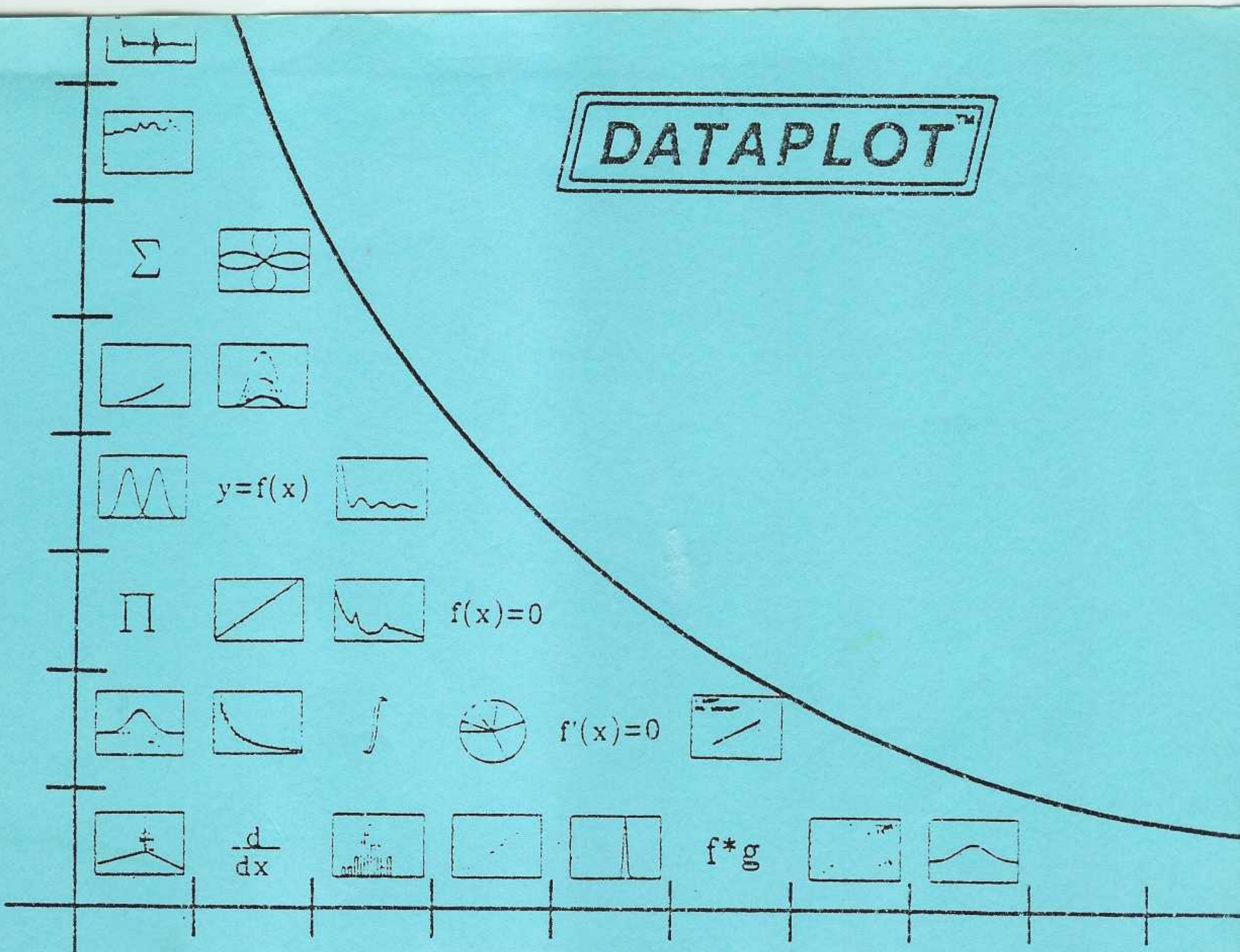
DATAPLOT™



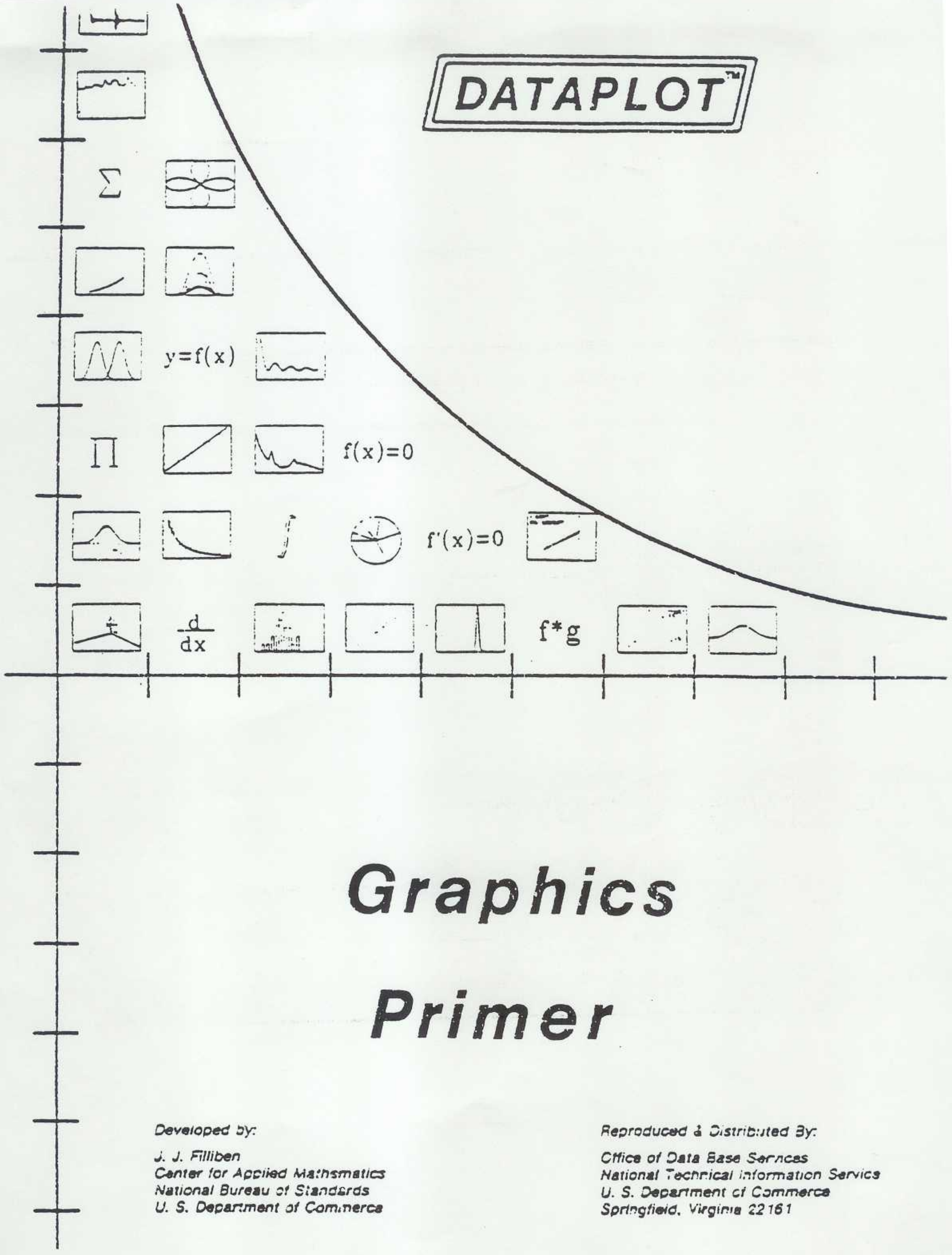
Graphics Primer

Developed by:

J. J. Filliben
Center for Applied Mathematics
National Bureau of Standards

Reproduced & Distributed By:

Office of Data Base Services
National Technical Information Service
U. S. Department of Commerce



Graphics Primer

Developed by:

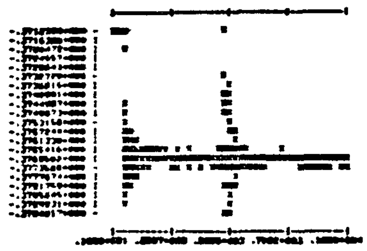
J. J. Filliben
Center for Applied Mathematics
National Bureau of Standards
U. S. Department of Commerce

Reproduced & Distributed By:

Office of Data Base Services
National Technical Information Service
U. S. Department of Commerce
Springfield, Virginia 22161

Plot Data at a Discrete Terminal

Problem



Data exists on a file ABC as a series of (x,y) pairs. Two numbers—an x and a y—are on each line image. There is an indeterminate (and unimportant) number of line images in the file. Read in the (x,y) data points from the file in a format-free fashion. Plot the data on a narrow-width (72 character), discrete terminal (e.g., TI 700, Hazeltine, Omron). (Note that DATAPLOT has discrete (and batch) analogues to almost all of the continuous plots as shown in the remaining examples.) See figure for the discrete terminal plot.

DATAPLOT Program

```
READ file name X Y
DISCRETE
PLOT Y X
```

Program Description

<u>READ ABC. X Y</u>
carries out a format-free read of data from file ABC; the file name (ABC) (on most sytems) must terminate with a period. The data is read into variables X and Y; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

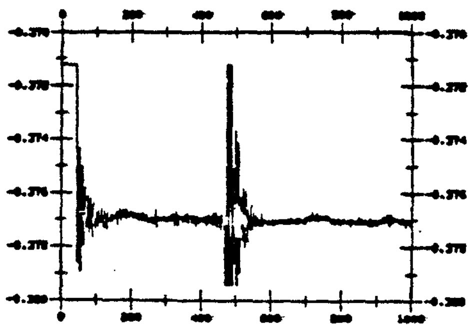
<u>DISCRETE</u>
specifies that all succeeding plots will be in a narrow-width, non-continuous format as would be suitable for a narrow-width/non-Tektronix/ alphanumeric terminal (e.g., Texas Instrument Silent 700). The discrete plots will be 70 characters wide and 24 characters deep.

<u>PLOT Y X</u>
generates a plot of variable Y (vertically) versus variable X (horizontally). The plot limits will be automatic and float with the data. Because of the prior DISCRETE command, such a plot will be discrete.

Plot Data

Problem

Data exists on a file ABC as a series of (x,y) pairs. Two numbers—an x and a y—are on each line image. There is an indeterminate (and unimportant) number of line images in the file. Read in the (x,y) data points from the file in a format-free fashion. Plot them. See figure for the output.



DATAPLOT Program

```
     READ ABC. X Y
     PLOT Y X
```

Program Description

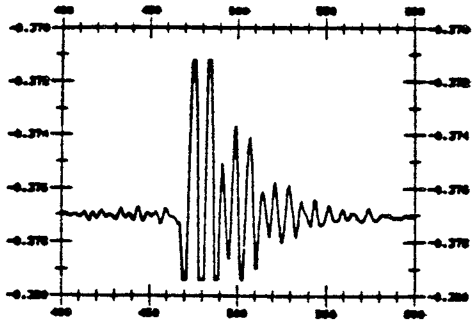
<u>READ ABC. X Y</u>
carries out a format-free read of data from file ABC; the file name (ABC) (on most sytems) must terminate with a period. The data is read into variables X and Y; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

<u>PLOT Y X</u>
generates a plot of variable Y (vertically) versus variable X (horizontally). The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

Plot a Subset of the Data

Problem

Read in (x,y) points from a file ABC. Plot them but restrict the plot to only those points between x = 400 to x = 600. This demonstrates the subsetting feature and will here have the effect of "blowing up" the plot. In general, subsetting may be done on any variable (not just those involved in the plot), may be done for combinations of variables, and may be appended to all graphics and analysis commands.



DATAPLOT Program

```
READ ABC. X Y
PLOT Y X SUBSET X 400 TO 600
```

Program Description

<u>READ ABC. X Y</u>
carries out a format-free read of data from file ABC; the file name (ABC) (on most sytems) must terminate with a period. The data is read into variables X and Y; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

<u>PLOT Y X SUBSET X 400 TO 600</u>
generates a plot of variable Y (vertically) versus variable X (horizontally); only those values of the X and Y variables which correspond to values of the X variable between 4 and 6 (inclusively) are included in the plot. The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

Plot a Mixture of Data and Functions

Problem

Read in (x,y) pairs from file ABC. Plot these points as discrete X's. Superimpose a continuous functional trace with a solid line.

DATAPLOT Program

```
READ ABC. X Y
.
LET A0 = -181
LET A1 = 142
LET B1 = 2.8
LET FUNCTION F = (A0+A1*X)/(1+B1*X)
.
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y X AND
PLOT F FOR X = 1.65 .1  3.3
```

Program Description

READ ABC. X Y
carries out a format-free read of data from file ABC; the file name (ABC) (on most sytems) must terminate with a period. The data is read into variables X and Y; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

.
is a null command--it visually separates chunks of DATAPLOT code.

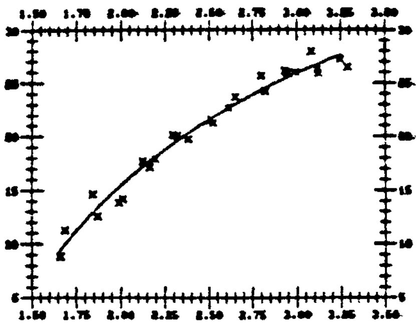
LET A0 = -181
defines the parameter A0 and assigns to it the value -181.

LET A1 = 142
defines the parameter A1 and assigns to it the value 142.

LET A2 = 2.8
defines the parameter A2 and assigns to it the value 2.8.

LET FUNCTION F = (A0+A1*X)/(1+B1*X)
*defines the function F and assigns to it the 18-character string (A0+A1*X)/(1+B1*X)*

CHARACTERS X BLANK
specifies (for future plots) that the first trace will have X's as plot characters and the second trace will have blanks as plot characters (that is, the second trace will have no characters at the plot points).



LINES BLANK SOLID
specifies (for future plots) that the first trace will have blanks as plot line types (that is, the first trace will have no connecting line between plot points), and the second trace will have solid connecting lines between the plot points.

PLOT Y X AND
PLOT F FOR X = 1.65 .1 3.3
generates a plot with 2 traces. Due to the prior CHARACTERS and LINES commands, the first trace will have X's as plot characters with blank (= no) connecting lines, and the second trace will have blanks as plot characters with solid connecting lines. The first trace is a plot of variable Y (vertically) versus variable X (horizontally); The second trace is a plot of the function F evaluated at a series of X values starting with X = 1.65, at increments of .1, and ending with X = 3.3 (that is, X = 1.65, 1.75, 1.85, ..., 3.15, 3.25). Note that these internally-generated X values at which the function F is evaluated have nothing to do with the variable X that was read in. The X values for the function evaluation are temporary and internally-generated. After the multi-trace plot is generated, the Y and X variables will be unchanged. The AND suffix at the end of PLOT Y X AND is important-- it tells DATAPLOT to generate only 1 plot, but on that 1 plot to have 2 traces--a plot of Y versus X, and then (on the same plot) to superimpose a plot of the function F. When the AND is included, the following occurs--

1) the usual screen pre-erase occurs;

2) the plot of Y versus X is generated;

3) the plot of the function F is superimposed

If the AND were omitted, then the following would occur-

1) the usual screen pre-erase occurs for the first plot;

2) the plot of Y vesus X is generated;

3) the usual screen pre-erase occurs for the second plot (thereby wiping out the Y versus X plot);

4) the plot of the function F is generated.

The default plot type is continuous; the default axis will be neat and float with the data/function.

Note that the function could have been explicitely included in the PLOT statement as opposed to an indirect reference via F. Thus an alternate form for

```
PLOT Y X AND
PLOT F FOR X = 1.65 .1 3.3
```

is

```
PLOT Y X AND
PLOT (A0+A1*X)/(1+B1*X) FOR X = 1.65 .1 3.3
```

Plot Multiple Data Traces

Problem

Data exists on a file ABC for several variables (data vectors) with corresponding points on the same line image, and with some indeterminate (and unimportant) number of line images. Read in the data points for the several different variables. Plot them (6 traces will result for this example). Have all traces solid except for the first trace which is to have a dotted line.

DATAPLOT Program

```
READ ABC. X Y1 Y2 Y3 Y4 Y5 Y6
LINES DOTTED SOLID SOLID SOLID SOLID SOLID
PLOT Y1 Y2 Y3 Y4 Y5 Y6 VERSUS X
```

Program Description

READ ABC. X Y1 Y2 Y3 Y4 Y5 Y6
carries out a format-free read of data from file ABC; the file name (ABC) (on most sytems) must terminate with a period. The data is read into variables X, Y1, Y2, Y3, Y4, Y5, and Y6; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y1; the third number is read into the variable Y2; the fourth number is read into the variable Y3; the fifth number is read into the variable Y4; the sixth number is read into the variable Y5; the seventh number is read into the variable Y6. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

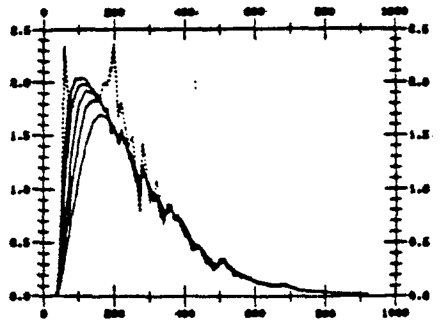
LINES DOTTED SOLID SOLID SOLID SOLID SOLID
specifies (for future plots) that the first trace will have dotted line type; and the second through sixth traces will have solid line types. The allowable line types in DATAPLOT have 2-character abbreviations (e.g., BL for BLANK, SO for SOLID, DO for DOTTED, DA for DASHED, DA1 for DASH1, DA2 for DASH2, ..., DA20 for DASH20. Thus an alternate form for

```
LINES DOTTED SOLID SOLID SOLID SOLID SOLID
```

is

```
LINES DO SO SO SO SO SO
```

PLOT Y1 Y2 Y3 Y4 Y5 Y6 VERSUS X
generates a multi-trace plot with 6 traces—Y1 versus X, Y2 versus X, Y3 versus X, Y4 versus X, Y5 versus X, and Y6 versus X. Due to the prior use of the LINES DOTTED SOLID SOLID SOLID SOLID SOLID command, the first trace (Y1 VERSUS X) will have dotted connecting line, while the remaining 5



traces will each have solid connecting line. The default plot type is continuous; the default character type is blank for all traces; the default axis limits is neat and float with the data.

There are numerous alternate forms for generating multi-trace plots in DATAPLOT, for example,

```
PLOT Y1 Y2 Y3 Y4 Y5 Y6 VERSUS X
```

or

```
PLOT Y1 Y2 Y3 Y4 Y5 Y6 VS X
```

or

```
PLOT Y1 Y2 Y3 Y4 Y5 Y6 VS. X
```

or

```
PLOT Y1 VERSUS X AND
PLOT Y2 VERSUS X AND
PLOT Y3 VERSUS X AND
PLOT Y4 VERSUS X AND
PLOT Y5 VERSUS X AND
PLOT Y6 VERSUS X
```

or

```
PLOT Y1 X AND
PLOT Y2 X AND
PLOT Y3 X AND
PLOT Y4 X AND
PLOT Y5 X AND
PLOT Y6 X
```

or

```
PLOT Y1 Y2 Y3 VERSUS X Y4 Y5 Y6 VERSUS X
```

or

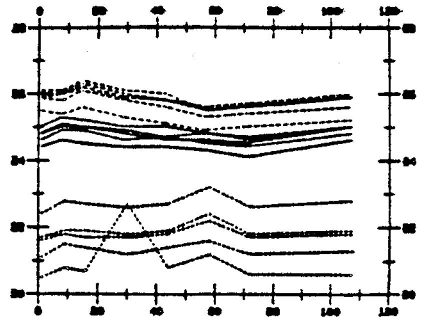
```
PLOT Y1 Y2 VERSUS X Y3 Y4 VERSUS X Y5 Y6 VERSUS X
```

and so forth.

Generate a Multi-Trace Plot to Emphasize Between-Group Effect

Problem

A response variable Y is dependent on 3 independent variables--time X, operator OP, and replication (within operator) REP. Data exists on a file ABC for these 4 variables consisting of the 4 corresponding values on a given line image. Read in the data. Suppose it is known that the operator variable will have 3 distinct values (1, 2, and 3), and the replication variable will have 5 distinct values (1 to 5). Generate a multi-trace plot of Y versus X consisting of 3 x 5 = 15 distinct traces. Emphasize between-operator effects by having the 5 traces for operator 1 solid, the 5 traces for operator 2 dashed, and the 5 traces for operator 3 dotted.



DATAPLOT Program

```
        READ ABC. Y X OP REP
        LET TAG = 5*(OP-1)+REP
        LINES SO SO SO SO SO DA DA DA DA DA DO DO DO DO DO
        PLOT Y X TAG
```

Program Description

READ ABC. Y X OP REP
carries out a format-free read of data from file ABC; The data is read into variables Y, X, OP, and REP; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X; the third number is read into the variable OP; the fourth number is read into the variable REP.

LET TAG = 5(OP-1)+REP*
forms a new variable TAG. TAG has the same number of elements as OP and REP. An individual element in TAG is computed by taking an individual element in OP, subtracting 1 from it, then multiplying by 5, and then adding the corresponding individual element in the variable REP. The rationale for forming TAG is that our ultimate objective is to form a plot which has 15 traces in it--where each trace represents a unique OP and REP combination. Since there are 3 distinct OP values (1, 2, and 3), and since there are 5 distinct REP values (1 through 5), then there are 3 x 5 = 15 distinct combinations. Since the 3-argument form for the PLOT command will be used to generate the multi-trace plot, we use this LET command to form the third argument needed in that PLOT statement. The TAG variable will have 15 unique values--1 through 15. For distinct values of OP and REP, the computed distinct value of TAG is as follows--

OP	REP	TAG
1	1	1
1	2	2
1	3	3
1	4	4
1	5	5
2	1	6
2	2	7
2	3	8
2	4	9
2	5	10
3	1	11
3	2	12
3	3	13
3	4	14
3	5	15

The above represents the distinct values of OP, REP, and TAG; the actual number of elements in TAG will be the same as the number of elements in OP and REP variables (= the number of elements in the Y and X variables).

LINES SO SO SO SO SO DA DA DA DA DA DO DO DO DO DO
defines (for future plots) the 15 lines types to be associated with the 15 traces which will be generated (when the PLOT command is entered). The first 5 traces will be solid; the next 5 traces will be dashed; the last 5 traces will be dotted. Note that SO is an acceptable abbreviated form for SOLID, DA is an abbreviated form for DASH or DASHED, and DO is an abbreviated form for DOT or DOTTED. The abbreviated form was used above so that the LINES command statement would fit on one line image; when many traces are being specified, it is common to use the abbreviated forms for line (and character) specifications.

<u>PLOT Y X TAG</u>
generates the multi-trace plot; 15 traces are generated. Each trace consists of a plot of the Y variable (vertically) versus the X variable (horizontally); however, each individual trace is restricted to only a subset of Y and X values. The subset is defined via the contents of the TAG variable. For the first trace, the following procedure is executed--

1) *the entire TAG variable is scanned and the minimum value of TAG is identified (in this case, the minimum value of TAG is 1);*

2) *all entries in the Y and X variables which correspond to values in the TAG variable of 1 are extracted;*

3) *this extracted subset of Y and X values is plotted;*

4) *the line type for this first trace will be dictated by the first entry of a previous LINES command-- in this case, the first entry was SO (= SOLID), and so a solid line will result; Similarly, the character type for the first trace will be dictated by the first entry of a previous CHARACTERS command-- since the CHARACTERS command was not previously entered, the default will be used, namely to have blank plot characters.*

For the second trace, the following procedure is executed--

1) *the entire TAG variable is scanned and the next larger value of TAG is identified (in this case, the next largerr value of TAG is 2);*

2) *all entries in the Y and X variables which correspond to values in the TAG variable of 2 are extracted;*

3) *this extracted subset of Y and X values is plotted;*

4) *the line type for this second trace will be dictated by the second entry of a previous LINES command-- in this case, the second entry was SO (= SOLID), and so a solid line will result; Similarly, the character type for the second trace will be dictated by the second entry of a previous CHARACTERS command-- since the CHARACTERS command was not previously entered, the default will be used, namely to have blank plot characters.*

An so forth up to the fifteenth trace; for the fifteenth trace, the following procedure is executed--

1) *the entire TAG variable is scanned and the largest value of TAG is identified (in this case, the minimum value of TAG is 15);*

2) *all entries in the Y and X variables which correspond to values in the TAG variable of 15 are extracted;*

3) *this extracted subset of Y and X values is plotted;*

4) *the line type for this fifteenth trace will be dictated by the fifteenth entry of a previous LINES command-- in this case, the fifteenth entry was DO (= DOTTED), and so a dotted line will result; Similarly, the character type for the fifteenth trace will be dictated by the fifteenth entry of a previous CHARACTERS command-- since the CHARACTERS command was not previously entered, the default will be used, namely to have blank plot characters. Note that the identical plot would have been*

Note that the identical plot would have been formed if we had defined the TAG variable via

```
LET TAG = 4*(OP-1) + REP
```

or

```
LET TAG = 10*(OP-1) + REP
```

The TAG variable as we have defined it (via LET TAG = 5(OP-1) + REP) will have sequential integer values 1 through 15. In general, the TAG variable need not have sequential integer values, nor need it even have integer values. The important point is that however the TAG variable is defined, the smallest value of TAG corresponds to the first trace to be plotted, the next larger value corresponds to the second trace to be plotted, and so forth. If TAG had values 1 to 5, 11 to 15, and 21 to 25 (as would result from LET TAG = 10*(OP-1) + REP), or values 1 to 5, 101 to 105, 201 to 205 (as would result from LET TAG = 100*(OP-1) + REP), the procedure for forming the plot would still be the same-- the TAG variable would be scanned from smallest values to largest values, and each larger value would use the next available lines and characters specification. The resulting plot would be identical in all 3 cases.*

In this example, we have used the name TAG to represent the name of the variable that is the third argument in the PLOT command; the choice of TAG was by personal preference--any name may have been used.

BAR PATTERN HORIZONTAL
specifies (for future Bar Plots) that the pattern within the bar is a horizontal line. DATAPLOT provides 11-within bar patterns--

BAR PATTERN BLANK	*no pattern*
BAR PATTERN VERTICAL	*vertical*
BAR PATTERN HORIZONTAL	*horizontal*
BAR PATTERN D1	*diagonal (up)*
BAR PATTERN D2	*diagonal (down)*
BAR PATTERN D1D2	*diagonals (up and down)*
BAR PATTERN VED1	*vertical and diagonal (up)*
BAR PATTERN VED2	*vertical and diagonal (down)*
BAR PATTERN HOD1	*horizontal and diagonal (up)*
BAR PATTERN HOD2	*horizontal and diagonal (down)*
BAR PATTERN VEDO	*vert., hor., & both diagonals*

In practice, the first 6 are heavily used and the last 5 are rarely used.

The default bar pattern is blank; that is, no pattern.

BAR PLOT Y1 X1
generates a bar plot. The Y1 variable will be plotted vertically and the X1 variable will be plotted horizontally. The bars will be centered at each of the values of the X1 variable (which will be a bit left of the values in the X variable). Due to prior settings of the BAR PATTERN and BAR WIDTH commands, the bars will have horizontal stripes (at the default spacing of 3), and the width of each bar will be .2 Due to the prior XLIMITS and YLIMITS commands, the axis limits will be 1974 to 1980. Due to the prior setting of the FONT command, the title and labels will be written out in triplex italic font with default upper case. Due to the prior entry of the PRE-ERASE OFF command, the screen will not automatically erase before this bar plot; however, due to the prior entry of the ERASE command, and erase will have been "manually" caused. The net effect is that the screen will be clear as this first bar plot (involving the horizontal bars only) is plotted.

BAR PATTERN D1D2
specifies (for future Bar Plots) that the pattern within the bar is criss-cross diagonal lines.

BAR PLOT Y2 X2
generates a second bar plot. The Y2 variable will be plotted vertically and the X2 variable will be plotted horizontally. The bars will be centered at each of the values of the X2 variable (which will be a bit right of the values in the X variable). Due to prior settings of the BAR PATTERN and BAR WIDTH commands, the bars will have criss-cross stripes (at the default spacing of 3), and the width of each bar will be .2 Due to the prior XLIMITS and YLIMITS commands, the axis limits will be 1974 to 1980 (which is identical to the limits for the first bar plot). Due to the prior setting of the FONT command, the title and labels will be written out in triplex italic font of the PRE-ERASE OFF command, the screen will not automatically erase before this bar plot; and so this second bar plot will be overlayed atop the first bar plot.

MOVE 20 80
moves the cursor to a point 20% across the screen and 80% of the way up the screen. The rationale for this command is to move the cursor to a point where a text string will be genrated (via the subseqeunt TEXT command).

TEXT HORIZONTAL STRIPE = MAINFRAME
generates the text string

HORIZONTAL STRIPE = MAINFRAME

on the screen at the current cursor location (= (20,80)). Due the previous FONT command, the text string will be in triplex italic font. Since no previous CASE, JUSTIFICATION, HEIGHT, and WIDTH commands were entered, then the text string will be in the default upper case, left justified, with a height of 3 (= 3% of screen height) and a width of 2 (= 2% of screen width).

MOVE 20 76
moves the cursor to a point 20% across the screen and 76% of the way up the screen. The rationale for this command is to move the cursor to a point where another text string will be generated (via the subseqeunt TEXT command).

TEXT CRISS-CROSS STRIPE = MINICOMPUTERS
generates the text string

CRISS-CROSS STRIPE = MINICOMPUTERS

on the screen at the current cursor location (= (20,76)). Due the previous FONT command, the text string will be in triplex italic font. Since no previous CASE, JUSTIFICATION, HEIGHT, and WIDTH commands were entered, then the text string will be in the default upper case, left justified, with a height of 3 (= 3% of screen height) and a width of 2 (= 2% of screen width).

COPY
copies (to the local hardcopy unit) the current contents of the screen. Anytime a COPY command is encountered in a DATAPLOT program, the screen is immediately copied.

Generate a Plot to Emphasize Spread Due to Replication

Problem

A response variable Y is dependent on a single independent variable X. Data exists on a file consisting of corresponding values of Y and X on a given line image. Note that replication may exist—that is, a given value of X may have several values of Y (for example, X = .5 could have 3 replicates associated with it—86, 88, and 94). Replicated Y values are indicated not by multiple Y values on the same line image, as in

```
86 88 94 .5
```

rather, they are indicated by multiple line images with a common X value, as in

```
86   .5
88   .5
94   .5
```

In the above data, there is replication at X = .5 (namely, Y = 86, 88, and 94).

Read in the data into variables Y and X.

Plot the data. Emphasize the spread between replicates for each fixed X value by having these replicates for a given X connected by individual solid traces. Have the data values themselves displayed by using x's as a plot character. Have the height of the x's equal to 1.75% of the total vertical screen height.

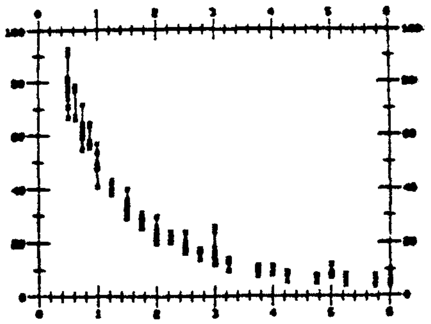
DATAPLOT Program

```
READ ABC. Y X
CHARACTERS X ALL
CHARACTER SIZE 1.75 ALL
PLOT Y X X
```

Program Description

<u>READ ABC. Y X</u>
carries out a format-free read of data from file ABC; The data is read into variables Y and X; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X.

<u>CHARACTERS X ALL</u>
specifies (for future plots) that all traces have x's at the plot points (the default is to have blanks at all plot points of all traces). The rationale for setting the plot characters to x's is that we would like to see the individual data points.



<u>CHARACTER SIZE 1.75 ALL</u>
specifies (for future plots) that the height of all plot characters (x's in this case) be 1.75% of total vertical screen distance. The default character size is 3 (= 3% of total vertical screen distance). The size has been changed from the default because (for the data set used in the example) the x's were too crowded—they dominated the plot too much and were a distraction to the main objective—study the variation due to replication. A character size of 1.75 was specified to illustrate that non-integer size may be specified. More commonly one would find that a charactr size of 2 or 1 would be visually adequate.

<u>PLOT Y X X</u>
generates the desired multi-trace plot where an individual trace consists of a line connecting all replicated values at a given x value. Generating a plot to emphasize replication is really a special case of the 3-argument PLOT command for generating multi-trace plots. Each trace consists of a plot of the Y variable (vertically) versus the X variable (horizontally); however, each individual trace is restricted to only a subset of Y and X values. The subset is defined via the thrid argument of the PLOT command— namely the X variable. Any variable may be used as this thrid argument; usually it is not Y or X, but it can be, and in this instance, the choice of X for the thrid variable allows the desired plot to be generated.

For the first trace, the following procedure is executed—

1) *the entire X variable is scanned and the minimum value of X is identified (in this case, the minimum value of X is .5);*

2) *all entries in the Y and X variables which correspond to values in the X variable of .5 are extracted;*

3) this extracted subset of Y and X values is plotted;

4) the line type for this first trace will be dictated by the first entry of a previous LINES command-- since the LINES command was not previously entered, the default will be used, namely to have solid line type; Similarly, the character type for the first trace will be dictated by the first entry of a previous CHARACTERS command-- since the previous CHARACTERS command had specified x's for the plot character type for all plot points of all traces, then the plot character for the first trace will be an x. Further, the size for the characters in the first trace will be dictated by the first entry of a previous CHARACTER SIZE command-- since the previous CHARACTER SIZE command had specified 1.75 for the plot character size for all plot points of all traces, then the plot character size for the first trace will be 1.75% of total vertical screen height. characters.

For the second trace, the following procedure is executed--

1) the entire X variable is scanned and the next larger value of X is identified (in this case, the minimum value of X is .6);

2) all entries in the Y and X variables which correspond to values in the X variable of .6 are extracted;

3) this extracted subset of Y and X values is plotted;

4) the line type for this second trace will be dictated by the second entry of a previous LINES command-- since the LINES command was not previously entered, the default will be used, namely to have solid line type; Similarly, the character type for the second trace will be dictated by the second entry of a previous CHARACTERS command-- since the previous CHARACTERS command had specified x's for the plot character type for all plot points of all traces, then the plot character for the second trace will be an x. Further, the size for the characters in the second trace will be dictated by the second entry of a previous CHARACTER SIZE command-- since the previous CHARACTER SIZE command had specified 1.75 for the plot character size for all plot points of all traces, then the plot character size for the second trace will be 1.75% of total vertical screen height.

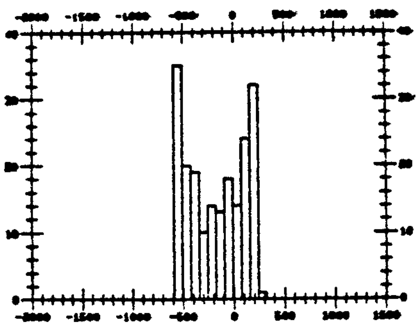
A similar procedure is followed for the formation of all traces. Note that for this example, we do not need to know ahead of time how many individual traces there will be. Since the CHARACTERS and CHARACTER SIZES commands specified settings for all such traces (via use of the keyword ALL in the commands), then the settings will hold regardless of how many traces result.

From a data analysis point of view, the plot shows that the usual constant-variation assumption does not hold for this data set. Note that the variation within replicates is much larger for small values of X than for large values of X. As a consequence of this, one would not carry out a fit (for example) of this data set without proper weighting or variable transformation.

Generate a Histogram

Problem

Data exists on a file as a series of values (1 value per line image). Read in the data. Generate a histogram. Have the class limits automatically determined. The histogram is a graphical data analysis technique for displaying distributional information in a data set. See figure 8.



DATAPLOT Program

```
READ ABC. X
HISTOGRAM X
```

Program Description

READ ABC. X
carries out a format-free read of data from file ABC. The first number on each line image is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

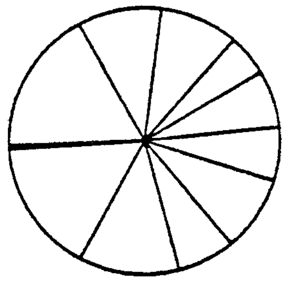
HISTOGRAM X
generates a histogram. The HISTOGRAM command (as with all DATAPLOT Graphics category commands) makes use of current settings for plot character type and line type. For the usual continuous histogram, the proper setting is blank plot characters and solid line types; since this is the default, the CHARACTERS and LINES commands need not be used. The class width is automatically computed (as 0.3 x s where s is the sample standard deviation for the data); to override the default, the analkyst would enter the CLASS WIDTH command before the HISTOGRAM command. The first class and the last class are also automatically computed (as xbar +- 6 x s, where xbar is the sample mean of the data). To override this, the analyst enters the CLASS LOWER and CLASS UPPER commands, respectively. In carrying out analysis graphics, the analyst rarely needs to use anything other than the default settings for the classes; for the vast majority of data sets, the default settings will yield histograms with a near-maximal amount of information. For persentation graphics, the analyst may find him/herself adjusting the class settings so as to generate a histogram with pre-determined settings.

From an interpretational point of view, this data set has a symmetric distribution, though decidely non-normal and bimodal.

Generate a Pie Chart

Problem

Data exists on a file as a series of values (1 value per line image). Read in the data. Generate a pie chart. Have the class limits automatically determined. The pie chart is a graphical data analysis technique for displaying distributional information in a data set. See figure 9.



DATAPLOT Program

```
READ ABC. X
PIE CHART X
```

Program Description

READ ABC. X
carries out a format-free read of data from file ABC. The first number on each line image is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

PIE CHART X
generates a pie chart. The PIE CHART command (as with all DATAPLOT Graphics category commands) makes use of current settings for plot character type and line type. Pie charts invariably have no plot characters and solid lines--since this is the DATAPLOT default the CHARACTERS BLANK and LINES SOLID commands need not be entered.

As with the histogram, the class width for the pie chart is automatically computed (as 0.3 x s where s is the sample standard deviation for the data); to override the default, the analyst would enter the CLASS WIDTH command before the PIE CHART command. The first class and the last class are also automatically computed (as xbar +- 6 x s, where xbar is the sample mean of the data). To override this, the analyst enters the CLASS LOWER and CLASS UPPER commands, respectively. For analysis graphics, the analyst rarely changes the default settings. On the other hand, the pie chart is primarily a presentation graphics (as opposed to analysis graphics) technique. For such persentation graphics, the analyst may find him/herself adjusting the class settings so as to generate a pie chart with pre-determined settings.

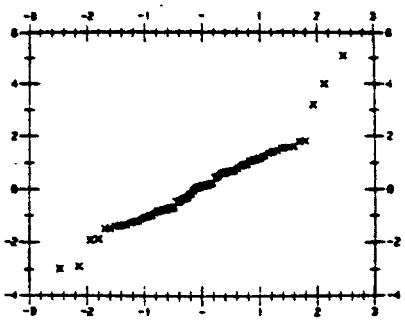
The pie chart is read in a clockwise fashion starting at the 9 o'clock position.

From an interpretational point of view, this pie chart tells us the same as the histogram--namely that the first and last classes have high frequency and the midlle classes have lower frequency.

Generate a Normal Probability Plot

Problem

Data exists on a file as a series of values (1 value per line image). Read in the data. Generate a probability plot for the normal distribution. The probability plot for a given distribution is a graphical data analysis technique for determining if the given distribution provides a good distributional fit to the data. The vertical axis is the ordered raw data. The horizontal axis is the order statistic medians from the given distribution. The linearity of the resulting plot is of interest—the more linear the plot, the better the distributional fit.



DATAPLOT Program

```
READ ABC. X
CHARACTERS X
LINES BLANK
NORMAL PROBABILITY PLOT X
```

Program Description

READ ABC. X
carries out a format-free read of data from file ABC. The first number on each line image is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

CHARACTERS X
specifies (for future plots) that the trace will have X's as plot characters.

LINES BLANK
specifies (for future plots) that the trace will have blanks as plot line types (that is, the trace will have no connecting line between plot points).

NORMAL PROBABILITY PLOT X
generates a normal probability plot of the data in the variable X. Due to the prior entry of the CHARACTERS X command and the LINES BLANK command, the plot will appear with x's as plot characters and with no connecting lines. The default plot type is continuous; the default axis limits will be neat and float with the data.

From a data analysis point of view, this normal probability plot is near-linear for most of the data; the largest 3 values appear to outliers or from a different distributional model.

DATAPLOT has capabilitities for generating probability plots for over 20 distributions and distributional families. Thus to generate a logistic probability plot of the same data set, one could enter

```
LOGISTIC PROBABILITY PLOT X
```

and to generate a probability plot from the Weibull distribution with shape parameter 3.3, one defines (via LET) the value of GAMMA, and then enters the usual PROBABILITY PLOT command, as in

```
LET GAMMA = 3.3
WEIBULL PROBABILITY PLOT X
```

Generate a Probability Plot
Correlation Coefficient Plot

Problem

Data exists on a file as a series of values (1 value per line image). Read in the data. Generate a PPCC (probability plot correlation coefficient) plot for the Tukey lambda distribution family. The PPCC plot for a given distributional family is a graphical data analysis technique for determining which member of the family provides the best distributional fit to the data. The maximum probability plot correlation coefficient criterion is the measure of goodness of fit. The vertical axis is the probability plot correlation coefficient-- a measure of linearity in a probability plot. The horizontal axis is the parameter (in this case, lambda) which defines the various members of the distributional family. The lambda value where the PPCC attains a maximum is of interest. It allows the analyst to graphically estimate the value of the shape parametr (and hence the member of the distributional family) which has the most linear probability plot. See figure 11.

DATAPLOT Program

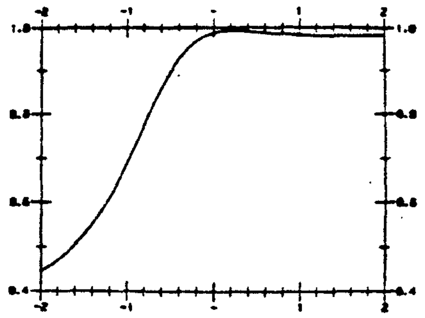
```
READ ABC. X
TUKEY LAMBDA PPCC PLOT X
```

Program Description

READ ABC. X
carries out a format-free read of data from file ABC. The first number on each line image is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

TUKEY LAMBDA PPCC PLOT
generates a Tukey PPCC plot of the variable X. The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

Interpretationlly, the PPCC plot maximizes near lambda = 0.1; Since lambda = 0.14 is approximately the normal distribution, then this plot suggests that of all the members of the Tueky lambda distributional family (a symmetric family), the normal distribution provides a near-best fit to the data. Long-tailed members of the Tukey family (values in which lambda , 0) provided poor fits to the data; Moderate and short-tailed members of the Tukey family provide near-equivalent fits to the data.



The PPCC plot may be generated for other distributions; besides the Tukey lambda family, there is also

t family;
chi-squard family;
gamma family;
extreme value type 2 family;
Weibull family;
Pareto family.

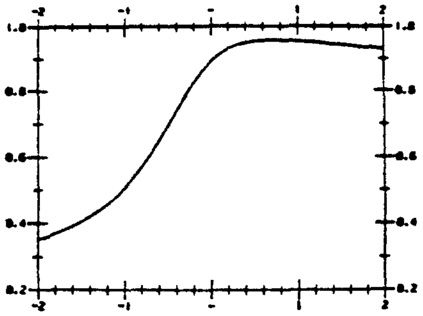
To generate a PPCC plot for the Weibull family, for example, one would enter

```
WEIBULL PPCC PLOT X
```

Generate a Normality Plot

Problem

Data exists on a file as a series of values (1 value per line image). Read in the data. Generate a normality plot for the Box-Cox transformation family. The normality plot for a given transformation family is a graphical data analysis technique for determining which member of the family (that is, which transformation of the raw data) will yield a transformed variable which is most closely normally distributed. The maximum normal probability plot correlation coefficient criterion is the measure of goodness of fit. The vertical axis is the normal probability plot correlation coefficient—a measure of linearity in a normal probability plot. The horizontal axis is the parameter (in this case, lambda) which defines the various members of the transformation family. The lambda value where the normal PPCC attains a maximum is of interest.



DATAPLOT Program

```
READ ABC. X
BOX-COX NORMALITY PLOT X
```

Program Description

<u>READ ABC. X</u>
carries out a format-free read of data from file ABC. The first number on each line image is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

<u>BOX-COX NORMALITY PLOT</u>
generates a Box-Cox normality plot of the variable X. The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

Interpretationally, the Box-Cox Normality Plot maximizes near lambda = 0. Since the Box-Cox transformation family is

```
T(x) = (x**lambda - 1)/lambda   if lambda not = 0

T(x) = log(x)                   if lambda     = 0
```

then the appearance of the plot indicates that a log transformation of the data would yield the closest approximation to normality.

Carry Out a 3-Factor Graphical ANOVA (GANOVA) for a 2-Cubed Full Factorial Design

Problem

Data exists on a file for a response variable Y and independent variables X1, X2, and X3; the 4 corresponding values of the variables are on each line image. Variables X1, X2, and X3 each have 2 levels (-1 and +1). There are 2 x 2 x 2 = 8 data lines. The data file ABC consists of the following 9 line images—

```
99   -1  -1  -1
237  -1   1  -1
149  -1  -1   1
186  -1   1   1
90    1  -1  -1
251   1   1  -1
93    1  -1   1
374   1   1   1
END OF DATA
```

Read in the data.

Carry out a graphical ANOVA (GANOVA) by generating a multi-trace plot of Y versus X1; imbed information regarding the 2 levels of X2 and the 2 levels of X3 by having 2 x 2 = 4 traces within the basic plot of Y versus X1. Emphasize the 2 levels of the X2 variable by having 2 different line types—

solid when X2 = -1
dotted when X2 = 1

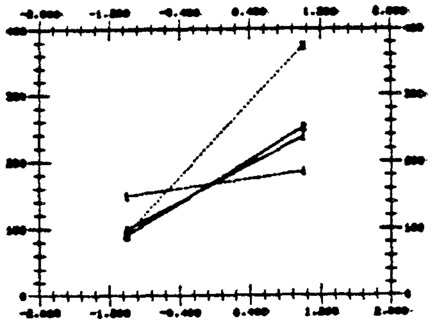
Emphasize the 2 levels of the X3 variable by having 2 different character types—

1 when X3 = -1
2 when X3 = 1

GANOVA is a valuable technique for determining whether an additive model is appropriate, and for determining which variables affect the response. It is a useful complement to the usual ANOVA (and in many respects, is superior to the usual ANOVA).

DATAPLOT Program

```
READ ABC. Y X1 X2 X3
LET ID = 2*(X2-1)+X3
LINES SOLID SOLID DOTTED DOTTED
CHARACTERS 1 2 1 2
PLOT Y X1 ID
```



Program Description

READ ABC. Y X1 X2 X3
carries out a format-free read of data from file ABC; The data is read into variables Y, X1, X2, and X3; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X1; the third number is read into the variable X2; the fourth number is read into the variable X3.

LET ID = 2(X2-1)+X3*
forms a new variable ID. ID has the same number of elements as X2 and X3. An individual element in ID is computed by taking an individual element in X2, subtracting 1 from it, then multiplying by 2, and then adding the corresponding individual element in the variable X3. The rationale for forming ID is that our ultimate objective is to form a plot which has 4 traces in it—where each trace represents a unique X2 and X2 combination. Since there are 3 distinct X2 values (-1 and 1), and since there are 2 distinct X3 values (-1 and 1), then there are 2 x 2 = 4 distinct combinations. Since the 3-argument form for the PLOT command will be used to generate the multi-trace plot, we use this LET command to form the third argument needed in that PLOT statement. The ID variable will have 4 unique values—1 through 4. For distinct values of X2 and X3, the computed distinct value of ID is as follows—

X2	X3	ID
-1	-1	1
-1	1	2
1	-1	3
1	1	4

The above represents the distinct values of X2, X3, and ID; the actual number of elements in ID will be the same as the number of elements in X2 and X3 variables (= the number of elements in the Y and X1 variables) = 8.

<u>LINES SOLID SOLID DOTTED DOTTED</u>
defines (for future plots) the 4 lines types to be associated with the 4 traces which will be generated (when the PLOT command is entered)—

Trace 1—solid
Trace 2—solid
Trace 3—dotted
Trace 4—dotted

<u>CHARACTERS 1 2 1 2</u>
defines (for future plots) the 4 plot characters to be associated with the 4 traces which will be generated (when the PLOT command is entered)—

Trace 1—1
Trace 2—2
Trace 3—1
Trace 4—2

<u>PLOT Y X ID</u>
generates the multi-trace plot; 4 traces are generated. Each trace consists of a plot of the Y variable (vertically) versus the X1 variable (horizontally); however, each individual trace is restricted to only a subset of Y and X1 values. The subset is defined via the contents of the ID variable. For the first trace, the following procedure is executed—

1) the entire ID variable is scanned and the minimum value of ID is identified (in this case, the minimum value of ID is 1);

2) all entries in the Y and X1 variables which correspond to values in the ID variable of 1 are extracted (there are 2 such values);

3) this extracted subset of Y and X1 values is plotted;

4) the line type for this first trace will be dictated by the first entry of a previous LINES command— in this case, the first entry was SOLID, and so a solid line will result; Similarly, the character type for the first trace will be dictated by the first entry of a previous CHARACTERS command— in this case, the first entry was 1, and so 1's will appear at the plot points of the first trace.

For the second trace, the following procedure is executed—

1) the entire ID variable is scanned and the next larger value of ID is identified (in this case, the next larger value of ID is 2);

2) all entries in the Y and X1 variables which correspond to values in the ID variable of 2 are extracted (there are 2 such values);

3) this extracted subset of Y and X1 values is plotted;

4) the line type for this second trace will be dictated by the second entry of a previous LINES command— in this case, the second entry was SOLID, and so a solid line will result; Similarly, the character type for the second trace will be dictated by the second entry of a previous CHARACTERS command— in this case, the first entry was 2, and so 2's will appear at the plot points of the second trace.

A similar procedure for the third and fourth traces will yield a third trace which has a dotted connecting line and 1's at the plot points, and a fourth trace which has a dotted connecting line with 2's at the plot points.

From a data analysis point of view, the GANOVA reveals that factor X1 is significant (for all 4 traces, the response is higher at X1 = +1 than at X1 = -1). Further, an additive model is not appropriate—the criss-crossing of the lines indicates an X2 interaction with X3. Note also the increased variablility within the 2 dotted lines (that is, when X2 = +1) as opposed to within the 2 solid lines (that is, when X2 = -1).

Carry Out a 3-Factor Graphical ANOVA (GANOVA) for a General Full Factorial Design

Problem

Data exists on a file for a response variable Y and independent variables X1 (5 levels), X2 (3 levels), and X3 (2 levels). The 4 corresponding values of the variables are on each line image. There are 5 x 3 x 2 = 30 data lines. The data file ABC consists of the following 31 line images—

```
116   100  1  1   1
97    220  1  1   1
84    475  1  1   1
58    715  1  1   1
70    870  1  1   1
331   100  2  1   2
293   220  2  1   2
272   475  2  1   2
262   715  2  1   2
293   870  2  1   2
570   100  3  1   3
556   220  3  1   3
497   475  3  1   3
611   715  3  1   3
653   870  3  1   3
181   100  1  2   4
133   220  1  2   4
104   475  1  2   4
92    715  1  2   4
106   870  1  2   4
376   100  2  2   5
332   220  2  2   5
324   475  2  2   5
411   715  2  2   5
445   870  2  2   5
768   100  3  2   6
725   220  3  2   6
786   475  3  2   6
892   715  3  2   6
1355  870  3  2   6
END OF DATA
```

(This data is drawn from Hamaker, 1971). Read in the data into variables Y, X1, X2, and X3.

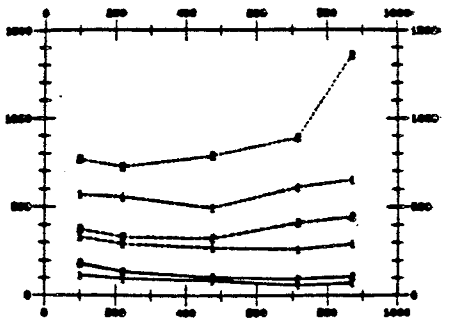
Carry out a graphical ANOVA (GANOVA) by generating a multi-trace plot of Y versus X1; imbed information regarding the 3 levels of X2 and the 2 levels of X3 by having 3 x 2 = 6 traces within the plot of Y versus X1. Emphasize the 3 levels of the X2 variable by having 3 different line types—

solid when X2 = 1
dashed when X2 = 2
dotted when X3 = 3

Emphasize the 2 levels of the X3 variable by having 2 different character types—

1 when X3 = 1
2 when X3 = 2

Graphical ANOVA (GANOVA) is a valualble tool for determining if and how variables affect the response.



DATAPLOT Program

```
READ ABC. Y X1 X2 X3
LET ID = 2*(X2-1)+X3
LINES SOLID SOLID DASHED DASHED DOTTED DOTTEL
CHARACTERS 1 2 1 2 1 2
PLOT Y X1 ID
```

Program Description

__READ ABC. Y X1 X2 X3__
carries out a format-free read of data from file ABC; The data is read into variables Y, X1, X2, and X3; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X1; the third number is read into the variable X2; the fourth number is read into the variable X3.

__LET ID = 2(X2-1)+X3__*
forms a new variable ID. ID has the same number of elements as X2 and X3. An individual element in ID is computed by taking an individual element in X2, subtracting 1 from it, then multiplying by 2, and then adding the corresponding individual element in the variable X3. The rationale for forming ID is that our ultimate objective is to form a plot which has 6 traces in it—where each trace represents a unique X2 and X2 combination. Since there are 3 distinct X2 values (1, 2, and 3), and since there are 2 distinct X3 values (1 and 2), then there are 3 x 2 = 6 distinct combinations. Since the 3-argument form for the PLOT command will be used to generate the multi-trace plot, we use this LET command to form the third argument needed in that PLOT statement. The ID variable will have 6 unique values—1 through 6. For distinct values of X2 and X3, the computed distinct value of ID is as follows—

X2	X3	ID
1	1	1
1	2	2
2	1	3
2	2	4
3	1	5
3	2	6

The above represents the distinct values of X2, X3, and ID; the actual number of elements in ID will be the same as the number of elements in X2 and X3 variables (= the number of elements in the Y and X1 variables) = 30.

LINES SOLID SOLID SOLID SOLID DOTTED DOTTED DOTTED defines (for future plots) the 6 lines types to be associated with the 6 traces which will be generated (when the PLOT command is entered)--

Trace 1--solid
Trace 2--solid
Trace 3--dashed
Trace 4--dashed
Trace 5--dotted
Trace 6--dotted

CHARACTERS 1 2 1 2 1 2 defines (for future plots) the 6 plot characters to be associated with the 6 traces which will be generated (when the PLOT command is entered)--

Trace 1--1
Trace 2--2
Trace 3--1
Trace 4--2
Trace 5--1
Trace 6--2

PLOT Y X ID generates the multi-trace plot; 4 traces are generated. Each trace consists of a plot of the Y variable (vertically) versus the X1 variable (horizontally); however, each individual trace is restricted to only a subset of Y and X1 values. The subset is defined via the contents of the ID variable. For the first trace, the following procedure is executed--

1) the entire ID variable is scanned and the minimum value of ID is identified (in this case, the minimum value of ID is 1);

2) all entries in the Y and X1 variables which correspond to values in the ID variable of 1 are extracted (there are 5 such values);

3) this extracted subset of Y and X1 values is plotted;

4) the line type for this first trace will be dictated by the first entry of a previous LINES command-- in this case, the first entry was SOLID, and so a solid line will result; Similarly, the character type for the first trace will be dictated by the first entry of a previous CHARACTERS command-- in this case, the first entry was 1, and so 1's will appear at the plot points of the first trace.

For the second trace, the following procedure is executed--

1) the entire ID variable is scanned and the next larger value of ID is identified (in this case, the next larger value of ID is 2);

2) all entries in the Y and X1 variables which correspond to values in the ID variable of 2 are extracted (there are 5 such values);

3) this extracted subset of Y and X1 values is plotted;

4) the line type for this second trace will be dictated by the second entry of a previous LINES command-- in this case, the second entry was SOLID, and so a solid line will result; Similarly, the character type for the second trace will be dictated by the second entry of a previous CHARACTERS command-- in this case, the first entry was 2, and so 2's will appear at the plot points of the second trace.

A similar procedure for traces 3 through 6 yielded

trace 3--dashed with plot character 1
trace 4--dashed with plot character 2
trace 5--dotted with plot character 1
trace 6--dotted with plot character 2

From a data analysis point of view, the GANOVA reveals many important facts--

1) there exists an outlier (note the point in the upper right corner);

2) factor X1 is not significant (note the flatness of the response for all 6 traces);

3) factor X2 is significant note how dotted higher than dashed higher than solid);

4) factor X3 is significant (note how 2 higher than 1 for solid, for dashed, for dotted);

5) variability increases for different levels of factor X2 (note narrow spread between 2 dotted lines, wider spread between 2 dashed lines, and wide spread between 2 dotted lines).

Graphical ANOVA is one of the most important tools for the data analyst.

Carry Out a Nested Graphical ANOVA (Nested CANOVA) of a 3-Factor Experiment

Problem

Data exists on a file for a response variable Y and independent variables X1 (6 levels), X2 (2 levels), and X3 (3 levels); data is missing at the following point— (X1 = 5, X2 = 1, X3 = 2); and at the following 3 points (X1 = 5, X2 = 2, X3 = 1, 2, and 3). Each line image has the corresponding values of each variable. There are (6 x 2 x 3) - 4 = 32 data lines. The data file ABC consists of the following 33 line images—

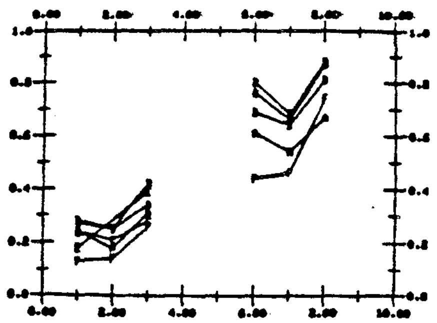
```
.24   1   1   1
.18   1   1   2
.31   1   1   3
.61   1   2   1
.54   1   2   2
.67   1   2   3
.28   2   1   1
.25   2   1   2
.42   2   1   3
.69   2   2   1
.64   2   2   2
.81   2   2   3
.24   3   1   1
.21   3   1   2
.28   3   1   3
.76   3   2   1
.66   3   2   2
.87   3   2   3
.27   4   1   1
.25   4   1   2
.34   4   1   3
.80   4   2   1
.68   4   2   2
.88   4   2   3
.18   5   1   1
.39   5   1   3
.13   6   1   1
.14   6   1   2
.26   6   1   3
.44   6   2   1
.46   6   2   2
.74   6   2   3
END OF DATA
```

Read in the data; place them in variables Y, X1, X2, and X3.

Carry out a nested graphical ANOVA (nested CANOVA). Emphasize the 2 levels of X2 by splitting the horizontal axis into 2 halves—

left half for X1 = 1
right half for X1 = 2

Emphasize the 3 levels of X3 by having X3 on the horizontal axis, but nested within each X2 value. Thus (when X2 = 1) there will be 3 levels of X3 on the left half of the horizontal axis; and (when X2 = 2) there will be 3 levels of X3 on the right half of the horizontal axis.



Emphasize the 6 levels of X1 by connecting values within a level by a solid line and by incorporating 6 distinct character types—

A when X1 = 1
B when X1 = 2
C when X1 = 3
D when X1 = 4
F when X1 = 5
G when X1 = 6

The net effect will be that all levels of X1 and X3 will be visually nested within each of the 2 levels of X2. Note that the 6 levels of X1 and the 2 levels of X2 would result in 6 x 2 = 12 traces; however, since data is missing for (X1 = 5, X2 = 2), only 11 traces will result. See figure 15.

DATAPLOT Program

```
READ ABC. Y X1 X2 X3
LET X23 = 5*(X2-1)+X3
LET TAG = 2*(X2-1)+X1
CHARACTERS A B C D E F A B C D F
PLOT Y X23 TAG
```

Program Description

__READ ABC. Y X1 X2 X3__
carries out a format-free read of data from file ABC; The data is read into variables Y, X1, X2, and X3; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X1; the third number is read into the variable X2; the fourth number is read into the variable X3.

LET X23 = 5(X2-1)+X3*
forms a new variable X23. The rationale for forming X23 is that we wish to form a new horizontal axis variable for the plot which will incorporate both the 2 values for X2 and the 3 values for X3 The final plot will involve Y (vertically) versus this new combined variable X23 (horizontally). The X23 will be the second argument (the horizontal axis argument) in the 3-argument form for the PLOT command. An individual element in X23 is computed by taking an individual element in X2, subtracting 1 from it, then multiplying by 5, and then adding the corresponding individual element in the variable X3. Since there are 2 distinct X2 values (1 and 2), and since there are 3 distinct X3 values (1, 2, and 3), then there are 2 x 3 = 6 distinct combinations. The X23 variable will have 6 distinct values--1 to 3 and 6 to 8. For distinct values of X2 and X3, the computed value of X23 is as follows--

X2	X3	X23
1	1	1
1	2	2
1	3	3
2	1	6
2	2	7
2	3	8

A factor of 5 was chosen in the definition of X23-- X23 = 5(X2-1)+X3. This was deliberate--it caused the X23 value to take on values 6, 7, and 8 when X2 = 2, with the net effect that there will be a visual separation on the final plot between the left half (X1 = 1) and the right half (X1 = 2) of the horizontal axis. Other factors besides 5 could have been used. X23 has 6 distinct values, but has many more actual values. The actual number of elements in X23 will be the same as the number of elements in X2 and X3 variables (= the number of elements in the Y and X1 variables) = 32.*

LET TAG = 6(X2-1)+X1*
forms a new variable TAG. The rationale for forming TAG is that our ultimate objective is to form a plot which has 6 traces in it--where each trace represents a unique X2 and X2 combination. Since there are 3 distinct X2 values (1, 2, and 3), and since there are 2 distinct X1 values (1 and 2), then there are 3 x 2 = 6 distinct combinations. Since the 3-argument form for the PLOT command will be used to generate the multi-trace plot, we use this LET command to form the third argument needed in that PLOT statement. An individual element in TAG is computed by taking an individual element in X2, subtracting 1 from it, then multiplying by 6, and then adding the corresponding individual element in the variable X1. For distinct values of X2 and X1, the computed distinct value of TAG is as follows--

X2	X1	TAG
1	1	1
1	2	2
1	3	3
1	4	4
1	5	5
1	6	6
2	1	7
2	2	8
2	3	9
2	4	10
2	5	11
2	6	12

The TAG variable would have 12 unique values--1 through 12; note, however, that since the data from (X2 = 2, X1 = 5) is missing, then it will in fact only have 11 values--1 through 10, and 12. The above represents the distinct values of X2, X1, and TAG; the actual number of elements in TAG will be the same as the number of elements in X2 and X1 variables (= the number of elements in the Y and X1 variables) = 32.

CHARACTER A B C D E F A B C D F
defines (for future plots) the 11 plot characters to be associated with the 11 traces which will be generated (when the PLOT command is entered)--

Trace 1	*--A*
Trace 2	*--B*
Trace 3	*--C*
Trace 4	*--D*
Trace 5	*--E*
Trace 6	*--F*
Trace 7	*--A*
Trace 8	*--B*
Trace 9	*--C*
Trace 10	*--D*
Trace 11	*--F*

Note that the character for the eleventh trace was set to F (rather than E) because of the realization that data was missing for (X2 = 2, X1 = 5).

PLOT Y X23 TAG
generates the multi-trace plot; 11 traces are generated. Each trace consists of a plot of the Y variable (vertically) versus the X23 variable (horizontally); however, each individual trace is restricted to only a subset of Y and X23 values. The subset is defined via the contents of the TAG variable. For the first trace, the following procedure is executed--

1) the entire TAG variable is scanned and the minimum value of TAG is identified (in this case, the minimum value of TAG is 1);

2) all entries in the Y and X23 variables which correspond to values in the TAG variable of 1 are extracted (there are 3 such values);

3) this extracted subset of Y and X23 values is plotted;

4) the line type for this first trace will be dictated by the first entry of a previous LINES command-- in this case, since the LINES command was not previously entered, then the default (= solid line) will be used, thus the 3 plot points will be connected by solid lines. Similarly, the character type for the first trace will be dictated by the first entry of a previous CHARACTERS command-- in this case, the first entry was A, and so A's will appear at the 3 plot points of the first trace.

A similar procedure was used for traces 2 to 11. The traces have the following characteristics--

Trace	TAG	Character	Line
1	1	A	SOLID
2	2	B	SOLID
3	3	C	SOLID
4	4	D	SOLID
5	5	E	SOLID
6	6	F	SOLID
7	7	A	SOLID
8	8	B	SOLID
9	9	C	SOLID
10	10	D	SOLID
11	12	F	SOLID

From a data analysis point of view, the nested GANOVA reveals many important facts--

1) factor X2 is significant
(note how the left half of the plot is lower than the right half of the plot);

2) factor X3 is significant
(note how level 3 of X3 is uniformly higher than level 1 of X3);

3) factor X1 is significant
(note how level xx of X1 is lower than the other levels of X1).

Generate a Discrete Contour Plot to Assess 2-Dimensional Homogeneity

Problem

A physical specimen has a measured response at equi- spaced points on its surface. The response Z and the coordinate values X and Y are on a file ABC. Read in the data; place them in variables X, Y, and Z. Carry out a check to see if the response is homogeneous (2-dimension randomness) over the surface. Do so by computing the median response, and then generating a discrete contour plot in which all values smaller than the median have plot character -, and all values greater have +. If homogeneous, there should be a random mix of -'s and +'s; a structured, non-random mix of -'s and +'s is indicative of lack of homogeneity.

DATAPLOT Program

```
READ ABC. Z X Y
LET MED = MEDIAN Z
CHARACTERS - +
LINES BLANK ALL
PLOT Y X SUBSET Z , MED AND
PLOT Y X SUBSET Z .= MED
```

Program Description

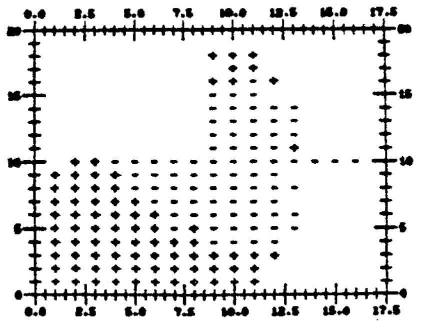
READ ABC. Z X Y
carries out a format-free read of data from file ABC; The data is read into variables Z, X, and Y. the first number on each line image of the file is read into the variable Z; the second number on each line is read into the variable X; the third number is read into the variable Y.

LET MED = MEDIAN Z
computes the median of the values in the Z variable, and places this median value in the parameter MED. This form of the LET command illustrates the use of available sub-commands for LET; the sub-command used here is MEDIAN, other such sub-commands are MEAN, STANDARD DEVIATION, RANGE, AUTOCORRELATION, etc. There are over 20 such sub-commands which are useful for computing statistics and other numeric summaries of the data in a variable.

CHARACTERS - +
specifies (for future plots) that

trace 1 have -'s at the plot points
trace 2 have +'s at the plot points

(the default is to have blanks at all plot points of all traces).



LINES BLANK ALL
specifies (for future plots) that all traces have blank (= no) connecting lines between plot points. (the default is to have solid connecting lines between plot points). The rationale for setting the plot lines to blank is that we would like to see patterns of individual -'s and +'s at the data points, and the lines (in this case) would only "get in the way".

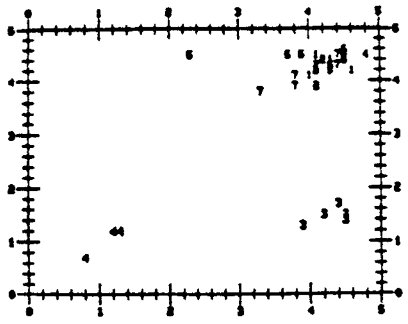
PLOT Y X SUBSET Z , MED AND
PLOT Y X SUBSET Z .= MED
generates a plot with 2 "traces". The first trace is a plot of variable Y (vertically) versus variable X (horizontally); however, only a subset of all (X,Y) combinations will be plotted, namely, only those X and Y values for which the corresponding Z value is less than the median value of Z. Due to the prior CHARACTERS and LINES commands, the first trace will have -'s as plot characters with blank (= no) connecting lines. The second trace is a plot of variable Y (vertically) versus variable X (horizontally); however, only a subset of all (X,Y) combinations will be plotted, namely, only those X and Y values for which the corresponding Z value is greater than or equal to the median value of Z. Due to the prior CHARACTERS and LINES commands, the second trace will have +'s as plot characters with blank (= no) connecting lines. The net result is a discrete contour plot in which the -'s indicate surface points for which the response is less than the median, and +'s indicate surface points for which the response is not less than the median. The AND suffix at the end of PLOT Y X SUBSET Z , MED is important-- it tells DATAPLOT to generate only 1 plot, but on that 1 plot to have 2 traces; rather than having 2 successive plots, each with only a single trace. The discrete contour plot is thus seen as a simple special case of multi-trace plotting in DATAPLOT.

From a data analysis point of view, one would conclude that the homogeneity hypothesis would be rejected--note the regions of segmentation for both the -'s and the +'s.

Generate a Youden Plot for Interlaboratory Analysis

Problem

Two specimens of a material are circulated to 7 laboratories. A given laboratory makes 5 measurements Y1 for specimen 1, and 5 measurements Y2 for specimen 2. The data exists on a file ABC. A given line image has 4 values—response Y1 for specimen 1, response Y2 for specimen 2, laboratory identification LAB, and replication identification REP. There are 7 x 5 = 35 line images. Read the data into variables Y1, Y2, LAB and REP. Carry out an interlaboratory analysis via a Youden plot. The vertical axis is Y1; the horizontal axis is Y2. The plot character is laboratory.



DATAPLOT Program

```
READ ABC. Y1 Y2 LAB REP
CHARACTERS 1 2 3 4 5 6 7
LINES BLANK ALL
PLOT Y1 Y2 LAB
```

Program Description

READ ABC. Y1 Y2 LAB REP
carries out a format-free read of data from file ABC; The data is read into variables Y1, Y2, LAB, and REP. the first number on each line image of the file is read into the variable Y1; the second number on each line is read into the variable Y2; the third number is read into the variable LAB; the fourth number is read into the variable REP.

CHARACTERS 1 2 3 4 5 6 7
specifies (for future plots) that

trace 1 have 1's at the plot points;
trace 2 have 2's at the plot points;
trace 3 have 3's at the plot points;
trace 4 have 4's at the plot points;
trace 5 have 5's at the plot points;
trace 6 have 6's at the plot points;
trace 7 have 7's at the plot points;

(the default is to have blanks at all plot points of all traces).

LINES BLANK ALL
specifies (for future plots) that all traces have blank (= no) connecting lines between plot points. (The default is to have solid connecting lines between plot points). The rationale for setting the plot lines to blank is that a Youden plot is a discrete-appearing plot in which no use is made of connect lines between plot points—they are omitted because they only "get in the way". The interpretation of the Youden plot is based on the relative location of the laboratories— and such laboratory identification is specified via plot characters—not line types.

PLOT Y1 Y2 LAB
generates a multi-trace plot; due to the character and line settings, this multi-trace will be a Youden plot. 7 traces are generated. Each trace consists of a plot of the Y1 variable (vertically) versus the Y2 variable (horizontally); however, each individual trace is restricted to only a subset of Y1 and Y2 values. The subset is defined via the contents of the LAB variable. For the first trace, the following procedure is executed—

1) the entire LAB variable is scanned and the minimum value of LAB is identified (in this case, the minimum value of LAB is 1);

2) all entries in the Y1 and Y2 variables which correspond to values in the LAB variable of 1 are extracted (there are 5 such values);

3) this extracted subset of Y1 and Y2 values is plotted;

4) the character type for the first trace will be dictated by the first entry of a previous CHARACTERS command— in this case, the first entry was 1, and so 1's will appear at the 5 plot points of the first trace. Similarly, the line type for this first trace will be dictated by the first entry of a previous LINES command— in this case, the first (and all) entry was BLANK, and thus the 5 plot points will not be connected by any lines.

A similar procedure was used for traces 2 to 7. The traces have the following characteristics--

Trace	*LAB*	*Character*	*Line*
1	*1*	*1*	*BLANK*
2	*2*	*2*	*BLANK*
3	*3*	*3*	*BLANK*
4	*4*	*4*	*BLANK*
5	*5*	*5*	*BLANK*
6	*6*	*6*	*BLANK*
7	*7*	*7*	*BLANK*

From a data analysis point of view, the Youden plot indicates

1) a between-lab effect
(note how laboratory 4 is displaced down the diagonal away from the body of other laboratories);

2) a within-lab effect
(note how laborary 3 is displaced off the diagonal);

3) an outlier
(note how laboratory 5 has one value displaced away from other replicates of laboratory 5).

Note that the above analysis never made direct or indirect used of the REP variable. This being the case, it could have been omitted from the READ statement with no effect on the remainder of the analysis.

Generate a Box Plot

Problem

A response variable Y is dependent on a single independent variable X. Data exists on a file ABC consisting of a value of Y and the corresponding value of X on the same line image. Replication exists in the data--that is, a given value of X will have several values of Y (for example, X = 580 could have 14 replicates associated with it, X = 630 could have 10 replicates; X = 710 could have 12 replicates, etc.) Each replicate occupies a different line in the data file; thus, for example, the 14 replicates at X = 580 would be indicated by 14 separate data lines where each line had 2 values, and where the X value on each line was 580.

Read in the data into variables Y and X.

The ultimate objective is to determine if the reponse variable Y differs for different values of X. The usual statistical technique for determining this is 1-way analysis of variance. In an analysis of variance context, the various distinct levels (580, 630, 710, etc.) of X would be termed "treatments". Generate a box plot-- the box plot is a graphical data analysis technique for assessing whether significant differences exist between treatments.

DATAPLOT Program

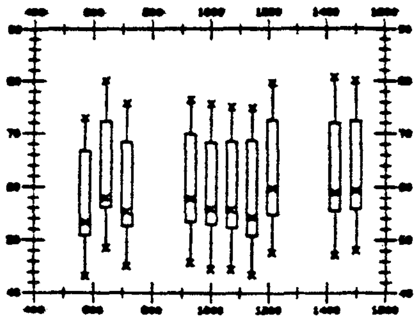
```
READ ABC. Y X
CHARACTERS BOX PLOT
LINES BOX PLOT
BOX PLOT Y X
```

Program Description

READ ABC. Y X
carries out a format-free read of data from file ABC; the data is read into variables Y and X. The first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X.

CHARACTERS BOX PLOT
specifies (for future plots) the character settings for the typical box plot. The typical box plot is made up of 20 components; rather than have the analyst specify the character settings for these 20 components individually, the CHARACTERS BOX PLOT specifies these first 20 character settings automatically.

LINES BOX PLOT
specifies (for future plots) the line settings for the typical box plot. The typical box plot is made up of 20 components; rather than have the analyst specify the line-type settings for these 20 components individually, the LINES BOX PLOT specifies these first 20 line type settings automatically.



BOX PLOT Y X
generates a box plot. The box plot is basically a plot of Y (vertically) versus X (horizontally); however, graphical summarization (in the form of a box) is used in place of the Y values. Consider an individual X value (that is, an individaul treatement)--it will have several replicate Y values associated with it. Rather than plot all of these individual Y values, the box plot graphically summarizes these values, and plots the summary form. In particular, the box plot reduces all replicated data points (for a given X value) into 5 points--

1) *a minimum value;*
2) *a 25% point;*
3) *a 50% point (that is, the median);*
4) *a 75% point;*
5) *a maximum value.*

The minimum, median, and maximum values are all indicated on the box plot by x's. A line will be drawn from the 25% point down to the minimum; a second line will be drawn from the 75% point up to the maximum. A box will be drawn from the 25 point to the 75% point--this box represents the "body" of the distribution. This procedure is repeated for each distinct X value. The proper character and line settings are automatically taken care of by previous entry of the CHARACTERS BOX PLOT and LINES BOX PLOT commands.

From a data analysis point of view, the box plot (for this data set) would indicate that the X variable is not significant. If X were significant, then the boxes for the various levels of X would be considerably displaced from one another (that is, they would not overlap). When there is considerable overlap (as in this example), it is indicative of non-significance.

Generate an I Plot

Problem

A response variable Y is dependent on a single independent variable X. Data exists on a file ABC consisting of a value of Y and the corresponding value of X on the same line image. Replication may exist in the data—that is, a given value of X may have several values of Y; each replicate occupies a different line in the data file.

Read the data in; place the data in variables Y and X.

Generate an I plot. The vertical bar on the I plot will extend from the min- imum response to the maximum response within a treatment. The X within the bar will be at the median response within a treatment. The I plot is a graphical data analysis technique which (like the box plot) may be considered as a graphical alternative to 1-way analysis of variance)—it assesses whether the response is significantly affected by different treatment (that is, by different levels of X).

DATAPLOT Program

```
READ ABC. Y X
CHARACTERS I PLOT
LINES I PLOT
I PLOT Y X
```

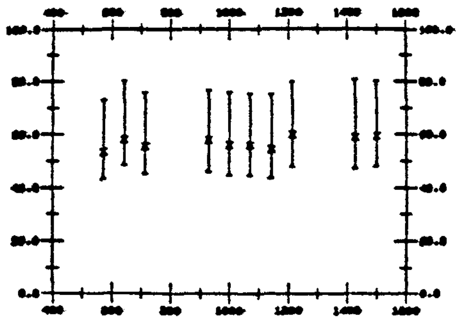
Program Description

<u>READ ABC. Y X</u>
carries out a format-free read of data from file ABC; the data is read into variables Y and X. The first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X.

<u>CHARACTERS I PLOT</u>
specifies (for future plots) the character settings for the typical I plot. The typical I plot is made up of 3 character components—

the character for the minimum;
the character for the median;
the character for the maximum.

Rather than have the analyst specify the character settings for these components individually, the CHARACTERS I PLOT specifies these character settings automatically.



<u>LINES I PLOT</u>
specifies (for future plots) the line settings for the typical I plot. The typical I plot is made up of 2 line components—

the line type from the minimum to the median;
the line type from the median to the maximum

Rather than have the analyst specify the line-type settings for these components individually, the LINES I PLOT specifies these line type settings automatically.

<u>I PLOT Y X</u>
generates a I plot. The I plot is basically a plot of Y (vertically) versus X (horizontally); however, graphical summarization (in the form of a vertical line) is used in place of the Y values. Consider an individual X value (that is, an individaul treatement)—it will have several replicate Y values associated with it. Rather than plot all of these individual Y values, the I plot graphically summarizes these values, and plots the summary form. In particular, the I plot reduces all replicated data points (for a given X value) into 3 points—

1) a minimum value;
2) a median value;
3) a maximum value.

The minimum and maximum values are indicated on the I plot by a small horizontal bar. The median value is indicated by an x. A line extends from the minimum value to the maximum value. The I-like appearance of the figure is the basis for the term "I plot". This procedure is repeated for each distinct X value. The proper character and line settings are automatically taken care of by previous entry of the CHARACTERS I PLOT and LINES I PLOT commands.

From a data analysis point of view, the I plot (for this data set) would indicate that the X variable is not significant. If X were significant, then the vertical bars for the various levels of X would be considerably displaced from one another (that is, they would not overlap). When there is considerable overlap (as in this example), it is suggestive of non-significance. A more stringent graphical procedure that the analyst should consider applying is the box plot.

The above example illustrated the use of the I plot in a 1-way ANOVA context. The most common usage of the I plot, is not, however in this graphical analogue to ANOVA. Rather, the most common application of the I plot is in a presentation graphics context for displaying estimated values of a set of physical parameters along with computed uncertainty limits. In this latter context, the data set is artificially constructed so that each treatment only has 3 replicates--the lower uncertainty limit, the estimated physical parameter value, and the upper uncertainty limit. Since there are only 3 values per "treatment", it is precisely these 3 values which will appear on the final plot.

Generate a Control Chart

Problem

Data exists on a file ABC as a series of line images with 2 values per line image. The first value is a response variable value; the second value is the set (e.g., time) identifier for a given response. To use control charts, the various sets must have more than 1 observation per set, and for strict validity should have the same number of observations for all sets. Each replicate occupies a different line in the data file.

Read in the data; place the data in variables Y and X.

Generate a mean control chart. The control chart (in general) is a graphical data analysis technique for assessing whether significant shifts in location or dispersion have occurred in a measurement process. The plotted points are set means; the center line in a mean control chart is the grand mean; the upper and lower lines are 95% limits for the set means.

DATAPLOT Program

```
READ ABC. Y X
CHARACTERS CONTROL CHART
LINES CONTROL CHART
MEAN CONTROL CHART Y X
```

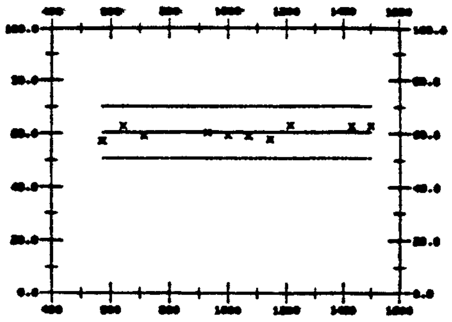
Program Description

READ ABC. Y X
carries out a format-free read of data from file ABC; the data is read into variables Y and X. The first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X.

CHARACTERS CONTROL CHART
specifies (for future plots) the character settings for the typical control chart. The typical control chart is made up of 4 character components—

the character for the set means;
the character for the grand mean line;
the character for the lower control limit line;
the character for the upper control limit line;

Rather than have the analyst specify the character settings for these components individually, the CHARACTERS CONTROL CHART specifies these character settings automatically— the raw data will have x's as characters, and the 3 control lines will have blanks as characters.



LINES CONTROL CHART
specifies (for future plots) the line settings for the typical control chart. The typical control chart is made up of 4 line components—

the line type for the set means;
the line type for the grand mean line;
the line type for the lower control limit line;
the line type for the upper control limit line.

Rather than have the analyst specify the line-type settings for these components individually, the LINES CONTROL CHART specifies these line type settings automatically— the line type for the raw data is blank; the line type for the grand mean line is solid; the line type for the 2 limit lines is dotted.

MEAN CONTROL CHART Y X
generates a mean control chart. The mean control chart is basically a plot of Y (vertically) versus X (horizontally); however, summarization and control lines are used in place of the Y values. Consider an individual X value (that is, an individual set)—it will have several replicate Y values associated with it. Rather than plot all of these individual Y values, the mean control chart graphically summarizes these values, and plots the summary form. In particular, the control chart reduces all replicated data points (for a given X value) into 1 point—

1) the mean for the set.

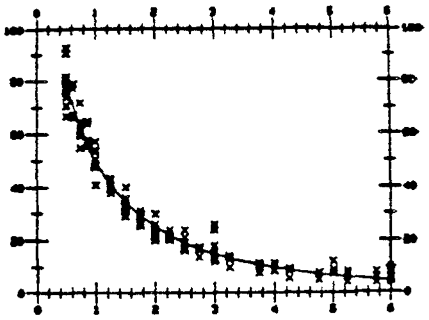
This mean value is then plotted in place of the individual data points for the set. In addition, the mean control chart augments this raw plot of mean values by 3 control lines which are based on global properties of the full data set; in particular,

Generate a Superimposed Plot of Raw Data and Fitted Values from a Non-Linear Fit

Problem

Data exists on a file ABC as a series of line images with 2 values per line image (response variable Y and indepen- dent variable X). Read in the data; place them in variables Y and X.

Theory suggests a non-linear relationship between Y and X. Carry out a least squares fit. Generate a plot of the raw data Y (as discrete x's) and the superimposed fitted curve (as a solid line) versus X. Such a superimposed plot is the first step in assessing goodness of fit.



DATAPLOT Program

```
READ ABC. Y X
.
LET ALPHA = .1
LET A = .001
LET B = .01
FIT Y = EXP(-ALPHA*X)/(A+B*X)
.
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
```

Program Description

READ ABC. Y X
carries out a format-free read of data from file ABC; The data is read into variables Y and X; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

.
is a null command—it visually separates chunks of DATAPLOT code.

LET ALPHA = .1
defines the parameter ALPHA and assigns to it the value .1. ALPHA is here being defined in preparation for the upcoming non-linear fit.

LET A = .001
defines the parameter A and assigns to it the value .001. A is here being defined in preparation for the upcoming non-linear fit.

LET B = .01
defines the parameter B and assigns to it the value .01. B is here being defined in preparation for the upcoming non-linear fit.

*FIT Y = EXP(-ALPHA*X)/(A+B*X)*
*carries out a least squares fit of Y as a function of X based on the non-linear model Y = EXP(-ALPHA*X)/(A+B*X). The current values of ALPHA, A, and B are used as starting values for the non-linear fit. At the end of the fit, the best-fit values placed into the parameters ALPHA, A, and B. Predicted values from the fit are computed and placed in the variable PRED. Residual values (= Y - PRED) are computed and placed in the variable RES. The residual standard deviation is placed in the parameter RESSD; the residual degrees of freedom is placed in RESDF. Replication is automatically checked for; if existent, the replication standard deviation is computed and placed in REPSD; the replication degrees of freedom is placed in REPDF; the lack of fit F test cdf value is placed in the parameter LOFCDF. All of the above (PRED, RES, RESSD, RESDF, REPSD, REPDF, LOFCDF) are automatically computed by DATAPLOT whenever any DATAPLOT modeling command (FIT, PRE-FIT, EXACT ... FIT, SPLINE FIT, SMOOTH, ANOVA, MEDIAN POLISH) is executed. These variables/parameters may be subsequently used by the analyst in any way whatsoever. They are usually used for further model-verification and residual analysis.*

CHARACTERS X BLANK
specifies (for future plots) that the first trace will have X's as plot characters, and the second trace will have blanks as plot characters (that is, the second trace will have no characters at the plot points).

LINES BLANK SOLID
specifies (for future plots) that the first trace will have blanks as plot line types (that is, the first trace will have no connecting line between plot points), and the second trace will have solid connecting lines between the plot points.

1) a grand mean is computed based on data from all sets;

2) a lower control limit for an individual set mean is computed;

3) a upper control limit for an individual set mean is computed.

The set means appear as x's; the grand mean line appears as a horizontal solid line; the 2 limit lines appear as horizontal dotted lines. The proper character and line settings are automatically taken care of by previous entry of the CHARACTERS CONTROL CHART and LINES CONTROL CHART commands.

From a data analysis point of view, the control chart (for this data set) would indicate the process is "in control" as far as location displacement is concerned. If the process were "out of control", then the set mean values would typically drift out of the control limit lines. To check for displacement and drifting in regard to within-set variation, the analyst would use a standard deviation control chart or a range control chart.

PLOT Y PRED VERSUS X
generates a plot with 2 traces--

1) Y versus X;
2) PRED versus X.

that is,

1) a trace with the raw data Y (vertically) versus the variable X (horizontally);

2) a trace with the predicted values PRED (vertically) versus the variable X (horizontally);

Due to the prior CHARACTERS and LINES commands, the 2 traces will have the following characteristics--

Trace	Contents	Characters	Lines
1	Y vs X	X	BLANK
2	PRED vs X	BLANK	SOLID

that is, the first trace will have X's as plot characters with blank (= no) connecting lines, and the second trace will have blanks as plot characters with solid connecting lines.

The default plot type is continuous; the default axis will be neat and float with the data.

There are numerous alternate forms for generating the same PLOT Y PRED VERSUS X plot; the most obvious option is that VERSUS may be universally replaced by VS or VS., as in

```
PLOT Y PRED VS X

PLOT Y PRED VS. X
```

Some other forms are

```
PLOT Y VS X PRED VS X

PLOT Y VERSUS X AND
PLOT PRED VERSUS X

PLOT Y X AND
PLOT PRED X
```

The FIT command has alternate forms also; rather than

```
FIT Y = EXP(-ALPHA*X)/(A+B*X)
```

the analyst could have entered

```
LET FUNCTION F = EXP(-ALPHA*X)/(A+B*X)
FIT Y = F
```

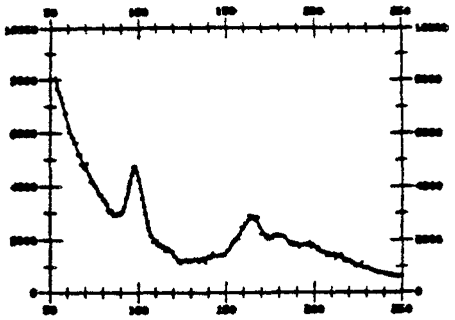
Starting values are a help in terms of speeding convergence for non-linear fits; in the absence of starting values, all fits will start with starting values of unity for all parametrzers. Starting values are not needed for linear, polynomial, and multi-linear fits.

Generate a Superimposed Plot of Raw Data and Fitted Values from a Spline Fit

Problem

Data exists on a file ABC as a series of line images with 2 values per line image (response variable and independent variable). The knots for such a fit are on a second file DEFG (1 knot point per line image). Read in the data from file ABC; place the data in variables Y and X. Read in the knots from file DEFG; place the knots in variable KNOTS.

Carry out a least squares spline fit using the specified knots. Generate a plot of the raw data Y (as discrete x's) and the superimposed fitted curve (as a solid line) versus X.



DATAPLOT Program

```
READ ABC. Y X
READ DEFG. KNOTS
.
CUBIC SPLINE FIT Y X KNOTS
.
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
```

Program Description

READ ABC. Y X
carries out a format-free read of data from file ABC; The data is read into variables Y and X; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

READ DEFG. KNOTS
carries out a format-free read of data from file DEFG; the first number on each line image of the file is read into variable KNOTS. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

.
is a null command—it visually separates chunks of DATAPLOT code.

SPLINE FIT Y X KNOTS
carries out a least squares cubic spline fit of Y as a function of X with knot points as specified by the variable KNOTS. At the end of the fit, the best-fit values for the 4 cubic coefficients for each of the spline intervals are placed in the following parameters—

Interval	*Parameters*
1	A10, A11, A12, A13
2	A20, A21, A22, A23
3	A30, A31, A32, A33
4	A40, A41, A42, A43
5	A50, A51, A52, A53
etc.	

Predicted values from the fit are computed and placed in the variable PRED. Residual values (= Y - PRED) are computed and placed in the variable RES. The residual standard deviation is placed in the parameter RESSD; the residual degrees of freedom is placed in RESDF. Replication is automatically checked for; if existent, the replication standard deviation is computed and placed in REPSD; the replication degrees of freedom is placed in REPDF; the lack of fit F test cdf value is placed in the parameter LOFCDF. All of the above (PRED, RES, RESSD, RESDF, REPSD, REPDF, LOFCDF) are automatically computed by DATAPLOT whenever any DATAPLOT modeling command (FIT, PRE-FIT, EXACT ... FIT, SPLINE FIT, SMOOTH, ANOVA, MEDIAN POLISH) is executed. These variables/parameters may be subsequently used by the analyst in any way whatsoever. They are usually used for further model-verification and residual analysis. The default spline fit degree is cubic. If a non-cubic spline fit is desired, then specify if directly,as in

```
QUADRATIC SPLINE FIT Y X KNOTS
```

or indirectly via the POLYNOMIAL DEGREE command, as in

```
POLYNOMIAL DEGREE 2
SPLINE FIT Y X KNOTS
```

.
is a null command—it visually separates chunks of DATAPLOT code.

CHARACTERS X BLANK
specifies (for future plots) that the first trace will have X's as plot characters, and the second trace will have blanks as plot characters (that is, the second trace will have no characters at the plot points).

LINES BLANK SOLID
specifies (for future plots) that the first trace will have blanks as plot line types (that is, the first trace will have no connecting line between plot points), and the second trace will have solid connecting lines between the plot points.

PLOT Y PRED VERSUS X
generates a plot with 2 traces--

1) Y versus X;
2) PRED versus X.

that is,

1) a trace with the raw data Y (vertically) versus the variable X (horizontally);

2) a trace with the predicted values PRED (vertically) versus the variable X (horizontally);

Due to the prior CHARACTERS and LINES commands, the 2 traces will have the following characteristics--

Trace	Contents	Characters	Lines
1	Y vs X	X	BLANK
2	PRED vs X	BLANK	SOLID

that is, the first trace will have X's as plot characters with blank (= no) connecting lines, and

the second trace will have blanks as plot characters with solid connecting lines.

The default plot type is continuous; the default axis will be neat and float with the data.

Generate a Superimposed Plot of Raw Data and Fitted Values from a Spline Fit

Problem

Data exists on a file ABC as a series of line images with 2 values per line image (response variable and independent variable). The knots for such a fit are on a second file DEFG (1 knot point per line image). Read in the data from file ABC; place the data in variables Y and X. Read in the knots from file DEFG; place the knots in variable KNOTS.

Carry out a least squares spline fit using the specified knots. Generate a plot of the raw data Y (as discrete x's) and the superimposed fitted curve (as a solid line) versus X.

DATAPLOT Program

```
READ ABC. Y X
READ DEFG. KNOTS
.
CUBIC SPLINE FIT Y X KNOTS
.
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
```

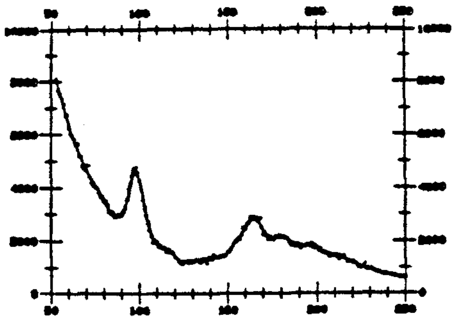
Program Description

READ ABC. Y X
carries out a format-free read of data from file ABC; The data is read into variables Y and X; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

READ DEFG. KNOTS
carries out a format-free read of data from file DEFG; the first number on each line image of the file is read into variable KNOTS. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

.
is a null command—it visually separates chunks of DATAPLOT code.

SPLINE FIT Y X KNOTS
carries out a least squares cubic spline fit of Y as a function of X with knot points as specified by the variable KNOTS. At the end of the fit, the best-fit values for the 4 cubic coefficients for each of the spline intervals are placed in the following parameters—



Interval	*Parameters*
1	A10, A11, A12, A13
2	A20, A21, A22, A23
3	A30, A31, A32, A33
4	A40, A41, A42, A43
5	A50, A51, A52, A53
etc.	

Predicted values from the fit are computed and placed in the variable PRED. Residual values (= Y - PRED) are computed and placed in the variable RES. The residual standard deviation is placed in the parameter RESSD; the residual degrees of freedom is placed in RESDF. Replication is automatically checked for; if existent, the replication standard deviation is computed and placed in REPSD; the replication degrees of freedom is placed in REPDF; the lack of fit F test cdf value is placed in the parameter LOFCDF. All of the above (PRED, RES, RESSD, RESDF, REPSD, REPDF, LOFCDF) are automatically computed by DATAPLOT whenever any DATAPLOT modeling command (FIT, PRE-FIT, EXACT ... FIT, SPLINE FIT, SMOOTH, ANOVA, MEDIAN POLISH) is executed. These variables/parameters may be subsequently used by the analyst in any way whatsoever. They are usually used for further model-verification and residual analysis. The default spline fit degree is cubic. If a non-cubic spline fit is desired, then specify if directly,as in

```
QUADRATIC SPLINE FIT Y X KNOTS
```

or indirectly via the POLYNOMIAL DEGREE command, as in

```
POLYNOMIAL DEGREE 2
SPLINE FIT Y X KNOTS
```

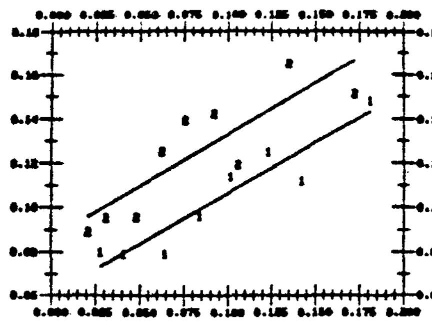
.
is a null command—it visually separates chunks of DATAPLOT code.

Generate a Superimposed Plot of Raw Data and Fitted Values from 2 Fits

Problem

Data exists on a file ABC as a series of line images with 3 values per line image (response variable Y, and independent variables X and LAB). Read in the data; place them in variables Y, X, and LAB.

Fit a line to the data from lab 1; fit a line to the data from lab 2. Generate a plot showing raw data and fitted curves from both labs. Lab 1 data has character 1; lab 2 has character 2. The 2 fitted lines are solid and dotted.



DATAPLOT Program

```
READ ABC. Y X LAB
.
FIT Y = A1+B1*X SUBSET LAB 1
LET PRED1 = PRED
.
FIT Y = A2+B2*X SUBSET LAB 2
.
CHARACTERS 1 BLANK 2 BLANK
LINES BLANK SOLID BLANK DOTTED
PLOT Y PRED1 VS X SUBSET LAB 1 AND
PLOT Y PRED  VS X SUBSET LAB 2
```

Program Description

READ ABC. Y X LAB
carries out a format-free read of data from file ABC; The data is read into variables Y and X; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X; the third number on each line is read into variable LAB. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

.
is a null command—it visually separates chunks of DATAPLOT code.

*FIT Y = A1+B1*X SUBSET LAB 1*

*carries out a least squares fit of Y as a function of X based on the linear model Y = A1+B1*X. Not all of the data in Y and X are included in the fit however; only those data points in Y and X which correspond to values in the LAB variable equal to 1 are included in the fit. At the end of the fit, the best-fit values placed into the parameters A1 and B1. Predicted values from the fit are computed and placed in the variable PRED. Residual values (= Y - PRED) are computed and placed in the variable RES. The other usual fit parameters (RESSD, RESDF, REPSD, REPDF, and LOFCDF) are also computed.*

LET PRED1 = PRED
defines the variable PRED1 and fills that variable with the current contents of the variable PRED. PRED1 is a variable (rather than a parameter) because the right side of the equation PRED1 = PRED involves a variable. This copy of PRED into PRED1 was done in anticipation of the next fit (to the LAB 2 data) which will have the side effect that the predicted values in PRED will be updated.

*FIT Y = A2+B2*X SUBSET LAB 2*
*carries out a least squares fit of Y as a function of X based on the linear model Y = A2+B2*X. Only those data points in Y and X which correspond to values in the LAB variable equal to 2 are included in the fit. At the end of the fit, the best-fit values placed into the parameters A2 and B2. Predicted values from the fit are computed and placed in the variable PRED. Residual values (= Y - PRED) are computed and placed in the variable RES. The other usual fit parameters (RESSD, RESDF, REPSD, REPDF, and LOFCDF) are also computed.*

CHARACTERS 1 BLANK 2 BLANK
specifies (for future plots) that

```
trace 1 has character 1
trace 2 has character blank
trace 3 has character 2
trace 4 has character blank
```

LINES BLANK SOLID BLANK SOLID specifies (for future plots)

```
trace 1 has blank (= no) connecting lines
trace 2 has solid        connecting lines
trace 3 has blank (= no) connecting lines
trace 4 has solid        connecting lines
```

PLOT Y PRED1 VERSUS X SUBSET LAB 1 AND
PLOT Y PRED VERSUS X SUBSET LAB 2
generates a plot with 4 traces—

1) Y versus X (but restricted to LAB 1 data);
2) PRED1 versus X (but restricted to LAB 1 data);
3) Y versus X (but restricted to LAB 2 data);
4) PRED versus X (but restricted to LAB 2 data).

that is,

1) a trace with the raw data Y (vertically) versus the variable X (horizontally), where the Y and X is restricted to those values corresponding to 1 in the LAB variable;

2) a trace with the predicted values PRED1 (vertically) versus the variable X (horizontally), where the PRED1 and X is restricted to those values corresponding to 1 in the LAB variable;

3) a trace with the raw data Y (vertically) versus the variable X (horizontally), where the Y and X is restricted to those values corresponding to 2 in the LAB variable;

4) a trace with the predicted values PRED (vertically) versus the variable X (horizontally), where the PRED and X is restricted to those values corresponding to 2 in the LAB variable.

Due to the prior CHARACTERS and LINES commands, the 4 traces will have the following characteristics—

Trace	*Contents*		*Characters*	*Lines*
1	*Y vs X*	*(LAB 1 only)*	*1*	*BLANK*
2	*PRED1 vs X*	*(LAB 1 only)*	*BLANK*	*SOLID*
3	*Y vs X*	*(LAB 2 only)*	*2*	*BLANK*
4	*PRED2 vs X*	*(LAB 2 only)*	*BLANK*	*SOLID*

that is, the first trace will have 1's as plot characters with blank (= no) connecting lines, the second trace will have blanks as plot characters with solid connecting lines; the third trace will have 2's as plot characters with blank connecting lines, and the fourth trace will have blanks as plot characters with solid connecting lines.

The default plot type is continuous; the default axis will be neat and float with the data.

Generate a Residual Plot

Problem

Data exists on a file ABC as a series of line images with 2 values per line image (response variable Y and indepen- dent variable X). Read in the data; place them in variables Y and X.

Theory suggests a linear relationship between Y and X. Carry out a least squares fit. Generate a plot (with discrete, unconnected x's) of the fit residuals (vertically) versus values of the independent variable (horizontally). Such a plot is drawn from a battery of graphical procedures known as residual analysis—they are essential for assessing goodness of fit of a general model. See figure 24.

DATAPLOT Program

```
READ ABC. Y X
.
FIT Y = A+B*X
.
CHARACTERS X
LINES BLANK
PLOT RES X
```

Program Description

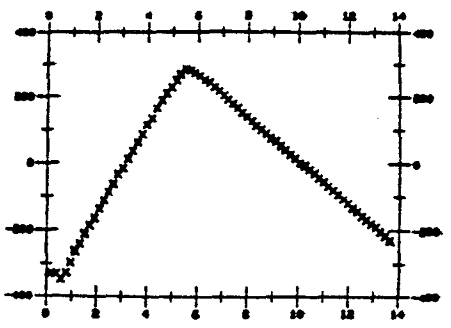
<u>READ ABC. Y X</u>
carries out a format-free read of data from file ABC; The data is read into variables Y and X; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

<u>.</u>
is a null command—it visually separates chunks of DATAPLOT code.

*<u>FIT Y = A+B*X</u>*
*carries out a least squares fit of Y as a function of X based on the linear model Y = A+B*X. At the end of the fit, the best-fit values placed into the parameters A and B. Predicted values from the fit are computed and placed in the variable PRED. Residual values (= Y - PRED) are computed and placed in the variable RES. The other usual fit parameters (RESSD, RESDF, REPSD, REPDF, and LOFCDF) are also computed.*

<u>CHARACTER X</u>
specifies (for future plots) that the trace will have x's at the plot points.

<u>LINES BLANK</u>
specifies (for future plots) that the trace will have blank lines (that is no connecting lines).



<u>PLOT RES X</u>
generates a plot of the fit residuals RES versus the variable X. The variable RES is plotted vertically; the variable X is plotted horizontally. Due to the prior entry of the CHARACTERS X command and the LINES BLANK command, the plot will appear with x's as plot characters and with no connecting lines. The default plot type is continuous; the default axis limits will be neat and float with the data.

From a data analysis point of view, any structure in the residual plot indicates that the model (linear in this case) may be improved upon. The well-defined structure in this residual plot suggests that a linear spline model would be provide a better fit to the data. A knot at X = 6 is suggested from the plot. To carry out such a fit, one would enter

```
LET KNOTS(1) = 6
LINEAR SPLINE FIT Y X KNOTS
```

This would define the first element of the variable KNOTS to be 6. It would then carry out a linear spline fit of Y on X using the knots in the variable KNOTS.

]

Generate a Lag Plot

Problem

Data exists on a file ABC as a series of values (1 value of a variable per line image). Read in the data; place it in the variable X.

Generate a lag plot with lag = 1. Use the symbol x as the plot character; have no connecting lines. The lag plot (with lag = 1) is a plot of X(i) versus X(i-1). It is a graphical data analysis technique for assessing autocorrelation structure in time series.

DATAPLOT Program

```
READ ABC. X
.
CHARACTERS X
LINES BLANK
LAG 1 PLOT X
```

Program Description

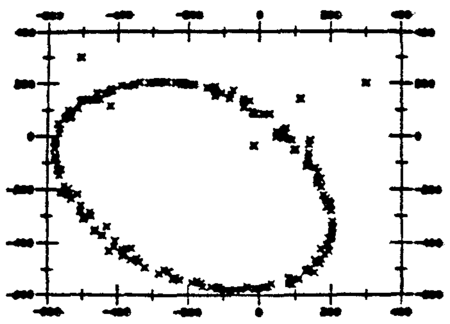
READ ABC. X
carries out a format-free read of data from file ABC. The first number on each line image of the file is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

.
is a null command—it visually separates chunks of DATAPLOT code.

CHARACTER X
specifies (for future plots) that the trace will have x's at the plot points.

LINES BLANK
specifies (for future plots) that the trace will have blank lines (that is no connecting lines).

LAG 1 PLOT X
generates a lag 1 plot of the data in the variable X. X(i) is plotted vertically; X(i-1) is plotted horizontally. Due to the prior entry of the CHARACTERS X command and the LINES BLANK command, the plot will appear with x's as plot characters and with no connecting lines. The default plot type is continuous; the default axis limits will be neat and float with the data.



From a data analysis point of view, any structure in the lag plot indicates that autocorrelation structure exists in the data and/or (for time series data) a more complicated model (e.g., autoregressive, cyclic, etc.) is appropriate. The well-defined circular structure in this lag plot indicates an underlying cyclic model. The next
SYSTEM WARNING - MAX TIME
step in the analysis would be to call on various time series analysis techniques (e.g., spectral plots, complex demodulation plots, etc.) to gain more detailed insight into this cyclic structure.

Lag plots are available for lags other than 1. For example, to generate a lag plot with a lag of 3, one would enter

```
LAG 3 PLOT X
```

When more than one variable is available, lag plots may be used to check for cross-correlation structure between the 2 series. For example, to generate a lag plot (with lag = 12) of the data in X and Y, one would enter

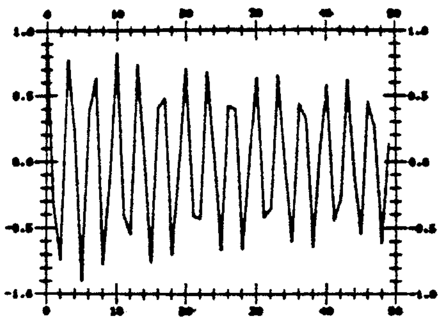
```
LAG 12 PLOT X Y
```

Generate an Autocorrelation Plot

Problem

Data exists on a file ABC as a series of values (1 value of a variable per line image). Read in the data; place it in the variable X.

Generate an autocorrelation plot. The autocorrelation plot is a plot of the correlation between X(i) and X(i+h) versus lag h. The correlations take on values between -1 and +1. The autocorrelation plot is a graphical data analysis technique for assessing autocorrelation structure in time series.



DATAPLOT Program

```
READ ABC. X
AUTOCORRELATION PLOT X
```

Program Description

READ ABC. X
carries out a format-free read of data from file ABC. The first number on each line image of the file is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

AUTOCORRELATION PLOT X
generates an autocorrelation plot of the data in the variable X. The correlation between X(i) and X(i+h) is plotted vertically; the lag h is plotted horizontally. The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

From a data analysis point of view, the type of structure in the autocorrelation plot indicates the type of model (e.g., autoregressive, cyclic, etc.) which would be appropriate for the data. The cyclic structure in this autocorrelation plot indicates an underlying cyclic model. The next step in the analysis would be to call on other time series analysis techniques (e.g., spectral plots, complex demodulation plots, etc.) to gain more detailed insight into this cyclic structure.

When more than one variable is available, the autocorrelation plot generalizes to the cross-correlation plot— this checks cross-correlation structure between the 2 series. For example, to generate a cross-correlation plot of the data in Y and X, one would enter

```
CROSS-CORRELATION PLOT Y X
```

Generate a Spectral Plot

Problem

Data exists on a file ABC as a series of values (1 value of a variable per line image). Read in the data; place it in the variable X.

Generate a spectral plot. The spectral plot is a frequency domain plot of power (= variation) versus frequency (0 to .5). The spectral power function is the smoothed Fourier transform of the autocorrelation function. The spectral plot is a graphical data analysis technique for assessing autocorrelation and cyclic structure in time series.

DATAPLOT Program

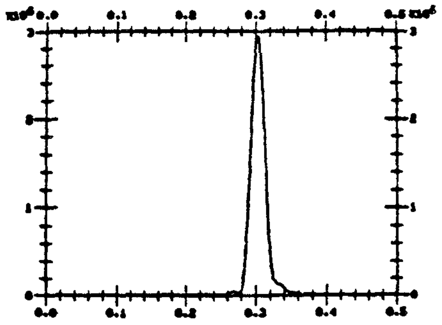
```
READ ABC. X
SPECTRAL PLOT X
```

Program Description

READ ABC. X
carries out a format-free read of data from file ABC. The first number on each line image of the file is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

SPECTRAL PLOT X
generates a spectral plot of the data in the variable X. The variance component at each frequency is plotted vertically; the frequency (0 to .5--cycles per unit time where the unit time is assumed to be 1) is plotted horizontally. The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

From a data analysis point of view, the type of structure in the autocorrelation plot indicates the location of peaks in the spectral plot indicates the dominant frequency for underlying cyclic models. For this data, the peak at frequency = .3 (approximately) indicates an underlying cycle of about .3 (that is, an underlying period in the data of slighly more than 3 observations). The next step in this analysis would be to call on other time series analysis techniques (such as complex demodulation phase plots) to determine if this frequency estimate is constant over the entire domain of the data, or to carry out a non-linear fit with an underlying cyclic model.



When more than one variable is available, the spectral plot generalizes to various cross-spectral plots-- this checks frequency structure between the 2 series. For example, to generate a cross-spectral plot of the data in Y and X, one would enter

```
CROSS-SPECTRAL PLOT Y X
```

Generate a Complex Demodulation Plot

Problem

Data exists on a file ABC as a series of values (1 value of a variable per line image). Read in the data; place it in the variable X.

Generate a complex demodulation phase plot at the demodulation frequency of 0.3. The complex demodulation phase plot is a plot of locally-estimated generation-frequency versus time. The complex demoduulation amplitude plot is a plot of locally-estimated amplitude versus time. The demodulation phase and amplitude plots are graphical data analysis techniques for assessing (among other things) whether generation-frequency and amplitude are constant over the entire time domain when the analyst suspects that the data has been generated from a single-cycle model. See figure 28.

DATAPLOT Program

```
READ ABC. X
DEMODULATION FREQUENCY .3
COMPLEX DEMODULATION PLOT X
```

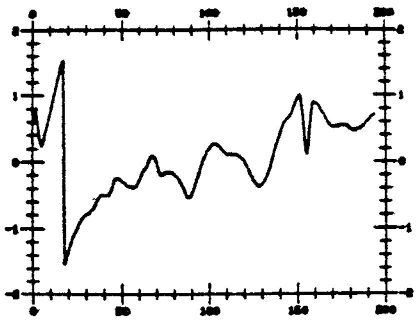
Program Description

READ ABC. X

carries out a format-free read of data from file ABC. The first number on each line image of the file is read into the variable X. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

DEMODULATION FREQUENCY .3
specifies (for future complex demodulations) that the demodulation frequency is .3. This demodulation frequency specification is in anticipation of the upcoming complex demodulation plot--which requires a prior-defined demodulation frequency.

COMPLEX DEMODULATION PHASE PLOT X
generates a spectral plot of the data in the variable X. A local (in time) estimate of the generation-frequency is plotted vertically; time is plotted horizontally (it is assumed that the data in X were collected equi-spaced in time with time increment = 1). The default plot type is continuous; the default axis limits will be neat and float with the data; the default character



type is blank; the default line type is solid. In general, complex demodulation plots should be generated with discrete x's and no connecting lines--this is to accomodate angle wrap-around which occasionally occurs in the frequency calculations. For this data set and this demodulation frequency, shifting to a discrete representation of the data was not necessary.

From a data analysis point of view, the type of structure in the complex demodulation phase plot indicates

1) How close is our estimate of the generation-frequency?

2) Is the generation-frequency constant over the entire time-domain of the data?

Note that the generation-frequency estimates (vertically) range within the usual principal domain of -pi/2 to +pi/2, that is, from about -1.6 to +1.6. The analyst should note when angular wrap-around (= the frequency estimate running off the top of the principal domain and then continuing at the bottom of the principal domain) occurs. For this data, there is a single wrap-around at about time = 20 on the horizontal axis. The fact that there is only 1 wrap-around indicates that our demodulation frequency estimate of .3 is fairly close. The fact that the complex demodulation plot phase plot has positive slope indicates that a better estimate for the generation-frequency would be slighly higher than .3 The generally-moderate ripples in the plot indicates that the underlying generation-frequency is fairly constant over the entire time domain of the data.

The next step in this analysis would be to redefine the demodulation frequency to be slightly higher than .3 (such as .31), via

```
DEMODULATION FREQUENCY .31
```

and then generate another complex demodulation phase plot via

```
COMLEX DEMODULATION PHASE PLOT X
```

This is repeated until the complex demodulation phase plot has as little (= near-zero) slope as possible. Another course of action would be to carry out a similar check of the constancy (over the entire time domain) of the of the amplitude in a single-cycle model, as via

```
COMPLEX DEMODULATION AMPLITUDE PLOT X
```

Yet a third course of action would be to make use of the above-estimated generation-frequency and amplitude estimates and carry out a non-linear fit of the data with the single-cycle model, as in

```
LET N = NUMBER X
LET T = SEQUENCE 1 1 N
.
LET CONST = MEAN X
LET AMP = best estimate
LET FREQ = best estimate
FIT X = CONST + AMP * SIN(2*3.14159*FREQ*T+PHASE)
```

The above code would

1) *compute the number of observations in X and place this number in the parameter N;*

2) *Form a variable T which has the sequence 1, 2, 3, ..., N;*

3) *Compute the mean of X and place it in the parameter CONST;*

4) *Define a parameter AMP with our best estimate of the amplitude (as obtained from the complex demodulation amplitude plot);*

5) *Define a parameter FREQ with our best estimate of the generation-frequency (as obtained from the complex demodulation phase plot);*

6) *Carry out a least-squares fit of X as a function of T via the non-linear model X = CONST + AMP * SIN(2*3.14159*FREQ*T + PHASE)*

Plot a Function

Problem

*Plot the Normal N(0,1) density function. This function has the form f(x) = (1/sqrt(2*pi)) * exp(-0.5*x**2) .*

DATAPLOT Program

Three different programs are presented to generate this plot. Program 1 is

```
      PLOT (1/SQRT(2*PI))*EXP(-0.5*X**2) FOR X = -3 .1 3
```

Program 2 is

```
      LET FUNCTION F1 = 1/SQRT(2*PI)
      LET FUNCTION F2 = EXP(-0.5*X**2)
      LET FUNCTION G = F1*F2
      PLOT G FOR X = -3 .1 3
```

Program 3 is

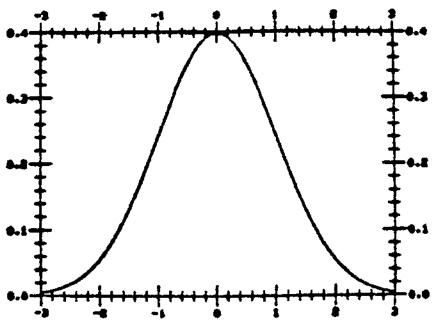
```
      PLOT NORPDF(X) FOR X = -3 .1 3
```

Program Description

*PLOT (1/SQRT(2*PI))*EXP(-0.5*X**2) FOR X = -3 .1 3*
in program 1 generates the desired plot. The function will be evaluated at a series of X values starting with -3, at increments of .1, and ending at 3 (that is, at X = -3, -2.9, -2.8, ..., 2.8, 2.9, 3). Note that the X values for the function evaluation are temporary and local to the PLOT statement only; no X variable as such is created, nor does such a PLOT statement affect an X variable which the analyst may happen to have. Note that the function includes a reference to PI— this is an automatically-provided DATAPLOT parameter with the value 3.1415926. It may be used like any other DATAPLOT parameter. The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

*LET FUNCTION F1 = 1/SQRT(2*PI)*
*in progam 2 defines the function F1 and assigns to it the 12-character string 1/SQRT(2*PI)*

*LET FUNCTION F2 = EXP(-0.5*X**2)*
*defines the function F2 and assigns to it the 14-character string EXP(-0.5*X**2)*



*LET FUNCTION G = F1*F2*
*defines the function G and assigns to it the 12+14+1+2+2 = 31 character string (1/SQRT(2*PI))*(EXP(-0.5*X**2)) 12 characters came from F1; 14 characters came from F2; 1 character came from the * in the desired operation G = F1*F2; 2 characters came from automatically-provided parentheses around F1; 2 characters came from automatically-provided parentheses around F2. The LET FUNCTION command thus allows not only the definition of individual functions, but the concatonation and step-wise composition of more complicated functions from simpler ones.*

PLOT F FOR X = -3 .1 3
generates the desired plot. DATAPLOT recognizes G as a function and inserts the full character string in its place for the generation of the plot.

PLOT NORPDF(X) FOR X = -3 .1 3
in program 3 also generates the desired plot. This PLOT statement makes use of the fact that DATAPLOT has a wide assortment of built-in intrinsic library functions which may be called on in any PLOT, 3D-PLOT, FIT, PRE-FIT, or LET statements. In particular, the built-in function for the normal N(0,1) probability density function is NORPDF(.) Other available probability-type functions include

normal	probability density function	NORPDF(X)
normal	cumulative distribution function	NORCDF(X)
normal	percent point function	NORPPF(P)
t	probability density function	TPDF(X,NU)
t	cumulative distribution function	TCDF(X,NU)
t	percent point function	TPPF(P,NU)
chi-squared	probability density function	CHSPDF(X,NU)
chi-squared	cumulative distribution function	CHSCDF(X,NU)
chi-squared	percent point function	CHSPPF(P,NU)
F	probability density function	FPDF(X,NU1,NU2)
F	cumulative distribution function	FCDF(X,NU1,NU2)
F	percent point function	FPPF(P,NU1,NU2)

Plot a Bessel Function

Problem

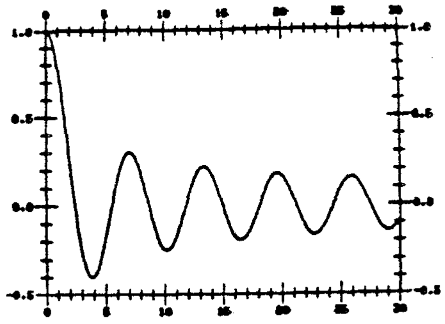
Plot J0(x)—a member of the family of Bessel functions of the first kind.

DATAPLOT Program

```
PLOT BESSJ0(X) FOR X = 0 .1 30
```

Program Description

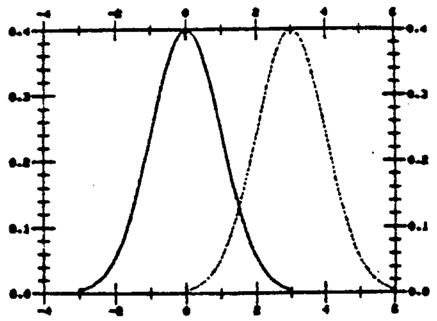
<u>*PLOT BESSJ0(X) FOR X = 0 .1 30*</u>
generates the desired plot. The function will be evaluated at a series of X values starting with 0, at increments of .1, and ending at 30 (that is, at X = 0, .1, .2, ..., 29.8, 29.9, 30). Note that the X values for the function evaluation are temporary and local to the PLOT statement only; no X variable as such is created, nor does such a PLOT statement affect an X variable which the analyst may happen to have. This PLOT statement makes use of the fact that DATAPLOT has a wide assortment of built-in intrinsic library functions which may be called on in any PLOT, 3D-PLOT, FIT, PRE-FIT, or LET statements. In particular, the built-in function for the Bessel J0(x) function is BESSJ0(.). Available Bessel functions are J0(x), J1(x), ..., J10(x), and are referenced as BESSJ0(.), BESSJ1(.), ..., BESSJ10(X). For the plot itself, the default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.



Plot Multiple Functions

Problem

*Plot 2 Normal N(.,1) density functions—one centered at 0 and the other centered at 3. The first function has the form f(x) = (1/sqrt(2*pi)) * exp(-0.5*x**2) ; the second function has the form f(x) = (1/sqrt(2*pi)) * exp(-0.5*(x-3)**2) . Have the first trace solid and the second trace dotted.*



DATAPLOT Program

Three different programs are presented to generate this plot. Program 1 is

```
LINES SOLID DOTTED
PLOT (1/SQRT(2*PI))*EXP(-0.5*X**2) FOR X = -3 .1 3 AND
PLOT (1/SQRT(2*PI))*EXP(-0.5*(X-3)**2) FOR X = 0 .1 6
```

Program 2 is

```
LINES SOLID DOTTED
LET FUNCTION F0 = 1/SQRT(2*PI)
LET FUNCTION F1 = EXP(-0.5*X**2)
LET FUNCTION F2 = EXP(-0.5*(X-3)**2)
LET FUNCTION G1 = F0*F1
LET FUNCTION G2 = F0*F2
PLOT G1 FOR X = -3 .1 3 AND
PLOT G2 FOR X = 0 .1 6
```

Program 3 is

```
LINES SOLID DOTTED
PLOT NORPDF(X) FOR X = -3 .1 3 AND
PLOT NORPDF(X-3) FOR X = 0 .1 6
```

Program Description

LINES SOLID DOTTED
in all 3 programs specifies (for future plots) that the first trace wll have solid connecting lines between plot points, and the second trace will have dotted connecting lines between plot points.

*PLOT (1/SQRT(2*PI))*EXP(-0.5*X**2) FOR X = -3 .1 3 AND*
*PLOT (1/SQRT(2*PI))*EXP(-0.5*(X-3)**2) FOR X = 0 .1 6*
in program 1 generates the desired plot with the 2 traces— one trace for each function. The first function will be evaluated at a series of X values starting with -3, at increments of .1, and ending at 3 (that is, at X = -3, -2.9, -2.8, ..., 2.8, 2.9, 3). The second function will be evaluated at a series of X values starting with 0, at increments of .1, and ending at 6 (that is, at X = 0, .1, .2, ..., 5.8, 5.9, 6). Note that the X values for each of the function evaluations are temporary and local to the PLOT statement only; no X variable as such is created, nor does such a PLOT statement affect an X variable which the analyst may happen to have.

The AND suffix at the end of the first PLOT statement is important— it tells DATAPLOT to generate only 1 plot, but on that 1 plot to have 2 traces—a plot of the first function, and then (on the same plot) to superimpose a plot of the second function. When the AND is included, the following occurs—

1) *the usual screen pre-erase occurs;*
2) *the trace for the first function is generated;*
3) *the trace for the second function is superimposed.*

If the AND were omitted, then the following would occur-

1) *the usual screen pre-erase for the first plot occurs;*
2) *the trace for the first function is generated;*
3) *the usual screen pre-erase for the second plot occurs (thereby wiping out the first function trace);*
4) *the trace for the second function is generated.*

Note that the functions includes a reference to PI— this is an automatically-provided DATAPLOT parameter with the value 3.1415926. It may be used like any other DATAPLOT parameter.

The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type (for both traces) is blank; due to the prior LINES command, the first trace will have solid connecting lines and the second trace will have dotted connecting lines.

LET FUNCTION F0 = 1/SQRT(2*PI)
in program 2 defines the function F0 and assigns to it the 12-character string 1/SQRT(2*PI)

LET FUNCTION F1 = EXP(-0.5*X**2)
defines the function F1 and assigns to it the 14-character string EXP(-0.5*x**2)

LET FUNCTION F2 = EXP(-0.5*(X-3)**2)
defines the function F2 and assigns to it the 18-character string EXP(-0.5*(X-3)**2)

LET FUNCTION G1 = F0*F1
defines the function G1 and assigns to it the 12+14+1+2+2 = 31 character string (1/SQRT(2*PI))*(EXP(-0.5*X**2)) 12 characters came from F0; 14 characters came from F1; 1 character came from the * in the desired operation G1 = F0*F2; 2 characters came from automatically-provided parentheses around F0; 2 characters came from automatically-provided parentheses around F1. The LET FUNCTION command thus allows not only the definition of individual functions, but the concatonation and step-wise composition of more complicated functions from simpler ones.

LET FUNCTION G2 = F0*F2
defines the function G2 and assigns to it the 12+18+1+2+2 = 35 character string (1/SQRT(2*PI))*(EXP(-0.5*(X-3)**2)) 12 characters came from F0; 18 characters came from F2; 1 character came from the * in the desired operation G2 = F0*F2; 2 characters came from automatically-provided parentheses around F0; 2 characters came from automatically-provided parentheses around F2.

PLOT G1 for x = -3 .1 3 AND
PLOT G2 FOR X = 0 .1 6
generates the desired plot. DATAPLOT recognizes G1 and G2 as functions and inserts the full character strings in their place for the generation of the plot.

PLOT NORPDF(X) FOR X = -3 .1 3 AND
PLOT NORPDF(X-3) FOR X = 0 .1 6 in program 3 also generates the desired plot. This PLOT statement makes use of the fact that DATAPLOT has a wide assortment of built-in intrinsic library functions which may be called on in any PLOT, 3D-PLOT, FIT, PRE-FIT, or LET statements. In particular, the built-in function for the normal N(0,1) probability density function is NORPDF(.) Other available probability-type functions include

normal	probability density function	NORPDF(X)
normal	cumulative distribution function	NORCDF(X)
normal	percent point function	NORPPF(P)
t	probability density function	TPDF(X,NU)
t	cumulative distribution function	TCDF(X,NU)
t	percent point function	TPPF(P,NU)
chi-squared	probability density function	CHSPDF(X,NU)
chi-squared	cumulative distribution function	CHSCDF(X,NU)
chi-squared	percent point function	CHSPPF(P,NU)
F	probability density function	FPDF(X,NU1,NU2)
F	cumulative distribution function	FCDF(X,NU1,NU2)
F	percent point function	FPPF(P,NU1,NU2)

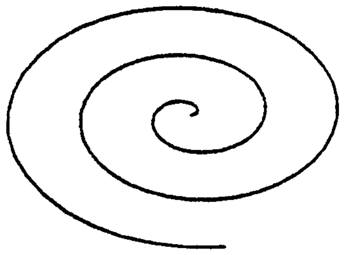
Plot Polar Coordinate Function

Problem

Generate the polar coordinate function (a spiral)

*radius = 2*theta*

for theta = 0 (10) 1000 degrees.



DATAPLOT Program

```
LET THETA = SEQUENCE 0 10 1000
LET R = 2*THETA
.
TRIGONOMETRIC UNITS DEGREES
LET Y = R*SIN(THETA)
LET X = R*COS(THETA)
.
FRAME OFF
PRE-SORT OFF
PLOT Y X
```

Program Description

The strategy for generating this (and all) polar coordinate function plots is to define the THETA variable for the desired values of theta, compute the R variable as dictated by the desired polar function, transform the polar THETA and R values into Cartesian Y and X values via the usual

*Y = R*sin(THETA)*
*X = R*cos(THETA)*

and then plot Y versus X.

LET THETA = SEQUENCE 0 10 1000
defines a variable X and places in that variable the values 0, 10, 20, 30, ..., 990, 1000. This makes use of the SEQUENCE sub-capability under the LET command.

*LET R = 2*THETA*
defines a variable R. R has the same number of elements as THETA. An individual element in R is computed by taking an individual element in THETA and multiplying it by 2.

.
is a null command--it visually separates chunks of DATAPLOT code.

TRIGONOMETRIC UNITS DEGREES
specifies (for all future calculations) that the units for trigonemetric calculations is degrees; the default is radians. Another available system of units is grads (popular in Europe). A synonym command for TRIGONOMETRIC UNITS DEGREES is TRIG UNITS DEGREES, or simply DEGREES.

*LET Y = R*SIN(THETA)*
defines a varaible Y. Y has the same number of elements as THETA and R. An individual element in Y is computed by taking an individual element in THETA, taking the sine of it, and then multiplying the result by the corresponding individual element in R. The rationale for computing Y is that it is part of the standard transformation in going from polar coordinates to Cartesian coordinates.

*LET X = R*COS(THETA)*
defines a varaible X. X has the same number of elements as THETA and R. An individual element in X is computed by taking an individual element in THETA, taking the cosine of it, and then multiplying the result by the corresponding individual element in R. The rationale for computing X is that it too is part of the standard transformation in going from polar coordinates to Cartesian coordinates.

FRAME OFF
turns off (for future plots) the frame (axis lines Z and surrounding box) that would typically circumscribe the usual Y versus X plot. As one would expect, when the frame is turned off, so too are the tic marks on the frame and the tic mark labels adjacent to the frame. The rationale for turning the frame off is that it makes little sense in a polar coordinate context.

PRE-SORT OFF
turns off (for future plots) the default sorting (based on the horizontal axis variable) of the plot points. Such sorting before the plot is automatic and it relieves the analyst of the needless chore of having data in variables sorted before a plot. When the pre-sort switch is ON, then DATAPLOT will (before a plot) take the data in the variables in whatever order they are, sort them (according to the horizontal axis variable) and then plot them. This allows the analyst to

generate the same plot regardless of the order of the data within variables. (Note that the variables themselves remain unchanged; DATAPLOT does all sorting in its own scratch area). For the vast majority (in excess of 99%) of plots, such pre-plot sorting is appropriate, and so the default for the pre-sort switch is ON. However, for certain applications (for example, polar function plots and maps) such sorting is not appropriate. In those cases in which we wish the points to be plotted in exactly the order in which they reside in the variables, then the pre-sort switch should be turned off.

<u>PLOT Y X</u>
generates a plot of variable Y (vertically) versus variable X (horizontally). The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

Plot a Function and Its Derivatives

Problem

*Plot the Normal N(0,1) density function, along with its first and second derivatives. This function has the form f(x) = (1/sqrt(2*pi)) * exp(-0.5*x**2) . Have the line types solid, dotted and dashed, respectively.*

DATAPLOT Program

```
LET FUNCTION F = (1/SQRT(2*PI))*EXP(-0.5*X**2)
LET FUNCTION D1 = DERIVATIVE F WRT X
LET FUNCTION D2 = DERIVATIVE D1 WRT X
.
LINES SOLID DOT DASH
PLOT F FOR X = -3 .1 3 AND
PLOT D1 FOR X = -3 .1 3 AND
PLOT D2 FOR X = -3 .1 3
```

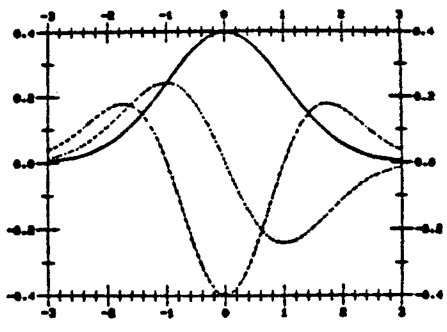
Program Description

*LET FUNCTION F = (1/SQRT(2*PI))*EXP(-0.5*X**2)*
*defines the function F and assigns to it the 29-character string (1/SQRT(2*PI))*EXP(-0.5*X**2)*

LET FUNCTION D1 = DERIVATIVE F
defines the function D1, determines the exact analytic derivative of function F, and places in function D1 the character string for that derived derivative. The WRT X in this command line indicates that the differentiation should be done "with respect to X"; in general, the SYSTEM WARNING - MAX PAGES differentiation may be done with respect to any variable or parameter which appears in the function string. This command statement demonstrates the use of the DERIVATIVE sub-capability under the LET command. The LET FUNCTION command thus allows not only the definition of functions, but also allows certain important mathematical operations (such as differentiation) to be applied to functions. The ability to take symbolic derivatives finds many useful applications in a scientific/research environment.

LET FUNCTION D2 = DERIVATIVE D1
defines the function D2, determines the exact analytic derivative of function D1, and places in function D2 the character string for that derived derivative. This command statement is also a demonstration of the use of the DERIVATIVE sub-capability under the LET command.

.
is a null command—it visually separates chunks of DATAPLOT code.



LINES SOLID DOTTED DASHED
specifies (for future plots) that the first trace wll have solid connecting lines between plot points, the second trace will have dotted connecting lines between plot points, and the third trace will have dashed connecting lines between plot points.

PLOT F FOR X = -3 .1 3 AND
PLOT D1 FOR X = -3 .1 3 AND
PLOT D2 FOR X = -3 .1 3

generates the desired plot with the 3 traces— one trace for each function. DATAPLOT recognizes F, D1, and D2 as functions and inserts the full character strings in their place for the generation of the plot. Each of the 3 functions will be evaluated at a series of X values starting with -3, at increments of .1, and ending at 3 (that is, at X = -3, -2.9, -2.8, ..., 2.8, 2.9, 3). Note that the X values for each of the function evaluations are temporary and local to the PLOT statement only; no X variable as such is created, nor does such a PLOT statement affect an X variable which the analyst may happen to have.

The AND suffix at the end of the first and second PLOT statement is important— it tells DATAPLOT to generate 1 plot with 3 superimposed traces, as opposed to 3 separate plots (each with only 1 trace). When the AND is included, the following occurs—

1) the usual screen pre-erase occurs;

2) the trace for the first function is generated;

3) the trace for the second function is superimposed;

4) the trace for the third function is superimposed.

If the AND were omitted, then the following would occur—

1) *the usual screen pre—erase for the first plot occurs;*

2) *the trace for the first function is generated;*

3) *the usual screen pre—erase for the second plot occurs (thereby wiping out the first function trace);*

4) *the trace for the second function is generated;*

5) *the usual screen pre-erase for the third plot occurs (thereby wiping out the second function trace);*

6) *the trace for the third function is generated;*

Note that the functions includes a reference to PI— this is an automatically-provided DATAPLOT parameter with the value 3.1415926. It may be used like any other DATAPLOT parameter.

The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type (for both traces) is blank; due to the prior LINES command, the first trace will have solid connecting lines, the second trace will have dotted connecting lines, and the third trace will have dashed connecting lines.

Plot 2 Functions and Their Convolution

Problem

Convolve 2 functions. The first function is

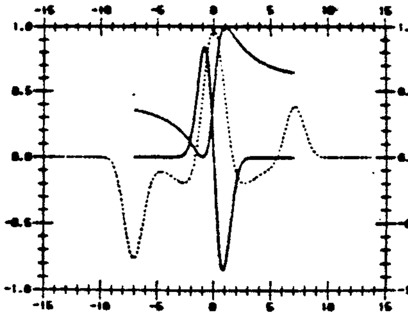
f1(x) = ((x+1)**2) / ((x**2)+1)/2

The second function is

f2(x) = 2**((-(x+1.3/2)**2)/(A**2))

- 2**((-(x-1.3/2)**2)/(A**2))

with the parameter a taking on the value .85. The domain of interest for the functions is -7 to 7. Carry out a numerical convolution with the functions being evaluated over the full interval (-7 to 7) and at increments of .1. Plot the 2 evaluated functions and the resulting convolution. Have the line types solid, solid, and dotted, respectively.



DATAPLOT Program

```
LET FUNCTION F1 = ((X+1)**2)/((X**2)+1)/2
LET FUNCTION F2 = 2**((-(X+1.3/2)**2)/(A**2))-2**((-(X-1.3/2)**2)/(A**2))
LET A = .85
.
LET XMIN = -7
LET XINC = .1
LET XMAX = 7
LET X = SEQUENCE XMIN XINC XMAX
.
LET Y1 = F1
LET Y2 = F2
LET Y3 = CONVOLUTION Y1 Y2
LET Y3 = Y3/XINC
.
LET X3MIN = 2*XMIN
LET X3MAX = 2*XMAX
LET X3 = SEQUENCE X3MIN XINC X3MAX
.
LINES SOLID SOLID DOTTED
PLOT Y1 Y2 VS X AND
PLOT Y3 VS X3
```

Program Description

The general strategy here is to define a sequence of values (in X) for which the 2 functions are to be evaluated, evaluate the 2 functions (that is, form variables Y1 and Y2), form a variable (Y which contains the convolution values, and then plot the 3 variables. Note that the following code is general in the sense that after functions F1 and F2 are defined (along with parameter A), and after the interval limits XMIN and XMAX are defined (along with the evaluation increment XINC), then the remaining lines of DATAPLOT are unchanged and may be used as is for other functions, intervals and increments.

<u>LET FUNCTION F1 = ((X+1)**2)/((X**2)+1)/2</u>
defines the function F1 and assigns to it the 23-character string ((X+1)**2)/((X**2)+1)/2

<u>LET FUNCTION F2 = 2**((-(X+1.3/2)**2)/(A**2))-2**((-(X-1.3/2)**2)/(A**2))</u>
defines the function F2 and assigns to it the 55-character string
2**((-(X+1.3/2)**2)/(A**2))-2**((-(X-1.3/2)**2)/(A**2))

<u>LET A = .85</u>
defines a parameter A and assigns to it the value .85. A will be used later on in the evaluation of the function F2. Note that A need not be defined prior to defining the function F2 (A was just a symbol in the function); the character string in F2 remains unchanged even after the LET A = .85 command. However, A must be defined prior to the evaluation of F2 (as will occur later on in the LET Y2 = F2 command).

<u>.</u>
is a null command—it visually separates chunks of DATAPLOT code.

<u>LET XMIN = -7</u>
defines a parameter XMIN and assigns to it the value -7. The rationale for defining XMIN is that it will specify the lower limit of the interval of interest over which the 2 functions will be evaluated.

LET XINC = .1
defines a parameter XINC and assigns to it the value .1. The rationale for defining XINC is that it will specify the increment at which the 2 functions will be evaluated.

LET XMAX = 7
defines a parameter XMAX and assigns to it the value 7. The rationale for defining XMAX is that it will specify the upper limit of the interval of interest over which the 2 functions will be evaluated.

LET X = SEQUENCE XMIN XINC XMAX
defines a variable X and places in that variable the values -7, -6.9, -6.8, ..., 6.8, 6.9, 7. This makes use of the SEQUENCE sub-capability under the LET command.

Note that the above 4 commands--

```
LET XMIN = -7
LET XINC = .1
LET XMAX = 7
LET X = XMIN XINC XMAX
```

could have been here replaced by 1 command, namely,

```
LET X = SEQUENCE -7 .1 7
```

We have, for this example, used the previous form for 2 reasons--

1) to illustrate the general DATAPLOT feature that parameters may replace numbers in most DATAPLOT command statements;

2) to promote the use of standardized coding in which particulars to a problem are defined "up front" and then referred to in unchanging chunks of standardized code thereafter.

The rationale for defining the X variable at all is that this variable defines the points at which the 2 functions of interest will be evaluated.

LET Y1 = F1
defines a variable Y1. DATAPLOT recognizes F1 as a function and inserts the full character string in its place for the formation of the variable Y1. Y1 has the same number of elements as X. An element of Y1 is formed by taking an individual element in X and carrying out the specified function evaluation. The command

```
LET Y1 = F1
```

is exactly equivalent to

```
LET Y1 = ((X+1)**2)/((X**2)+1)/2
```

LET Y2 = F2
defines a variable Y2. DATAPLOT recognizes F2 as a function and inserts the full character string in its place for the formation of the variable Y2. Y2 has the same number of elements as X. An element of Y2 is formed by taking an individual element in X and carrying out the specified function evaluation. The command

```
LET Y2 = F2
```

is exactly equivalent to

```
LET Y2 = 2**((-(X+1.3/2)**2)/(A**2))-2**((-(X-1.3/2)**2)/(A**2))
```

Note that the value of A must be pre-defined before this function evaluation can take place--it was done above via the LET A = .85 statement.

LET Y3 = CONVOLUTION Y1 Y2
defines a variable Y3 and places in that variable the values resulting from the numerical convolution of the values in Y1 and Y2. The CONVOLUTION sub-capability is but one of a wide variety of sub-capabilities under the LET command. It is a general property of numerical convolutions that the output variable (Y3) typically has a broader domain than either of the 2 functions being convolved. The net effect is that the output variable Y3 does not have the same (it has considerably more) number of elements as the input variables Y1 and Y2.

In regard to the question of what X values correspond to the computed Y3 values, this will be dealt with below in the formation of the X3 variable.

Note also that the CONVOLUTION sub-capability implicitely assumes a unit increment in the X values (which we know is not true in this example and is rarely true in general). The net result is that the values in Y3 will be off by a scale factor. This scale factor adjustment is straightforward and is done in the LET Y3 = Y3/XINC statement below.

LET Y3 = Y3/XINC
redefines the variable Y3. An individaul element in Y3 is redefined by taking an old element in Y3 and dividing it by the value of XINC (= .1 in this example). This adjustment of the Y3 values is done so as to effect the proper scaling of the convolution output due to non-unity increments in the evaluated functions.

*<u>LET X3MIN = 2*XMIN</u>*
defines a parameter X3MIN and assigns to it the value obtained by taking the value in the parameter XMIN and multiplying it by 2. The rationale for defining X3MIN is that it will specify the lower limit of the interval of interest over which the convolution function has been evaluated; in other words, it will be the x value corresponding to the first element in the Y3 variable. In this example, X3MIN will have the value -14.

*<u>LET X3MAX = 2*XMAX</u>*
defines a parameter X3MAX and assigns to it the value obtained by taking the value in the parameter XMAX and multiplying it by 2. The rationale for defining X3MAX is that it will specify the upper limit of the interval of interest over which the convolution function has been evaluated; in other words, it will be the x value corresponding to the last element in the Y3 variable. In this example, X3MAX will have the value 14.

<u>LET X3 = SEQUENCE X3MIN XINC X3MAX</u>
defines a variable X3 and places in that variable the values -14, -13.9, -13.8, ..., 13.8, 13.9, 14. This makes use of the SEQUENCE sub-capability under the LET command. With the execution of this command statement, we now have a sequence of x values that corresponds exactly with the values in the Y3 variable. Note that Y3 and X3 will, by construction, have the same number of elements (although Y3 was not involved directly in the definition of X3). With the construction of X3, we now have a meaningful variable against which Y3 may be plotted.

<u>LINES SOLID SOLID DASHED</u>
specifies (for future plots) that the first trace wll have solid connecting lines between plot points, the second trace will have solid connecting lines between plot points, and the third trace will have dotted connecting lines between plot points.

<u>PLOT Y1 Y2 VERSUS X AND</u>
<u>PLOT Y3 X</u> generates the desired plot with 3 traces. The first trace is a plot of variable Y1 (vertically) versus variable X (horizontally); The second trace is a plot of variable Y2 (vertically) versus variable X (horizontally); The third trace is a plot of variable Y3 (vertically) versus variable X3 (horizontally). The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type (for all 3 traces) is blank; due to the prior LINES command, the first 2 traces will have solid connecting lines and the third trace will have dotted connecting lines.

The AND suffix at the end of PLOT Y1 Y2 VERSUS X AND is important-- it tells DATAPLOT to generate only 1 plot, but on that 1 plot to have 3 traces, as opposed to having 2 plots (the first with 2 traces and the second with 1 trace). When the AND is included, the following occurs--

1) the usual screen pre-erase occurs;

2) the trace of Y1 versus X is generated;

3) the trace of Y2 versus X is superimposed;

4) the trace of Y3 versus X3 is superimposed.

If the AND were omitted, then the following would occur-

1) the usual screen pre-erase occurs for the first plot;

2) the trace of Y1 vesus X is generated;

3) the trace of Y2 vesus X is superimposed;

4) the usual screen pre-erase occurs for the second plot (thereby wiping out the first plot);

5) the trace of Y3 vesus X3 is generated.

Note that the AND suffix may be used to link any form of the DATAPLOT PLOT (and 3D-PLOT) command. In this example, we have linked a "VERSUS" form with a "2 argument" form. Other forms include the "3 argument" form and the "function" form. Any or all of these basic forms may be linked via the AND suffix, if the analyst so desires.

Note also that the PLOT command has the flexibility to allow the generation of the above 3 traces in a variety of other ways, for example, the identical plot would result from any of the following--

```
PLOT Y1 VERSUS X AND
PLOT Y2 VERSUS X AND
PLOT Y3 VERSUS X3

PLOT Y1 X AND
PLOT Y2 X AND
PLOT Y3 X3

PLOT Y1 Y2 VERSUS X Y3 VERSUS X3

PLOT Y1 VERSUS X Y2 VERSUS X Y3 VERSUS X3
```

Note also that VERSUS may be uniformly abbreviated to VS or VS. in any of the above statements.

Generate a Differential Equation Phase Diagram

Problem

Generate a phase diagram for the differential equation associated with the simple pendulum. The phase diagram (= phase portrait) is a graphical technique for displaying the nature of the solution for a differential equation. The phase diagram of a differential equation y'' = f(y',y) or y' = f(y) is a plot of y' (vertically) versus y horizontally. The individual phase paths (= trajectories = integral curves) represent solution contours for the differential equation. For the simple pendumlum, the solution contours are given by the function

```
     +sqrt(0.5*cos(2*pi*t + c)
```

and

```
     -sqrt(0.5*cos(2*pi*t + c)
```

Different values of c represent different contours. Plot contours for c = .5, .6, 1, and 4. Plot the contours over the range t = 0 (.1) 3.

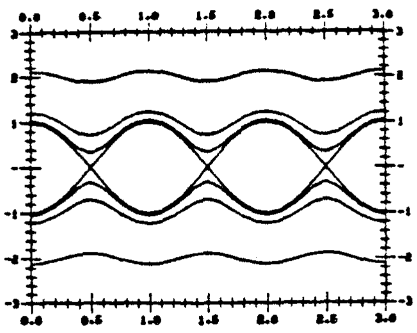
DATAPLOT Program

Two different programs are presented to generate the plot. Program 1 is

```
     LET FUNCTION F = 0.5*COS(2*PI*T)
     PLOT  SQRT(F+0.5) FOR T = 0 .1 3 AND
     PLOT -SQRT(F+0.5) FOR T = 0 .1 3 AND
     PLOT  SQRT(F+0.6) FOR T = 0 .1 3 AND
     PLOT -SQRT(F+0.6) FOR T = 0 .1 3 AND
     PLOT  SQRT(F+1.0) FOR T = 0 .1 3 AND
     PLOT -SQRT(F+1.0) FOR T = 0 .1 3 AND
     PLOT  SQRT(F+4.0) FOR T = 0 .1 3 AND
     PLOT -SQRT(F+4.0) FOR T = 0 .1 3
```

Program 2 is

```
     LET FUNCTION F = 0.5*COS(2*PI*T)
     .
     SERIAL READ C
     .5 .5 .6 .6 1 1 8 8
     END OF READ
     .
     LOOP FOR K = 1 1 8
     LET CONST=C(K)
     PLOT ((-1)**K) * SIN(F+CONST) FOR T = 0 .1 3 AND
     END OF LOOP
     PLOT 0 FOR T = 0 0 0
```



Program Description

Program 1 is straightforward—a "brute force" approach.

*<u>LET FUNCTION F = 0.5*COS(2*PI*T)</u>*
*in program 1 defines the function F and assigns to it the 15-character string 0.5*COS(2*PI*T)*

```
PLOT  SQRT(F+0.5) FOR T = 0 .1 3 AND
PLOT -SQRT(F+0.5) FOR T = 0 .1 3 AND
PLOT  SQRT(F+0.6) FOR T = 0 .1 3 AND
PLOT -SQRT(F+0.6) FOR T = 0 .1 3 AND
PLOT  SQRT(F+1.0) FOR T = 0 .1 3 AND
PLOT -SQRT(F+1.0) FOR T = 0 .1 3 AND
PLOT  SQRT(F+4.0) FOR T = 0 .1 3 AND
PLOT -SQRT(F+4.0) FOR T = 0 .1 3
```
generates the desired plot with 8 traces.

DATAPLOT recognizes F in each PLOT statement as a function and inserts the full character string in its place for the generation of the trace. Note how the function F may be used in the PLOT statement as part of yet a more complicated function; F is here being used as a "kernal", and the various contours are being generated by adjusting the sign and the constant c about the kernal. Each function will be evaluated at a series of T values starting with 0, at increments of .1, and ending at 3 (that is, at X = 0, .1, .2, ..., 2.8, 2.9, 3). Note that the T values for each of the function evaluations are temporary and local to the PLOT statement only; no T variable as such is created, nor does such a PLOT statement affect a T variable which the analyst may happen to have.

The AND suffix at the end of all (but the last) PLOT statement is important- it tells DATAPLOT to generate only 1 plot, but on that 1 plot to have 8 superimposed traces, as opposed to having 8 separate plots, each with only 1 trace.

Note that the functions includes a reference to PI— this is an automatically-provided DATAPLOT parameter with the value 3.1415926. It may be used like any other DATAPLOT parameter.

The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type (for both traces) is blank; the default line type (for all traces) is solid.

Program 2 is a more elegant way of generating the same plot. The advantage of program 1 is that it is straighforward. The disadvantage of program 1 is that if the analyst wanted 20 contour lines instead of 8, then the number of lines of code would increase accordingly. The advantage of program 2 is that the number of lines of code remains the same regardless of how many contour lines are desired; the disadvantage of program 2 is that is a bit more elegant, less straightforward, makes use of more advanced DATAPLOT features (e.g, the LOOP command), and is slightly more difficult to follow.

```
        LET FUNCTION F = 0.5*COS(2*PI*T)
        .
        SERIAL READ C
        .5 .5 .6 .6 1 1 8 8
        END OF READ
        .
        LOOP FOR K = 1 1 8
        LET CONST=C(K)
        PLOT ((-1)**K) * SIN(F+CONST) FOR T = 0 .1 3 AND
        END OF LOOP
        PLOT 0 FOR T = 0 0 0
```

*LET FUNCTION F = 0.5*COS(2*PI*T)*
*in program 2 (as in program 1) defines the function F and assigns to it the 15-character string 0.5*COS(2*PI*T)*

.
is a null command—it visually separates chunks of DATAPLOT code.

SERIAL READ C
initiates a read and defines a variable C. The first word after SERIAL READ is C; since C does not have a period following it, then this indicates to DATAPLOT that the data to be read in will be supplied directly from the terminal (if this program is being entered interactively), or will be directly in the program (if this program is being pre-stored and run all at once). In any event, the lack of a period after C tells DATAPLOT that C is not the name of a mass storage file/subfile, but rather is the name of the variable being created when the data is read in. The word SERIAL in SERIAL READ indicates that the data will be read into the variable C in a sequential fashion proceeding across each data line image—the first number found on the first line will be read into the first element of C, the second number (iif existent) on the first line will be read into the second element of C, and so forth. The read is format-free; successive data values on a line image need not be in certain columns; they need only be separated by at least one blank. The ultimate number of observations in C need not be pre-defined—it is dictated by the data set.

.5 .5 .6 .6 1 1 8 8
is a data line image. These 8 numbers will be read into the first 8 elements of the variable C. These 8 values will ultimately be used in froming the 8 traces on the final plot.

END OF READ
terminates the read. The variable C thus ends up with 8 values. DATAPLOT will provide summary feedback information as to what was read in so that the analyst may be assured that the read went as expected. A synonym to END OF DATA is END DATA.

LOOP FOR K = 1 1 8
initiates a loop in DATAPLOT. The variable K is defined as a parameter in the usual DATAPLOT sense, and so may be used (and even changed!) within the loop. K will take on the values 1, at increments of 1, and ending with 8; that is, 1, 2, 3, ..., 7, 8. The end of the loop is specified by and END OF LOOP (or END LOOP) command. DATAPLOT handles loops by first storing the entire loop, and then executing the loop the desied number of times. Thus the loop will not, in fact, be executed until the END OF LOOP command has been entered. Loops may be nested 6 deep, though it is rare in practice for the nesting to go further than 2 deep.

LET CONST=C(K)
defines a parameter CONST and assigns to it the k-th element in the variable C. The parameter CONST is thus seen to change each time through the loop. Note the ability to reference individual elements in a variable and "copy" them out into parameters—this provides flexibility of referencing for the analyst.

*PLOT ((-1)**K) * SIN(F+CONST) FOR T = 0 .1 3 AND*
*generates the desired 8 traces. Note how CONST (which changes each time through the loop) is imbedded in the function. Note also how the loop variable K is being used in ((-1)**K) so as to generate the + contour and the complementary - contour. Finally, note the AND suffix at the end of each PLOT statement— as in program 1, this allows the analyst to string together PLOT statements with the net effect of generating 1 plot with 8 traces, as opposed to 8 plots with 1 trace each.*

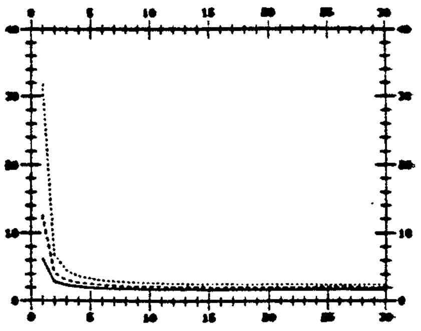
END OF LOOP
terminates the loop. When this statement is encountered, DATAPLOT transfers program control to the first statement after the LOOP FOR K = 1 1 8. In DATAPLOT, the loop parameter (K in this case) is compared to the loop limits before the loop is executed. In this example, the loop will be passed through 8 times. A synonym for END OF LOOP is END LOOP.

PLOT 0 FOR X = 0 0 0
generates a dummy trace and will result in a point appearing on the plot at x = 0 and y = 0. Formally, the statement PLOT 0 FOR X = 0 0 0 tells DATAPLOT to plot the function f(x) = 0 over the x domain starting at 0, at increments of 0, and ending at 0. Plotting such a constant function (f(x) = 0) is a special case of the general capability of plotting functions. The net result on the plot will be a "line" drawn from (x = 0, y = 0) to (x = 0, y = 0)--that is from the start point to itself. Since the starting point and ending point of this line are the same, the line will appear as a point--and will thus be virtually unnoticeable. Why then are we plotting a point whose sole purpose is to be unnoticeable? The answer lies in the PLOT statement within the loop. Note in program 1 how the final PLOT did not have an AND suffix whereas the other PLOT statements did. This indicated to DATAPLOT that the eighth trace was in fact the final trace. However, note in program 2 how the PLOT statement in the loop does have an AND suffix, regardless of whether it is trace 1, trace 2, or trace 8. To finish off the plot, it is necessary to indicate to DATAPLOT that the final trace has been drawn-- this is done by including a PLOT statement with no AND suffix. Such a PLOT statement here is the "junk" trace PLOT 0 FOR X = 0 0 0. The net result is that the plot will be properly finished off by this ninth PLOT command, and yet this ninth trace will in fact add nothing to the plot.

Generate a Multi-Trace Plot of Percent Point Functions

Problem

Compute and generate the 95 percent points for the members of the t distribution family for degrees of freedom parameter nu = 1 (1) 30. Do similarly for the 97.5 and 99 percent points. Plot the 3 functions versus nu. Have the 3 traces solid, dashed, and dotted.



DATAPLOT Program

```
LET NU = SEQUENCE 1 1 30
LET Y1 = TPPF(.95,NU)
LET Y2 = TPPF(.975,NU)
LET Y3 = TPPF(.99,NU)
.
LINES SOLID DASHED DOTTED
PLOT Y1 Y2 Y3 VS NU
```

Program Description

The general strategy here is to define a sequence of values (in NU) for which the 3 percent point functions are to be evaluated, evaluate the 3 functions (that is, form variables Y1, Y2, and Y3) making use of the built-in DATAPLOT library function TPPF, and then plot the 3 variables.

LET NU = SEQUENCE 1 1 30
defines a variable NU and places in that variable the values 1, 2, 3, ..., 28, 29, 30. This makes use of the SEQUENCE sub-capability under the LET command.

LET Y1 = TPPF(.95,NU)
defines a variable Y1. Y1 has the same number of elements as NU. An element in Y1 is formed by taking an individual element in NU, accessing the built-in DATAPLOT library function TPPF to compute the percent point of the t distribution for the specified probability value (.95) and for the specified degrees of freedom (NU). The variable NU will thus result in the 95% point of the t distribution for the NU values 1, 2, 3, ..., 28, 29, 30. Since NU is a variable, then Y1 is a variable; if NU were a parameter (or number), then Y1 would be a parameter. DATAPLOT has a wide assortment of built-in intrinsic library functions which may be called on in any PLOT, 3D-PLOT, FIT, PRE-FIT, or LET statements. Other available probability-type functions include

normal	probability density function	NORPDF(X)
normal	cumulative distribution function	NORCDF(X)
normal	percent point function	NORPPF(P)
t	probability density function	TPDF(X,NU)
t	cumulative distribution function	TCDF(X,NU)
t	percent point function	TPPF(P,NU)
chi-squared	probability density function	CHSPDF(X,NU)
chi-squared	cumulative distribution function	CHSCDF(X,NU)
chi-squared	percent point function	CHSPPF(P,NU)
F	probability density function	FPDF(X,NU1,NU2)
F	cumulative distribution function	FCDF(X,NU1,NU2)
F	percent point function	FPPF(P,NU1,NU2)

LET Y2 = TPPF(.975,NU)
defines a variable Y2. Y2 has the same number of elements as NU. An element in Y2 is formed by taking an individual element in NU, accessing the built-in DATAPLOT library function TPPF to compute the percent point of the t distribution for the specified probability value (.975) and for the specified degrees of freedom (NU). The variable NU will thus result in the 97.5% point of the t distribution for the NU values 1, 2, 3, ..., 28, 29, 30.

LET Y3 = TPPF(.99,NU)
defines a variable Y3. Y3 has the same number of elements as NU. An element in Y3 is formed by taking an individual element in NU, accessing the built-in DATAPLOT library function TPPF to compute the percent point of the t distribution for the specified probability value (.99) and for the specified degrees of freedom (NU). The variable NU will thus result in the 99% point of the t distribution for the NU values 1, 2, 3, ..., 28, 29, 30.

.
is a null command--it visually separates chunks of DATAPLOT code.

LINES SOLID DASHED DOTTED
specifies (for future plots) that the first trace wll have solid connecting lines between plot points, the second trace will have dashed connecting lines between plot points, and the third trace will have dotted connecting lines between plot points. Note that an alternate form for the SOLID specification is SO; an alternate form for the DASHED specification is DASH or DA, and an alternate form for DOTTED specification is DOT or DO, as in

```
LINES SOLID DASH DOT
```

or

```
LINES SO DA DO
```

PLOT Y1 Y2 Y3 VERSUS X
generates the desired plot with 3 traces. The first trace is a plot of variable Y1 (vertically) versus variable X (horizontally); The second trace is a plot of variable Y2 (vertically) versus variable X (horizontally); The third trace is a plot of variable Y3 (vertically) versus variable X (horizontally). The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type (for all 3 traces) is blank; due to the prior LINES command, the first trace will have solid connecting lines; the second trace will haved dashed connecting lines; and the third trace will have dotted connecting lines.

Generate a 3-Dimensional Data Plot

Problem

(x,y,z) coordinates exist on a data file ABC as a series of values (3 values per line image). These data points may define any general 3-dimensional entity—a line in space, a plane, a surface, a figure, etc. The figure for this example is a box. The data file consisted of the following 18 line images—

```
0 0 0
0 0 1
1 0 1
1 0 0
0 0 0
0 1 0
0 1 1
0 0 1
0 0 0
1 0 0
1 1 0
1 1 1
1 0 1
1 1 1
0 1 1
0 1 0
1 1 0
END OF DATA
```

Read in the data; place it in variables X, Y, and Z.

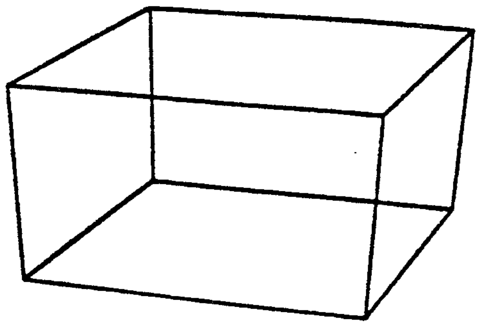
Generate a 3-dimensional plot. Position the eye at (x = 2, y = 5, z = 3). Omit the reference axes (= frame).

DATAPLOT Program

```
READ ABC. X Y Z
FRAME OFF
EYE COORDINATES 2 5 3
3D-PLOT Z X Y
```

Program Description

READ ABC. X Y Z
carries out a format-free read of data from file ABC. The data is read into variables X, Y, and Z; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y; the third number on each line image is read into the variable Z. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).



FRAME OFF
turns off (for future plots) the usual 4-sided frame that would appear on 2-dimensional plots, and the usual 3-prong axis that would appear on 3-dimensional plots. The rationale for turning the frame off is that we here desire to have the 3-dimensional figure "hang in space" with no reference to a coordinate system.

EYE COORDINATES 2 5 3
specifies (for future 3-dimensional plots) the position of the analyst's eye in viewing the 3-dimensional figure. The eye coordinates defined here are (x = 2,y = 5,z = 3). Such specification requires some prior knowledge on the part of the analyst as to what the magnitude of the units are which specify the figure to be plotted. In this case, we know that each of the variables X, Y, and Z range between 0 to 1, and so, as a first pass we here specify the eye coordinates to be, say, x = 2, y = 5, and z = 3. Since the appearance of the 3-d plot is so strongly dependent on the analyst's viewing position, the eye coordinates specification must always be made before the 3-dimensional plot is actually generated. If the eye position is not specified, then DATAPLOT will automatically compute an eye position based on the limits of the data; in particular, the eye coordinates (xeye,yeye,zeye) will be

```
xeye = xmax + 3*(xmax - xmin)
yeye = ymax + 3*(ymax - ymin)
zeye = zmax + 3*(zmax - zmin)
```

where

xmin is the minimum of the X variable
xmax is the maximum of the X variable

ymin is the minimum of the Y variable
ymax is the maximum of the Y variable

zmin is the minimum of the Z variable
zmax is the maximum of the Z variable

3D-PLOT Z X Y
generates a 3-dimensional plot of the variable Z (vertically) versus the the variable X (as the first axis in the horizontal plane), and the variable Y (as the second axis in the horizontal plane). The variable Y (as the second horizontal axis). The figure will be formed in exactly the order in which the data resides in the variables X, Y, and Z. Note that the 3-argument form for the 3D-PLOT command is a strict generalization of the 2-argument form for the PLOT command. The first argument is always the vertical axis variable; this makes good sense in a data analysis context where the most important variable is the "response" variable which is the one which invariably appears on the vertical axis.

The default plot type is continuous; the default character type is blank; the default line type is solid.

Synonym commands for 3-D-PLOT are 3-D PLOT, 3DPLOT, and 3DPLOT.

Generate 3-Dimensional Multiple Data Traces

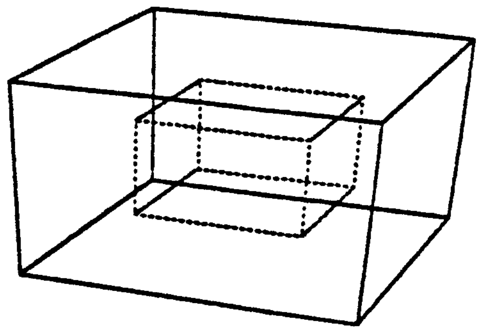
Problem

Coordinate data exists on a file ABC as (x,y,z) triples along with a fourth variable—a tag variable—which identifies the "trace" that a given (x,y,z) triple belongs to. A trace may define any general 3-dimensional entity—a line, a plane, a surface, a figure, etc. In this example, each "trace" is a figure (a box), and the data file contains information regarding 2 such figures. The data file consisted of the following 35 line images--

```
0 0 0           1
0 0 1           1
1 0 1           1
1 0 0           1
0 0 0           1
0 1 0           1
0 1 1           1
0 0 1           1
0 0 0           1
1 0 0           1
1 1 0           1
1 1 1           1
1 0 1           1
1 1 1           1
0 1 1           1
0 1 0           1
1 1 0           1
 .25  .25  .25  2
 .25  .25  .75  2
 .75  .25  .75  2
 .75  .25  .25  2
 .25  .25  .25  2
 .25  .75  .25  2
 .25  .75  .75  2
 .25  .25  .75  2
 .25  .25  .25  2
 .75  .25  .25  2
 .75  .75  .25  2
 .75  .75  .75  2
 .75  .25  .75  2
 .75  .75  .75  2
 .25  .75  .75  2
 .25  .75  .25  2
 .75  .75  .25  2
END OF DATA
```

Read in the data; place the data in variables X, Y, Z, and TAG.

Generate a 3-dimensional plot. The plot will consist of 2 "traces" (boxes)—one within the other. Have the first trace solid and the second trace dotted— the outer box will be solid and the inner box will be dotted. Omit the reference axes (= frame). Position the eye at (x = 2, y = 5, z = 3).



DATAPLOT Program

```
READ ABC. X Y Z TAG
LINES SOLID DOTTED
FRAME OFF
EYE COORDINATES 2 5 3
3-D PLOT Z X Y TAG
```

Program Description

<u>READ ABC. X Y Z TAG</u>
carries out a format-free read of data from file ABC. The data is read into variables X, Y, Z, and TAG; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y; the third number on each line image is read into the variable Z; the fourth number on each line image is read into the variable TAG. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

<u>LINES SOLID DOTTED</u>
specifies (for future plots) that the first trace will have solid connecting lines between plot points, and the second trace will have dashed connecting lines between plot points.

<u>FRAME OFF</u>
turns off (for future plots) the usual 4-sided frame lines that would appear on 2-dimensional plots, and the usual 3-prong axis that would appear on 3-dimensional plots. The rationale for turning the frame off is that we here desire to have the 3-dimensional figure "hang in space" with no reference to a coordinate system.

EYE COORDINATES 2 5 3
specifies (for future 3-dimensional plots) the position of the analyst's eye in viewing the 3-dimensional figure. The eye coordinates defined here are (x = 2,y = 5,z = 3). Such specification requires some prior knowledge on the part of the analyst as to what the magnitude of the units are which specify the figure to be plotted. In this case, we know that each of the variables X, Y, and Z range between 0 to 1, and so, as a first pass we here specify the eye coordinates to be, say, x = 2, y = 5, and z = 3. Since the appearance of the 3-d plot is so strongly dependent on the analyst's viewing position, the eye coordinates specification must always be made before the 3-dimensional plot is actually generated. If the eye position is not specified, then DATAPLOT will automatically compute an eye position based on the limits of the data; in particular, the eye coordinates (xeye,yeye,zeye) will be

```
xeye = xmax + 3*(xmax - xmin)
yeye = ymax + 3*(ymax - ymin)
zeye = zmax + 3*(zmax - zmin)
```

where

```
xmin is the minimum of the X variable
xmax is the maximum of the X variable

ymin is the minimum of the Y variable
ymax is the maximum of the Y variable

zmin is the minimum of the Z variable
zmax is the maximum of the Z variable
```

3D-PLOT Z X Y TAG
generates the multi-trace 3-dimensional plot; 2 traces are generated. Each trace consists of a plot of the Z variable (vertically) versus the variable X (as the first axis in the horizontal plane), and versus the variable Y (as the second axis in the horizontal plane). however, each trace is restricted to only a subset of Z, X and Y values. The subset is defined via the contents of the TAG variable. For the first trace, the following procedure is executed--

1) *the entire TAG variable is scanned and the minimum value of TAG is identified (in this case, the minimum value of TAG is 1);*

2) *all entries in the Z, X, and Y variables which correspond to values in the TAG variable of 1 are extracted;*

3) *this extracted subset of Z, X, and Y values is plotted;*

4) *the line type for this first trace will be dictated by the first entry of a previous LINES command-- in this case, the first entry was SOLID, and so a solid line will result; Similarly, the character type for the first trace will be dictated by the first entry of a previous CHARACTERS command-- since the CHARACTERS command was not previously entered, the default will be used, namely to have blank plot characters.*

For the second trace, the following procedure is executed--

1) *the entire TAG variable is scanned and the next larger value of TAG is identified (in this case, the next largerr value of TAG is 2);*

2) *all entries in the Z, X, and Y variables which correspond to values in the TAG variable of 2 are extracted;*

3) *this extracted subset of Z, X, and Y values is plotted;*

4) *the line type for this second trace will be dictated by the second entry of a previous LINES command-- in this case, the second entry was DOTTED, and so a dotted line will result; Similarly, the character type for the second trace will be dictated by the second entry of a previous CHARACTERS command-- since the CHARACTERS command was not previously entered, the default will be used, namely to have blank plot characters.*

This procedure of using successively larger values of the TAG variable to define traces within the plot would continue until the largest value in TAG was noted and used. For this data set, the TAG variable only had 2 distinct values (1 and 2) and so only 2 traces resulted.

Note that it is the relative size of the elements in the TAG variable which is important in defining the traces, not the absolute size. Thus the identical plot would form if the elements of TAG were transformed by any monotonic increasing trasnformation. If the 1's and 2's in TAG became 100's and 500's, or became 0.4's and 23.7's, or became -1's and +15's, etc., the identical plot would form. In general, the TAG variable need not have sequential integer values, nor need it even have integer values. The important point is that however the TAG variable is defined, the smallest value of TAG corresponds to the first trace to be plotted, the next larger value corresponds to the second trace to be plotted, and so forth.

In this example, we have used the name TAG to represent the name of the variable that is the fourth argument in the PLOT command; the choice of TAG was by personal preference--any name may have been used.

Within a given trace, the figure will be formed in exactly the order in which the data resides in the variables X, Y, and Z.

Note that the 4-argument form for the 3D-PLOT command is a strict generalization of the 3-argument form for the PLOT command. The first argument is always the vertical axis variable; the middle argument(s) are always the horizontal plane variables, and the last argument is always the trace-identifier variable. Although not demonstrated here, each of the various forms of the 2-dimensional PLOT command generalizes to corresponding for thems for 3-dimension 3D-PLOT command (including the use of the AND suffix).

The default plot type is continuous; the default character type is blank; due to the prior LINES command, the first trace will have solid connecting lines, and the second trace will haved dotted connecting lines.

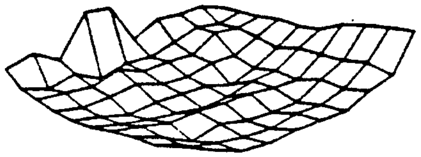
Synonym commands for 3-D-PLOT are 3-D PLOT, 3DPLOT, and 3DPLOT.

Generate a 3-Dimensional Data Surface

Problem

A response variable Z has been measured at equi-spaced values of 2 independent variables X and Y. Z, X, and Y values have been collected on a file ABC with 3 numbers per line. Read in the data; place the data in variables Z, X, and Y.

Examine the nature of the response surface by generating a cross-hatch 3-dimensional plot of the surface. Omit the reference axes (= frame). Position the eye at (x = 20, y = 30 z = 53). See figure 39.



DATAPLOT Program

```
READ ABC. Z X Y
FRAME OFF
EYE COORDINATES 20 30 53
3D-PLOT Z X Y X AND
3D-PLOT Z X Y Y
```

Program Description

READ ABC. Z X Y
carries out a format-free read of data from file ABC. The data is read into variables Z, X, and Y; the first number on each line image of the file is read into the variable Z; the second number on each line is read into the variable X; the third number on each line image is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

FRAME OFF

turns off (for future plots) the usual 4-sided frame lines that would appear on 2-dimensional plots, and the usual 3-prong axis that would appear on 3-dimensional plots. The rationale for turning the frame off is that we here desire to have the 3-dimensional figure "hang in space" with no reference to a coordinate system.

EYE COORDINATES 20 30 53
specifies (for future 3-dimensional plots) the position of the analyst's eye in viewing the 3-dimensional figure. The eye coordinates defined here are (x = 20,y = 30,z = 53). Such specification requires some prior knowledge on the part of the analyst as to what the magnitude of the (x,y,z) units are for the figurers to be plotted. Since the appearance of the 3-d plot is so strongly dependent on the analyst's viewing position, the eye coordinates specification must always be made before the 3-dimensional plot is actually generated. If the eye position is not specified, then DATAPLOT will automatically compute a default eye position based on the minima and maxima of the X, Y, and Z variables.

3D-PLOT Z X Y X AND
3D-PLOT Z X Y Y
generates the desired multi-trace, cross-hatched, 3-dimensional plot. Consider the general 4-argument form for the 3D-PLOT command--

```
3D-PLOT Z X Y TAG
```

for some trace-identifier variable TAG. This would generate a 3-dimensional plot of the Z variable (vertically) versus the variable X (as the first axis in the horizontal plane), and versus the variable Y (as the second axis in the horizontal plane)-- but each trace would consist only of a subset of Z, X, and Y values, where the subset would be defined by each distinct value in the TAG variable. Thus if the TAG variable had 10 distinct values, then 10 traces would result.

Note that any variable may be used for the fourth argument in the 3D-PLOT command; usually it is not the variables directly involved in the plot (Z, X, or Y), but it can be. For the problem at hand, the use of X or Y as the fourth argument is precisely what is needed; thus if the variable X has 10 distinct values (as it does in this example), then

```
3D-PLOT Z X Y X
```

will generate 10 traces (one trace per distinct X value), and these 10 traces are the cross-hatch lines for the X direction. Similarly, if the variable Y has 10 distinct values (as it does in this example), then

```
3D-PLOT Z X Y Y
```

will also generate 10 traces (one trace per distinct Y value), and these 10 traces are the cross-hatch lines in the Y direction.

Thus

```
3D-PLOT Z X Y X
```

alone would generate a plot with 10 distinct traces in one direction; and

```
3D-PLOT Z X Y Y
```

alone would generate a plot with 10 distinct traces in the perpendicular direction.

The combination of the X-direction traces and the Y-direction traces will give us the desired full cross-hatching. To superimpose the 10 traces from 3D-PLOT Z X Y X and the 10 traces from 3D-PLOT Z X Y Y onto one and the same plot, the usual DATAPLOT feature of linking plots via the AND suffix is used, as in

```
3D-PLOT Z X Y X AND
3D-PLOT Z X Y Y
```

The desired 20-trace cross-hatched, 3-dimensional plot will result. It is seen, therefore, that the generation of a cross-hatched, 3-dimensional surface plot is simply a special case of the 4-argument 3D-PLOT command, in combination with the AND suffix for linking plots together.

The default plot type is continuous; the default character type (for all 20 traces) is blank; the default line type (for all 20 traces) is solid.

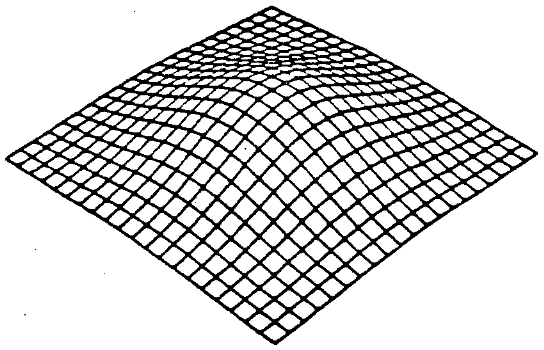
Generate a 3-dimensional Function Plot

Problem

Generate the bivariate normal N(0,0,1,1,0) density function. This function has the form

```
f(x,y) = (1/2*pi) * exp(-0.5*((x**2) + (y**2))
```

Position the eye at (x = 10, y = 10, z = 11). Omit the reference axes.



DATAPLOT Program

Two different programs are presented to generate this plot. Program 1 is

```
FRAME OFF
EYE COORDINATES 10 10 11
3D-PLOT (1/(2*PI))*EXP(-0.5*((X**2)+Y**2)) FOR X = -2 .2 2 FOR Y = -2 .2 2
```

Program 2 is

```
LET FUNCTION E = -0.5*((X**2)+(Y**2))
LET FUNCTION F = (1/(2*PI))*EXP(E)
FRAME OFF
EYE COORDINATES 10 10 11
3D-PLOT F FOR X = -2 .2 2 FOR Y = -2 .1 2
```

Program Description

FRAME OFF
in both programs turns off (for future plots) the usual 4-sided frame lines that would appear on 2-dimensional plots, and the usual 3-prong axis that would appear on 3-dimensional plots. The rationale for turning the frame off is that we here desire to have the 3-dimensional figure "hang in space" with no reference to a coordinate system.

EYE COORDINATES 10 10 11
in both programs specifies (for future 3-dimensional plots) the position of the analyst's eye in viewing the 3-dimensional figure. The eye coordinates specified here are (x = 10, Y = 10, z = 11). Such specification requires some prior knowledge on the part of the analyst as to what the magnitude of the (x,y,z) units are for the figurers to be plotted. Since the appearance of the 3-d plot is so strongly dependent on the analyst's viewing position, the eye coordinates specification must always be made before the 3-dimensional plot is actually generated. If the eye position is not specified, then DATAPLOT will automatically compute a default eye position based on the minima and maxima of the X, Y, and Z variables.

*3D-PLOT (1/(2*PI))*EXP(-0.5*((X**2)+Y**2)) FOR X = -2 .2 2 FOR Y = -2 .2 2*
in program 1 generates the desired plot. The function will be evaluated at a series of X values starting with -2, at increments of .2, and ending at 2 (that is, at X = -2, -1.8, -1.6, ..., 1.6, 1.8, 2), and for a series of Y values starting with -2, at increments of .2, and ending at 2 (that is, at Y = -2, -1.8, -1.6, ..., 1.6, 1.8, 2). Note that the evaluation is done in a dual fashion--for each given X value, all of the Y values are computed and plotted (thereby resulting in cross-hatch traces in one direction), and for each given Y value, all of the X values are computed and plotted (thereby resulting in cross-hatch traces in the other direction). Note that the X and Y values for the function evaluation are temporary and local to the PLOT statement only; no X or Y variable as such is created, nor does such a PLOT statement affect an X or Y variable which the analyst may happen to have.

Note that the function includes a reference to PI-- this is an automatically-provided DATAPLOT parameter with the value 3.1415926. It may be used like any other DATAPLOT parameter.

The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type (for all traces) is blank; the default line type (for all traces) is solid.

LET FUNCTION E = -0.5((X**2)+(Y**2))*
in program 2 defines the function E and assigns to it the 20-character string -0.5((X**2)+(Y**2))*

LET FUNCTION F = (1/(2*PI))*EXP(E)
defines the function F and assigns to it the 34-character string (1/(2*PI))*EXP(-0.5*((X**2)+Y**2)) Note how DATAPLOT recognizes E as a function and inserts the full character string in its place in the definition of the function F. The LET FUNCTION command thus allows not only the definition of individual functions, but the concatonation and step-wise composition of more complicated functions from simpler ones.

3D-PLOT F FOR X = -2 .2 2 FOR Y = -2 .2 2
in program 2 generates the desired plot. DATAPLOT recognizes F as a function and inserts the full character string in its place for the generation of the plot. This plot statement is thus exactly equivalent to the 3D-PLOT statement in program 1.

Generate a Boxed Title

Problem

Go to the middle of the screen and generate the centered string--

graphics

in upper case triplex italic. Surround it with a box.

DATAPLOT Program

```
FONT TRIPLEX ITALIC
CASE UPPER
JUSTIFICATION CENTER
HEIGHT 5
WIDTH 4
.
ERASE
MOVE 50 50
TEXT GRAPHICS
BOX 30 45 70 59
```

Program Description

The general procedure (with diagrammatic graphics) is to

1) *experiment with sizing and positioning of text, boxes, etc. by entering DATAPLOT commands interactively;*

2) *(after satisfied with sizing, positioning, etc.) write the program (via the local editor) out onto a file/subfile;*

3) *execute the program as a whole (via the DATAPLOT CALL command).*

<u>*FONT TRIPLEX ITALIC*</u>
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

<u>*CASE UPPER*</u>
specifies (for future text strings) that the case is upper (as opposed to lower). The CASE UPPER command is, in fact, unnecessary since this is the default. The CASE command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

<u>*JUSTIFICATION CENTER*</u>
specifies (for future text strings) that the output from the TEXT command be center-justified (as opposed to left-justified or right-justified). The default is left-justified. The JUSTIFICATION command applies only to output resulting from the TEXT command, since output resulting from the TITLE and LABEL commands is automatically center-justified, and output resulting from the LEGEND command is automatically left-justified.

<u>*HEIGHT 5*</u>
specifies (for future text strings) that the output from the TEXT command have a height equal to 5% of vertical sceen height. The HEIGHT command applies only to output resulting from the TEXT command. The size of output resulting from the TITLE, LABEL, and LEGEND commands are controlled via the

```
TITLE SIZE
...LABEL SIZE
LEGEND ... SIZE
```

commands, respectively, as in

```
TITLE SIZE 2
XLABEL SIZE 1.5
LEGEND 3 SIZE 2.5
```

For all cases (text, titles, labels, and legends), note that the character sizing for all 7 Hershey character fonts is controllable on a continuous scale, whereas for the default Tektronix font (which is hardware-generated), the character sizing is limited to only 4 sizes (on large-screen terminals such as the Tektronix 4014, 4054, and

4114), and even to just 1 size (on small-screen terminals such as the Tektronix 4010, 4027, and 4052). For all text output (resulting from TEXT, TITLE, LABEL, and LEGEND efault character height is 3 (= 3% of total commands, the default character size is 3 (= 3% of total screen height).

WIDTH 4
specifies (for future text strings) that the output from the TEXT command have a width equal to 4% of horizontal sceen width. The WIDTH command applies only to output resulting from the TEXT command. The width of output resulting from the TITLE, LABEL, and LEGEND commands is automatically set to 3/4 of the character height which in turn is controlled by the

```
TITLE SIZE
...LABEL SIZE
LEGEND ... SIZE
```

commands, respectively. For all cases (text, titles, labels, and legends), note that the character width for all 7 Hershey character fonts is controllable on a continuous scale, whereas for the default Tektronix font (which is hardware-generated), the character width is limited to only 4 sizes (on large-screen terminals such as the Tektronix 4014, 4054, and 4114), and even to just 1 size (on small-screen terminals such as the Tektronix 4010, 4027, and 4052). For all text output (resulting from TEXT, TITLE, LABEL, and LEGEND efault character height is 3 (= 3% of total commands, the default character width is 2 (= 2% of total screen width).

Note the existence of the HW command which is an abbreviated combination of both the HEIGHT and WIDTH commands, thus

```
HEIGHT 5
WIDTH 4
```

may be alternately specified as

```
HW 5 4
```

.
is a null command--it visually separates chunks of DATAPLOT code.

ERASE
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased.

MOVE 50 50
moves the cursor to a point 50% of the way across the screen and 50% of the way up the screen. MOVE 50 50 will thus place the cursor in the middle of the screen. Note that DATAPLOT has standardized screen coordinates-- it is simple, easy to use, and universal for all output devices. Rather than use a screen coordinates system based on English units, metric units, device picture points, etc. (which change from device to device), DATAPLOT uses a percentage coordinates system. To determine the x-cordinate for a point, simply ask "How far across (0 to 100%) the screen is the point? To determine the y-coordinates for a point, simply ask "How far (0 to 100%) up the screen is the point?" All coordinate values are 0 to 100%; decimal values such as 23.5, 56.27, and 34.693 are permitted). Thus

(0,0)	= lower left corner of screen
(100,0)	= lower right corner of screen
(0,100)	= upper left corner of screen
(100,100)	= upper right corner of screen
(50,50)	= middle of screen

TEXT GRAPHICS
generates the text string

```
graphics
```

on the screen. Due to the previous FONT, CASE, JUSTIFICATION, HEIGHT, and WIDTH commands, the text string will be in triplex italic, upper case, center-justified (at the current cursor position, namely (50,50)), with height = 5% of screen height and width = 4% of screen width.

BOX 30 45 70 59
draws a box with one corner at a point 30% of the way across the screen and 45% of the way up the screen, and with the opposing corner 70% of the way across the screen and 59% of the way up the screen. This box will surround the text string as desired.

Note that the easiest way to obtain the coordinate values for the BOX command is to make a pass at putting up the various text strings interactively (where we are using the screen as a scratch area with no concern about screen clutter), and then making liberal use of the cross-hair in the following fashion--

1) enter the CROSS-HAIR (synonym = CH) command which will cause the cross-hair to appear on the screen;

2) position the cross-hair via the thumbwheel or joystick;

3) hit any key which will then cause the cross-hair to disappear and will report (in DATAPLOT 0 to 100 units) the latest position of the cross-hair.

Note that in step 3 above, the coordinates will not be reported if the feedback switch has been turned off. Thus one could leave the feedback switch on, or preferably one could augment the CROSS-HAIR (synonym = with 2 parameter values, such as

```
      CH X1 Y1
```

which will cause (in step 3 above) for the coordinates values to be placed into the DATAPLOT parameters X1 Y1. These parameters may be printed at a later time, as in

```
      WRITE X1 Y1
```

or may be used in other DATAPLOT commands, as in

```
      MOVE X1 Y1
```

or (if X2 and Y2 are also parameters obtained in the same way)

```
      BOX X1 Y1 X2 Y2
```

The ability to store coordinates in parameters and refer to them symbolically thereafter is a powerful diagrammatic feature of DATAPLOT.

Generate a Mathematical Equation

Problem

Go to a point 20 percent of the way across the screen and 50 percent of the way up the screen and write the following equation in lower case triplex:

y = integral sin(alpha x)dx + exp(bx)

$$y = \int \sin(\alpha x)dx + e^{(bx)}$$

DATAPLOT Program

```
FONT TRIPLEX
CASE LOWER
JUSTIFICATION LEFT
HEIGHT 5
WIDTH 4
.
ERASE
MOVE 20 50
TEXT Y = INTE()SIN(ALPH()X)DX + ESUP()(BX)
```

Program Description

The general procedure (with diagrammatic graphics) is to

1) *experiment with sizing and positioning of text, boxes, etc. by entering DATAPLOT commands interactively;*

2) *(after satisfied with sizing, positioning, etc.) write the program (via the local editor) out onto a file/subfile;*

3) *execute the program as a whole (via the DATAPLOT CALL command).*

FONT TRIPLEX
specifies (for future text strings) that the font be triplex. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

The last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

CASE LOWER
specifies (for future text strings) that the case is lower (as opposed to upper). The default case is upper. The CASE command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

JUSTIFICATION LEFT
specifies (for future text strings) that the output from the TEXT command be left-justified (as opposed to center-justified or right-justified). Note that the JUSTIFICATION LEFT command is, in fact, unnecessary, since the default is left-justified. The JUSTIFICATION command applies only to output resulting from the TEXT command, since output resulting from the TITLE and LABEL commands is automatically center-justified, and output resulting from the LEGEND command is automatically left-justified.

HEIGHT 5
specifies (for future text strings) that the output from the TEXT command have a height equal to 5% of vertical sceen height. The HEIGHT command applies only to output resulting from the TEXT command. The size of output resulting from the TITLE, LABEL, and LEGEND commands are controlled via the

```
TITLE SIZE
...LABEL SIZE
LEGEND ... SIZE
```

commands, respectively, as in

```
TITLE SIZE 2
XLABEL SIZE 1.5
LEGEND 3 SIZE 2.5
```

For all cases (text, titles, labels, and legends), note that the character sizing for all 7 Hershey character fonts is controllable on a continuous scale, whereas for the default Tektronix font (which is hardware-generated), the character sizing is limited to only 4 sizes (on large-screen *terminals such as the Tektronix 4014, 4054, and 4114), and even to just 1 size (on small-screen terminals such as the Tektronix 4010, 4027, and 4052). For all text output (resulting from TEXT, TITLE, LABEL, and LEGEND efault character height is 3 (= 3% of total commands, the default character size is 3 (= 3% of total screen height).*

__WIDTH 4__
specifies (for future text strings) that the output from the TEXT command have a width equal to 4% of horizontal sceen width. The WIDTH command applies only to output resulting from the TEXT command. The width of output resulting from the TITLE, LABEL, and LEGEND commands is automatically set to 3/4 of the character height which in turn is controlled by the

```
TITLE SIZE
...LABEL SIZE
LEGEND ... SIZE
```

commands, respectively. For all cases (text, titles, labels, and legends), note that the character width for all 7 Hershey character fonts is controllable on a continuous scale, whereas for the default Tektronix font (which is hardware-generated), the character width is limited to only 4 sizes (on large-screen terminals such as the Tektronix 4014, 4054, and 4114), and even to just 1 size (on small-screen terminals such as the Tektronix 4010, 4027, and 4052). For all text output (resulting from TEXT, TITLE, LABEL, and LEGEND efault character height is 3 (= 3% of total commands, the default character width is 2 (= 2% of total screen width).

Note the existence of the HW command which is an abbreviated combination of both the HEIGHT and WIDTH commands, thus

```
HEIGHT 5
WIDTH 4
```

may be alternately specified as

```
HW 5 4
```

__.__
is a null command—it visually separates chunks of DATAPLOT code.

__ERASE__

erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased.

__MOVE 20 50__
moves the cursor to a point 20% of the way across the screen and 50% of the way up the screen. MOVE 20 50 will thus place the cursor in the middle of the screen. Note that DATAPLOT has standardized screen coordinates-- it is simple, easy to use, and universal for all output devices. Rather than use a screen coordinates system based on English units, metric units, device picture points, etc. (which change from device to device), DATAPLOT uses a percentage coordinates system. To determine the x-cordinate for a point, simply ask "How far across (0 to 100%) the screen is the point? To determine the y-coordinates for a point, simply ask "How far (0 to 100%) up the screen is the point?" All coordinate values are 0 to 100%; decimal values such as 23.5, 56.27, and 34.693 are permitted). Thus

```
(0,0)     = lower left corner of screen
(100,0)   = lower right corner of screen
(0,100)   = upper left corner of screen
(100,100) = upper right corner of screen
(50,50)   = middle of screen
```

__TEXT Y = INTE()SIN(ALPH()X)DX + ESUP()(BX)__
generates the desired text string consisting of the specified mathematical equation.

```
y = integral sin(alpha x)dx + exp(bx)
```

on the screen. Due to the previous FONT, CASE, JUSTIFICATION, HEIGHT, and WIDTH commands, the text string will be in triplex, lower case, left-justified (at the current cursor position, namely (20,50)), with height = 5% of screen height and width = 4% of screen width.

Note the use in this example of the DATAPLOT in-line text sub-commands

INTE() to draw an integral;

ALPH() to write the Greek letter alpha.

SUP() to shift to superscript;

DATAPLOT has in-line text sub-commands which allow the analyst to output a wide variety of mathematical symbols, Greek characters, and typographical symbols/operations. These in-line sub-commands may be used with any of the "fancy script" fonts (that is, with all of the fonts except the Tektronix font). The sub-commands are at most 4 characters long (as in INTE and ALPH). These sub-commands all have an appended () which declares to DATAPLOT that these strings are not to be written out literally, but rather are sub-commands which generate specialized output. Refer to the appendix for the full list of in-line text sub-commands. These sub-commands may be used in any TEXT, TITLE, LABEL, or LEGEND command.

Generate a Word Chart

Problem

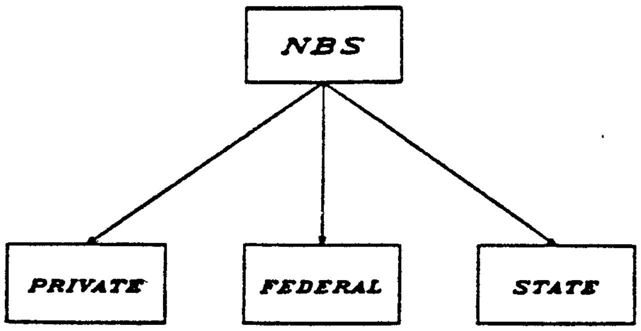
Generate a word chart. Have a box with

NBS

in it at one level; have 3 boxes with

PRIVATE
FEDERAL
STATE

in them at a second level. Have arrows going from the first box to the other 3 boxes. Have all text in upper case triplex italic.



DATAPLOT Program

```
FONT TRIPLEX ITALIC
JUSTIFICATION CENTER
.
FEEDBACK OFF
ERASE
.
HEIGHT 4
WIDTH 3
MOVE 50 80
TEXT NBS
.
HEIGHT 3
WIDHT 2
MOVE 20 40
TEXT PRIVATE
.
MOVE 50 40
TEXT FEDERAL
.
MOVE 80 40
TEXT STATE
.
BOX 40 75 60 88
BOX 10 35 30 48
BOX 40 35 60 48
BOX 70 35 90 48
.
HEIGHT .5
WIDTH 1
ARROW 50 75 20 48
ARROW 50 75 50 48
ARROW 50 75 80 48
.
COPY
```

Program Description

The general strategy is to write the text, then draw the boxes, and then draw the arrows. The general procedure (with diagrammatic graphics) is to

1) *experiment with sizing and positioning of text, boxes, etc. by entering DATAPLOT commands interactively;*

2) *(after satisfied with sizing, positioning, etc.) write the program (via the local editor) out onto a file/subfile;*

3) *execute the program as a whole (via the DATAPLOT CALL command).*

<u>FONT TRIPLEX ITALIC</u>
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

<u>JUSTIFICATION CENTER</u>
specifies (for future text strings) that the output from the TEXT command be center-justified (as opposed to left-justified or right-justified). The default is left-justified. The JUSTIFICATION command applies only to output resulting from the TEXT command, since output resulting from the TITLE and LABEL commands is automatically center-justified, and output resulting from the LEGEND command is automatically left-justified.

<u>.</u>
is a null command--it visually separates chunks of DATAPLOT code.

<u>FEEDBACK OFF</u>
suppresses the usual feedback messages that result from most DATAPLOT commands. The rationale for the FEEDBACK OFF command is that as soon as we erase the screen (which is the next command in this program), we want nothing to appear on the screen except the desired design. If we did not turn off the feedback, we would get feedback message superimposed on top of the design.

In general, when the feedback switch is turned off, then the messages from all "secondary" DATAPLOT commands are suppressed. In particular, the usual feedback messages from commands in the following categories are suppressed--

```
        Plot Control
        Support
        Output Device
```

Output from the Graphics category is not suppressed; output from the Analysis category (except for the LET command) is not suppressed; and output from the Diagrammatic Graphics category (except for FONT, CASE, JUSTIFICATION, HEIGHT, WIDTH, and CROSS-HAIR) is not suppressed. Unfortunately, for our program, we do have a few HEIGHT and WIDTH commands in the code after the ERASE, and therefore we must suppress their usual feedback messages.

<u>ERASE</u>
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased.

<u>HEIGHT 4</u>
specifies (for future text strings) that the output from the TEXT command have a height equal to 4% of vertical screen height. The HEIGHT command applies only to output resulting from the TEXT command. The size of output resulting from the TITLE, LABEL, and LEGEND commands are controlled via the

```
        TITLE SIZE
        ...LABEL SIZE
        LEGEND ... SIZE
```

commands, respectively, as in

```
        TITLE SIZE 2
        XLABEL SIZE 1.5
        LEGEND 3 SIZE 2.5
```

For all cases (text, titles, labels, and legends), note that the character sizing for all 7 Hershey character fonts is controllable on a continuous scale, whereas for the default Tektronix font (which is hardware-generated), the character sizing is limited to only 4 sizes (on large-screen terminals such as the Tektronix 4014, 4054, and 4114), and even to just 1 size (on small-screen terminals such as the Tektronix 4010, 4027, and 4052). For all text output (resulting from TEXT, TITLE, LABEL, and LEGEND efault character height is 3 (= 3% of total commands, the default character size is 3 (= 3% of total screen height).

<u>WIDTH 3</u>
specifies (for future text strings) that the output from the TEXT command have a width equal to 3% of horizontal screen width. The WIDTH command applies only to output resulting from the TEXT command. The width of output resulting from the TITLE, LABEL, and LEGEND commands is automatically set to 3/4 of the character height which in turn is controlled by the

```
        TITLE SIZE
        ...LABEL SIZE
        LEGEND ... SIZE
```

commands, respectively. For all cases (text, titles, labels, and legends), note that the character width for all 7 Hershey character fonts is controllable on a continuous scale, whereas for the default Tektronix font (which is hardware-generated), the character width is limited to only 4 sizes (on large-screen terminals such as the Tektronix 4014, 4054, and 4114), and even to just 1 size (on small-screen terminals such as the Tektronix 4010, 4027, and 4052). For all text output (resulting from TEXT, TITLE, LABEL, and LEGEND efault character height is 3 (= 3% of total commands, the default character width is 2 (= 2% of total screen width).

Note the existence of the HW command which is an abbreviated combination of both the HEIGHT and WIDTH commands, thus

```
        HEIGHT 4
        WIDTH 3
```

may be alternately specified as

```
        HW 4 3
```

MOVE 50 80
moves the cursor to a point 50% of the way across the screen and 80% of the way up the screen. MOVE 50 80 will thus place the cursor in the middle of the screen horizontally and up toward the top of the screen vertically. Note that DATAPLOT has standardized screen coordinates-- it is simple, easy to use, and universal for all output devices. Rather than use a screen coordinates system based on English units, metric units, device picture points, etc. (which change from device to device), DATAPLOT uses a percentage coordinates system. To determine the x-cordinate for a point, simply ask "How far across (0 to 100%) the screen is the point? To determine the y-coordinates for a point, simply ask "How far (0 to 100%) up the screen is the point?" All coordinate values are 0 to 100%; decimal values such as 23.5, 56.27, and 34.693 are permitted). Thus

(0,0)	*= lower left corner of screen*
(100,0)	*= lower right corner of screen*
(0,100)	*= upper left corner of screen*
(100,100)	*= upper right corner of screen*
(50,50)	*= middle of screen*

TEXT NBS
generates the text string

NBS

on the screen. Due to the previous FONT, JUSTIFICATION, HEIGHT, and WIDTH commands, the text string will be in triplex italic, center-justified (at the current cursor position, namely (50,80)), with height = 4% of screen height and width = 3% of screen width. Since the CASE command was not used previously, the text string will be in the default upper case.

HEIGHT 3
specifies (for future text strings) that the output from the TEXT command have a height equal to 3% of vertical screen height.

WIDTH 2
specifies (for future text strings) that the output from the TEXT command have a width equal to 2% of horizontal screen width.

MOVE 20 40
moves the cursor to a point 20% of the way across the screen and 40% of the way up the screen.

TEXT PRIVATE
generates the text string

PRIVATE

on the screen. Due to the previous FONT, JUSTIFICATION, HEIGHT, and WIDTH commands, the text string will be in triplex italic, center-justified (at the current cursor position, namely (20,40)), with height = 3% of screen height and width = 2% of screen width. Since the CASE command was not used previously, the text string will be in the default upper case.

MOVE 50 40
moves the cursor to a point 50% of the way across the screen and 40% of the way up the screen.

TEXT FEDERAL
generates the text string

FEDERAL

on the screen. Due to the previous FONT, JUSTIFICATION, HEIGHT, and WIDTH commands, the text string will be in triplex italic, center-justified (at the current cursor position, namely (50,40)), with height = 3% of screen height and width = 2% of screen width. Since the CASE command was not used previously, the text string will be in the default upper case.

MOVE 80 40
moves the cursor to a point 80% of the way across the screen and 40% of the way up the screen.

TEXT STATE
generates the text string

STATE

on the screen. Due to the previous FONT, JUSTIFICATION, HEIGHT, and WIDTH commands, the text string will be in triplex italic, center-justified (at the current cursor position, namely (80,40)), with height = 3% of screen height and width = 2% of screen width. Since the CASE command was not used previously, the text string will be in the default upper case.

BOX 40 75 60 88
draws a box with one corner at a point 40% of the way across the screen and 75% of the way up the screen, and with the opposing corner 60% of the way across the screen and 88% of the way up the screen. This box will surround the text string NBS.

Note that the easiest way to obtain the coordinate values for the BOX command is to make a pass at putting up the various text strings interactively (where we are using the screen as a scratch area with no concern about screen clutter), and then making liberal use of the cross-hair in the following fashion--

1) *enter the CROSS-HAIR (synonym = CH) command which will cause the cross-hair to appear on the screen;*

2) *position the cross-hair via the thumbwheel or joystick;*

3) *hit any key which will then cause the cross-hair to disappear and will report (in DATAPLOT 0 to 100 units) the latest position of the cross-hair.*

Note that in step 3 above, the coordinates will not be reported if the feedback switch has been turned off. Thus one could leave the feedback switch on, or preferably one could augment the CROSS-HAIR (synonym = with 2 parameter values, such as

```
        CH X1 Y1
```

which will cause (in step 3 above) for the coordinates values to be placed into the DATAPLOT parameters X1 Y1. These parameters may be printed at a later time, as in

```
        WRITE X1 Y1
```

or may be used in other DATAPLOT commands, as in

```
        MOVE X1 Y1
```

or (if X2 and Y2 are also parameters obtained in the same way)

```
        BOX X1 Y1 X2 Y2
```

The ability to store coordinates in parameters and refer to them symbolically thereafter is a powerful diagrammatic feature of DATAPLOT.

BOX 10 35 40 38
draws a box with one corner at a point 10% of the way across the screen and 35% of the way up the screen, and with the opposing corner 40% of the way across the screen and 38% of the way up the screen. This box will surround the text string PRIVATE.

BOX 40 35 60 48
draws a box with one corner at a point 40% of the way across the screen and 35% of the way up the screen, and with the opposing corner 60% of the way across the screen and 48% of the way up the screen. This box will surround the text string FEDERAL.

BOX 70 35 90 48
draws a box with one corner at a point 70% of the way across the screen and 35% of the way up the screen, and with the opposing corner 90% of the way across the screen and 48% of the way up the screen. This box will surround the text string STATE.

HEIGHT .5
specifies (for future text strings) that the output from the TEXT command have a height equal to .5% of vertical screen height. In addition, the HEIGHT command allows us to control the size of features of other diagrammatic graphics features; in particular—

the size (across the shaft) of an arrow head;

the size (across the circuit line) of a resistor wrinkle;

the size of a bar in a capacitor.

For our problem, we will be drawing arrows and so we wish to specify the characteristics of the arrow head. HEIGHT .5 tells DATAPLOT to have the size (across the shaft) of the arrow head to be .5% of vertical screen height.

WIDTH 1
specifies (for future text strings) that the output from the TEXT command have a width equal to 1% of horizontal screen width. In addition, the WIDTH command allows us to control the size of features of other diagrammatic graphics features; in particular—

the size (along the shaft) of an arrow head;

the size (along the circuit line) of a resistor wrinkle;

the spacing between the bars in a capacitor.

For our problem, we will be drawing arrows and so we wish to specify the characteristics of the arrow head. WIDTH 1 tells DATAPLOT to have the size (along the shaft) of the arrow head to be 1% of horizontal screen width. The net result between the HEIGHT .5 and WIDTH 1 commands is that future arrows will have arrow heads which are twice as long along the shaft as they are across the shaft; that is, the arrows will be fairly pointed as opposed to being fairly blunt.

ARROW 50 75 20 48
draws an arrow with the back of the arrow at at a point 50% of the way across the screen and 75% of the way up the screen, and with the front of the arrow 20% of the way across the screen and 48% of the way up the screen. This arrow will proceed from the NBS box to the PRIVATE box. The coordinates for this arrow are best obtained by use of the cross-hair as descrived above, or by sinply noting the coordinates of the existing boxes.

ARROW 50 75 50 48
draws an arrow with the back of the arrow at at a point 50% of the way across the screen and 75% of the way up the screen, and with the front of the arrow 50% of the way across the screen and 48% of the way up the screen. This arrow will proceed from the NBS box to the FEDERAL box.

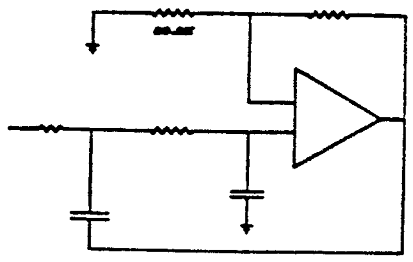
ARROW 50 75 80 48
draws an arrow with the back of the arrow at at a point 50% of the way across the screen and 75% of the way up the screen, and with the front of the arrow 80% of the way across the screen and 48% of the way up the screen. This arrow will proceed from the NBS box to the STATE box.
COPY copies (to the local hardcopy unit) the current contents of the screen. Anytime a COPY command is encountered in a DATAPLOT program, the screen is immediately copied. Multiple copies are made by appending the desired number, as in

```
        COPY 2
```

Generate an Electrical Diagram

Problem

Generate an electrical diagram, complete with appropriately positioned text strings.



DATAPLOT Program

```
FEEDBACK OFF
ERASE
.
HEIGHT .6
WIDTH .6
.
MOVE 20 65
GROUND 20 63
.
DRAW 20 65 20 70 27 70
RESISTOR 27 70 33 70
.
DRAW 33 70 47 70
RESISTOR 47 70 53 70
.
DRAW 53 70 60 70 60 30 20 30 20 35
CAPACITOR 20 35 20 36
.
DRAW 20 36 20 50 17 50
RESISTOR 17 50 13 50
.
DRAW 13 50 10 50
CIRCLE 10 50 9.7 50
.
MOVE 20 50
DRAW 20 50 27 50
RESISTOR 27 50 33 50
DRAW 33 50 46 50
.
MOVE 40 70
DRAW 40 70 40 55 46 55
MOVE 46 52.5
AMP 46 52.5 57 52.5
DRAW 57 52.5 60 52.5
.
MOVE 40 50
DRAW 40 50 40 40
CAPACITOR 40 39
.
DRAW 40 39 40 35
GROUND 40 35 40 33
.
HEIGHT 1.5
WIDTH 1
FONT TRIPLEX ITALIC
JUSTIFICATION CENTER
MOVE 30.2 66.5
TEXT 39.2K
.
COPY
```

Program Description

This program contains many coordinate references. These coordinates were obtained by heavy use of the cross-hair (via the CROSS-HAIR command) in an initial pass at interactively constructing the diagram.

Note also that all of the coordinates in this program are given explicitely as numbers, as in

```
DRAW 40 50 60 75
```

It is at times convenient and time-saving (especially in updating large, complicated diagrams) to store such numbers in DATAPLOT parameters via the LET command, as in

```
LET X1 = 40
LET Y1 = 50
LET X2 = 60
LET X2 = 75
```

or via the CROSS-HAIR (synonym = CH) command, as in

```
CH X1 Y1
CH X2 Y2
```

and then refer to them symbolically afterwards, as in

```
DRAW X1 Y1 X2 Y2
```

Because of the repeated use and similarity of the various commands in this electrical diagram program, we will selectively describe only one of each distinct command.

FEEDBACK OFF
suppresses the usual feedback messages that result from most DATAPLOT commands. The rationale for the FEEDBACK OFF command is that as soon as we erase the screen (which is the next command in this program), we want nothing to appear on the screen except the desired design. If we did not turn off the feedback, we would get feedback message superimposed on top of the design.

ERASE
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased.

HEIGHT .6
specifies (for future text strings) that the output from the TEXT command have a height equal to .6% of vertical screen height. In addition, the HEIGHT command allows us to control the size of features of other diagrammatic graphics features; in particular--

the size (across the shaft) of an arrow head,

the size (across the circuit line) of a resistor wrinkle;

the size of a bar in a capacitor.

WIDTH .6
specifies (for future text strings) that the output from the TEXT command have a width equal to .6% of horizontal screen width. In addition, the WIDTH command allows us to control the size of features of other diagrammatic graphics features; in particular--

the size (along the shaft) of an arrow head;

the size (along the circuit line) of a resistor wrinkle;

the spacing between the bars in a capacitor.

MOVE 20 65
moves the cursor to a point 20% of the way across the screen and 65% of the way up the screen. Note the standardized DATAPLOT screen coordinate system

GROUND 20 63
draws a ground starting at the current cursor location and ending at a point 20% of the way across the screen and 63% of the way up the screen). For the remainder of this example, such a coordinate will be referred to as (x = 20,y = 63), or simply (20,63).

The standardized (0 to 100 in both the horizontal and vertical directions) is such that

(0,0)	*= lower left corner of screen*
(100,0)	*= lower right corner of screen*
(0,100)	*= upper left corner of screen*
(100,100)	*= upper right corner of screen*
(50,50)	*= middle of screen*

GROUND 40 35 40 33
draws a ground with its top at (40,35) and its bottom at (40,33).

DRAW 20 65 20 70 27 70
draws a line segment from (20,65) to (20,70), and then from (20,70) to (27,70).

RESISTOR 27 70 33 70
draws a resistor from (27,70) to (33,70). The wrinkles on the resistor have a height and width as specified via prior entry of the HEIGHT and WIDTH commands, respectively.

CAPACITOR 20 35 20 36
draws a capacitor with the midpoint of one bar at (20,35) and the midpoint of the other bar at (20,36) The height of the 2 bars is specified via prior entry of the HEIGHT command.

CIRCLE 10 50 9.7 50
draws a circle with one end of the diameter at (10,50) and the other end of the diameter at (9.7,50).

AMP 46 52.5 57 52.5
draws an amplifier with the midpoint of the input side at (46,52.5) and the point on the output side at (57,52.5).

FONT TRIPLEX ITALIC
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

JUSTIFICATION CENTER
specifies (for future text strings) that the output from the TEXT command be center-justified (as opposed to left-justified or right-justified). The default is left-justified. The JUSTIFICATION command applies only to output resulting from the TEXT command, since output resulting from the TITLE and LABEL commands is automatically center-justified, and output resulting from the LEGEND command is automatically left-justified.

<u>TEXT 39.2X</u>
generates the text string

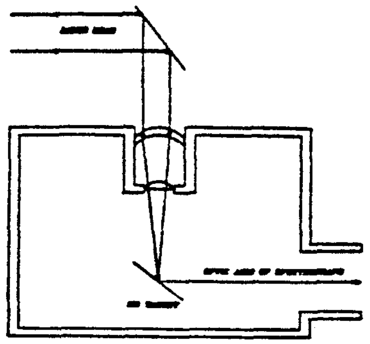
```
       39.2X
```

on the screen at the current cursor position. The text string will make use of settings as specified via previous entry of FONT, JUSTIFICATION, CASE, HEIGHT, and WIDTH command. If none of these had been previously entered, then the default settings of Tektronix font, left-justified, upper case, 3% height, and 2% width will be used. In this physical diagram example, FONT TRIPLEX ITALIC, JUSTIFICATION CENTER, HEIGHT 1.5, and WIDTH 1 had been previsouly entered, and so the text string appeared in triplex italic font, center-justified, upper case, with 1.5% height, and 1% width.

<u>COPY</u>
copies (to the local hardcopy unit) the current contents of the screen. Anytime a COPY command is encountered in a DATAPLOT program, the screen is immediately copied. Multiple copies are made by appending the desired number, as in

```
       COPY 2
```

Generate a Physical Diagram



Problem

Generate a physical diagram, complete with appropriately positioned text strings.

***DATAPLOT* Program**

```
     FEEDBACK OFF
     ERASE
     .
     DRAW 90 10 80 10 80 5 20 5 20 60 45 60 45 45
     DRAW 45 45 47 45 47 43 43 43 43 58 22 58 22 7 78 7 78 12 90 12 90 10
     DRAW 90 28 78 28 78 58 57 58 57 43 53 43
     DRAW 53 43 53 45 55 45 5 60 80 60 80 30 90 30 90 28
     .
     ARC 45 57 50 60 55 57
     ARC 45 55 50 58 55 55
     ARC 47 44 50 46 53 44
     .
     DRAW 47 44 53 44
     DRAW 55 75 45 92
     DRAW 45 25 55 15
     .
     HEIGHT .5
     WIDTH 1
     ARROW 20 90 30 90 46.5 90 46.7 57 50 20 90 20
     ARROW 20 80 30 80 52 80 51.95 57.4 50 20
     .
     FONT TRIPLEX ITALIC
     HEIGHT 2
     WIDTH 1
     MOVE 30 85
     TEXT LASER BEAM
     .
     MOVE 44 13
     TEXT NO TARGET
     .
     MOVE 59 22
     TEXT OPTIC AXIS OF SPECTROGRAPHY
     .
     COPY
```

Program Description

Because of the repeated use and similarity of the various commands in this program, we will selectively describe only one of each distinct command.

<u>FEEDBACK OFF</u>
suppresses the usual feedback messages that result from most DATAPLOT commands. The rationale for the FEEDBACK OFF command is that as soon as we erase the screen (which is the next command in this program), we want nothing to appear on the screen except the desired design. If we did not turn off the feedback, we would get feedback message superimposed on top of the design.

<u>ERASE</u>
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased.

<u>.</u>
is a null command--it visually separates chunks of DATAPLOT code.

<u>DRAW 90 10 80 10 80 5 20 5 20 60 45 60 45 45</u>
draws a line segment from (90,10) to (80,10), and then onto (80,5), then to (20,5), then to (20,60), then to (45,60), and then to (45,45).

<u>ARC 45 57 50 60 55 57</u>

draws an arc (of a circle) through the 3 points (45,57), (50,60), and (55,57). The arc starts at (45,57) and finishes at (55,57).

<u>HEIGHT .5</u>
specifies (for future text strings) that the output from the TEXT command have a height equal to .5% of vertical screen height. In addition, the HEIGHT command allows us to control the size of features of other diagrammatic graphics features; in particular--

the size (across the shaft) of an arrow head;

the size (across the circuit line) of a resistor wrinkle;

the size of a bar in a capacitor.

WIDTH 1
specifies (for future text strings) that the output from the TEXT command have a width equal to 1% of horizontal screen width. In addition, the WIDTH command allows us to control the size of features of other diagrammatic graphics features; in particular--

the size (along the shaft) of an arrow head;

the size (along the circuit line) of a resistor wrinkle;

the spacing between the bars in a capacitor.

ARROW 20 90 30 90 46.5 90 46.7 57 50 20 90 20
draws a series of 5 arrows with the first arrow extending (back to front) from (20,90) to (30,90); the second arrow extending from (30,90) to (46.5,90); the third arrow extending from (46.5,90) to (46.7,57); the fourth arrow extending from (46.7,57) to (50,20), and the fifth arrow extending from (50,20) to (90,20). Due to the prior entry of the HEIGHT .5 and WIDTH 1 commands, the arrow heads of each of the 5 arrows will be twice as long along the shaft as they are across the shaft; that is, the arrow heads will be fairly pointed as opposed to being fairly blunt.

FONT TRIPLEX ITALIC
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

MOVE 30 85
moves the cursor to the point (30,85), that is, to a point 30% of the way across the screen and 85% of the way up the screen.

TEXT LASER BEAM
generates the text string

LASER BEAM

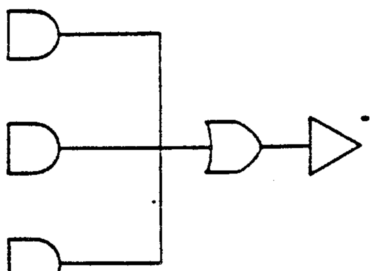
on the screen at the current cursor position. The text string will make use of settings as specified via previous entry of FONT, JUSTIFICATION, CASE, HEIGHT, and WIDTH command. If none of these had been previously entered, then the default settings of Tektronix font, left-justified, upper case, 3% height, and 2% width will be used. In this physical diagram example, FONT TRIPLEX ITALIC, HEIGHT 2, and WIDTH 1 had been previsouly entered, and so the text string appeared in triplex italic font, left-justified, upper case, with 2% height, and 1% width.

COPY
copies (to the local hardcopy unit) the current contents of the screen. Anytime a COPY command is encountered in a DATAPLOT program, the screen is immediately copied.

Generate a Logic Diagram

Problem

Generate a simplified logic diagram.



DATAPLOT Program

Two programs will be presented, the first will use explicit coordinates, and the second will use parameter coordinates. Program 1 is

```
        FEEDBACK OFF
        ERASE
        .
SYSTEM WARNING - MAX TIME
        AND 20 20 30 20
        AND 20 50 30 50
        AND 20 80 30 80
        .
        DRAW 30 20 50 20 50 50 30 50 50 50 50 80 30 80
        .
        DRAW 50 50 60 50
        OR 60 50 70 50
        .
        DRAW 70 50 80 50
        AMP 80 50 90 50
        .
        COPY
```

Program 2 is

```
        FEEDBACK OFF
        ERASE
        .
        LET DELX = 10
        LET DELY = 30
        .
        LET X1 = 20
        LET X2 = X1+DELX
        LET X3 = X2+2*DELX
        LET X4 = X3+DELX
        LET X5 = X4+DELX
        LET X6 = X5+DELX
        LET X7 = X6+DELX
        .
        LET Y1=20
        LET Y2=Y1+DELY
        LET Y3=Y2+DELY
        .
        AND X1 Y1 X2 Y1
        AND X1 Y2 X2 Y2
        AND X1 Y3 X2 Y3
        .
        DRAW X2 Y1 X3 Y1 X3 Y2 X2 Y2 X3 Y2 X3 Y3 X2 Y2
        .
        DRAW X3 Y2 X4 Y2
        OR X4 Y2 X5 Y2
        .
        DRAW X5 Y2 X6 Y2
        AMP X6 Y2 X7 Y2
        .
        COPY
```

Program Description

Program 1 has explicit, numeric coordinates; the advantage of program 1 is that it is short and explicit; the disadvantage is that most of the coordinates would need to be changed if it were desired to shrink or "blow up" the diagram, to move the diagram, or to change the spacing between components in the diagram.

Program 2 has changeable, parametric coordinates; the advantage of program 1 is that only a few lines need be changed (defining DELX, DELY, X1, and X2) if it were desired to shrink, "blow up", or move the diagram, or to change the spacing between components of the diagram. The disadvantage is that it is longer and slightly more difficult to understand.

Because of the repeated use and similarity of the various commands in these programs, we will selectively describe only one of each distinct command.

FEEDBACK OFF
suppresses the usual feedback messages that result from most DATAPLOT commands. The rationale for the FEEDBACK OFF command is that as soon as we erase the screen (which is the next command in this program), we want nothing to appear on the screen except the desired design. If we did not turn off the feedback, we would get feedback message superimposed on top of the design.

ERASE
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased.

.
is a null command—it visually separates chunks of DATAPLOT code.

AND 20 20 30 20
draws an AND gate with the middle of the input side at (20,20) and the output point at (30,20); that is, the middle of the input side is at a point 20% of the way across the screen and 20% of the way up the screen, while the output point is 30% of the way across the screen and 20% of the way up the screen.

DRAW 30 20 50 20 50 50 30 50 50 50 50 80 30 80
draws a line segment form (30,20) to (50,20), and then onto (50,50), then to (30,50), then to (50,50), then to (50,80), and then to (30,80).

OR 60 50 70 50
draws an OR gate with the middle of the input side at (60,50) and the output point at (70,50)

AMP 80 50 90 50
draws an amplifier with the middle of the input side at (80,50) and the output point at (90,50).

COPY
copies (to the local hardcopy unit) the current contents of the screen. Anytime a COPY command is encountered in a DATAPLOT program, the screen is immediately copied.

LET DELX = 10
defines a parameter DELX and assigns to it the value 10.

LET X1 = 20
defines a parameter X1 and assigns to it the value 20.

LET X2 = X1+DELX
defines a parameter X2 and assigns to it the value obtained by summing the current contents of the parameters X1 and DELX.

*LET X3 = X2+2*DELX*
defines a parameter X3 and assigns to it the value obtained by taking the current contents of parameter X3, multiplying it by 2, and adding it to the current contents of parameter X2.

AND X1 Y1 X2 Y1
draws an AND gate with the middle of the input side at (X1,Y1) and with the output point at (X2,Y1). As expected, when this command is executed, DATAPLOT will replace the parameter names with the current parameter values.

Generate a Chemical Equation

Problem

Generate a chemical equation.

$${}_{7}N^{13} \longrightarrow {}_{6}C^{13} + \beta^{-} + \nu$$

DATAPLOT Program

```
        FONT TRIPLEX ITALIC
        .
        FEEDBACK OFF
        ERASE
        .
        HW 4 3
        MOVE 20 50
        TEXT SUB()7UNSB()NSUP()13UNSP()
        .
        HW .5 1
        ARROW 32 51 40 51
        .
        HW 4 3
        TEXT SUB()6UNSB()CSUP()13UNSP() + LC()BETA()SUP()+UNSP() + NU()
        .
        COPY
```

Program Description

FONT TRIPLEX ITALIC
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

- *Tektronix (the default)*
- *Simplex*
- *Duplex*
- *Triplex*
- *Triplex Italic*
- *Complex*
- *Simplex Script*
- *Complex Script*

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

.
is a null command--it visually separates chunks of DATAPLOT code.

FEEDBACK OFF
suppresses the usual feedback messages that result from most DATAPLOT commands. The rationale for the FEEDBACK OFF command is that as soon as we erase the screen (which is the next command in this program), we want nothing to appear on the screen except the desired design. If we did not turn off the feedback, we would get feedback message superimposed on top of the design.

ERASE
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased.

HW 4 3
specifies (for future text strings) that the output from the TEXT command have a height equal to 4% of vertical screen height and a width equal to 3% of horizontal screen width.

In addition, the height specification allows us to control the size of features of other diagrammatic graphics features; in particular--

the size (across the shaft) of an arrow head;

the size (across the circuit line) of a resistor wrinkle;

the size of a bar in a capacitor.

and the width specification allows us to control

the size (along the shaft) of an arrow head;

the size (along the circuit line) of a resistor wrinkle;

the spacing between the bars in a capacitor.

In this example, HW 4 3 is used to control the size of characters in the TEXT example. Note that HW 4 3 is exactly equivalent to

```
HEIGHT 4
WIDTH 3
```

MOVE 20 50

moves the cursor to a point 20% of the way across the screen and 50% of the way up the screen.

TEXT SUB()7UNSB()NSUP()13UNSP()

generates the text string which constitutes the left side of the chemical equation. The text string will make use of settings as specified via previous entry of FONT, JUSTIFICATION, CASE, HEIGHT, and WIDTH commands. If none of these had been previously entered, then the default settings of Tektronix font, left-justified, upper case, 3% height, and 2% width will be used. In this physical diagram example, FONT TRIPLEX ITALIC, HEIGHT 4, and WIDTH 3 had been previously entered, and so the text string appeared in triplex italic font, left-justified, upper case, with 4% height, and 3% width.

Note the use in this example of the DATAPLOT in-line text sub-commands

SUB() to shift to subscript

UNSB() to shift out of subscript

SUP() to shift to superscript;

UNSP() to shift out of superscript.

DATAPLOT has in-line text sub-commands which allow the analyst to output a wide variety of mathematical symbols, Greek characters, and typographical symbols/operations. These in-line sub-commands may be used with any of the "fancy script" fonts (that is, with all of the fonts except the Tektronix font). The sub-commands are at most 4 characters long (as in UNSB and UNSP). These sub-commands all have an appended () which declares to DATAPLOT that these strings are not to be written out literally, but rather are sub-commands which generate specialized output. Refer to the appendix for the full list of in-line text sub-commands. These sub-commands may be used in any TEXT, TITLE, LABEL, or LEGEND command.

HW .5 1

specifies (for future text strings) that the output from the TEXT command have a height equal to .5% of vertical screen height and a width equal to 1% of horizontal screen width. It also controls the size of arrow heads, resistor wrinkles, and capacitors (as described earlier). In this example, HW .5 1 is used to control the size of the arrow head in the succeeding ARROW command.

ARROW 32 51 40 51

draws an arrow. The start of the arrow is at (32,51) and the end of the arrow is at (40,51). Due to the prior entry of the HW .5 1 command, the arrow head will be twice as long along the shaft as they are across the shaft; that is, the arrow head will be fairly pointed as opposed to being fairly blunt.

HW 4 3

specifies (for future text strings) that the output from the TEXT command have a height equal to 4% of vertical screen height and a width equal to 3% of horizontal screen width.

TEXT SUB()6UNSB()CSUP()13UNSP() + LC()BETA()SUP()+UNSP() + NU()

generates the text string which constitutes the right side of the chemical equation. The text string will make use of settings as specified via previous entry of the FONT, JUSTIFICATION, CASE, HEIGHT, and WIDTH commands. In this physical diagram example, FONT TRIPLEX ITALIC and HW 4 3 had been previously entered, and so the text string appeared in triplex italic font, left-justified, upper case, with 4% height, and 3% width.

Note the use in this example of the DATAPLOT in-line text sub-commands

SUB() to shift to subscript

UNSB() to shift out of subscript

SUP() to shift to superscript;

UNSP() to shift out of superscript;

LC() to shift to lower case;

BETA() to write out the Greek letter beta;

NU() to write out the Greek letter nu.

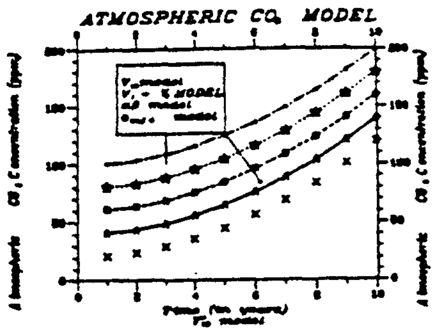
COPY

copies (to the local hardcopy unit) the current contents of the screen. Anytime a COPY command is encountered in a DATAPLOT program, the screen is immediately copied.

Generate a Presentation-Graphics Plot

Problem

Analysis graphics is graphics which is used to interrogate a data set; little concern is had for changing the DATAPLOT defaults for the fonts, tic marks, etc. the like. Presentation graphics is graphics which is used to convey the anlysis conclusions in a quality suitable for a journal or slide persentation. This example deals with presentation graphics per se. Generate a multi-trace persentation-graphics-quality plot with a variety of plot character types and line types. Have the titles and labels in Hershey triplex italic.



DATAPLOT Program

```
FONT TRIPLEX ITALIC
TIC MARK POSITION OUTSIDE
.
TITLE ATMOTPHERIC COSUB()2UNSB() MODEL
YLABEL ALC()TMOSTPHERIC UC()COSUB()2UNSB() CLC()ONCENTRATION (PPM)
XLABEL TLC()IME (IN YEARS)
CHARACTERS X A BOX START TRIANGLE
LINES BLANK SOLID DASHED DOTTED DASH2
.
PLOT X**2 + 20  FOR X = 1 1 10 AND
PLOT X**2 + 40  FOR X = 1 1 10 AND
PLOT X**2 + 60  FOR X = 1 1 10 AND
PLOT X**2 + 80  FOR X = 1 1 10 AND
PLOT X**2 + 100 FOR X = 1 1 10
```

Program Description

FONT TRIPLEX ITALIC
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

TIC MARK POSITION OUTSIDE
specifies (for future plots) that the tic marks are outside the frame (as opposed to through the frame, or inside the frame). The default tic mark position is through. Equivalent command statements to TIC MARK POSITION OUSIDE are

```
TIC MARK POSITION OUT
TIC MARKS OUTSIDE
TIC MARKS OUT
TICS OUTSIDE
TICS OUT
```

Similar equivalent statements exist for the TIC MARK POSITION THROUGH statement and the TIC MARK POSITION INSIDE statement.

TITLE ATMOTPHERIC COSUB()2UNSB() MODEL

specifies (for future plots) the title. The title appears above the top frame line of the plot and is automatically centered.

Note the use in this example of the DATAPLOT in-line text sub-commands

LC() to shift to lower case.

SUB() to shift to subscript

UNSB() to shift out of subscript

Generate a Map

Problem

Map information exists on a file ABC as (x,y) coordinate pairs. Two numbers—an x and a y—are on each line image. The (x,y) pairs are ordered in the sense that they exist in the file in the same sequence that they are to be drawn. Read in the data; place then in variables X and Y. Plot the map.

DATAPLOT Program

```
READ ABC. X Y
FRAME CORNER COORDINATES 15 20 70 80
FRAME OFF
PRE-SORT OFF
PLOT Y X
```

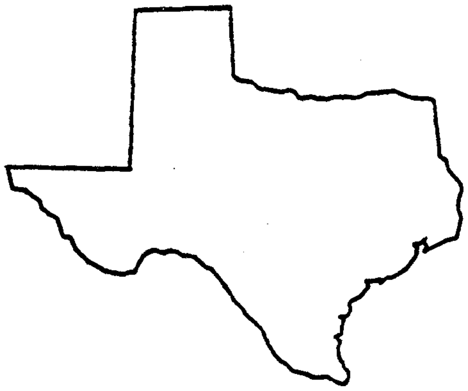
Program Description

READ ABC. X Y
carries out a format-free read of data from file ABC. The data is read into variables X and Y; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

FRAME CORNER COORDINATES 15 20 70 85
specifies (for future plots) that the usual 4-sided plot frame has opposing corners at (15,20) and (70,85)—that is, the lower left corner is 15% of the way across the screen and 20% of the way up the screen, and has the upper right corner is 70% of the way across the screen and 85% of the way up the screen. The net result is that the 4-sided plot frame will (on a Tektronix terminal) becomes square and so the map coordinates will not be distorted by the default rectangular frame corner coordinates of (15,20) and (85,85).

FRAME OFF
turns off (for future plots) the frame (axis lines Z and surrounding box) that would typically circumscribe the usual Y versus X plot. As one would expect, when the frame is turned off, so too are the tic marks on the frame and the tic mark labels adjacent to the frame. The rationale for

turning the frame off is that it makes little sense in a map context.



PRE-SORT OFF
turns off (for future plots) the default sorting (based on the horizontal axis variable) of the plot points. Such sorting before the plot is automatic and it relieves the analyst of the needless chore of having data in variables sorted before a plot. When the pre-sort switch is ON, then DATAPLOT will (before a plot) take the data in the variables in whatever order they are, sort them (according to the horizontal axis variable) and then plot them. This allows the analyst to generate the same plot regardless of the order of the data within variables. (Note that the variables themselves remain unchanged; DATAPLOT does all sorting in its own scratch area). For the vast majority (in excess of 99%) of plots, such pre-plot sorting is appropriate, and so the default for the pre-sort switch is ON. However, for certain applications (for example, polar function plots and maps) such sorting is not appropriate. In those cases in which we wish the points to be plotted in exactly the order in which they reside in the variables, then the pre-sort switch should be turned off.

PLOT Y X
generates a plot of variable Y (vertically) versus variable X (horizontally). The default plot type is continuous; the default axis limits will be neat and float with the data; the default character type is blank; the default line type is solid.

DATAPLOT has an extensive set of in-line text sub-commands which allow the analyst to output a wide variety of mathematical symbols, Greek characters, and typographical symbols/operations. These in-line sub-commands may be used with any of the "fancy script" fonts (that is, with all of the fonts except the Tektronix font). The sub-commands are at most 4 characters long (as in UNSB and UNSP). These sub-commands all have an appended () which declares to DATAPLOT that these strings are not to be written out literally, but rather are sub-commands which generate specialized output. Refer to the appendix for the full list of in-line text sub-commands. These sub-commands may be used in any TEXT, TITLE, LABEL, or LEGEND command.

<u>YLABEL ALC()TMOSTPHERIC UC()COSUB()2UNSB() CLC()ONCENTRATION (PPM)</u> *specifies (for future plots) the vertical axis labels. These labels appear on the left and right frame lines and are centered. If the analyst wishes to have a vertical axis label on th left side only, then the analyst should replace YLABEL with Y1LABEL, as in*

```
    Y1LABEL ALC()TMOSTPHERIC UC()COSUB()2UNSB() CLC()ONCENTRATION (PPM)
```

This example also makes use of in-line text commands, namely,

LC()	*to shift to lower case;*
UC()	*to shift to upper case;*
SUB()	*to shift to subscript*
UNSB()	*to shift out of subscript*

<u>XLABEL UC()TLC()IME (IN YEARS)</u> *specifies (for future plots) the horizontal axis label. This label appears under the bottom frame line and is centered. This example also makes use of in-line text commands, namely,*

LC()	*to shift to lower case;*
UC()	*to shift to upper case;*

Note that a synonym for XLABEL is X1LABEL, as in

```
    X1LABEL UC()TLC()IME (IN YEARS)
```

DATAPLOT provides space for 3 lines of centered labeling under the bottom frame line; XLABEL (or X1LABEL) specifies the first line; X2LABEL specifies the second line; and X3LABEL specifies the third line. All titles and labels remain in effect until overridden by another TITLE, YLABEL, XLABEL, etc. command. if additional annotiation is needed on plots, the analyst should use the LEGEND command or the TEXT command, as in

```
    LEGEND 1 COORDINATES 20 80
    LEGEND 1 ABC
```

or

```
    MOVE 20 80
    TEXT ABC
```

The legend commands would specify that the coordinates of the first character of the legend ABC is (20,80)--that is, 20% of the way across the screen and 80% of the way up the screen, and that such a legend should be printed out on all succeeding plots, until overridden via another LEGEND command.

The MOVE and TEXT command specifies an immediate move to the same location (20,80) and a write of the text string ABC.

<u>LINES BLANK SOLID DASHED DOTTED DASH2</u> *specifies (for future plots) the lines types that will appear between the plot points for the first 5 traces, namely,*

trace 1 will have blank (= no) connecting lines;

trace 2 will have solid connecting lines;

trace 3 will have dashed connecting lines;

trace 4 will have dotted connecting lines;

trace 5 will have another dash-dot pattern for its connecting line.

Note that in addition to BLANK, SOLID, DASHED, and DOTTED, there are an additional 20 dash-dot patterns which may be defined via the LINES command. These 20 specifications are defined via DASH1, DASH2, DASH3, ..., DASH20. See appendix xx for details.
Note that alternate forms exists for many of the line specifications, namely, BLANK may be written as BL, NONE, or NO; SOLID may be written as SO; DASHED may be written as DASH or DA; DOTTED may be written as DOT or DO; DASH1 may be written as DA1; DASH2 may be written as DA2; and so forth up to DASH20 may be written as DA20.

<u>CHARACTERS X A BOX STAR TRIANGLE</u> *specifies (for future plots) the characters that will appear at the plot points for the first 5 traces, namely,*

trace 1 will have X's at the plot points;

trace 2 will have A's at the plot points;

trace 3 will have boxes at the plot points;

trace 4 will have stars at the plot points;

trace 5 will have triangles at the plot points.

The choice of plot characters used here indicates the variety that DATAPLOT allows, for example,

the 26 alphabetic characters;

the 10 numeric characters;

any other keyboard character;

special characters, such as

box	*via BOX or SQUARE;*
circle	*via circle or o*
start	*via STAR*
diamond	*via DIAMOND*
triangle	*via TRIANGLE*
reverse triangle	*via TRIREV*
arrow	*via ARROW*

For a complete list of avaialbe characters, see appendix xxx.

```
PLOT X**2 + 20  FOR X = 1 1 10 AND
PLOT X**2 + 40  FOR X = 1 1 10 AND
PLOT X**2 + 60  FOR X = 1 1 10 AND
PLOT X**2 + 80  FOR X = 1 1 10 AND
PLOT X**2 + 100 FOR X = 1 1 10
```

generates a plot with 5 traces-- one plot for each of the 5 quadratic functions. Each function will be evaluated at a series of X values starting with 1, at increments of 1, and ending at 10 (that is, at X = 1, 2, 3, ..., 9, 10). The AND suffix at the end of the first 4 PLOT statments is important-- it tells DATAPLOT to link the PLOT statements so that DATAPLOT will generate 1 plot with 5 traces, as opposed to 5 plots eith 1 trace each.

Due to prior CHARACTERS and LINES commands, the 5 traces will be as follows--

trace 1 will have X's as the plot character and will have no connecting lines;

trace 2 will have A's as the plot character and will have solid connecting lines;

trace 3 will have boxes as plot character and will have dashed connecting lines;

trace 4 will have stars as plot character and will have dotted connecting lines;

trace 5 will have triangles as plot character and will have dash-dot connecting lines.

This plot illustrate typical output for presentation graphics. One other item which some analysts find preferable is to have the tic marks (and the tic mark labels) removed from the top frame line and the right frame line. To carry out this, the analyst should enter (somewhere before the PLOT commands) the following--

```
X2TIC MARKS OFF
Y2TIC MARKS OFF
```

which will delete the tic marks (and tic mark labels) on the upper frame line and the right frame lines, respectively. Another way to accomplish the same, is to simply enter

```
IMPLEMENT 1
```

The IMPLEMENT command with an argument of 1 will cause the upper and right tics (and tic labels) to be omitted.

If one turns off the tics on top and to the right, one also usually omits the vertical axis label on the right, this may be done by changing the YLABEL command to Y1LABEL, as in

```
Y1LABEL ALC()TMOSPHERIC OC()COSUB()2UNSB() CLC()ONCENTRATION (PPM)
```

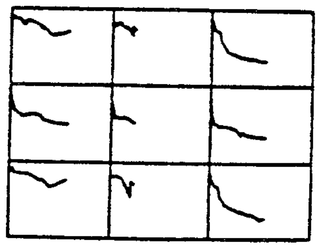
or by using the YLABEL command as is and simply blanking out the right vertical lable via Y2LABEL with no entry, as in

```
YLABEL ALC()TMOSPHERIC OC()COSUB()2UNSB() CLC()ONCENTRATION (PPM)
Y2LABEL
```

Generate Multiple Plots—All on the Same Page

Problem

Data exists on a file ABC for a response variable Y and 3 indepenent variables--time X (many levels), laboratory LAB (3 levels), and material MAT (3 levels). Read the data in; place the data into variables Y, X, LAB, and MAT.
Carry out an analysis to determine the effect of the 3 variables on the response. Do so by generating multiple plots (all on the same page) where each plot is a plot of Y versus X for some fixed value of the variable LAB and the variable MAT. To assist in the comparison across plots, have the vertical and horizontal axis limits the same for all plots. Since their are 3 levels for LAB and 3 levels for MAT, there will be 3 x 3 = 9 plots on the page.



DATAPLOT Program

```
READ ABC. Y X1 X2 X3
.
XLIMITS 0 4500
YLIMITS 1 9
TIC MARKS OFF
.
FEEDBACK OFF
PRE-ERASE OFF
ERASE
.
. GENERATE THE 3 PLOTS ON THE TOP ROW (LAB 1)
.
FRAME CORNER COORDINATES 20 60 40 80
PLOT Y X SUBSET LAB 1 SUBSET MAT 1
FRAME CORNER COORDINATES 40 60 60 80
PLOT Y X SUBSET LAB 1 SUBSET MAT 2
FRAME CORNER COORDINATES 60 60 80 80
PLOT Y X SUBSET LAB 1 SUBSET MAT 3
.
. GENERATE THE 3 PLOTS ON THE MIDDLE ROW (LAB 2)
.
FRAME CORNER COORDINATES 20 40 40 60
PLOT Y X SUBSET LAB 2 SUBSET MAT 1
FRAME CORNER COORDINATES 40 40 60 60
PLOT Y X SUBSET LAB 2 SUBSET MAT 2
FRAME CORNER COORDINATES 60 40 80 60
PLOT Y X SUBSET LAB 2 SUBSET MAT 3
.
. GENERATE THE 3 PLOTS ON THE BOTTOM ROW (LAB 3)
.
FRAME CORNER COORDINATES 20 20 40 40
PLOT Y X SUBSET LAB 3 SUBSET MAT 1
FRAME CORNER COORDINATES 40 20 60 40
PLOT Y X SUBSET LAB 3 SUBSET MAT 2
FRAME CORNER COORDINATES 60 20 80 40
PLOT Y X SUBSET LAB 3 SUBSET MAT 3
.
COPY
```

Program Description

READ ABC. Y X LAB MAT
carries out a format-free read of data from file ABC. The data is read into variables Y, X, LAB, and MAT; the first number on each line image of the file is read into the variable Y; the second number on each line is read into the variable X; the third number is read into the variable LAB; the fourth number is read into the variable MAT.

The read terminates when an END OF DATA line image
The read terminates when an END OF DATA line image

is encountered (or when a system end of file is encountered).

.
is a null command—it visually separates chunks of DATAPLOT code.

XLIMITS 0 4500
specifies (for future plots) that the limits for the horizontal axis will be fixed at 0 and 4500. The rationale for this command statement is to assure that all 9 plots have identical horizontal axis limits--this will faciliate the intercomparison of plots. If this command were not put in, then the DATAPLOT default (neat limits which float with the data) could result in different horizontal axis limits for each different plot.

YLIMITS 0 9
specifies (for future plots) that the limits for the vertical axis will be fixed at 1 and 9. As before, the rationale for this command statement is to assure that all 9 plots have identical vertical axis limits—this will faciliate the intercomparison of plots. If this command were not put in, then the DATAPLOT default (neat limits which float with the data) could result in different vertical axis limits for each different plot.

TIC MARKS OFF
specifies (for future plots) that the tic marks be omitted on all 4 frame lines. Other variations of this command are--

```
XTIC MARKS OFF   omits tics on both   horizontal frames;
X1TIC MARKS OFF  omits tics on bottom horizontal frame;
X2TIC MARKS OFF  omits tics on top    horizontal frame;

YTIC MARKS OFF   omits tics on both   vertical   frames;
Y1TIC MARKS OFF  omits tics on left   vertical   frame;
Y2TIC MARKS OFF  omits tics on right  vertical   frame;
```

Note that when the tic marks are omitted, the tic mark labels are also automatically omitted. The rational for omitting the tic marks and tic mark labels in this example is that we wish to have the plot frames abutt up against one another. This being the case, the plots look "cleaner" if the tic marks (and tic mark labels) are omitted. The default is for tic marks and tic mark labels to be included. In all ...TIC MARKS commands, the word MARKS may be omitted, if the analyst so desires, as in

```
XTIC OFF
X1TIC OFF
X2TIC OFF

YTIC OFF
Y1TIC OFF
Y2TIC OFF
```

FEEDBACK OFF
suppresses the usual feedback messages that result from most DATAPLOT commands. The rationale for the FEEDBACK OFF command is that as soon as we erase the screen (which will be done with the ERASE command), we want nothing to appear on the screen except the desired plots. If we did not turn off the feedback, we would get feedback message superimposed on top of the plots. In particular, the FRAME CORNER COORDINTES command would generate such a message as it adjusted frame sizes.

In general, when the feedback switch is turned off, then the messages from all "secondary" DATAPLOT commands are suppressed. In particular, the usual feedback messages from commands in the following categories are suppressed—

Plot Control
Support
Output Device

Output from the Graphics category is not suppressed; output from the Analysis category (except for the LET command) is not suppressed; and output from the Diagrammatic Graphics category (except for FONT, CASE, JUSTIFICATION, HEIGHT, WIDTH, and CROSS-HAIR) is not suppressed. Unfortunately, for our program, we do have a few HEIGHT and WIDTH commands in the code after the ERASE, and therefore we must suppress their usual feedback messages.

PRE-ERASE OFF
specifies (for future plots) that pre-erase switch be turned off. The net effect of this is that whenever a plot is generated (by any command within the DATAPLOT Graphics category, e.g., PLOT, 3D-PLOT, HISTOGRAM, SPECTRUM, PIE CHART, ... PROBABILITY PLOT, LAG ... PLOT, etc.) the default screen erasure which occurs before any such plot will not take place.

The rationale in this example for turning off such auto automatic pre-plot screen erasing is that in essense we are generating 9 separate plots (as opposed to 9 traces within a single plot), and the usual operation (with the pre-erase switch on) is to have an automatic screen erasure at the beginning of each of those 9 plots. The net effect, of course, is the erasure for plot 2 will also erase plot 1, the erasure for plot 3 will also erase plot 2, etc. The basic problem is that many terminals allow only full-screen erasure (that is, we cannot selectively erase only part of the screen). To get around this problem, we recommend that the method presented in this program, namely, to turn off the automatic pre-plot erasure (via the PRE-ERASE OFF command) and then to "manually" erase the screen before the first plot only (via the ERASE command to be described next).

ERASE
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased. The ERASE command will only be used once in this program-- to erase the screen before the formation of the first plot. We will not, of course, be erasing the screen before any of the remaining 8 plots.

. GENERATE THE 3 PLOTS ON THE TOP ROW (LAB 1)
is a non-executing comment. This command statement illustrates another use of the . command. chunks of DATAPLOT code. . is a null command--it visually separates It is used here to insert a comment statement. As with all DATAPLOT commands, the . command must be followed by at least 1 space before the rest of the command line (note the space between . and the G in the word GENERATE).

FRAME CORNER COORDINATES 20 60 40 80
specifies (for future plots) that the usual 4-sided plot frame has opposing corners at (20,60) and (40,80)--that is, the lower left corner is 20% of the way across the screen and 60% of the way up the screen, and has the upper right corner 40% of the way across the screen and 80% of the way up the plot. The net result is that the 4-sided plot frame will (until overridden by another FRAME CORNER COORDINATES command) appear in the upper left part of the screen. Due to prior use of the FEEDBACK OFF command, the usual feedback message from the FRAME CORNER COORDINATES command will be suppressed (so as to not clutter up the screen). The default frame corner coordinates are (15,20) and (85,85). As with most DATAPLOT commands, only the first 4 characters of each command word are needed to uniquely define the command (thus FRAME could be shortened to FRAM, CORNER could be shortened to CORN, and COORDINATES could be shortened to COOR). Such abbreviations, though generally permitted, are not encouraged due to the fact that they deviate from the full English syntax and so make programs a bit more difficult to understand after-the-fact.

PLOT Y X SUBSET LAB 1 SUBSET MAT 1
generates a plot of variable Y (vertically) versus variable X (horizontally); however, only those values in the X and Y variables which correspond to values in the LAB variable equal to 1 and (simultaneously) values in the MAT variable equal to 1) will be included in the plot. The net effect is that the plot of Y versus X will be restricted to that intersection set for which LAB = 1 and MAT = 1. This plot will appear in the upper left region of the screen. This method of using qualifications at the end of the command line (in this case, at the end of a PLOT command line) is an easy-to-use, and powerful This qualification involved the SUBSET keyword; other qualficiations may include the EXCEPT keyword and the FOR keyword. The qualfication statements may become quite complicated--involving unions and intersections of sets). The qualification for this example involved 2 subsets; however, more subsets could have been specified. SUBSET/EXCEPT/FOR qualifications may be appended to any command in the Graphics category (PLOT, 3D-PLOT, HISTOGRAM, PIE CHART, SPECTRAL PLOT, ...PROBABILITY PLOT, etc.) and any command in the Analysis category (FIT, LET, SMOOTH, SUMMARY, ANOVA, etc.)

As usual, since neither the CHARACTERS or LINES command had been entered, the default character is blank, and the default line type is solid.

FRAME CORNER COORDINATES 40 60 60 80
specifies (for future plots) that the usual 4-sided plot frame has opposing corners at (40,60) and (60,80)--that is, the lower left corner is 40% of the way across the screen and 60% of the way up the screen, and has the upper right corner 60% of the way across the screen and 80% of the way up the plot. The net result is that the 4-sided plot frame will appear in the upper middle part of the screen.

PLOT Y X SUBSET LAB 1 SUBSET MAT 2
generates a plot of variable Y (vertically) versus variable X (horizontally); however, only those values in the X and Y variables which correspond to values in the LAB variable equal to 1 and (simultaneously) values in the MAT variable equal to 2) will be included in the plot. The net effect is that the plot of Y versus X will be restricted to that intersection set for which LAB = 1 and MAT = 2. This plot will appear in the upper middle region of the screen.

As usual, since neither the CHARACTERS or LINES command had been entered, the default character is blank, and the default line type is solid.

The remaining FRAME CORNER COORDINATES and PLOT commands generate the remaining 7 plots in a similar fashion. For all 9 plots, the FRAME CORNER COORDINATES commands placed the frames as follows--

```
FRAME CORNER COORDINATES 20 60 40 80  upper  left
FRAME CORNER COORDINATES 40 60 60 80  upper  middle
FRAME CORNER COORDINATES 60 60 80 80  upper  right

FRAME CORNER COORDINATES 20 40 40 60  middle left
FRAME CORNER COORDINATES 40 40 60 60  middle middle
FRAME CORNER COORDINATES 60 40 80 60  middle right

FRAME CORNER COORDINATES 20 20 40 40  lower  left
FRAME CORNER COORDINATES 40 20 60 40  lower  middle
FRAME CORNER COORDINATES 60 20 80 40  lower  right
```

Other frame positions are, of course, possible. The ones here were chosen for simplicity and because they gave the desired output.

The 9 PLOT commands with the subset qualifications generated a plot within each of the 9 frame areas. We used prior knowledge that the LAB variable had values 1, 2, and 3, and that the MAT variable had values 1, 2, and 3. If the LAB and MAT variables took on other values, we would have changed the SUBSET qualifications accordingly.

COPY
copies (to the local hardcopy unit) the current contents of the screen. Anytime a COPY command is encountered in a DATAPLOT program, the screen is immediately copied. Multiple copies are made by appending the desired number, as in

```
COPY 2
```

To augment such multi-plots with text information, the analyst typically uses the TEXT command (for example, to indicate Laboratories 1, 2, and 3; and Material 1, 2, and 3 on the multi-plot). The positioning for the text strings are usually done via an earlier pass at the plot with the CROSS-HAIR (CH) command in order to get the needed coordinate information for each text string. A FONT command would specify the desired font, and a series of MOVE and TEXT commands would overlay the desired descriptive information. These commands should be inserted immediately before the COPY command in the program above.

Generating multiple plots on a single page is a powerful analytic tool for intercomparing the effect of various factors. The example presented above is fairly simple in that it consists of a response variable and only 3 factors; each sub-plot consists of only a single trace. This multi-plot tool becomes even more powerful in the simultaneous examination of the effect of 4, 5, and 6 factors. This is done by conveying information about additional factors by augmenting each sub-plot with not just 1 trace, but several traces, whereby information about factors 4, 5, and 6 may be conveyed in the usual graphical ANOVA fashion (GANOVA) via plot characters and line types. Yet additional information could be conveyed by use of color.

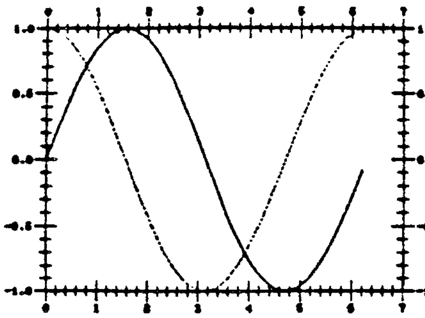
In summary, if the analyst had a response variable and 5 independent variables (factors), then the effect of these 5 factors could be examined via GANOVA in conjuntion with multiple plots per page.
SYSTEM WARNING - MAX PAGES
Information about the factors could be conveyed as follows--

response variable	*--vertical axis of each sub-plot;*
factor 1	*--horizontal axis of each sub-plot;*
factor 2	*--plot characters within each sub-plot;*
factor 3	*--line types within each sub-plot;*
factor 4	*--rows of sub-plots;*
factor 5	*--columns of sub-plots.*

Generate a Color Plot

Problem

Generate a multi-trace plot of 2 functions— a sine and a cosine. Have one trace green and the second trace yellow. Photographic restrictions preclude the illustration of the color plot. See figure 52.



DATAPLOT Program

```
COLOR ON
LINES SOLID DOTTED
LINE COLORS GREEN YELLOW
.
PLOT SIN(X) FOR X = 0 .1 6.3 AND
PLOT COS(X) FOR X = 0 .1 6.3
```

Program Description

COLOR ON
specifies that the terminal we are working on is capable of color output. The default color setting is off. If color is specified, the assumed terminal is the Tektronix 4027 (or equivalent). If color is specified, the default colors for all lines, characters, title, frame lines, tic marks, labels, etc. is red, and the default color for the background (= region within the frame lines) and margin (= region outside the frame lines) is blue.

LINES SOLID DOTTED
specifies (for future plots) that the first trace will have solid connecting lines; and the second trace will have dotted connecting lines. The default line types for all traces is solid.

LINE COLORS GREED YELLOW
specifies (for future plots) that the lines for trace 1 will be green, and the lines for trace 2 will be yellow. The default color for all lines is red.

.
is a null command--it visually separates chunks of DATAPLOT code.

PLOT SIN(X) FOR X = 0 .1 6.28 AND
PLOT COS(X) FOR X = 0 .1 6.28
*generates the desired plot with the 2 traces-- one trace for the sine function and one trace for the cosine function. Both functions will be evaluated at a series of X values starting with 0, at increments of .1, and ending at 6.28 (that is, at X = 0, .1, .2, .3, ..., 6.0, 6.1, 6.2). (6.28 was chosen as a rough approximation to 2*pi— the plots should thus yield about 1 cycle for both the sine and cosine) functions). Note that the X values for each of the function evaluations are temporary and local to the PLOT statement only; no X variable as such is created, nor does such a PLOT statement affect an X variable which the analyst may happen to have.*

The AND suffix at the end of the first PLOT statement is important-- it tells DATAPLOT to generate 1 plot with 2 traces, as opposed to 2 separate plots--each with only 1 trace. Due to prior entry of the COLOR, LINES, and LINE COLORS commands, the plot will be color, with trace 1 being a solid green line, and the second trace being a dotted yellow line. The frame lines, tic marks, and tic mark labels will be default red; background and the margin will be default blue.

Generate a Solid-Fill Bar Plot

Problem

Data exists on a file ABC as a series of (x,y) pairs. Two numbers--an x and a y--are on each line image. The x value represents a horizontal axis value and the y represents the corresponding vertical axis value. There are 5 data lines. The data file consists of the following 6 line images--

```
1975   .253
1976   1.141
1977   2.168
1978   5.077
1979   4.857
END OF DATA
```

Read in the data; place the data in variables X and Y. Generate a bar plot of the data. Have the bars filled. Have the title

MAINFRAME COMPUTER USAGE

Have the vertical axis label

CPU HOURS (IN HUNDREDS)

Have the horizontal axis label

YEAR

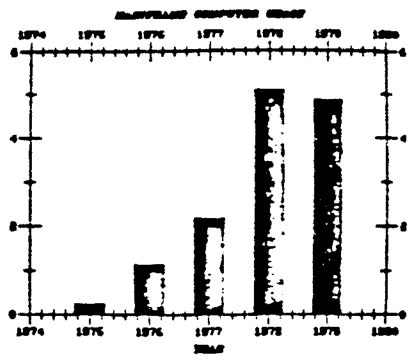
Have all titles and labels in triplex italic font.

DATAPLOT Program

```
READ ABC. X Y
.
HARDCOPY ON
FONT TRIPLEX ITALIC
.
TITLE MAINFRAME COMPUTER USAGE
YLABEL CPU HOURS (IN HUNDREDS)
XLABEL YEAR
.
BAR PATTERN VERTICAL
BAR SPACING .1
BAR WIDTH .5
BAR PLOT Y X
```

Program Description

<u>READ ABC. X Y</u>
carries out a format-free read of data from file ABC; The data is read into variables X and Y; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).



<u>.</u>
is a null command--it visually separates chunks of DATAPLOT code.

<u>HARDCOPY ON</u>
specifies (for future plots) that whenever a plot is generated from any command within the DATAPLOT Graphics category (e.g., PLOT, 3D-PLOT, HISTOGRAM, PIE CHART, SPECTRUM, BAR PLOT, etc.), as soon as the plot appears on the screen, it will be automatically copied onto the local hardcopy unit. The HARDCOPY command need be entered only once to cause the automatic hardcopying of all plots that appear on the screen. This is a very convenient feature which allows the analyst to run stored DATAPLOT programs which generate many plots-- and have all of the plots copied as they are produced. The HARDCOPY ON switch setting will remain as such until overridden by a HARDCOPY OFF command. To specify more than one hardcopy, simply append the desired number at the end of the command, as in

```
HARDCOPY ON 2
```

<u>FONT TRIPLEX ITALIC</u>
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

TITLE MAINFRAME COMPUTER USAGE
specifies (for future plots) the title. The title appears above the top frame line of the plot and is automatically centered. The title will appear above all succeeding plots until overridden by another TITLE command. To delete a current title, enter TITLE with no entry, as in

```
TITLE
```

YLABEL CPU USAGE (IN HUNDREDS)
specifies (for future plots) the vertical axis labels. These labels appear on the left and right frame lines and are centered. If the analyst wishes to have a vertical axis label on th left side only, then the analyst should replace YLABEL with Y1LABEL, as in

```
Y1LABEL CPU USAGE (IN HUNDREDS)
```

XLABEL YEAR
specifies (for future plots) the horizontal axis label. This label appears under the bottom frame line and is centered. Note that a synonym for XLABEL is X1LABEL, as in

```
X1LABEL ABC
```

DATAPLOT provides space for 3 lines of centered labeling under the bottom frame line; XLABEL (or X1LABEL) specifies the first line; X2LABEL specifies the second line; and X3LABEL specifies the third line. All titles and labels remain in effect until overridden by another TITLE, YLABEL, XLABEL, etc. command. if additional annotiation is needed on plots, the analyst should use the LEGEND command or the TEXT command, as in

```
LEGEND 1 COORDINATES 20 80
LEGEND 1 ABC
```

or

```
MOVE 20 80
TEXT ABC
```

The legend commands would specify that the coordinates of the first character of the legend ABC is (20,80)--that is, 20% of the way across the screen and 80% of the way up the screen, and that such a legend should be printed out on all succeeding plots, until overridden via another LEGEND command.

The MOVE and TEXT command specifies an immediate move to the same location (20,80) and a write of the text string ABC.

BAR PATTERN VERTICAL
specifies (for future Bar Plots) that the pattern within the bar is a vertical line. DATAPLOT provides 11-within bar patterns--

BAR PATTERN BLANK	*no pattern*
BAR PATTERN VERTICAL	*vertical*
BAR PATTERN HORIZONTAL	*horizontal*
BAR PATTERN D1	*diagonal (up)*
BAR PATTERN D2	*diagonal (down)*
BAR PATTERN D1D2	*diagonals (up and down)*
BAR PATTERN VED1	*vertical and diagonal (up)*
BAR PATTERN VED2	*vertical and diagonal (down)*
BAR PATTERN HOD1	*horizontal and diagonal (up)*
BAR PATTERN HOD2	*horizontal and diagonal (down)*
BAR PATTERN VEDD	*vert., hor., & both diagonals*

In practice, the first 6 are heavily used and the last 5 are rarely used.

Why are we specifying a bar pattern if we want to generate a solid fill bar? The rationale for this is that DATAPLOT has 2 ways of controlling the contents of bars in a bar plot--

bar pattern--the type of pattern;

bar spacing--the spacing between lines in the pattern (the spacing is in the usual 0 to 100 DATAPLOT screen percentage units; the default spacing is 3).

For any given bar pattern, the spacing of the pattern can be controlled to be sparse (for large bar spacings--such as 5) or dense (for small bar spacings such as .5). In the extreme, a bar spacing which is very small (such as .1) will have the net effect that the resulting plot will have solid fill. Thus DATAPLOT handles solid-fill bars as a special case of the more general bar pattern and bar spacing capabilites. For simplicity and efficiency (since the bar will be solid anyway), it is recommended that a simple pattern (such as vertical or horizontal) be used whenever a solid fill is desired.

The default bar pattern is blank; that is, no pattern.

<u>BAR SPACING .1</u>
specifies (for future bar plots) that the spacing between lines of a pattern within a bar be .1; that is, the spacing be such that the distance from one line in a pattern to the adjacent line in a pattern be equal to .1% of vertical screen height. For almost all terminals, such a small spacing will result in the bar being filled solid, as desired. On some terminals, the spacing could be broadened at bit (e.g., BAR SPACING .3) with the same net result-- a solid-appearing bar. On extremely high-resolution terminals, the bar spacing may need to be made even smaller (e.g., BAR SPACING .05) to generate solid fill plots. By way of example, the Tekronix 4014 and 4114 (with resolution of 4096 by 3124 picture points) will yield solid fill bars with a bar spacing of .1).

The default bar spacing is 3.

<u>BAR WIDTH .5</u>
specifies (for future bar plots) that the total width of each bar be .5 (in units of the horizontal axis variable). The BAR WIDTH command allows the analyst control how isolated each bar is. Does the analyst want "fat" bars which abutt up against one another with no gap at all between adjacvcent bars; or does the analyst want "skinney" bars which have a large gap between adjacent bars? Note that the units of the BAR WIDTH command is the units of the horizontal axis variable.

In this case, since the horizontal axis variable ranges from 1975 through 1979, a bar spacing of .5 will cause each of the bars to have a width of .5 unit (and therefore a gap of .5 units). If the bar width has been set to .7, then the bar width would have been .7 unit, (and therefore a gap of .3 units); If the bar width has been set to 1, then the bar width would have been 1 unit, (and therefore a gap of 0 units--that is, the bars would abutt up agains one another).

The default bar width is such that the bars will abutt up against one another.

<u>BAR PLOT Y X</u>
generates a bar plot. i The Y variable will be plotted vertically and the X variable will be plotted horizontally. The bars will be centered at each of the values of the X variable. Due to prior settings of the BAR PATTERN, BAR SPACING, and BAR WIDTH commands, the bars will have vertical stripes at a pspacing of .1 (and so will be solid), and the width of each bar will be .5. Since no XLIMITS or YLIMITS commands had been previosuly entered, the axis limits will be neat and float with the data. Due to the prior setting of the FONT command, the title and labels will be written out in triplex italic font with default upper case.

TITLE MAINFRAME COMPUTER USAGE
specifies (for future plots) the title. The title appears above the top frame line of the plot and is automatically centered. The title will appear above all succeeding plots until overridden by another TITLE command. To delete a current title, enter TITLE with no entry, as in

```
TITLE
```

YLABEL CPU USAGE (IN HUNDREDS)
specifies (for future plots) the vertical axis labels. These labels appear on the left and right frame lines and are centered. If the analyst wishes to have a vertical axis label on th left side only, then the analyst should replace YLABEL with Y1LABEL, as in

```
Y1LABEL CPU USAGE (IN HUNDREDS)
```

XLABEL YEAR
specifies (for future plots) the horizontal axis label. This label appears under the bottom frame line and is centered. Note that a synonym for XLABEL is X1LABEL, as in

```
X1LABEL ABC
```

DATAPLOT provides space for 3 lines of centered labeling under the bottom frame line; XLABEL (or X1LABEL) specifies the first line; X2LABEL specifies the second line; and X3LABEL specifies the third line. All titles and labels remain in effect until overridden by another TITLE, YLABEL, XLABEL, etc. command. if additional annotiation is needed on plots, the analyst should use the LEGEND command or the TEXT command, as in

```
LEGEND 1 COORDINATES 20 80
LEGEND 1 ABC
```

or

```
MOVE 20 80
TEXT ABC
```

The legend commands would specify that the coordinates of the first character of the legend ABC is (20,80)--that is, 20% of the way across the screen and 80% of the way up the screen, and that such a legend should be printed out on all succeeding plots, until overridden via another LEGEND command.

The MOVE and TEXT command specifies an immediate move to the same location (20,80) and a write of the text string ABC.

BAR PATTERN VERTICAL
specifies (for future Bar Plots) that the pattern within the bar is a vertical line. DATAPLOT provides 11-within bar patterns--

BAR PATTERN BLANK	no pattern
BAR PATTERN VERTICAL	vertical
BAR PATTERN HORIZONTAL	horizontal
BAR PATTERN D1	diagonal (up)
BAR PATTERN D2	diagonal (down)
BAR PATTERN D1D2	diagonals (up and down)
BAR PATTERN VED1	vertical and diagonal (up)
BAR PATTERN VED2	vertical and diagonal (down)
BAR PATTERN HOD1	horizontal and diagonal (up)
BAR PATTERN HOD2	horizontal and diagonal (down)
BAR PATTERN VEDO	vert., hor., & both diagonals

In practice, the first 6 are heavily used and the last 5 are rarely used.

Why are we specifying a bar pattern if we want to generate a solid fill bar? The rationale for this is that DATAPLOT has 2 ways of controlling the contents of bars in a bar plot--

bar pattern--the type of pattern;

bar spacing--the spacing between lines in the pattern (the spacing is in the usual 0 to 100 DATAPLOT screen percentage units; the default spacing is 3).

For any given bar pattern, the spacing of the pattern can be controlled to be sparse (for large bar spacings--such as 5) or dense (for small bar spacings such as .5). In the extreme, a bar spacing which is very small (such as .1) will have the net effect that the resulting plot will have solid fill. Thus DATAPLOT handles solid-fill bars as a special case of the more general bar pattern and bar spacing capabilites. For simplicity and efficiency (since the bar will be solid anyway), it is recommended that a simple pattern (such as vertical or horizontal) be used whenever a solid fill is desired.

The default bar pattern is blank; that is, no pattern.

BAR SPACING .1
specifies (for future bar plots) that the spacing between lines of a pattern within a bar be .1; that is, the spacing be such that the distance from one line in a pattern to the adjacent line in a pattern be equal to .1% of vertical screen height. For almost all terminals, such a small spacing will result in the bar being filled solid, as desired. On some terminals, the spacing could be broadened at bit (e.g., BAR SPACING .3) with the same net result-- a solid-appearing bar. On extremely high-resolution terminals, the bar spacing may need to be made even smaller (e.g., BAR SPACING .05) to generate solid fill plots. By way of example, the Tekronix 4014 and 4114 (with resolution of 4096 by 3124 picture points) will yield solid fill bars with a bar spacing of .1).

The default bar spacing is 3.

BAR WIDTH .5
specifies (for future bar plots) that the total width of each bar be .5 (in units of the horizontal axis variable). The BAR WIDTH command allows the analyst control how isolated each bar is. Does the analyst want "fat" bars which abutt up against one another with no gap at all between adjacvcent bars; or does the analyst want "skinney" bars which have a large gap between adjacent bars? Note that the units of the BAR WIDTH command is the units of the horizontal axis variable.

In this case, since the horizontal axis variable ranges from 1975 through 1979, a bar spacing of .5 will cause each of the bars to have a width of .5 unit (and therefore a gap of .5 units). If the bar width has been set to .7, then the bar width would have been .7 unit, (and therefore a gap of .3 units); If the bar width has been set to 1, then the bar width would have been 1 unit, (and therefore a gap of 0 units--that is, the bars would abutt up agains one another).

The default bar width is such that the bars will abutt up against one another.

BAR PLOT Y X
generates a bar plot. i The Y variable will be plotted vertically and the X variable will be plotted horizontally. The bars will be centered at each of the values of the X variable. Due to prior settings of the BAR PATTERN, BAR SPACING, and BAR WIDTH commands, the bars will have vertical stripes at a pspacing of .1 (and so will be solid), and the width of each bar will be .5. Since no XLIMITS or YLIMITS commands had been previosuly entered, the axis limits will be neat and float with the data. Due to the prior setting of the FONT command, the title and labels will be written out in triplex italic font with default upper case.

Generate a Bar Plot with Multiple Patterns

Problem

Data exists on a file ABC as a series of 3 variables-- a horizontal axis variable, and 2 response variables. There are 5 data lines. The data file consists of the following 6 line images--

```
1975   .253     0.069
1976   1.141    1.595
1977   2.168    3.136
1978   5.077    7.001
1979   4.857    10.732
END OF DATA
```

Read in the data; place the data in variables X, Y1, and Y2. Generate a bar plot of the data. Have the bars for the Y1 data with a horizontal stripe; have the bars for the Y2 data with a criss-cross stripe. Have the title

COMPUTER USAGE

Have the vertical axis label

CPU HOURS (IN HUNDREDS)

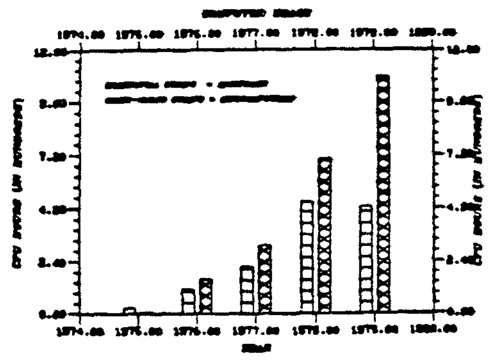
Have the horizontal axis label

YEAR

Have a legend (in upper right) which says

HORIZONTAL STRIPE = MAINFRAME
CRISS-CROSS STRIPE = MINICOMPUTERS

Have all titles, labels, and legends in triplex italic font.



DATAPLOT Program

```
READ ABC. X Y1 Y2
.
LET X1 = X-0.25
LET X2 = X+0.25
.
FONT TRIPLEX ITALIC
.
TITLE COMPUTER USAGE
YLABEL CPU HOURS (IN HUNDREDS)
XLABEL YEAR
.
FEEDBACK OFF
PRE-ERASE OFF
ERASE
.
XLIMITS 1974 1980
YLIMITS 0 6
BAR WIDTH .2
.
BAR PATTERN HORIZONTAL
BAR PLOT Y1 X1
.
BAR PATTERN D1D2
BAR PLOT Y2 X2
.
MOVE 20 80
TEXT HORIZONTAL STRIPE  = MAINFRAME
MOVE 20 76
TEXT CRISS-CROSS STRIPE = MINICOMPUTERS
.
COPY
```

Program Description

READ ABC. X Y
carries out a format-free read of data from file ABC; The data is read into variables X and Y; the first number on each line image of the file is read into the variable X; the second number on each line is read into the variable Y. The read terminates when an END OF DATA line image is encountered (or when a system end of file is encountered).

.
is a null command--it visually separates chunks of DATAPLOT code.

LET X1 = X-0.25
forms a new variable X1. X1 has the same number of elements as X. An individual element in X1 is computed by taking an individual element in X and subtracting 0.25 from it. The rationale for forming X1 is that our ultimate objective is to form a bar plot where we have 2 bars (side-by-side) at each year. The bars corresponding to the Y1 data will be slightly to the left of the year mark, while the bars corresponding to the Y2 data will be slightly to the right of the year mark. The easiest way to create such a displacement is to create an artificial variable X1 which will be slightly smaller than the year, and a variable X2 which will be slightly larger than the year. The final bar plot commands will thus be BAR PLOT Y1 X1 and BAR PLOT Y2 X2.

LET X2 = X+0.25
forms a new variable X2. X2 has the same number of elements as X. An individual element in X2 is computed by taking an individual element in X and adding 0.25 to it.

FONT TRIPLEX ITALIC
specifies (for future text strings) that the font be triplex italic. Eight fonts are available in DATAPLOT--

Tektronix (the default)
Simplex
Duplex
Triplex
Triplex Italic
Complex
Simplex Script
Complex Script

the last 7 are "fancy script" fonts and are based on Hershey character sets. The default font is Tektronix--the hardware-character font that would appear on the various Tektronix terminals. The FONT command specifications affects output from all future TEXT, TITLE, LABEL, and LEGEND commands.

TITLE COMPUTER USAGE
specifies the title. The title appears above the top frame line of the plot and is automatically centered. The title will appear above all succeeding plots until overridden by another TITLE command. To delete a current title, enter TITLE with no entry, as in

```
TITLE
```

YLABEL CPU USAGE (IN HUNDREDS)
specifies (for future plots) the vertical axis labels. These labels appear on the left and right frame lines and are centered. If the analyst wishes to have a vertical axis label on th left side only, then the analyst should replace YLABEL with Y1LABEL, as in

```
Y1LABEL CPU USAGE (IN HUNDREDS)
```

XLABEL YEAR
specifies (for future plots) the horizontal axis label. This label appears under the bottom frame line and is centered. Note that a synonym for XLABEL is X1LABEL, as in

```
X1LABEL ABC
```

DATAPLOT provides space for 3 lines of centered labeling under the bottom frame line; XLABEL (or X1LABEL) specifies the first line; X2LABEL specifies the second line; and X3LABEL specifies the third line. All titles and labels remain in effect until overridden by another TITLE, YLABEL, XLABEL, etc. command. If additional annotiation is needed on plots, the analyst should use the LEGEND command or the TEXT command, as in

```
LEGEND 1 COORDINATES 20 80
LEGEND 1 ABC
```

or

```
MOVE 20 80
TEXT ABC
```

LEGEND 1 COORDINATES 20 80 would specify (for future plots) the starting point of the first legend. In this case, the starting point (= the location of the first character of the legend) would be a point 20% across the screen and 80% up the screen.

The MOVE and TEXT command specifies an immediate move to the same location (20,80) and a write of the text string ABC.

FEEDBACK OFF
suppresses the usual feedback messages that result from most DATAPLOT commands. The rationale for the FEEDBACK OFF command is that as soon as we erase the screen (which will be done with the ERASE command), we want nothing to appear on the screen except the desired plots. If we did not turn off the feedback, we would get feedback message superimposed on top of the plots. In particular, the BAR PATTERN and BAR WIDTH commands would generate such messages as they were being executed. In general, when the feedback switch is turned off, then the messages from all "secondary" DATAPLOT commands are suppressed. In particular, the usual feedback messages from commands in the following categories are suppressed--

Plot Control
Support
Output Device

Output from the Graphics category is not suppressed; output from the Analysis category (except for the LET command) is not suppressed; and output from the Diagrammatic Graphics category (except for FONT, CASE, JUSTIFICATION, HEIGHT, WIDTH, and CROSS-HAIR) is not suppressed. Unfortunately, for our program, we do have a few HEIGHT and WIDTH commands in the code after the ERASE, and therefore we must suppress their usual feedback messages.

PRE-ERASE OFF
specifies (for future plots) that pre-erase switch be turned off. The net effect of this is that whenever a plot is generated (by any command within the DATAPLOT Graphics category, e.g., PLOT, 3D-PLOT, HISTOGRAM, SPECTRUM, BAR PLOT, ... PROBABILITY PLOT, LAG ... PLOT, etc.) the default screen erasure which occurs before any such plot will not take place.

The rationale in this example for turning off such auto automatic pre-plot screen erasing is that in essense we are generating 2 separate plots (as opposed to 2 traces within a single plot), and the usual operation (with the pre-erase switch on) is to have an automatic screen erasure at the beginning of each of those 2 plots. The net effect, of course, is the erasure for bar plot 2 will erase bar plot 1. The basic problem is that only the PLOT and 3D-PLOT commands in DATAPLOT have the ability to be "strunk together" via the AND suffix. Thus if we wish to superimpose 2 spectral plots, or 2 probaiblity plots, or 2 bar plots, etc. we do so by generating 2 separate plot. If the pre-plot erasure is suppressed and a few other details attended to, then the net result

will be the second plot superimposed atop the first plot.

ERASE
erases the screen. Anytime an ERASE command is encountered in a DATAPLOT program, the screen is immediately erased. The ERASE command will only be used once in this program-- to erase the screen before the formation of the first plot. We will not, of course, be erasing the screen before any of the remaining 8 plots.

XLIMITS 1974 1980
specifies (for future plots) that the limits for the horizontal axis will be fixed at 1974 and 1980. The rationale for this command statement is to assure that both bar plots have identical horizontal axis limits--this will allow the second bar plot to be superimposed exactly atop the first bar plot. If this command were not put in, then the DATAPLOT default (neat limits which float with the data) could result in different horizontal axis limits for the second bar plot--which would cause erroneous results when the second bar plot was overlayed atop the first bar plot.

YLIMITS 0 6
specifies (for future plots) that the limits for the vertical axis will be fixed at 0 and 6. As before, the rationale for this command statement is to assure that both bar plots have identical vertical axis limits--this will allow the second bar plot to be superimposed exactly atop the first bar plot. If this command were not put in, then the DATAPLOT default (neat limits which float with the data) could result in different vertical axis limits for the second bar plot--which would cause erroneous results when the second bar plot was overlayed atop the first bar plot.

BAR WIDTH .2
specifies (for future bar plots) that the total width of each bar be .2 (in units of the horizontal axis variable). The BAR WIDTH command allows the analyst control how isolated each bar is. Does the analyst want "fat" bars which abutt up against one another with no gap at all between adjacvcent bars; or does the analyst want "skinney" bars which have a large gap between adjacent bars? Note that the units of the BAR WIDTH command is the units of the horizontal axis variable.

In this case, since the horizontal axis variable ranges from 1975 through 1979, a bar spacing of .2 will cause each of the bars to have a width of .2 unit (and therefore a gap of .8 units).

The default bar width is such that the bars will abutt up against one another.