Introduction and Overview

\[ \sum \]

\[ y = f(x) \]

\[ \Pi \]

\( f(x) = 0 \)

\( f'(x) = 0 \)

\( \frac{d}{dx} f \)

\( f * g \)
NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards1 was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities2 — Radiation Research — Chemical Physics — Analytical Chemistry — Materials Science

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:


THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

1Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
2Some divisions within the center are located at Boulder, CO 80303.
Preface

The DATAPLOT language was designed and developed in 1976 in response to data analysis problems encountered in the scientific/research environment at the National Bureau of Standards. At such time it became increasingly evident that the 3 highest priority computing activities of the scientist/engineer were

1. graphics (especially continuous);
2. fitting (especially non-linear);
3. operations with functions.

Up to that time, the generation of such continuous graphics, the carrying out of such non-linear fits, and the application of such function operations was typically done in a subroutine-dependent, batch-oriented fashion which was time-consuming and non-conducive to the continuity of thought which is so important in scientific investigations. The need was seen, therefore, for the development of a

high-level,
English-syntax,
interactive

language which included single-command capabilities for

continuous graphics;
non-linear fitting;
function evaluation/transformations.

The realization of this goal occurred in early 1977 with the NBS implementation of the DATAPLOT language with its 3 kernel commands--

PLOT (for graphics);
FIT (for fitting);
LET (for function operations).

These 3 commands were from the beginning the core of the DATAPLOT language and their central role as such has become a distinctive mark of the language.

In addition to the above core activities, however, the language has extensive capabilities in

1) graphics (continuous or discrete);
2) fitting (non-linear or linear);
3) general data analysis;
4) mathematics.

DATAPLOT commands are high-level, English-syntax, and self-descriptive, such as

PLOT Y X
PLOT EXP(-X**2) FOR X = -3 .1 3
FIT Y = A+B*EXP(-ALPHA*X)
BOX PLOT Y X
ANOVA Y X1 X2 X3
LET FUNCTION F = DERIVATIVE EXP(-X**2)/(A+B*X) WRT X

Although DATAPLOT may be used in a batch environment, the language was designed for (and is most effectively used in) an interactive environment. DATAPLOT has proven to be a valuable time-saving tool from the early stages of an analysis through to the final presentation graphics for the scientist, engineer, data analyst, and general researcher alike. No programming experience is assumed in using DATAPLOT; it is used by both the novice programmer and by the veteran analyst.

This book should serve as a comprehensive reference for all DATAPLOT programmers. It is the author's intention that the layout of this book will

- facilitate the learning of the DATAPLOT language;
- encourage the use of the many features of the language;
- result in more thorough and insightful analyses; and
- result in significant savings of the analyst's time.

Indeed, the use of the DATAPLOT language as a computing tool in solving the variety of problems which arise daily in a research environment will allow the scientist to "think science" as opposed to "think computing".

The National Technical Information Service (NTIS) has been distributing the NBS-DATAPLOT tape since March of 1982. There have been 3 major releases--

March 1982--ANSI FORTRAN 66, General Version
July 1982--ANSI FORTRAN 77, General Version
June 1983--ANSI FORTRAN 77, Vax-Specific Version

The language defined in this book is consistent with the source code released from NBS to NTIS for distribution as of September 1983. Prior versions may differ slightly from specifications outlined in this manual. In particular, the LOOP command, the IF command, the VALU() sub-command, the bar plot patterns/spacing, and the symbolic differentiation do not apply for earlier versions.

The NBS source code tape which NTIS distributes is portable-out of the 600 ANSI FORTRAN subroutines (= 200,000 lines of FORTRAN code), only a few (less than 10) of the subroutines need be changed in going from one computer to the next. Known implementations exist on Univac, Vax, Honeywell, and Prime computers; other implementations are in progress. For distribution information, contact
This book has had the benefit of constructive and insightful comments from a number of different individuals—both inside and outside of NBS. Particular thanks are extended to my NBS colleagues Stefan Leigh and Roland DeWit whose excellent reviewing and editing served to substantially improve earlier versions of this manual. Additionally, the comments of my colleagues in the Center for Applied Mathematics at NBS have been extremely valuable in regard to both this manual and the DATA PLOT language. In spite of the above, however, there no doubt still remain discrepancies, errors, omissions, and other technical ambiguities in this manual. In such case, the author welcomes the opportunity of receiving feedback from readers, and will be glad to make corrections and clarifications. Please direct such questions and comments to

James J. Filliben
Statistical Engineering Division
Center for Applied Mathematics
Administration Building, Room A-337
National Bureau of Standards
Washington, D. C. 20234
301-921-3651

James J. Filliben
May 1984
Disclaimer

The purpose of the manual is to give the analyst a broad overview of the structure, capabilities, and features of the DATAPLOT language.

Certain commercial brand names (e.g., Tektronix, Calcomp, Texas Instruments, etc.) are mentioned in this document. This does not constitute an endorsement of such products by either the author or the United States Government.

The features and capabilities described in this primer are for version 84/7 of DATAPLOT. Most (but not all) descriptions also hold for prior versions.

Neither the author nor any branch of the United States Government may be held liable for errors arising from the use of the DATAPLOT language or the implemented DATAPLOT system.

Suggestions and comments are welcome. Please forward them to

James J. Filliben
National Bureau of Standards
Administration Building, Room A-340
Washington, D.C. 20234
301-921-3651
Changes from Previous Versions

The National Technical Information Service (NTIS) has distributed 3 prior versions of DATAPLOT—

Version 82/3 FORTRAN 66 (General)
Version 82/7 FORTRAN 77 (General)
Version 83/5 FORTRAN 77 (Vax)

Version 84/7 will be released from NTIS shortly (July 1984), and will be primarily distinguished from earlier versions in that it will have the benefit of code updates (for certain uncovered bugs) and capability enhancements (for added analysis power).

The features and capabilities described in this manual are for version 84/7 of DATAPLOT. Most (but not all) descriptions also hold for the prior released versions (82/3 and 82/7).

Commands available in 84/7 but not in earlier versions include—

READ PARAMETER for reading parameters from a file;
READ FUNCTION for reading functions from a file;
LOOP for looping;
IF for conditional execution of a block of statements;
LIST for listing a subprogram on a file.

The MACRO ON, MACRO OFF, and MACRO EXECUTE commands of previous versions are described in this manual as the CREATE ON, CREATE OFF, and CALL commands. The MACRO commands will still work in 84/7, but the preferable commands are CREATE and CALL.

The ANGLE UNITS command of previous versions is described in this manual as the TRIG UNITS command—either will work in 84/7.

The COMMENT command in previous versions (to write out descriptive text) is described in this manual as a sub-capability under the WRITE command, so that

COMMENT CALIBRATION ANALYSIS

and

WRITE "CALIBRATION ANALYSIS"

both have the same effect in 83/10.

The analytic differentiation sub-capability under the LET FUNCTION command has been reinstated for version 84/7, but was omitted from earlier versions.

New data analysis capabilities which have been added to the 84/7 version are

1) The BOX PLOT command has been generalized (via the FENCE command) so that values beyond the inner and outer fences may be indicated.

2) Robust (3RSR) smoothing and Hamming smoothing have been added.

3) Stem and Leaf diagrams have been added.

4) The CONTROL CHART command has been generalized to cover the unequal-sample size case.

James J. Filliben
May 1984
Table of Contents

1 General

What is DATAPLOT? 1-1
Capabilities 1-1
Getting Into and Out of DATAPLOT 1-2
Getting Started with DATAPLOT 1-3
4 Typical Problems 1-5
4 DATAPLOT Solutions 1-7
Programming with DATAPLOT 1-11
English-Syntax Language 1-12
Free-Format Language 1-12
Declaration-Free Language 1-13
Structured Language 1-13
Punctuation 1-14
Language Components 1-14

2 Command Categories

Commands 2-1
Graphics Commands 2-2
Analysis Commands 2-3
Diagrammatic Graphics Commands 2-4
Plot Control Commands 2-5
Support Commands 2-6
Output Device Commands 2-8
Keywords 2-9
3 Parameters, Variables, and Functions

Arithmetic Operations 3-1
Relational Operations 3-1
Numbers 3-2
Parameters
LET 3-3
Variables
LET, READ, and SERIAL READ 3-4
Functions
LET FUNCTION 3-5
Evaluating Functions
LET 3-5
Sub-commands Under the LET Command
Statistics, Mathematics, Random Numbers, Manipulation
LET 3-6
Built-in Library Functions
LET, LET FUNCTION, PLOT, 3D-PLOT, FIT, and PRE-FIT 3-12
Creating Parameters, Variables, and Functions
LET, LET FUNCTION, READ, and SERIAL READ 3-15
Copying Parameters, Variables, and Functions
LET 3-15
Deleting Parameters, Variables, and Functions
DELETE 3-16
Assigning Multiple Names to a Variable
NAME 3-18
Creating Data Internally
LET, READ, and SERIALY READ 3-19
4 Input/Output

Files and Subfiles 4-1
Input/Output 4-3
Reading in Columns of Data READ 4-4
Reading Data in Sequentially SERIAL READ 4-7
Reading in Parameters READ PARAMETER 4-8
Reading in Functions READ FUNCTION 4-9
Writing Out Parameters, Variables, and Functions WRITE 4-10
Skipping Over Lines in a Read SKIP 4-11
Restricting a Read to Certain Rows ROW LIMITS 4-11
Restricting a Read to Certain Columns COLUMN LIMITS 4-12
Terminating a Read END OF DATA 4-12

5 Program Control

Looping LOOP 5-1
Conditional Statements SUBSET, EXCEPT, FOR, and IF 5-3
Calling Subprograms CALL 5-5
Creating Subprograms CREATE 5-6
Listing Subprograms LIST 5-6
6 Miscellaneous

Restricting Plots and Analyses to Subsets
SUBSET, EXCEPT, and FOR

In-line Text Sub-commands
TITLE, ...LABEL, LEGEND, and TEXT

Defaults, Restrictions, and Settings

Dumping Status Information
STATUS

Dumping On-Line Documentation
HELP

Redimensioning Internal Storage
DIMENSION

Specifying the Terminal
TERMINAL

Specifying the Host
HOST

Specifying the Communications Link
COMMUNICATIONS LINK and BAUD RATE

Specifying Trigonometric Units
TRIGONOMETRIC UNITS

Echoing Commands
ECHO

Suppressing Printed Output
FEEDBACK and PRINTING

Diverting Graphics Output to Offline Devices
PENPLOTTER, ZETA, CALCOMP, and VERSATEC

Redefining I/O Units and Saving DATAPLOT Output
SET

Controlling The Terminal--Erasing, Hardcopying, etc.
PRE-ERASE, HARDCOPY, ERASE, and COPY

Activating Local Settings
IMPLEMENT
Entering Comment Lines

6-29

COMMEN T and .

Recommended Programming Practices

6-30

Interrupting, Saving, and Resuming A Run

SAVE and RESTORE

6-32

Starting Over

RESET

6-32

Communicating with the Host Operator

OPERATOR

6-33

Communicating with DATAPLOT Service Group

MESSAGE, NEWS, MAIL, and BUGS

6-34

DATAPLOT File Information for Implementation

6-35

Default Output Device Settings

6-37

Color Graphics

TERMINAL, COLOR, and PICTURE POINTS

6-37

Multiple Commands Per Line

TERMINATOR CHARACTER

6-38

Specifying Post-Plot Cursor Position and Size

CURSOR SIZE

6-38
General

What is DATAPLOT? 1-1
Capabilities 1-1
Getting Into and Out of DATAPLOT 1-2
Getting Started with DATAPLOT 1-3
4 Typical Problems 1-5
4 DATAPLOT Solutions 1-7
Programming with DATAPLOT 1-11
English-Syntax Language 1-12
Free-Format Language 1-12
Declaration-Free Language 1-13
Structured Language 1-13
Punctuation 1-14
Language Components 1-14
What is DATAPLOT?

DATAPLOT is an interactive high-level (free-format English-like syntax) language with extensive capabilities in

1) Graphics (continuous or discrete);
2) Fitting (non-linear or linear);
3) General Data Analysis;
4) Mathematics.

DATAPLOT was developed originally in 1977 in response to data analysis problems encountered at the National Bureau of Standards. DATAPLOT may be run either in batch mode or interactively, although it was primarily designed for (and is most effectively used in) an interactive environment. DATAPLOT graphics may appear on many different types of output devices.

Capabilities

DATAPLOT has a broad scope of capabilities--

1) raw graphics (2-D, 3-D, color, single/multiple plots per page, etc.);
2) analysis graphics (plotting data, plotting functions);
3) presentation graphics (special fonts, Greek, math symbols, word charts, etc.);
4) diagrammatic graphics (diagrams, schematics, etc.);
5) graphical data analysis (box plots, probability plots, histograms, control charts, lag plots, EDR, residual analysis, etc.);
6) time series plots (autocorrelation plots, spectral plots, demodulation plots, etc.);
7) fitting (non-linear, multi-linear, polynomial, spline, etc.);
8) general data analysis (ANOVA, median polish, robust smoothing, ANOP, etc.);
9) statistical/probability capabilities (.20 statistics, cdf's, ppf's, random numbers, etc.);
10) mathematics (roots, differentiation, integration, convolution, etc.);
11) subset analyses (as easy as full-set analyses);
12) extensive built-in Fortran-like library of math and probability functions.
Getting Into and Out of DATAPLOT

The procedure for accessing DATAPLOT varies from one computer to the next. For example, on a Univac computer, one enters

@DATA

On a Vax computer, one enters

DATA

Check with the local DATAPLOT service group to determine the local convention for accessing DATAPLOT. Upon entry into DATAPLOT, the DATAPLOT banner will appear, along with a version number, and dimension message about the current configuration, as in--

*****************************************************************
*                DATAPLOT                      *
*    INTERACTIVE GRAPHICAL DATA ANALYSIS LANGUAGE               *
*****************************************************************
VERSION 83/1

INTERNAL DATA STORAGE CAPACITY = 10,000 OBSERVATIONS
(1000 OBSERVATIONS PER VARIABLE X 10 VARIABLES)

To exit from a DATAPLOT program, one enters any of the following--

EXIT
END
STOP
HALT
BYE
Getting Started with DATAPLOT

Suppose the analyst has the following 5 observations on the variables X and Y--

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
</tr>
</tbody>
</table>

Enter a DATAPLOT program to carry out the following 4 operations--

1) Compute the mean of the Y data;
2) Plot Y versus X with titles and labels;
3) Carry out a least squares fit of Y on X with a linear model;
4) Generate a plot with the fitted values from the linear fit on top of the raw data.

To do so, one first invokes DATAPLOT (e.g., by entering @DATAPLOT on UNIVAC computers, and whatever is appropriate on other computers). Upon invoking DATAPLOT, a DATAPLOT banner and a few lines of messages will appear on the screen.

One is now ready to enter the DATAPLOT program. A program to carry out the above 4 operations is as follows--

```
. STEP 0--READ IN THE DATA
  READ X Y
  1 1
  2 3
  3 15
  4 18
  5 30
END OF DATA

. STEP 1--COMPUTE THE MEAN
  LET M = MEAN Y

. STEP 2--PLOT THE DATA
  TITLE CALIBRATION ANALYSIS
  YLABEL RESPONSE
  XLABEL FORCE
  PLOT Y X

. STEP 3--FIT THE DATA
  FIT Y=A*B*X+C*X**2

. STEP 4--GENERATE A SUPERIMPOSED PLOT AFTER THE FIT
  CHARACTERS X BLANK
  LINES BLANK SOLID
  PLOT Y FRED VERSUS X

EXIT
```

All lines beginning with . (period) are non-executing comment lines and so DATAPLOT will ignore them; they are included here for clarity. There are 15 such lines in the above program. Ignore them or enter them at your discretion. The existence of such comment lines is more common in a DATAPLOT program to be stored in a file (and run later in toto) than in a DATAPLOT program to be entered line by line and run interactively.

The READ statement will read data into variables X and Y. The read is format-free and so spacing between data values on a line is unimportant (but at least one space between data values must exist--3 15 could not be written as 315). Since there are only 5 data pairs, they are entered directly from the terminal (as opposed to reading the data from a file). The read terminates when an END OF DATA line is entered.

LET M = MEAN Y computes the mean of the data in the variable Y, prints out the value, and stores this value in the internal DATAPLOT parameter M. DATAPLOT has over 20 summary statistics (MEAN, MEDIAN, STANDARD DEVIATION, CORRELATION, etc.) which may be similarly invoked via the LET command.

TITLE CALIBRATION ANALYSIS defines the title to appear above subsequent plots; YLABEL RESPONSE defines the vertical axis label to appear on subsequent plots; XLABEL FORCE defines the horizontal axis label to appear on subsequent plots. PLOT Y X actually generates the plot--Y will be plotted vertically and X horizontally. The plot will have a solid line (the default) connecting the data points. The pre-defined titles and labels will be automatically written out on the plot. The plot limits will be automatic and neat.

FIT Y=A*B*X+C*X**2 carries out a least squares linear fit of Y on X. The coefficients and other results from the fit are be printed out. Predicted (=fitted) values from the fit are automatically placed in a variable PRED, and residuals from the fit are automatically placed in the variable RES. The above model (linear) is simple--DATAPLOT may carry out least squares fits for not only such polynomial and multi-linear models, but also for general non-linear models.

CHARACTERS X BLANK specifies that the first trace on subsequent plots should have X's as the plot character and the second trace should have blank (=no) plot character. LINES BLANK SOLID specifies that the first trace on subsequent plots should have a blank (=no) connecting line, and the second trace should have a solid connecting line. PLOT Y PRED VERSUS X generates the superimposed plot of the raw data Y and predicted values PRED as a function of X. Due to the prior CHARACTERS and LINES commands, the first trace (Y versus X) has X's as characters and no connecting line; the second trace (PRED VERSUS X) has no plot characters but solid connecting lines.

EXIT terminates the DATAPLOT run.
The output from the above DATAPLOT program is as follows:

The mean of the 5 observations = 0.13400000 + 002
The computed value of the parameter M = 0.13400000 + 002

If the analyst were running on an alphanumeric terminal (e.g., Texas Instrument Silent 700) then the code should be preceded by a DISCRETE command which instructs DATAPLOT that the terminal is essentially non-continuous (has no ability to draw a continuous straight line from any point on the screen to any other point on the screen) and therefore the plots should be suitably modified. Also, the BLANK in

CHARACTERS X BLANK

should be replaced by some non-blank character, as in

CHARACTERS X P

The 2 plots will thus appear as:

[Calibration Analysis Plots]

LEAST SQUARES NON-LINEAR FIT
SAMPLE SIZE N = 5
MODEL: Y = A + BX
NO REPLICATION CASE

ITERATION CONVERGENCE RESIDUAL PARAMETER
NUMBER MEASURE STANDARD ESTIMATES

1 10000000 + 001 0.10000000 + 001 0.10000000 + 001
2 0.0000000 + 002 0.0000000 + 002 0.0000000 + 002

FINAL PARAMETER ESTIMATES (APPROX. ST. DEV.)
A 0.0000000 + 002 (0.0000000 + 002)
B 7.3000000 (0.9013)

RESIDUAL STANDARD DEVIATION = 3.0713732541
RESIDUAL DEGREES OF FREEDOM = 3

[Calibration Analysis Plots]
4 Typical Problems

A computer language is a tool—a means of generating solutions to problems. Before delving into the details of the DATAPLOT language, let us first consider the types of problems that the scientist/engineer/research scientist typically encounters. This will provide motivation for how the computer language (as a tool) was developed. All computer languages have their own areas of strength. The choice of problems below serve as a frame of reference for both reader and developer alike as to the type of problems that DATAPLOT considers "important" and for which it has been designed to be strong in.

1) A Graphics Problem--
An analyst has a data set consisting of \((x, y)\) pairs. Plot the data. "Blow up" any interesting sub-regions of the plot.

2) A Non-Linear Fitting Problem--
An analyst has a data set consisting of \((x, y)\) pairs. Read the data into the computer. Plot them. Carry out a non-linear fit for the model

\[ y = \exp(-\alpha x)/(a + b x) \]

Generate a superimposed plot of raw data and predicted values from the fit. Generate a plot of residuals versus \(x\). Generate a normal probability plot of the residuals.

3) A Data Analysis Problem--
An analyst has data consisting of a response variable and 3 independent variables (factors). Determine if the factors affect the response. Determine if there is interaction between the factors. Carry out an analysis of variance. Carry out a graphical analysis of variance.

4) A Mathematics Problem--
An analyst wishes to examine the function \(x \exp(-x) + \sin(x^2)\) over the interval 0 to 1. Plot the function over the interval. Determine any roots in the interval. Determine its definite integral over the interval.

Several points are noteworthy--

1) Graphics as a Core Activity.
   Note the graphics component that exists in all of the above problems--graphics is a key activity in both data analysis and mathematics.

2) Time.
   These problems should all be solvable with less than 10 lines of code.

3) Number of Lines of Code.
   These problems should all take less than 10 minutes to solve.

4) Interactive Analysis.
   These problems should be solvable interactively so that in case some interesting tangent arises in the course of the solution, the analyst may immediately pursue it.

5) Graphics Quality.
   Ideally, the graphics should be all continuous and of manuscript-quality if so desired.

   On the other hand, if the analyst is working at a discrete terminal or in batch, neither the logic of the analysis nor the entered plot commands should be any different.

7) Subset Analyses.
   The analyst should not need to worry about whether the graphics, fitting, or data analysis is being carried out over the full data set or over any complicated subsets of the data. Carrying out analyses over subsets should not result in irrelevant (as far as the scientist is concerned) preliminary data extraction and manipulation. It should be as easy to carry out any (and all) graphics and analysis operations over a subset as it is to carry it out over the full set.

8) Sample Size.
   The analyst should not need to worry about whether the data set consists of 7 data points or 700 data points—the number of data points is a nuisance parameter that the analyst should not need to concern himself with.

9) Data Format.
   The analyst should not need to worry about how the data is formatted upon input—this also is an unimportant nuisance item.
10) Predicted Values and Residuals.
The analyst should not need to worry about predicted values and residuals from the fit—they should be automatically available for further analysis and plotting.

11) Scope of Capabilities.
The analyst should be able to fluidly glide from graphics to fitting to data analysis to mathematics activities with no interruption and within the context of the language.

12) Ease of Use.
The ultimate objective of the analyst is not in learning a computer language—it is in gaining insight into the problem at hand; thus the computer language should be natural, easy to learn, and easy to use. The language that the analyst uses should preserve the continuity of thought that is so important in scientific research.

13) English-Syntax.
Ideally, the language should correspond as close as possible to the English-language and mathematical representation of the solution. This will allow the analyst to "think science" as opposed to "think computing" and will eliminate an unnecessary mapping from conceptual solution to computer-language solution.
4 DATAPLOT Solutions

1. The Graphics Problem---
An analyst has a data set consisting of
\((x,y)\) pairs. Plot the data. *Blow up*
any interesting sub-regions of the plot.

The DATAPLOT program and output is as follows---

```
READ FILE1. X Y
PLOT Y X
PLOT Y X SUBSET X 400 TO 600
```
2. The Non-Linear Fitting Problem
An analyst has a data set consisting of \((x,y)\) pairs. Read the data into the computer. Plot them. Carry out a non-linear fit for the model

\[ y = \exp(-\alpha x)/(a+b x) \]

Generate a superimposed plot of raw data and predicted values from the fit. Generate a plot of residuals versus \(x\). Generate a normal probability plot of the residuals.

The DATAPLOT program and output is as follows--

```
READ FILE2. X Y
PLOT Y X
LET A=0
LET B = 1
LET ALPHA=.1
FIT Y = EXP(-ALPHA*X)/(A+B*X)
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
PLOT RES X
NORMAL PROBABILITY PLOT RES
```

```
LEAST SQUARES NON-LINEAR FIT
SAMPLE SIZE N  =  214
Y=EXP(-ALPHA*X)/(A+B*X)

REPLICATION STANDARD DEVIATION  =  3.281762669X10^-01
REPLICATION DEGREES OF FREEDOM  =  102
NUMBER OF DISTINCT SUBSETS  =  22

<table>
<thead>
<tr>
<th>ITERATION</th>
<th>CONVERGENCE</th>
<th>RESIDUAL</th>
<th>PARAMETER</th>
<th>MEASURE</th>
<th>STANDARD</th>
<th>ESTIMATES</th>
<th>DEVIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.19000-001</td>
<td>.45962-002</td>
<td>.10933-001</td>
<td>.88888-000</td>
<td>.88888-001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>.58909-002</td>
<td>.32787-003</td>
<td>.13197-001</td>
<td>.78965-000</td>
<td>.78965-001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.72866-001</td>
<td>.47043-001</td>
<td>.15786-000</td>
<td>.49103-000</td>
<td>.49103-001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.70508-001</td>
<td>.39644-001</td>
<td>.18662-000</td>
<td>.98389-000</td>
<td>.98389-001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.62648-003</td>
<td>.31617-001</td>
<td>.10562-000</td>
<td>.61376-000</td>
<td>.61376-001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FINAL PARAMETER ESTIMATES (APPROX ST. DEV.)
1 ALPHA  = .00000-000 ( .22883-001 )
2 A     = .01327-002 ( .3429-003 )
3 B     = .16526-001 ( .9827-003 )

RESIDUAL STANDARD DEVIATION  =  3.361673128
REPLICATION DEGREES OF FREEDOM  =  211
REPLICATION STANDARD DEVIATION  =  3.281762669
REPLICATION DEGREES OF FREEDOM  =  102
LACK OF FIT F RATIO  =  1.5474 = THE .02,0461% POINT OF THE F DISTRIBUTION WITH 10 AND 102 DEGREES OF FREEDOM
```
3. The Data Analysis Problem---
An analyst has data consisting of a response variable and 3 independent variables (factors). Determine if the factors affect the response. Determine if there is interaction between the factors. Carry out an analysis of variance. Carry out a graphical analysis of variance.

The DATAPLOT program and output is as follows---

READ FILE3, Y X1 X2 X3
ANNOVA Y X1 X2 X3
.
LET TAG = X2+2*(X3-1)
CHARACTERS 1 1 2 2
LINES SOLID DOTTED SOLID DOTTED
PLOT Y X1 TAG

```
+-----------------------------------------------------------------
+ 3-WAY ANALYSIS OF VARIANCE
+-----------------------------------------------------------------
+ NUMBER OF OBSERVATIONS       =   3
+ NUMBER OF FACTORS            =   2
+ NUMBER OF LEVELS FOR FACTOR 1 =   2
+ NUMBER OF LEVELS FOR FACTOR 2 =   3
+ RESIDUAL DEGREES OF FREEDOM  =   4
+ NUMBER DISTINCT CELLS       =   8
+-----------------------------------------------------------------
+ GRAND MEAN                   = 0.625643154875
+ GRAND STANDARD DEVIATION     = 0.038758078883
+-----------------------------------------------------------------
+ LEVEL-ID       N1  MEAN EFFECT SD(EFFECT)
+  FACTOR 1       -1.00000  4. 187.75000  -77.12500  23.10134
+                -1.00000  4. 262.00000  -77.12500  23.10134
+                1.00000  4. 160.25000  -15.62500  23.10134
+                1.00000  4. 200.50000  -15.62500  23.10134
+  FACTOR 2       -1.00000  4. 167.75000  -17.12500  23.10134
+                -1.00000  4. 282.00000  -17.12500  23.10134
+                1.00000  4. 167.75000  -17.12500  23.10134
+                1.00000  4. 282.00000  -17.12500  23.10134
+  FACTOR 3       -1.00000  4. 187.75000  -77.12500  23.10134
+                -1.00000  4. 262.00000  -77.12500  23.10134
+                1.00000  4. 160.25000  -15.62500  23.10134
+                1.00000  4. 200.50000  -15.62500  23.10134
+-----------------------------------------------------------------
+ MODEL
+-----------------------------------------------------------------
+ RESIDUAL STAND. DEVIATION
+-----------------------------------------------------------------
+ CONSTANT ONLY       = 90.2564315487
+ CONSTANT & FACTOR 1 ONLY = 50.2564315487
+ CONSTANT & FACTOR 2 ONLY = 185.2564315487
+ CONSTANT & ALL 3 FACTORS = 65.3404541816
+-----------------------------------------------------------------
+ NUM. LEVELS   F STAT.   F CDF
+-----------------------------------------------------------------
+ FACTOR 1       = 2  11.1450229025  0.071138
+ FACTOR 2       = 2  4.5747328177  0.46189
+ FACTOR 3       = 2  5.4952242296  0.08333
+-----------------------------------------------------------------
+ RESIDUAL STAND. DEVIATION = 65.3404541816
+ RESIDUAL DEGREES OF FREEDOM = 4
+-----------------------------------------------------------------
```
4. The Mathematics Problem---
An analyst wishes to examine the function \( x \cdot \exp(-x) + \sin(x^2) \) over the interval 0 to 3. Plot the function over the interval. Determine any roots in the interval. Determine its definite integral over the interval.

The DATAPLOT program and output is as follows---

```
LET FUNCTION F = X*EXP(-X)+SIN(X**2)
PLOT F FOR X = 0 .1 3
LET R = ROOTS F WRT X FOR X = 0 TO 3
LET INT = INTEGRAL F WRT X FOR X = 0 TO 3
```

```
ROOTS OF AN EQUATION
FUNCTION=(X*EXP(-X)+SIN(X**2))
ROOT VARIABLE = X
SPECIFIED LOWER LIMIT OF INTERVAL = 0
SPECIFIED UPPER LIMIT OF INTERVAL = 3
NUMBER OF ROOTS FOUND IN INTERVAL = 3
ROOT 1 = .00000000
ROOT 2 = .19537877+001
ROOT 3 = .2464136+001

THE NUMBER OF VALUES GENERATED FOR THE VARIABLE R = 3
THE FIRST COMPUTED VALUE OF R = .00000000 (ROW 1)
THE LAST (3RD) COMPUTED VALUE OF R = .2464136+001 (ROW 3)
THE CURRENT COLUMN FOR THE VARIABLE R = 1
THE CURRENT LENGTH OF THE VARIABLE R = 3
THE COMPUTED VALUE OF THE CONSTANT NROOTS = .3000000+001

INTEGRAL EVALUATION
FUNCTION=(X*EXP(-X)+SIN(X**2))
SPECIFIED LOWER LIMIT OF INTEGRAL = 0
SPECIFIED UPPER LIMIT OF INTEGRAL = 3
NUMBER OF VARIABLES OF INTEGRATION = 1

+-------------------------------+
<table>
<thead>
<tr>
<th>PARTITIONS</th>
<th>VALUE OF INTEGRAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.1574414+001</td>
</tr>
<tr>
<td>2</td>
<td>.1574414+001</td>
</tr>
</tbody>
</table>
+-------------------------------+

INTEGRAL VALUE = .1574414+001
THE COMPUTED VALUE OF THE CONSTANT INT = .1574414+001
```
Programming with DATAPLOT

DATAPLOT programs proceed sequentially from the beginning of the code to the end. The language is structured. There are no "go-to"-type statements and no statement labels.

Like BASIC, FORTRAN, etc., DATAPLOT has a series of low-level commands which specify to the computer that certain elementary operations (such as reading, writing, looping, etc.) should be carried out.

In addition, however, DATAPLOT goes beyond BASIC and FORTRAN in that it allows the analyst to also make use of a set of high-level commands (such as PLOT, FIT, HISTOGRAM, etc.) which permit the analyst to carry out a wide variety of graphics (continuous or discrete), non-linear fitting, data analysis, and mathematical operations directly.

Like BASIC, DATAPLOT is an interpretive language. Although the language may be used in a semi-interactive mode (running pre-stored programs at a terminal), and a batch mode (running pre-stored programs with remote output on the batch high-speed printer), it was primarily designed for (and is most effectively used in) an interactive environment.

Because of the existence of commonly-used high-level graphical and analytical capabilities, the typical DATAPLOT program is short (e.g., 5 to 20 lines). Usually there is only a main program with no need of subprograms. If subprograms are desired, they are accessed via the CALL command.

Input and output in DATAPLOT is format-free. The READ and SERIAL READ command enter data into DATAPLOT, the WRITE (with PRINT as a synonym) command allows the writing of data (back to the terminal screen or out to a file/subfile).

The DATAPLOT command vocabulary consists of over 100 commands. The analyst may use any command at any time. A given program for a given application typically makes use of a small subset of DATAPLOT commands (10 to 15 commands). The 3 most important DATAPLOT commands are PLOT, FIT, and LET. Other commonly-used commands are CHARACTERS, LINES, TITLE, ...LABEL (as in YLABEL, XLABEL, X2LABEL, and X3LABEL), FONT, TEXT, ECHO, HARDCOPY, READ, WRITE, and STATUS.

The universal separator between words on a DATAPLOT command line is a blank (1 or more blanks). The comma is never used in any DATAPLOT command syntax; when in doubt, use a blank rather than a comma, as in

```
WRITE A B C
```

rather than

```
WRITE A,B,C
```
**English-Syntax Language**

DATAPLOT is an English-syntax language. Many of the command statements are identical to their English-language counterparts, for example,

```
PLOT
FIT
HISTOGRAM
NORMAL PROBABILITY PLOT
ANOVA
SMOOTH
BOX PLOT
```

Thus to generate a plot of the function \( \frac{\sin(x)}{x} \) for the values starting with \( x = .1 \), at increments of .1, and ending with \( x = 12 \), one would enter

```
PLOT S\( \text{IN}(X)/X \) FOR X = .1 .1 1.2
```

And if the analyst has data in variables \( Y \) and \( X \) and wishes to carry out a least squares fit of \( Y \) on \( X \) with the model \( \hat{y} = a + b\exp(-c\cdot x) \), one would enter

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

And if the analyst has data in the variable \( X \) and wishes to form a new variable \( Y \) defined as the natural logarithm of \( X \), one enters

```
LET Y = LOG(X)
```

The net result is that DATAPLOT programs are easy to write, easy to understand, and easy to update. This is particularly important for the analyst with no prior programming experience.

**Free-Format Language**

Command statements appear free-format in columns 1 to 80 of a program line.

The usual separator between components of a command line is a blank (one or more blanks). Thus to generate a plot of \( x^{*}2 \) over the interval of \( x \) values starting with \( x = -3 \), at increments of .1, and ending with \( x = +3 \), the following is incorrect--

```
PLOT X**2 FOR X = -3 .1 3
```

while the following is correct--

```
PLOT X**2 FOR X = -3 .1 3
```

Note the spacing between PLOT, \( x^{*}2 \), FOR, \( X \), -3, .1, AND 3. The spacing here is 1 blank, but optionally could have been any number of blanks. The PLOT statement as given here starts in column 1, but optionally could have been any column.

Packing of characters is permitted only in defining and using functions. Thus

```
PLOT 3 + 2*EXP(-X) FOR X = 0 .1 6
```

is correct, as is

```
PLOT 3 + 2*EXP(-X) FOR X = 0 .1 6
```

and other variations.

If command statements are longer than 80 characters, then they may be extended onto the next line by appending \( * \) at the end of the first line, as in

```
PRE-FIT Y = A + B*EXP(-C*X) FOR A = 10 1 20 ...
           FOR B = 1 .1 2 FOR C = .5 .01 .6
```

Command statements longer than 2 full lines (160 characters) are not permitted.
Declaration-Free Language

The elements of the DATAPLOT language which the analyst may create, redefine, and operate on are:

- parameters = named constants
- variables = named vectors
- functions = named character strings

Names used for parameters/variables/functions may be up to 8 characters, must start with an alphabetic character, and may thereafter be any combination of the 26 alphabetic characters and the 10 numeric characters. Names which are longer than 8 characters may be used but only the first 8 characters are scanned and internally stored.

DATAPLOT is a declaration-free language—one need not pre-define parameter, variable, and function names in a separate section unto itself (as with ALGOL and PASCAL); in fact, one does not pre-define such elements at all—one simply introduces these elements along the way as needed by the program and as dictated by the analysis.

Once a name is defined (as a parameter, a variable, or a function), it remains that type throughout the DATAPLOT run (thus, for example, if X is used as a variable, it will remain a variable for the entire run). If the analyst chooses to change the use of a name (for example, to change a variable X to a function X) in the middle of a run, then the analyst must first delete the name (via the DELETE command, as in DELETE X), and then reuse the name in the desired fashion.

Structured Language

Program execution—whether in the main routine or within a subprogram—always flows from top to bottom, and so DATAPLOT is by design a structured language. DATAPLOT does not have statement labels and therefore is a "GO-TO-less" language. In practice, because of the existence of higher-level graphics/analysis commands, and because of the DATAPLOT feature of being able to append subset and conditionality qualifiers at the end of any high-level graphics and analysis command, the need for such branching has been virtually eliminated. The net result is a structured, top-to-bottom language structure which greatly facilitates the writing and updating of programs.

Branching within a (sub)program is not permitted; branching between subprograms may be done via the CALL command, as in

```
CALL ANALYSIS.
```

where ANALYSIS is the name of the (analyst-created) file where the subprogram resides (the period at the end of the file name tells DATAPLOT that ANALYSIS is the name of a mass storage file name as opposed to, for example, the name of a parameter, variable, or function).
Punctuation

The universal separator for components in a DATAPLOT command line is the space (= blank). A blank between the words of a command line are important because DATAPLOT uses such spaces as a separator. Spaces around relational operators (e.g., <, >, >=, etc.), arithmetic operators (+, -, *, /, and **), and within arithmetic expressions (e.g., $b**2-4*a*c$) may be included or excluded at the preference of the analyst. Readability considerations suggest that spaces be included around such operators and in such expressions.

Commas serve no purpose in DATAPLOT and may at times be the cause of syntax errors. There are no commands which call for the use of commas in DATAPLOT; when in doubt, leave a space rather than a comma. Blanks and spaces within mathematical and functional expressions are solely for visual convenience—they may be included or excluded at the discretion of the analyst; as in

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

Command statements appear free-format in columns 1 to 80 of a program line. If a statement is longer than this, it may be continued onto the next line by appending a ... at the end of the first line. Statements longer than 2 full lines (160 characters) are not permitted.

DATAPLOT also expects no punctuation in data files which are read via the READ and SERIAL READ commands. Adjacent numbers on a line image may be free-format but should be separated by at least one blank. Commas between numbers will usually be ignored by DATAPLOT, but if error messages are generated because of them on a READ, the commas should be replaced by a blank via the local editor. Alphabetic information up at the beginning of a data file may be skipped over via the SKIP command, as in

```
SKIP 5
```

which will cause subsequent READ and SERIAL READ commands to skip over all information on line images 1 to 5 of the file. The ROW LIMITS command allows the analyst to focus only on specified line images of a file, as in

```
ROW LIMITS 6 100
```

The COLUMN LIMITS command allows the analyst to focus only on certain column limits in reading a file, as in

```
COLUMN LIMITS 1 50
```

which would be a way of omitting non-numeric (or numeric) information beyond column 50.

Language Components

The DATAPLOT language consists of the following components--

1) commands;
2) arithmetic operators;
3) relational operators;
4) numbers;
5) parameters;
6) variables;
7) functions;
8) sub-commands under the LET command;
9) keywords;
10) in-line text sub-commands;
11) file and subfile references.
Command Categories

Commands 2-1
Graphics Commands 2-2
Analysis Commands 2-3
Diagrammatic Graphics Commands 2-4
Plot Control Commands 2-5
Support Commands 2-6
Output Device Commands 2-8
Keywords 2-9
Commands

There are 6 DATAPLOT command categories--

Graphics commands;
Analysis commands;
Diagrammatic Graphics commands;

Plot Control commands;
Support commands;
Output Device Commands.

The first 3 categories are of primary interest. The commands in these categories are all active in the sense that they all *do something*--a plot is generated, an analysis (e.g., an ANOVA or a least squares fit) is carried out, or a geometric figure (e.g., a box or circle) is drawn out on the screen. The commands in these categories could very well constitute an end in itself for the experimental objectives of the analyst.

On the other hand, the commands, in the last 3 categories are of secondary importance--they are rarely end objectives of an analysis; rather these commands typically play some intermediate role in assuring that the analyst carries out precisely the analysis as desired. Typical examples of such secondary commands are

1) reading data in from a file (reading data in is never the end objective--it is only the first step in an analysis);

2) specifying that calculations should be carried out in degrees rather than radians for trigonometric calculations;

3) specifying the title and labels for succeeding plots;

4) specifying line types (e.g., dotted) for succeeding plots;

5) specifying the desired graphics output device (e.g., the Xera plotter).

Clearly such commands are not ultimate objectives in an analysis; on the other hand, the absence of such commands would severely handcuff the analyst in carrying out the desired analysis.
## Graphics Commands

All commands in this category generate a plot. This category provides the analyst a set of graphical analysis tools for plotting data and functions. The most important (and most heavily-used) command in this category is PLOT. Other frequently-used commands are HISTOGRAM and ... PROBABILITY PLOT (especially NORMAL PROBABILITY PLOT).

### Command List

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLOT</td>
<td>Generate plot of var 4/ord func</td>
<td>N/A</td>
<td>PLOT Y X</td>
</tr>
<tr>
<td>3-D PLOT</td>
<td>Generate 3-dimensional plot of var 4/ord func</td>
<td>N/A</td>
<td>3-D PLOT X Y Z</td>
</tr>
<tr>
<td>... HISTOGRAM</td>
<td>Generate histogram-cum, rel, or cum rel</td>
<td>N/A</td>
<td>HISTOGRAM Y</td>
</tr>
<tr>
<td>... FREQUENCY PLOT</td>
<td>Generate frequency plot-cum, rel, or cum rel</td>
<td>N/A</td>
<td>FREQUENCY PLOT Y</td>
</tr>
<tr>
<td>PIE CHART</td>
<td>Generate pie chart</td>
<td>N/A</td>
<td>PIE CHART Y</td>
</tr>
<tr>
<td>PERCENT POINT PLOT</td>
<td>Generate percent point plot</td>
<td>N/A</td>
<td>PERCENT POINT PLOT Y</td>
</tr>
<tr>
<td>... PROBABILITY PLOT</td>
<td>Generate probability plot (24 distributions)</td>
<td>N/A</td>
<td>NORMAL PROBABILITY PLOT Y</td>
</tr>
<tr>
<td>... PPCC PLOT</td>
<td>Generate prob plot corr coeff plot (9 families)</td>
<td>N/A</td>
<td>WILKULL PPCC PLOT</td>
</tr>
<tr>
<td>... NORMALITY PLOT</td>
<td>Generate normality plot (Box-Cox family only)</td>
<td>N/A</td>
<td>BOX-COX NORMALITY PLOT Y</td>
</tr>
<tr>
<td>RUN SEQUENCE PLOT</td>
<td>Generate run sequence plot</td>
<td>N/A</td>
<td>RUN SEQUENCE PLOT Y</td>
</tr>
<tr>
<td>LAG ... PLOT</td>
<td>Generate lag plot for a given lag number</td>
<td>N/A</td>
<td>LAG 1 PLOT Y</td>
</tr>
<tr>
<td>... CORRELATION PLOT</td>
<td>Generate auto- or cross-correlation plot</td>
<td>N/A</td>
<td>AUTOCORRELATION PLOT Y</td>
</tr>
<tr>
<td>... SPECTRAL PLOT</td>
<td>Generate auto- or cross-, etc spectral plot</td>
<td>N/A</td>
<td>SPECTRAL PLOT Y</td>
</tr>
<tr>
<td>... PERIODOGRAM</td>
<td>Generate auto- or cross-periodogram</td>
<td>N/A</td>
<td>PERIODOGRAM Y</td>
</tr>
<tr>
<td>COMPLEX DEDUCTION ... PLOT</td>
<td>Generate complex demodulation amp or phase plot</td>
<td>N/A</td>
<td>COMPLEX DEDUCTION PHASE PLOT Y</td>
</tr>
<tr>
<td>BOX PLOT</td>
<td>Generate box plot</td>
<td>N/A</td>
<td>BOX PLOT Y X</td>
</tr>
<tr>
<td>I PLOT</td>
<td>Generate I plot</td>
<td>N/A</td>
<td>I PLOT Y X</td>
</tr>
<tr>
<td>... CONTROL CHART</td>
<td>Generate mean, sd, or range control chart</td>
<td>N/A</td>
<td>MEAN CONTROL CHART Y X</td>
</tr>
</tbody>
</table>
Analysis Commands

All commands in this category generate printed output. This category provides the analyst with a set of quantitative/statistical/mathematical tools which serve as an analytic complement to the graphical techniques of the prior category. The most important commands in this category are FIT and LET.

Analysis Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>LET</td>
<td>Define var &amp; param; calc stat; roots/diff/inv</td>
<td>N/A</td>
<td>LET Y=X<strong>X+100+EXP(-A*X</strong>2)</td>
</tr>
<tr>
<td>LET FUNCTION</td>
<td>Define &amp; operate on func; differentiate func</td>
<td>N/A</td>
<td>LET FUNCTION F1=AO/(1+B1*X**2)</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>Compute summary statistics</td>
<td>N/A</td>
<td>SUMMARY Y</td>
</tr>
<tr>
<td>... FIT</td>
<td>Perform least squares linear or non-linear fit</td>
<td>N/A</td>
<td>FIT Y=A+B<em>EXP(-C</em>X)</td>
</tr>
<tr>
<td>... PRE-FIT</td>
<td>Perform pre-fit analysis for starting values</td>
<td>N/A</td>
<td>PRE-FIT X**A FOR A = 1 .1 2</td>
</tr>
<tr>
<td>EXACT ... RATIONAL FIT</td>
<td>Perform exact rational function fit</td>
<td>N/A</td>
<td>EXACT 1/2 RATIONAL FIT Y2 X2 Y X</td>
</tr>
<tr>
<td>... SPLINE FIT</td>
<td>Perform spline fit</td>
<td>N/A</td>
<td>CUBIC SPLINE FIT Y X</td>
</tr>
<tr>
<td>... SMOOTH</td>
<td>Perform smoothing of equi-spaced data</td>
<td>N/A</td>
<td>CUBIC SMOOTH Y</td>
</tr>
<tr>
<td>ANOVA</td>
<td>Perform analysis of variance</td>
<td>N/A</td>
<td>ANOVA Y X1 X2</td>
</tr>
<tr>
<td>MEDIAN POLISH</td>
<td>Perform analysis of variance</td>
<td>N/A</td>
<td>ANOVA Y X1 X2</td>
</tr>
</tbody>
</table>
Diagrammatic Graphics Commands

The commands in this category are used for the superposition of text on any output, and for the construction of diagrams, schematics, and specialized charts. The most important commands in this category are `FONT`, `BM`, `JUSTIFICATION`, `ERASE`, `MOVE`, `TEXT`, and `COPY`.

## Diagrammatic Graphics Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FONT</code></td>
<td>Specify script font</td>
<td>N/A</td>
<td><code>FONT TRIPLEX ITALIC</code></td>
</tr>
<tr>
<td><code>HEIGHT</code></td>
<td>Specify height of letters</td>
<td>N/A</td>
<td><code>HEIGHT 10</code></td>
</tr>
<tr>
<td><code>WIDTH</code></td>
<td>Specify width of letters</td>
<td>N/A</td>
<td><code>WIDTH 7</code></td>
</tr>
<tr>
<td><code>BM</code></td>
<td>Specify height and width of letters</td>
<td>N/A</td>
<td><code>BM 30 7</code></td>
</tr>
<tr>
<td><code>JUSTIFICATION</code></td>
<td>Specify justification (left/center/right)</td>
<td>N/A</td>
<td><code>JUSTIFICATION CENTER</code></td>
</tr>
<tr>
<td><code>CASE</code></td>
<td>Specify script case (upper vs. lower)</td>
<td>N/A</td>
<td><code>CASE LOWER</code></td>
</tr>
<tr>
<td><code>ERASE</code></td>
<td>Erase current screen</td>
<td>N/A</td>
<td><code>ERASE</code></td>
</tr>
<tr>
<td><code>COPY</code></td>
<td>Copy the current screen onto local hardcopy</td>
<td>N/A</td>
<td><code>COPY</code></td>
</tr>
<tr>
<td><code>RING BELL</code></td>
<td>Ring bell</td>
<td>N/A</td>
<td><code>RING BELL 10</code></td>
</tr>
<tr>
<td><code>CROSS-BAIR (or CS)</code></td>
<td>Activate cross-bair</td>
<td>N/A</td>
<td><code>CROSS-BAIR A B</code></td>
</tr>
<tr>
<td><code>TEXT</code></td>
<td>Write out script text</td>
<td>N/A</td>
<td><code>TEXT GRAPHICS</code></td>
</tr>
<tr>
<td><code>MOVE</code></td>
<td>Move to given coordinates on the screen</td>
<td>N/A</td>
<td><code>MOVE 50 50</code></td>
</tr>
<tr>
<td><code>DRAW</code></td>
<td>Draw from present location to specified end</td>
<td>N/A</td>
<td><code>DRAW 50 50 60 60</code></td>
</tr>
<tr>
<td><code>POINT</code></td>
<td>Draw a point</td>
<td>N/A</td>
<td><code>POINT 50 50</code></td>
</tr>
<tr>
<td><code>ARROW</code></td>
<td>Draw an arrow</td>
<td>N/A</td>
<td><code>ARROW 50 50 60 60</code></td>
</tr>
<tr>
<td><code>TRIANGLE</code></td>
<td>Draw a triangle</td>
<td>N/A</td>
<td><code>TRIANGLE 50 50 60 50 55 60</code></td>
</tr>
<tr>
<td><code>BOX</code></td>
<td>Draw a box</td>
<td>N/A</td>
<td><code>BOX 50 50 60 60</code></td>
</tr>
<tr>
<td><code>HEXAGON</code></td>
<td>Draw a hexagon</td>
<td>N/A</td>
<td><code>HEXAGON 50 50 60 50</code></td>
</tr>
<tr>
<td><code>CIRCLE</code></td>
<td>Draw a circle</td>
<td>N/A</td>
<td><code>CIRCLE 50 50 60 50</code></td>
</tr>
<tr>
<td><code>Semi-Circle</code></td>
<td>Draw a semi-circle</td>
<td>N/A</td>
<td><code>Semi-Circle 50 50 60 50</code></td>
</tr>
<tr>
<td><code>ARC</code></td>
<td>Draw an arc</td>
<td>N/A</td>
<td><code>ARC 50 50 60 70 40</code></td>
</tr>
<tr>
<td><code>ELLIPSE</code></td>
<td>Draw an ellipse</td>
<td>N/A</td>
<td><code>ELLIPSE 50 50 60 45 70 50</code></td>
</tr>
<tr>
<td><code>OVAL</code></td>
<td>Draw an oval</td>
<td>N/A</td>
<td><code>OVAL 50 50 60 45 70 50</code></td>
</tr>
<tr>
<td><code>DIAMOND</code></td>
<td>Draw a diamond</td>
<td>N/A</td>
<td><code>DIAMOND 50 50 60 45 70 50</code></td>
</tr>
<tr>
<td><code>AMPLIFIER</code></td>
<td>Draw an amplifier</td>
<td>N/A</td>
<td><code>AMPLIFIER 50 50 60 50</code></td>
</tr>
<tr>
<td><code>CAPACITOR</code></td>
<td>Draw a capacitor</td>
<td>N/A</td>
<td><code>CAPACITOR 50 50 60 50</code></td>
</tr>
<tr>
<td><code>GROUND</code></td>
<td>Draw a ground</td>
<td>N/A</td>
<td><code>GROUND 50 50 60 50</code></td>
</tr>
<tr>
<td><code>INDUCTOR</code></td>
<td>Draw an inductor</td>
<td>N/A</td>
<td><code>INDUCTOR 50 50 60 50</code></td>
</tr>
<tr>
<td><code>RESISTOR</code></td>
<td>Draw a resistor</td>
<td>N/A</td>
<td><code>RESISTOR 50 50 60 50</code></td>
</tr>
<tr>
<td><code>AND</code></td>
<td>Draw an and box</td>
<td>N/A</td>
<td><code>AND 50 50 60 50</code></td>
</tr>
<tr>
<td><code>OR</code></td>
<td>Draw an or box</td>
<td>N/A</td>
<td><code>OR 50 50 60 50</code></td>
</tr>
<tr>
<td><code>NAND</code></td>
<td>Draw a nand box</td>
<td>N/A</td>
<td><code>NAND 50 50 60 50</code></td>
</tr>
<tr>
<td><code>NOR</code></td>
<td>Draw a nor box</td>
<td>N/A</td>
<td><code>NOR 50 50 60 50</code></td>
</tr>
</tbody>
</table>
## Plot Control Commands

The commands in this category allow the analyst to specify the details of plots which are generated in the Graphics command category. The most important commands in this category are \texttt{CHARACTERS}, \texttt{LINES}, \texttt{TITLE}, and \texttt{LABEL} (especially \texttt{TIABLE} and \texttt{XLABEL}).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{CHARACTERS}</td>
<td>Specify plot character types</td>
<td>all blank</td>
<td>\texttt{CHARACTERS A B CIRCLE STAR}</td>
</tr>
<tr>
<td>\texttt{CHARACTER SIZES}</td>
<td>Specify size (height) for characters on plots</td>
<td>red</td>
<td>\texttt{CHARACTERS SIZES 3 3 7}</td>
</tr>
<tr>
<td>\texttt{LINES}</td>
<td>Specify line types</td>
<td>all solid</td>
<td>\texttt{LINES SOLID DOT DASH DASH}</td>
</tr>
<tr>
<td>\texttt{TITLE}</td>
<td>Specify title at top of plot</td>
<td>no title</td>
<td>\texttt{TITLE SPECTROMETRIC ANALYSIS}</td>
</tr>
<tr>
<td>\texttt{TITLE SIZE}</td>
<td>Specify title size (height)</td>
<td>automatic</td>
<td>\texttt{TITLE SIZE 4}</td>
</tr>
<tr>
<td>\texttt{LABEL}</td>
<td>Specify labels at sides &amp; bottom of plot</td>
<td>no labels</td>
<td>\texttt{YLABEL RESPONSE}</td>
</tr>
<tr>
<td>\texttt{MINIMUM}</td>
<td>Specify minima for plot axes</td>
<td>automatic</td>
<td>\texttt{MINIMUM 250}</td>
</tr>
<tr>
<td>\texttt{MAXIMUM}</td>
<td>Specify maxima for plot axes</td>
<td>automatic</td>
<td>\texttt{MAXIMUM 300}</td>
</tr>
<tr>
<td>\texttt{LIMITS}</td>
<td>Specify limits (min and max) for plot axes</td>
<td>OFF = no bell</td>
<td>\texttt{LIMITS 250 300}</td>
</tr>
<tr>
<td>\texttt{BELT}</td>
<td>Specify automatic plot bell \texttt{(ON/OFF)}</td>
<td>ON = frame</td>
<td>\texttt{BELT ON}</td>
</tr>
<tr>
<td>\texttt{PRE-SORT}</td>
<td>Specify automatic pre-plot sort \texttt{(ON/OFF)}</td>
<td>ON = pre-sorted</td>
<td>\texttt{PRE-SORT OFF}</td>
</tr>
<tr>
<td>\texttt{SEQUENCE}</td>
<td>Specify auto seq numbering for plots \texttt{(ON/OFF)}</td>
<td>OFF = no seq num</td>
<td>\texttt{SEQUENCE ON}</td>
</tr>
<tr>
<td>\texttt{GRID}</td>
<td>Specify plot grid lines \texttt{(ON/OFF)}</td>
<td>OFF = no grid lines</td>
<td>\texttt{GRID OFF}</td>
</tr>
<tr>
<td>\texttt{LOG}</td>
<td>Specify axis scale as logarithmic \texttt{(ON/OFF)}</td>
<td>OFF = linear scale</td>
<td>\texttt{YLOG ON}</td>
</tr>
<tr>
<td>\texttt{FRAME}</td>
<td>Specify frame \texttt{(ON/OFF)}</td>
<td>ON = frame</td>
<td>\texttt{FRAME OFF}</td>
</tr>
<tr>
<td>\texttt{FRAME CORNER COORDINATES}</td>
<td>Specify plot frame location and shape</td>
<td>15 20 90 90</td>
<td>\texttt{CORNER COORDINATES 20 20 75 90}</td>
</tr>
<tr>
<td>\texttt{LEGEND ...}</td>
<td>Specify legend \texttt{(ON/OFF)}</td>
<td>no legend</td>
<td>\texttt{LEGEND 1 NO CATALYST}</td>
</tr>
<tr>
<td>\texttt{COORDINATES}</td>
<td>Specify legend positioning</td>
<td>20 85</td>
<td>\texttt{LEGEND 1 COORDINATES 75 85}</td>
</tr>
<tr>
<td>\texttt{SILK}</td>
<td>Specify size (height) for legends on plots</td>
<td>red</td>
<td>\texttt{LEGEND 1 COLOR BLUE}</td>
</tr>
<tr>
<td>\texttt{ARROW ...}</td>
<td>Specify location of arrows on plots</td>
<td>no default</td>
<td>\texttt{ARROW 2 CORIZAS 29 80 50 50}</td>
</tr>
<tr>
<td>\texttt{COORDINATES}</td>
<td>Specify location of boxes on plots</td>
<td>no default</td>
<td>\texttt{AXIS 1 COORDINATES 5 5 9 9}</td>
</tr>
<tr>
<td>\texttt{SEGMENT ...}</td>
<td>Specify location of line segments on plots</td>
<td>no default</td>
<td>\texttt{SEGMENT 4 30 80 50 50}</td>
</tr>
<tr>
<td>\texttt{TIC MARK}</td>
<td>Specify tic marks on plots \texttt{(ON/OFF)}</td>
<td>ON = tic marks</td>
<td>\texttt{TIC MARK OFF}</td>
</tr>
<tr>
<td>\texttt{TIC MARK COORDINATES}</td>
<td>Specify plot tic mark positioning</td>
<td>automatic</td>
<td>\texttt{TIC MARK COORDINATES 1500 2500}</td>
</tr>
<tr>
<td>\texttt{TIC MARK POSITION}</td>
<td>Specify plot tic mark positioning</td>
<td>through the frame</td>
<td>\texttt{TIC MARK POSITION INSIDE}</td>
</tr>
<tr>
<td>\texttt{TIC MARK SIZE}</td>
<td>Specify tic mark size</td>
<td>automatic</td>
<td>\texttt{TIC MARK SIZE 3}</td>
</tr>
<tr>
<td>\texttt{TIC MARK LABEL}</td>
<td>Specify tic mark labeling \texttt{(ON/OFF)}</td>
<td>automatic</td>
<td>\texttt{XTIC MARK LABEL MALE FEMALE}</td>
</tr>
<tr>
<td>\texttt{TIC MARK LABEL SIZE}</td>
<td>Specify tic mark labeling size (height)</td>
<td>automatic</td>
<td>\texttt{XTIC MARK LABEL SIZE 2}</td>
</tr>
<tr>
<td>\texttt{EYE COORDINATES}</td>
<td>Specify eye location for 3-dimensional plot</td>
<td>max = 5\texttt{range}</td>
<td>\texttt{EYE COORDINATES 20 20 50}</td>
</tr>
<tr>
<td>\texttt{ORIGIN COORDINATES}</td>
<td>Specify reference origin for 3-dimensional plot \texttt{(min,min,min)}</td>
<td>OFF = no pedestal</td>
<td>\texttt{ORIGIN COORDINATES 50 200 200}</td>
</tr>
<tr>
<td>\texttt{PEDESTAL}</td>
<td>Specify 3-d plot with pedestal \texttt{(ON/OFF)}</td>
<td>OFF = no pedestal</td>
<td>\texttt{PEDESTAL OFF}</td>
</tr>
<tr>
<td>\texttt{PEDESTAL SIZE}</td>
<td>Specify size for pedestal on plots</td>
<td>ON = no pedestal</td>
<td>\texttt{PEDESTAL SIZE 2000}</td>
</tr>
<tr>
<td>\texttt{VISIBLE}</td>
<td>Specify 3-d broken lines visibility \texttt{(ON/OFF)}</td>
<td>ON = visible</td>
<td>\texttt{VISIBLE OFF}</td>
</tr>
<tr>
<td>\texttt{CHARACTER COLORS}</td>
<td>Specify colors for characters on plots</td>
<td>red</td>
<td>\texttt{CHARACTER COLORS RED BLUE GREEN}</td>
</tr>
<tr>
<td>\texttt{LINE COLORS}</td>
<td>Specify colors for lines on plots</td>
<td>red</td>
<td>\texttt{LINE COLORS RED BLUE GREEN YELLOW}</td>
</tr>
<tr>
<td>\texttt{TITLE COLOR}</td>
<td>Specify color for title on plots</td>
<td>red</td>
<td>\texttt{TITLE COLOR ORANGE}</td>
</tr>
<tr>
<td>\texttt{LABEL COLOR}</td>
<td>Specify colors for labels on plots</td>
<td>red</td>
<td>\texttt{LABEL COLOR GREEN}</td>
</tr>
<tr>
<td>\texttt{FRAME COLOR}</td>
<td>Specify colors for frame on plots</td>
<td>red</td>
<td>\texttt{FRAME COLOR RED}</td>
</tr>
<tr>
<td>\texttt{TIC MARK COLOR}</td>
<td>Specify color of tic marks on plots</td>
<td>red</td>
<td>\texttt{TIC MARK COLOR BLUE}</td>
</tr>
<tr>
<td>\texttt{BACKGROUND COLOR}</td>
<td>Specify color of background on plots</td>
<td>blue</td>
<td>\texttt{BACKGROUND COLOR RED}</td>
</tr>
<tr>
<td>\texttt{MARGIN COLOR}</td>
<td>Specify color of margin on plots</td>
<td>blue</td>
<td>\texttt{MARGIN COLOR GREEN}</td>
</tr>
<tr>
<td>\texttt{ARROW ... COLOR}</td>
<td>Specify colors for arrows on plots</td>
<td>red</td>
<td>\texttt{ARROW 2 COLOR BLUE}</td>
</tr>
<tr>
<td>\texttt{SEGMENT ... COLOR}</td>
<td>Specify colors for line segments on plots</td>
<td>red</td>
<td>\texttt{SEGMENT 4 COLOR BLACK}</td>
</tr>
<tr>
<td>\texttt{BOX ... COLOR}</td>
<td>Specify color of box frame on plots</td>
<td>red</td>
<td>\texttt{BOX 7 COLOR GREEN}</td>
</tr>
<tr>
<td>\texttt{PEDESTAL COLOR}</td>
<td>Specify color of pedestal on 3-d plots</td>
<td>red</td>
<td>\texttt{PEDESTAL COLOR BLUE}</td>
</tr>
</tbody>
</table>
Support Commands

The commands in this category allow the analyst to carry out important secondary operations (such as input/output), and to specify a variety of settings which will allow an analysis to be tailored precisely to one's specifications. The most important commands in this category are READ, END OF DATA, WRITE, ECHO, EXIT, STATUS, and HELP.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>Read variables</td>
<td>Read from terminal</td>
<td>READ CALIB. Y X</td>
</tr>
<tr>
<td>SERIAL READ</td>
<td>Read variables serially</td>
<td>Read to terminal</td>
<td>SERIAL READ CALIB. Y X</td>
</tr>
<tr>
<td>SKIP</td>
<td>Specify number of header lines to skip for READ</td>
<td>0 = no lines</td>
<td>SKIP 5</td>
</tr>
<tr>
<td>ROW LIMITS</td>
<td>Specify row limits for READ and SERIAL READ</td>
<td>rows 1 to infinity</td>
<td>ROW LIMITS 50 100</td>
</tr>
<tr>
<td>COLUMN LIMITS</td>
<td>Specify column limits for READ and SERIAL READ</td>
<td>columns 1 to 132</td>
<td>COLUMN LIMITS 10 40</td>
</tr>
<tr>
<td>END OF DATA</td>
<td>Define end of data for READ and SERIAL READ</td>
<td>F/N/A</td>
<td>END OF DATA</td>
</tr>
<tr>
<td>MCRD</td>
<td>Specify auto echo of command lines (ON/OFF)</td>
<td>OFF = no echo</td>
<td>MCRD ON</td>
</tr>
<tr>
<td>FEEDBACK</td>
<td>Allow/suppress all feedback printing (ON/OFF)</td>
<td>ON = allow messages</td>
<td>FEEDBACK OFF</td>
</tr>
<tr>
<td>PRINTING</td>
<td>Allow/suppress all analysis printing (ON/OFF)</td>
<td>ON = printing</td>
<td>PRINTING ON</td>
</tr>
<tr>
<td>RESET</td>
<td><em>zero-out</em> all var, param, func, etc</td>
<td>N/A</td>
<td>RESET</td>
</tr>
<tr>
<td>SAVE</td>
<td>Dump all var/param/fun to mass storage</td>
<td>N/A</td>
<td>SAVE SCRATCH4.</td>
</tr>
<tr>
<td>RESTORE</td>
<td>Restore all saved var/param/fun from mass stg</td>
<td>N/A</td>
<td>RESTORE SCRATCH4.</td>
</tr>
<tr>
<td>EXIT</td>
<td>Exit from DATAPLOT</td>
<td>N/A</td>
<td>EXIT</td>
</tr>
<tr>
<td>STATUS</td>
<td>Print status of all lines, char, var, param</td>
<td>N/A</td>
<td>STATUS</td>
</tr>
<tr>
<td>DIMENSION</td>
<td>Specify dimensions of internal data storage</td>
<td>10 var x 1000 obs</td>
<td>DIMENSION 3D VARIABLES</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete variables or elements of a variable</td>
<td>N/A</td>
<td>DELETE Y(5) X(5) 73(LAB)5</td>
</tr>
<tr>
<td>RETAIN</td>
<td>Retain variables or elements of a variable</td>
<td>N/A</td>
<td>RETAIN Y X MAT SUBSET LAB 4</td>
</tr>
<tr>
<td>MAKE</td>
<td>Assign (equate) additional names to variables</td>
<td>N/A</td>
<td>MAKE Y X 1 .J 2 LAB 3</td>
</tr>
<tr>
<td>&quot;NAME&quot;</td>
<td>Insert a comment line in code</td>
<td>N/A</td>
<td>COMMENT DILLUION ANALYSIS . CARRY OUT ANALYSIS ON LAB 4</td>
</tr>
<tr>
<td>IMPLEMENT</td>
<td>Activate local change to DFPLT. implementation</td>
<td>As initialized</td>
<td>IMPLEMENT 3</td>
</tr>
<tr>
<td>PFR-ERASE</td>
<td>Specify auto pre-erase for plots (ON/OFF)</td>
<td>ON = pre-erase</td>
<td>PFR-ERASE ON</td>
</tr>
<tr>
<td>CREATE</td>
<td>Create a subroutine</td>
<td>N/A</td>
<td>CREATE PROC.</td>
</tr>
<tr>
<td>END OF CREATE</td>
<td>End creation of a subroutine</td>
<td>N/A</td>
<td>END OF CREATE</td>
</tr>
<tr>
<td>CALL</td>
<td>Execute a DATAPLOT subroutine from mass stg</td>
<td>N/A</td>
<td>CALL PROC.</td>
</tr>
<tr>
<td>LOOP</td>
<td>Initialize a loop</td>
<td>N/A</td>
<td>LOOP FOR A = 2 .15</td>
</tr>
<tr>
<td>END OF LOOP</td>
<td>Terminate a loop</td>
<td>N/A</td>
<td>END OF LOOP</td>
</tr>
<tr>
<td>IT</td>
<td>Define start of conditionally-executed code</td>
<td>N/A</td>
<td>IT FOR A &gt; 5</td>
</tr>
<tr>
<td>END OF IT</td>
<td>Define end of conditionally-executed code</td>
<td>N/A</td>
<td>END OF IT</td>
</tr>
<tr>
<td>TRIGONOMETRIC UNITS</td>
<td>Specify trigonometric units</td>
<td>radians</td>
<td>TRIGONOMETRIC UNITS DEGREES</td>
</tr>
<tr>
<td>RADIANS</td>
<td>Specify radians for trig calculations (ON/OFF)</td>
<td>ON = radians</td>
<td>RADIANS ON</td>
</tr>
<tr>
<td>DEGREES</td>
<td>Specify degrees for trig calculations (ON/OFF)</td>
<td>OFF = radians</td>
<td>DEGREES ON</td>
</tr>
<tr>
<td>GRADS</td>
<td>Specify grad for trig calculations (ON/OFF)</td>
<td>OFF = radians</td>
<td>GRADS ON</td>
</tr>
<tr>
<td>CLASS ...LOWER</td>
<td>Spec. first class lower lim for HISTOGRAM, etc</td>
<td>xbar - 6*s</td>
<td>CLASS LOWER -150</td>
</tr>
<tr>
<td>CLASS ...UPPER</td>
<td>Spec. last class upper lim for HISTOGRAM, etc</td>
<td>xbar + 6*s</td>
<td>CLASS UPPER 150</td>
</tr>
<tr>
<td>CLASS ...WIDTH</td>
<td>Spec. class width for HISTOGRAM, etc</td>
<td>0.3*s</td>
<td>CLASS WIDTH 10</td>
</tr>
<tr>
<td>DEMODULATION FREQUENCY</td>
<td>Specify frequency for complex demodulation</td>
<td>no default</td>
<td>DEMODULATION FREQUENCY .3</td>
</tr>
<tr>
<td>FILTER WIDTH</td>
<td>Specify filter width for SMOOTH</td>
<td>3</td>
<td>FILTER WIDTH 7</td>
</tr>
<tr>
<td>FIT CONSTRAINTS</td>
<td>Specify constraints for FIT &amp; PFR-FIT</td>
<td>OFF = no constraint</td>
<td>FIT CONSTRAINTS A . G B = 5</td>
</tr>
<tr>
<td>FIT ITERATIONS</td>
<td>Specify upper bound on iterations for FIT</td>
<td>50</td>
<td>FIT ITERATIONS 30</td>
</tr>
<tr>
<td>FIT POWER</td>
<td>Specify fit criterion power for PFR-FIT &amp; FIT</td>
<td>2 = least squares</td>
<td>FIT POWER 1</td>
</tr>
<tr>
<td>FIT STANDARD DEVIATION</td>
<td>Specify lower bound on rms sd for FIT</td>
<td>.0000005</td>
<td>FIT STANDARD DEVIATION .0001</td>
</tr>
<tr>
<td>KNOTS</td>
<td>Specify knots variable for SPLINE</td>
<td>OFF = no knots</td>
<td>KNOTS 10 30 40 60 80</td>
</tr>
<tr>
<td>WEIGHTS</td>
<td>Specify weights variable for FIT, PFR-FIT, etc</td>
<td>OFF = equi-weighted weights</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>COMMUNICATIONS LINK</td>
<td>Specify link (phone, network, etc.) to host the &quot;local&quot; link</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAUD RATE</td>
<td>Specify baud rate</td>
<td>9600</td>
<td></td>
</tr>
<tr>
<td>HOST</td>
<td>Specify host computer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURSOR COORDINATES</td>
<td>Specify cursor coordinates after a command</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURSOR SIZE</td>
<td>Specify cursor size after a plot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERASE DELAY</td>
<td>Specify delay factor for erase</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>HARDCOPY DELAY</td>
<td>Specify delay factor for hardcopy</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>HELP</td>
<td>Print short documentation for a given command</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMS</td>
<td>Print general news from DATAPILOT service org</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAIL</td>
<td>Print message from DATAPILOT service org to user</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HUGS</td>
<td>List known bugs</td>
<td>no default</td>
<td></td>
</tr>
<tr>
<td>OPERATOR</td>
<td>Send a message to the host console operator</td>
<td>no default</td>
<td></td>
</tr>
<tr>
<td>MESSAGE</td>
<td>Write message to DATAPILOT service organization</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>Display (wall clock) time and elapsed run time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROBE</td>
<td>Dump contents of underlying FORTRAN parameter</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td>Set contents of underlying FORTRAN parameter</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>TERMINATOR CHARACTER</td>
<td>Specify character to terminate commands</td>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>
Output Device Commands

The commands in this category deal with the specification of output devices. As with the previous category, the commands in this category are secondary in nature. The most important commands in this category are HARDCOPY, DISCRETE, BATCH, and TERMINAL.

### Output Device Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
</table>
| TERMINAL                    | Specify terminal model or power (ON/OFF)         | TEXTRONIX 4014                  | TERMINAL TEXTRONIX 4014 ON
| CONTINUOUS                  | Specify continuity (ON/OFF) for terminal         | ON 4096 BY 3024                 | CONTINUOUS ON PICTURE POINTS 72 24 COLOR ON |
| PICTURE POINTS              | Specify number of picture points for terminal    |                                 | DISCRETE NARROW-WIDTH ON DISCRETE NARROW-WIDTH ON |
| COLOR                       | Specify color (ON/OFF) for terminal              | OFF                             | DISCRETE WIDE-CARRIAGE ON DISCRETE WIDE-CARRIAGE ON |
| DISCRETE                    | Specify primary output--discr mw term (ON/OFF)   | OFF = contin term               | DISCRETE NARROW-WIDTH ON DISCRETE NARROW-WIDTH ON |
| DISCRETE NARROW-WIDTH       | Specify primary output--discr mw term (ON/OFF)   | OFF = contin term               | DISCRETE WIDE-CARRIAGE ON DISCRETE WIDE-CARRIAGE ON |
| BATCH                       | Specify primary output--batch (ON/OFF)           | OFF = contin term               | BATCH ON DISCRETE NARROW-WIDTH ON DISCRETE WIDE-CARRIAGE ON |
| DISCRETE WIDE-CARRIAGE      | Specify primary output--discr wc term (ON/OFF)   | OFF = contin term               | BATCH ON DISCRETE NARROW-WIDTH ON DISCRETE WIDE-CARRIAGE ON |
| HARDCOPY                    | Specify secondary output--local hdp (ON/OFF)     | OFF = no hdp outp               | HARDCOPY ON                                   |
| PENPLOTTER                  | Specify secondary output--penplotter (ON/OFF)    | OFF = no penpl outp             | PENPLOTTER ON                                 |
| CALCOP                    | Specify secondary output--Calcop (ON/OFF)        | OFF = no Calcop outp            | CALCOP ON                                    |
| VERSATEC                    | Specify secondary output--Versatec (ON/OFF)      | OFF = no Vers outp              | VERSATEC ON                                   |
| ZETA                        | Specify secondary output--Zeta (ON/OFF)          | OFF = no Zeta outp              | ZETA ON                                       |
| DEVICE ... POWER            | Specify power (ON/OFF) for a device              | ON for dev 1 (only)             | DEVICE 2 POWER ON ON for dev 1 (only) DEVICE 2 MANUFACTURER PR-80 |
| DEVICE ... MANUFACTURER     | Specify manufacturer for a device                | ON for dev 1 (only)             | DEVICE 2 MANUFACTURER PR-80 |
| DEVICE ... CONTINUOUS       | Specify continuous (ON/OFF) for a device         | ON for dev 1 (only)             | DEVICE 3 CONTINUOUS ON DEVICE 3 CONTINUOUS ON |
| DEVICE ... PICTURE POINTS   | Specify number of picture points for a device    | ON for dev 1 (only)             | DEVICE 3 PICTURE POINTS 4096 3024 |
| DEVICE ... COLOR            | Specify color (ON/OFF) for a device              | ON for dev 1 (only)             | DEVICE 3 COLOR ON ON for dev 1 (only) DEVICE 1 COLOR ON |
### Keywords

Keywords are reserved words which are not commands in themselves, but rather are special words which may appear at various points within a command line. Some keywords are built-in parameter names (such as PI, INFINITY, RESSD, etc.); some keywords are built-in variable names (such as PRED and RES); some keywords are components in other commands (such as VERSUS and AND for PLOT); some keywords are optional and powerful extensions to a variety of commands (such as SUBSET, EXCEPT, and FOR). The most important keywords are AND, VERSUS, SUBSET, EXCEPT, FOR, PRED, RES, and =.

#### Keywords

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>Used with PLOT, etc for multi-trace plots</td>
<td>N/A</td>
<td>PLOT Y X AND</td>
</tr>
<tr>
<td>VERSUS</td>
<td>Used with PLOT, etc for multi-trace plots</td>
<td>N/A</td>
<td>PLOT Y Y2 Y3 VERSUS X</td>
</tr>
<tr>
<td>SUBSET</td>
<td>Qualifier denoting subset of interest</td>
<td>N/A</td>
<td>PLOT Y SUBSET LAB 4</td>
</tr>
<tr>
<td>EXCEPT</td>
<td>Qualifier denoting excepted subset</td>
<td>N/A</td>
<td>FIT Y=A*X**B EXCEPT LAB 9</td>
</tr>
<tr>
<td>FOR</td>
<td>Qualifier denoting els or var of interest</td>
<td>N/A</td>
<td>PLOT SIN(X) FOR X = 0 .1 6.3</td>
</tr>
<tr>
<td>I</td>
<td>Var denoting dummy index; used in FOR</td>
<td>N/A</td>
<td>LET Y(I) = X(I+20) FOR I = 1 20</td>
</tr>
<tr>
<td>TO</td>
<td>Specify interval of values within a variable</td>
<td>N/A</td>
<td>PLOT Y X SUBSET LAB 4 TO 9</td>
</tr>
<tr>
<td>PI</td>
<td>Parameter with value 3.1415926</td>
<td>3.1415926</td>
<td>LET Y = SIN(2<em>PI</em>F)</td>
</tr>
<tr>
<td>INFINITY</td>
<td>Parameter with value &quot;infinity&quot;</td>
<td>largest real value</td>
<td>DELETE Y X FOR Y = 50 TO IN</td>
</tr>
<tr>
<td>PRED</td>
<td>Var with predicted values from FIT, etc</td>
<td>N/A</td>
<td>PLOT Y PRED VERSUS X</td>
</tr>
<tr>
<td>RES</td>
<td>Var with residuals from FIT, ANOVA, etc</td>
<td>N/A</td>
<td>PLOT RES X</td>
</tr>
<tr>
<td>RESSD</td>
<td>Param with res ad from FIT, ANOVA, etc</td>
<td>N/A</td>
<td>PRINT RESSD</td>
</tr>
<tr>
<td>RESDF</td>
<td>Param with res deg of freedom from FIT etc</td>
<td>N/A</td>
<td>PRINT RESDF</td>
</tr>
<tr>
<td>REPDF</td>
<td>Param with replication ad from FIT, ANOVA, etc</td>
<td>N/A</td>
<td>PRINT REPDF</td>
</tr>
<tr>
<td>LOPCDF</td>
<td>Param with rep deg of freedom from FIT etc</td>
<td>N/A</td>
<td>PRINT LOPCDF</td>
</tr>
<tr>
<td>WRT</td>
<td>&quot;With respect to&quot;; used with LET for roots, etc</td>
<td>N/A</td>
<td>LET FUNCTION O=DERIVATIVE F WRT X</td>
</tr>
<tr>
<td>CORP</td>
<td>Var where FIT, ANOVA, etc coef stored</td>
<td>N/A</td>
<td>PRINT CORP</td>
</tr>
<tr>
<td>CORPSD</td>
<td>Var where FIT, ANOVA, etc coef ad stored</td>
<td>N/A</td>
<td>PRINT CORPSD</td>
</tr>
<tr>
<td>DMBODF</td>
<td>Params with updated complex demodulation freq</td>
<td>N/A</td>
<td>PRINT DMBODF</td>
</tr>
<tr>
<td>ON</td>
<td>Set a switch to &quot;on&quot; position</td>
<td>N/A</td>
<td>CALCOMP ON</td>
</tr>
<tr>
<td>OFF</td>
<td>Set a switch to &quot;off&quot; position</td>
<td>N/A</td>
<td>CALCOMP OFF</td>
</tr>
<tr>
<td>AUTOMATIC</td>
<td>Set a switch to &quot;automatic&quot; position</td>
<td>N/A</td>
<td>LIMITS AUTOMATIC</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>Set a switch to &quot;default&quot; position</td>
<td>N/A</td>
<td>ETY COORDINATES DEFAULT</td>
</tr>
<tr>
<td>VERTICALLY</td>
<td>Rotate contents (but not frame) of plot</td>
<td>N/A</td>
<td>PLOT VERTICALLY X**2 FOR X=1 10</td>
</tr>
<tr>
<td>=</td>
<td>&quot;Equal&quot;; used in FIT, FRB-FIT, FOR, etc</td>
<td>N/A</td>
<td>FIT Y = A+B*LOG(-C-Y)</td>
</tr>
<tr>
<td>&lt;</td>
<td>&quot;Not equal to&quot;</td>
<td>N/A</td>
<td>LET Y = A/IF X &lt; 0</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&quot;Less than&quot;</td>
<td>N/A</td>
<td>FIT Y = 0<em>EXP(-A</em>X) SUBSET X &lt; 3</td>
</tr>
<tr>
<td>&gt;</td>
<td>&quot;Greater than&quot;</td>
<td>N/A</td>
<td>FIT Y = 0<em>EXP(-A</em>X) SUBSET X &gt; 3</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&quot;Greater than or equal to&quot;</td>
<td>N/A</td>
<td>DELETE Y X LAB SUBSET Y &gt; 100</td>
</tr>
<tr>
<td>;</td>
<td>Terminator character for a command</td>
<td>N/A</td>
<td>PLOT Y PRED VS X; PLOT RES X</td>
</tr>
<tr>
<td>...</td>
<td>Continue any statement onto next line</td>
<td>N/A</td>
<td>FIT Y = A0 + A1/X + A2/...</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
<td>N/A</td>
<td>X<strong>2 + A3/X</strong>3</td>
</tr>
</tbody>
</table>
Parameters
Variables
Functions

Arithmetic Operations 3-1
Relational Operations 3-1
Numbers 3-2
Parameters 3-3
LET 3-3
Variables 3-4
LET, READ, and SERIAL READ 3-4
Functions 3-5
LET FUNCTION 3-5
Evaluating Functions 3-5
LET 3-5
Subcommands Under the LET Command 3-6
Statistics, Mathematics, Random Numbers, Manipulation LET 3-6
Built-in Library Functions 3-12
LET, LET FUNCTION, PLOT, JD-PLT, FIT, and PRE-FIT 3-12
Creating Parameters, Variables, and Functions 3-15
LET, LET FUNCTION, READ, and SERIAL READ 3-15
Copying Parameters, Variables, and Functions 3-15
LET 3-15
Deleting Parameters, Variables, and Functions 3-16
DELETE 3-16
Assigning Multiple Names to a Variable 3-18
NAME 3-18
Creating Data Internally 3-19
LET, READ, and SERIALY READ 3-19
**Arithmetic Operations**

As with FORTRAN, the DATAPLOT language uses the following symbols for arithmetic operations—

+ addition  
- subtraction  
* multiplication  
/ division  
** exponentiation

Also, operations are performed left to right with priorities defined in a fashion identical to FORTRAN—

1) exponentiation  
2) multiplication and division  
3) addition and subtraction

Also as in FORTRAN, the order of operations may be altered by use of parentheses—with operations in parentheses being performed first.

The 3 most important DATAPLOT commands are

PLOT  
FIT  
LET

These 3 commands also have the most common occurrence of arithmetic operations, as in

PLOT \(10 + X^{**}2\) FOR \(X = 1\) TO \(10\)

which would generate a plot of \(10 + X^{**}2\) for the values \(X = 1\), at increments of 1, up to 10 (that is, \(X = 1, 2, ..., 10\)).

FIT \(Y = A + B \times \log(C + X^{**}2)\)

would carry out a non-linear fit of \(Y\) on \(X\) with the model \(A + B \times \log(C + X^{**}2)\)

LET \(A = X^{**}Y + Z^{**}2\)

would compute the parameter or variable \(A\) from the parameters or variables \(X, Y,\) and \(Z\). If \(X, Y,\) and \(Z\) were all parameters, then \(A\) would become a parameter. If any of the \(X, Y,\) or \(Z\) were variables, then \(A\) would become a variable.

It is also possible to use arithmetic operators in the definition of functions, as in,

LET FUNCTION \(F = X^{**}2\)  
LET FUNCTION \(G = 10 \times Y\)  
LET FUNCTION \(H = A + F/G\)

which would result in the creation of the functions \(F, G,\) and \(H\) where \(F = X^{**}2, \ G = 10 \times Y,\) and \(H = A + (X^{**}2)/(10 \times Y)\). Note that DATAPLOT would automatically provide the needed parentheses in the definition of \(H\).

**Relational Operations**

DATAPLOT has 6 relational operators—

\(=\) equality  
\(<\) inequality  
\(<=\) less than  
\(\geq\) or \(<=\) less than or equal to  
\(>\) greater than  
\(\geq\) or \(>=\) greater than or equal to

The spacing within such operators is important—using \(\geq\) instead of \(\geq\) will lead to a syntax error or possible erroneous results. Thus to generate a plot of \(Y\) versus \(X\) but with the plot restricted to those \(X\) and \(Y\) values for which the LAB variable is 7 or greater, the proper entry is

PLOT Y X SUBSET LAB \(\geq\) 7

while

PLOT Y X SUBSET LAB \(>\) 7

is incorrect.

Spacing around such relational operators is optional and at the analyst's discretion. The following are equivalent—

PLOT Y X SUBSET LAB \(\geq\) 7  
PLOT Y X SUBSET LAB \(>\) 7

FIT \(Y = A + B/X\)  
FIT \(Y = A + B/X\)

LET \(Y = (X^{**}\Lambda)/(\Lambda + 1)\)  
LET \(Y = (X^{**}\Lambda)/(\Lambda + 1)\)

Good programming practice and readability, however, suggest that relational operators be surrounded by a blank, as in

PLOT Y X SUBSET X \(<\) 100

rather than

PLOT Y X SUBSET X \(<\) 100
Numbers

Numbers are unnamed scalars. They may appear in a variety of different kinds of commands; note the 20 and 50 in

BOX 20 20 50 50

note the 0, .1 and 10 in

PLOT SIN(X) FOR X = 0 .1 10

note the 2.5 in

LET Y = X+2.5

note the 2 in

FIT Y = A+B*X+C*X**2

With the exception of the FIT and PRE-FIT commands, the general rule in DATAPLOT is that anywhere a number appears in a command line, it could equally well have been replaced by a parameter, as in the following analogues to the above--

LET X1 = 20
LET Y1 = 20
LET X2 = 50
LET Y2 = 50
BOX X1 Y1 X2 Y2

LET START = 0
LET INC = .1
LET STOP = 10
PLOT SIN(X) FOR X = START INC STOP

LET A = 2.5
LET Y = X+A

The FIT and PRE-FIT commands are an exception because the command lines

LET D = 2
FIT Y = A+B*X+C*X**D

will not be treated in the same manner as

FIT Y = A+B*X+C*X**2

In the first case, DATAPLOT will realize that D is a parameter and so (like all parameters appearing in a fit) will determine the least squares estimate for the parameter D along with the other parameters A, B, and C. However, in the second case (FIT Y = A+B*X+C*X**2), DATAPLOT will note the scalar number 2 and fit for the parameters A, B, and C only. Be aware of this distinction in carrying out fits and pre-fits.

All numbers are stored internally in DATAPLOT as single precision floating point. If the analyst wishes to specify a decimal number, as in

LET Y = X**2.378

then the decimal point and trailing decimal digits should of course be included. However, if the number happens to be an integer, then the analyst has the choice of including or excluding the trailing decimal point, and including or excluding any trailing zeros--thus the following are all equivalent--

LET Y = X**2
LET Y = X**2.0
LET Y = X**2.00

All such expressions will be stored and processed internally by DATAPLOT in an identical fashion--the analyst gains nothing by including the trailing decimal point and zeros. Simplicity dictates that the first form be used, but if the analyst prefers to use other forms, the results will be identical.

To define numbers with large exponents (for example, 7.4 raised to the 15th power), the analyst should use the ** (exponentiation) operator directly, as in

LET A = 7.4**15

The E format (as occurs in FORTRAN) is not permitted; thus

LET A = 7.4E15

is not a valid DATAPLOT command.

Negative exponents are handled in a similar fashion, as in

LET B = 7.4**(-15)
Parameters

LET

A parameter is a named scalar and may be defined
via the LET command, as in

LET A = 17.26
LET B = 3.97
LET C = -5.38
LET D = 2.4*10**(-8)
LET E = B**2-4*A*C
LET F = MEAN X

Two internally-provided parameters which the
analyist may use are

PI
INFINITY

PI has the value 3.14159265; INFINITY has the
value of the largest floating point number which
the user 's computer may store (that is,
INFINITY = machine infinity). PI and INFINITY may
be used at any time and like any other user-defined parameter, for example,

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

FIT Y = A+B*X**C EXCEPT X 100 TO INFINITY

Be wary of the use of PI in FIT and PRE-FIT
expressions; like all parameters, DATAPLOT will
attempt to determine least squares estimates for
it. Thus rather than use

FIT Y = AMP * SIN(2*PI*X)

one should explicitly use

FIT Y = AMP * SIN(2*3.14159265*PI*X)

With the exception of the above-mentioned FIT and
PRE-FIT exclusion, parameters may be substituted
anywhere in which numbers appear. For example,
suppose the analyst wished to override the usual
plot frame coordinates and specify that all
succeeding plots have lower left corner at (20,50)
and for the plot to be 30 units wide and 30 units
high (that is, have the upper right corner at
(50,80). This may be done explicitly by

FRAME CORNER COORDINATES 20 50 50 80

or alternatively

LET X1 = 20
LET Y1 = 50
LET X2 = 50
LET Y2 = 80
FRAME CORNER COORDINATES X1 Y1 X2 Y2

or

LET X1 = 20
LET Y1 = 50
LET X2 = X1+30
LET Y2 = Y1+30
FRAME CORNER COORDINATES X1 Y1 X2 Y2

This capability of substituting parameters for
numbers is especially convenient for diagram-
construction on terminals which have built-in
hardware for inputting screen coordinates via
cross-hair, light-pen, or equivalent. Suppose
it is desired to draw a line between 2 points on
the screen which the analyst will interactively
specify via the cross-hair. One way is to

1) raise the cross-hair (via CROSS-HAIR);
2) position it to the first point (via the
   thumbwheels);
3) input and print the coordinates (via
   hitting any key (on Tektronix terminals),
   or by hitting some pre-designated key (on
   other terminals);
4) raise the cross-hair (via CROSS-HAIR);
5) position it to the second point (via the
   thumbwheels);
6) print the coordinates (via hitting any
   key);
7) draw the line (via, for example,
   DRAW 20.4 35.3 78.4 80.5).

An easier way which avoids the handling of
absolute numbers and replaces it with the handling
of symbolic parameter names is

1) raise the cross-hair (via CROSS-HAIR X1 Y1);
2) position it to the first point (via the
   thumbwheels);
3) copy the coordinates into X1 and Y1 (via
   hitting any key);
4) raise the cross-hair (via CROSS-HAIR X2 Y2);
5) position it to the second point (via
   thumbwheels);
6) copy the coordinates into X2 and Y2 (via
   hitting any key);
7) draw the line (via DRAW X1 Y1 X2 Y2).
Variables

LET, READ, and SERIAL READ

A variable is a named vector (a named single-dimension array), and may be defined via the LET, READ, and SERIAL READ commands, as in:

```
LET Y = SEQUENCE 1 1 10
LET Z = PATTERN 1 2 3 FOR I = 1 1 9
LET U = NORMAL RANDOM NUMBERS FOR I = 1 1 100
LET X2 = X**2-LOG(Y)
LET Y2 = SQRT(X+Y**3)
```

or

```
READ X Y
1 1
2 4
3 9
4 16
5 25
END OF DATA
```

or

```
SERIAL READ X Y
1 1 2 4 3 9 4 16 5 25
END OF DATA
```

Variables are the most commonly-handled component in DATAPLOT, as in:

```
PLOT Y X
FIT Y = A+B*LOG(X+C)
LET X2 = LOG(X)
```

Names for variables, parameters, and functions may be of any length, but since only the first 8 characters are scanned and internally stored, no 2 names should be identical for the first 8 characters. Names must start with an alphabetic character, but may be any combination of alphabetic and numeric characters thereafter. It is the author's personal practice (which shows up throughout this manual) to follow the usual mathematical custom of using characters toward the end of the alphabet (X's, Y's, Z's, etc.) to represent variables, of using characters toward the beginning of the alphabet (A's, B's, C's, etc.) to represent parameters, and of using characters in the vicinity of F (e.g., F's, G's, H's, etc.) to represent functions. Note, however, that this is personal preference and not a DATAPLOT requirement.
Functions
LET FUNCTION

A function is a named character string, and may be defined via the LET FUNCTION command, as in

LET FUNCTION F = EXP(-0.5*X**2)
LET FUNCTION G = SIN(2*PI*X**2)
LET FUNCTION H = F+LOG(G)
LET FUNCTION F2 = DERIVATIVE F WRT X

Functions may be concatenated and built-up piece-by-piece, as in

LET FUNCTION NUM = EXP(-ALPHA*X)
LET FUNCTION DENOM = A+B*X
LET FUNCTION RATIO = NUM/DENOM

which is equivalent to

LET FUNCTION RATIO = EXP(-ALPHA*X)/(A+B*X)

Functions may be defined before (or after) the parameters and variables contained in them are created, as in the following example involving a variable transformation--

LET FUNCTION F = X**2

SERIAL READ X
1 2 3
END OF DATA

LET Y = F

This last statement (LET Y = F) is equivalent to

LET Y = X**2

and will (upon execution) result in the Y variable having the values 1, 4, and 9.

A more common example of functions being defined prior to use is in fitting--

LET FUNCTION F1 = A1+B1*SORT(X)
LET FUNCTION F2 = A2+B2*LOG(X)

READ X Y
1 1
2 1.5
3 2
4 2.3
5 2.5
END OF DATA

FIT Y = F1
FIT Y = F2

Evaluating Functions
LET

The LET FUNCTION command and the LET command carry out 2 distinctly different operations--the LET FUNCTION command allows the analyst to create functions; the LET command allows the analyst to carry out function evaluations.

For example, suppose it is desired to evaluate the function sqrt(1-0.3*X**2) over the region x = 0 (.01) 1. This may be done in a number of ways; the most direct way is

LET X = SEQUENCE 0 .01 1
LET Y = SQRT(1-0.3*X**2)
WRITE X Y
PLOT X Y

The first LET command makes use of the SEQUENCE sub-capability of the LET command to create the variable X with a sequence of 101 values in it-- .00, .01, .02, ..., .99, 1.00. The second LET command creates a variable Y (also with 101 elements) which has the desired function evaluation values in it. The WRITE command generates a list of X and Y values. The PLOT command will generate a plot of Y (vertically) versus X (horizontally).

A second way to evaluate the function would be

LET X = SEQUENCE 0 .01 1
LET Y = SQRT(1-0.3*X**2)
LET F = F
WRITE X Y
PLOT X Y

As before, the first LET statement would create the variable X with the specified sequence of 101 values. The LET FUNCTION command would then create the function F consisting of the following 16 character string--SQRT(1-0.3*X**2). Note that the LET FUNCTION command does not carry out a function evaluation--it merely creates a function. The LET Y = F command would then recognize F as a pre-defined function, replace the name F with the specified 16-character string, and carry out the function evaluation. As before, the WRITE command prints out the results of the function evaluation and the PLOT command plots out the results of the function evaluation.

Note that if our ultimate objective is to simply plot the function (rather than creating variables containing evaluated values of the function), then the above code could be shortened directly to

PLOT SQRT(1-0.3*X**2) FOR X = 0 .01 1

or

LET FUNCTION F = SQRT(1-0.3*X**2)
PLOT F FOR X = 0 .01 1
Sub-commands Under the LET Command

Statistics, Mathematics, Random Numbers, Manipulation

LET

The LET command is the single most powerful command in DATAPLOT. The most important capability of the LET command is carrying out function evaluations and variable transformations. Such evaluations/transformations are general--any Fortran-like expression may be used.

In addition, the LET command may also be used by the analyst to carry out a broad spectrum of statistical, mathematical, and manipulative operations. These operations are specified by inclusion of sub-commands under the LET command. These sub-commands fall into 4 general categories—

1. Computing Statistics
2. Performing Mathematical Operations
3. Generating Random Numbers
4. Miscellaneous
## SUB-COMMANDS UNDER THE LET COMMAND

### Computing Statistics on a Variable

*Input is a variable; output is a parameter*

<table>
<thead>
<tr>
<th>Description</th>
<th>Sub-Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute sample size (number of observations)</td>
<td>SIZE (or NUMBER)</td>
<td>LET A = SIZE Y</td>
</tr>
<tr>
<td>Compute sample mean</td>
<td>MEAN</td>
<td>LET A = MEAN Y</td>
</tr>
<tr>
<td>Compute sample median</td>
<td>MEDIAN</td>
<td>LET A = MEDIAN</td>
</tr>
<tr>
<td>Compute sample midrange</td>
<td>MIDRANGE</td>
<td>LET A = MIDRANGE Y</td>
</tr>
<tr>
<td>Compute sample midmean</td>
<td>MIDMEAN</td>
<td>LET A = MIDMEAN Y</td>
</tr>
<tr>
<td>Compute sample range</td>
<td>RANGE</td>
<td>LET A = RANGE Y</td>
</tr>
<tr>
<td>Compute sample standard deviation</td>
<td>STANDARD DEVIATION</td>
<td>LET A = STANDARD DEVIATION Y</td>
</tr>
<tr>
<td>Compute sample variance</td>
<td>VARIANCE</td>
<td>LET A = VARIANCE Y</td>
</tr>
<tr>
<td>Compute sample relative standard deviation</td>
<td>RELATIVE STANDARD DEVIATION</td>
<td>LET A = RELATIVE STANDARD DEVIATION Y</td>
</tr>
<tr>
<td>Compute sample standard deviation of the mean</td>
<td>STANDARD DEVIATION OF THE MEAN</td>
<td>LET A = STANDARD DEVIATION OF THE MEAN Y</td>
</tr>
<tr>
<td>Compute sample third central moment</td>
<td>THIRD CENTRAL MOMENT</td>
<td>LET A = THIRD CENTRAL MOMENT</td>
</tr>
<tr>
<td>Compute sample skewness (stand. 3rd cent. mom.)</td>
<td>SKEWNESS</td>
<td>LET A = SKEWNESS Y</td>
</tr>
<tr>
<td>Compute sample fourth central moment</td>
<td>FOURTH CENTRAL MOMENT</td>
<td>LET A = FOURTH CENTRAL MOMENT Y</td>
</tr>
<tr>
<td>Compute sample kurtosis (stand. 4th cent. mom.)</td>
<td>KURTOSIS</td>
<td>LET A = KURTOSIS Y</td>
</tr>
<tr>
<td>Compute sample minimum</td>
<td>MINIMUM</td>
<td>LET A = MINIMUM Y</td>
</tr>
<tr>
<td>Compute sample maximum</td>
<td>MAXIMUM</td>
<td>LET A = MAXIMUM Y</td>
</tr>
<tr>
<td>Compute sample lower quartile</td>
<td>LOWER QUARTILE</td>
<td>LET A = LOWER QUARTILE Y</td>
</tr>
<tr>
<td>Compute sample upper quartile</td>
<td>UPPER QUARTILE</td>
<td>LET A = UPPER QUARTILE Y</td>
</tr>
<tr>
<td>Compute sample lower hinge</td>
<td>LOWER HINGE</td>
<td>LET A = LOWER HINGE Y</td>
</tr>
<tr>
<td>Compute sample upper hinge</td>
<td>UPPER HINGE</td>
<td>LET A = UPPER HINGE Y</td>
</tr>
<tr>
<td>Compute sample autocovariance</td>
<td>AUTOCOVARIANCE</td>
<td>LET A = AUTOCOVARIANCE Y</td>
</tr>
<tr>
<td>Compute sample autocorrelation</td>
<td>AUTOCORRELATION</td>
<td>LET A = AUTOCORRELATION Y</td>
</tr>
<tr>
<td>Compute sample covariance</td>
<td>COVARIANCE</td>
<td>LET A = COVARIANCE Y X</td>
</tr>
<tr>
<td>Compute sample correlation</td>
<td>CORRELATION</td>
<td>LET A = CORRELATION Y X</td>
</tr>
<tr>
<td>Compute sample rank covariance</td>
<td>RANK COVARIANCE</td>
<td>LET A = RANK COVARIANCE Y X</td>
</tr>
<tr>
<td>Compute sample rank correlation</td>
<td>RANK CORRELATION</td>
<td>LET A = RANK CORRELATION Y X</td>
</tr>
</tbody>
</table>
Performing Mathematical Operations on a Variable
(Part 1—Input is a variable, output is a parameter)

<table>
<thead>
<tr>
<th>Description</th>
<th>Sub-Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute sum of elements in a var</td>
<td>SUM</td>
<td>LET A = SUM Y</td>
</tr>
<tr>
<td>Compute product of elements in a var</td>
<td>PRODUCT</td>
<td>LET A = PRODUCT Y</td>
</tr>
<tr>
<td>Compute integral of elements in a var</td>
<td>INTEGRAL</td>
<td>LET A = INTEGRAL Y X</td>
</tr>
</tbody>
</table>

Performing Mathematical Operations on a Variable
(Part 2—Input is a variable, output is a variable)

<table>
<thead>
<tr>
<th>Description</th>
<th>Sub-Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute cumulative sum of els in a var</td>
<td>CUMULATIVE SUM</td>
<td>LET Y2 = CUMULATIVE SUM Y</td>
</tr>
<tr>
<td>Compute cumulative products of els in a var</td>
<td>CUMULATIVE PRODUCT</td>
<td>LET Y2 = CUMULATIVE PRODUCT Y</td>
</tr>
<tr>
<td>Compute cumulative integrals of els in a var</td>
<td>CUMULATIVE INTEGRAL</td>
<td>LET Y2 = CUMULATIVE INTEGRAL Y X</td>
</tr>
<tr>
<td>Compute sequential differences of els in a var</td>
<td>SEQUENTIAL DIFFERENCE</td>
<td>LET Y2 = SEQUENTIAL DIFFERENCE Y</td>
</tr>
<tr>
<td>Sort the elements in a variable</td>
<td>SORT</td>
<td>LET Y2 = SORT Y</td>
</tr>
<tr>
<td>Rank the elements in a variable</td>
<td>RANK</td>
<td>LET Y2 = RANK Y</td>
</tr>
<tr>
<td>Code the elements in a variable</td>
<td>CODE</td>
<td>LET Y2 = CODE Y</td>
</tr>
<tr>
<td>Compute convolution of els in 2 var</td>
<td>CONVOLUTION</td>
<td>LET Y2 = CONVOLUTION Y X</td>
</tr>
</tbody>
</table>

Performing Mathematical Operations on a Function

<table>
<thead>
<tr>
<th>Description</th>
<th>Sub-Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute roots of a function</td>
<td>ROOTS</td>
<td>LET Y2 = ROOTS EXP(-X)-SIN(X) WRT X FOR X=0 TO 10</td>
</tr>
<tr>
<td>Compute derivative of a function</td>
<td>DERIVATIVE</td>
<td>LET A = DERIVATIVE SORT(EXP(-X**2)) WRT X FOR X=3</td>
</tr>
<tr>
<td>Compute definite integral of a function</td>
<td>INTEGRAL</td>
<td>LET A = INTEGRAL EXP(-X**2) WRT X FOR X = -3 TO 3</td>
</tr>
</tbody>
</table>
### Generating Random Numbers

<table>
<thead>
<tr>
<th>Description</th>
<th>Sub-Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate normal (N(0,1)) random numbers</td>
<td>NORMAL RANDOM NUMBERS</td>
<td>LET Y = NORMAL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate uniform ((0,1)) random numbers</td>
<td>UNIFORM RANDOM NUMBERS</td>
<td>LET Y = UNIFORM RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate logistic random numbers</td>
<td>LOGISTIC RANDOM NUMBERS</td>
<td>LET Y = LOGISTIC RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate double exponential random numbers</td>
<td>DOUBLE EXPONENTIAL RANDOM NUMBERS</td>
<td>LET Y = DOUBLE EXPONENTIAL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate Cauchy random numbers</td>
<td>CAUCHY RANDOM NUMBERS</td>
<td>LET Y = CAUCHY RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate Tukey lambda random numbers</td>
<td>TUKEY LAMDA RANDOM NUMBERS</td>
<td>LET Y = TUKEY LAMDA RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate semi-circular random numbers</td>
<td>SEMI-CIRCULAR RANDOM NUMBERS</td>
<td>LET Y = SEMI-CIRCULAR RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate triangular random numbers</td>
<td>TRIANGULAR RANDOM NUMBERS</td>
<td>LET Y = TRIANGULAR RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate lognormal random numbers</td>
<td>LOGNORMAL RANDOM NUMBERS</td>
<td>LET Y = LOGNORMAL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate halfnormal random numbers</td>
<td>HALFNORMAL RANDOM NUMBERS</td>
<td>LET Y = HALFNORMAL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate t random numbers</td>
<td>T RANDOM NUMBERS</td>
<td>LET Y = T RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate chi-squared random numbers</td>
<td>CHI-SQUARED RANDOM NUMBERS</td>
<td>LET Y = CHI-SQUARED RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate F random numbers</td>
<td>F RANDOM NUMBERS</td>
<td>LET Y = F RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate exponential random numbers</td>
<td>EXPONENTIAL RANDOM NUMBERS</td>
<td>LET Y = EXPONENTIAL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate gamma random numbers</td>
<td>GAMMA RANDOM NUMBERS</td>
<td>LET Y = GAMMA RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate beta random numbers</td>
<td>BETA RANDOM NUMBERS</td>
<td>LET Y = BETA RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate Weibull random numbers</td>
<td>WEIBULL RANDOM NUMBERS</td>
<td>LET Y = WEIBULL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate extreme value type 1 random numbers</td>
<td>EXTREME VALUE TYPE 1 RANDOM NUMBERS</td>
<td>LET Y = EXTREME VALUE TYPE 1 RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate extreme value type 2 random numbers</td>
<td>EXTREME VALUE TYPE 2 RANDOM NUMBERS</td>
<td>LET Y = EXTREME VALUE TYPE 2 RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate Pareto random numbers</td>
<td>PARETO RANDOM NUMBERS</td>
<td>LET Y = PARETO RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate binomial random numbers</td>
<td>BINOMIAL RANDOM NUMBERS</td>
<td>LET Y = BINOMIAL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate geometric random numbers</td>
<td>GEOMETRIC RANDOM NUMBERS</td>
<td>LET Y = GEOMETRIC RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate Poisson random numbers</td>
<td>POISSON RANDOM NUMBERS</td>
<td>LET Y = POISSON RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate negative binomial random numbers</td>
<td>NEGATIVE BINOMIAL RANDOM NUMBERS</td>
<td>LET Y = NEGATIVE BINOMIAL RANDOM NUMBERS FOR I = 1 1 50</td>
</tr>
<tr>
<td>Generate a random permutation</td>
<td>RANDOM PERMUTATION</td>
<td>LET Y = RANDOM PERMUTATION FOR I = 1 1 50</td>
</tr>
</tbody>
</table>

### Miscellaneous

<table>
<thead>
<tr>
<th>Description</th>
<th>Sub-Command</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate a sequence within a var</td>
<td>SEQUENCE</td>
<td>LET Y = SEQUENCE 1 1 10</td>
</tr>
<tr>
<td>Generate distinct elements from a var</td>
<td>DISTINCT</td>
<td>LET Y = DISTINCT X</td>
</tr>
<tr>
<td>Extract distinct elements from a var</td>
<td>DISTINCT</td>
<td>LET Y = DISTINCT X</td>
</tr>
</tbody>
</table>
For category 1 (Computing Statistics), the input is always a variable and the output is always a parameter. For example, if \( Y \) is a variable, then

\[
\text{LET } A = \text{MEAN } Y
\]

computes the mean of the data in the variable \( Y \) and places the result into the parameter \( A \).

For category 2 (Mathematical Operations), there are 3 sub-categories:

2.1) \( \text{input } = \text{variable}; \text{output } = \text{parameter}; \)

2.2) \( \text{input } = \text{variable}; \text{output } = \text{variable}; \)

2.3) \( \text{input } = \text{function}; \text{output } = \text{parameter or variable}; \)

An example of sub-category 2.1 is the \text{SUM} operation; if \( Y \) is a variable, then

\[
\text{LET } A = \text{SUM } Y
\]

computes the sum of all the elements in the variable \( Y \), and places that sum into the parameter \( A \).

An example of sub-category 2.2 is the \text{CUMULATIVE SUM} operation; if \( Y \) is a variable, then

\[
\text{LET } Y_2 = \text{CUMULATIVE SUM } Y
\]

computes the cumulative sum (= partial sum) of the elements in the variable \( Y \), and places the resulting cumulative sums into corresponding elements of the variable \( Y_2 \); thus

\[
Y_2(1) = Y(1) \\
Y_2(2) = Y(1) + Y(2) \\
Y_2(3) = Y(1) + Y(2) + Y(3) \\
\text{etc.}
\]

An example of sub-category 2.3 is the \text{ROOTS} operation; if \( F \) is a function, then

\[
\text{LET } Y = \text{ROOTS } F \text{ WRT } X \text{ FOR } X = 0 \text{ TO } 10
\]

determines all of the roots of the function \( X \) in the interval 0 to 10 and places those roots (if any found) into the elements of \( Y \). If no roots are found, then \( Y_3 \) will not be formed; if 1 root is found, then \( Y_3 \) will be a parameter; if 2 or more roots are found, then \( Y_3 \) will be a variable.

For category 3 (Computing Random Numbers), the \text{FOR} qualification at the end of the command tells \text{DATAPLOT} not only how many random numbers to generate but also where (into what elements of the variable) to place the random numbers. Thus

\[
\text{LET } Y = \text{NORMAL RANDOM NUMBERS FOR } I = 1 \text{ TO } 50
\]

generates 50 normal \( \mathcal{N}(0,1) \) random numbers, and places these numbers into the first 50 elements of the variable \( Y \).

If the analyst enters

\[
\text{LET } Y = \text{NORMAL RANDOM NUMBERS FOR } I = 101 \text{ TO } 200
\]

then this instructs \text{DATAPLOT} to generate enough random numbers so as to fill

\[
\text{element } 101 \\
\text{at increments of } 1 \\
\text{up to element } 200
\]

of the variable \( Y \) (that is, elements 101, 102, 103, 104, ..., 198, 199, 200 of the variable \( Y \); a total of 100 random numbers are generated.

If the analyst enters

\[
\text{LET } Y = \text{NORMAL RANDOM NUMBERS FOR } I = 101 \text{ TO } 200
\]

then this instructs \text{DATAPLOT} to generate enough random numbers so as to fill

\[
\text{element } 101 \\
\text{at increments of } 10 \\
\text{up to element } 200
\]

of the variable \( Y \) (that is, elements 101, 111, 121, 131, ..., 181, 191, of the variable \( Y \); a total of 10 random numbers are generated.

Random numbers may be generated for 24 distributions and distributional families. Note the distinction between a distribution and a distributional family. A distribution has only 1 prototype shape for the probability density function; this probability density function may be displaced or may be squeezed/expanded due to different location/scale parameters, but nevertheless, there is only one prototype shape for this function. Examples of distributions are

- Normal (= Gaussian)
- Uniform
- Logistic
- Double Exponential (= LaPlace)
- Cauchy
- Semi-Circular
- Triangular
- Lognormal
- Halfnormal
- Exponential
- Extreme Value Type 1
On the other hand, a distributional family represents not just 1 distribution, but rather a set of distributions. Each different distribution has its own prototype probability density function which changes depending on the value of the shape/tail length parameter for the family. Examples of distributional families are:

Tukey lambda

\[ t (= \text{Student's } t) \]

Chi-squared

\[ F \]

Gamma

Beta

Weibull

Extreme Value Type 2

Pareto

Binomial

Geometric

Poisson

Negative Binomial

When generating random numbers from a distribution, such as

\[ \text{LET } Y = \text{NORMAL RANDOM NUMBERS FOR } I = 1 \text{ I } 50 \]

then the one command line is sufficient. On the other hand, when generating random numbers from a member of a distributional family, then an additional command line is needed so as to specify what member of the family the random numbers are being drawn from. For example, to generate random numbers from the t distribution, the entry

\[ \text{LET } Y = \text{T RANDOM NUMBERS FOR } I = 1 \text{ I } 50 \]

would be incomplete because we have not specified which member of the t family is desired. To rectify this, we precede the above statement with an additional statement which defines the desired \( \nu \) value, as in

\[ \text{LET } \nu = 5 \]

\[ \text{LET } Y = \text{T RANDOM NUMBERS FOR } I = 1 \text{ I } 50 \]

or

\[ \text{LET } \nu = 20 \]

\[ \text{LET } Y = \text{T RANDOM NUMBERS FOR } I = 1 \text{ I } 50 \]

The shape/tail length parameters which need to be defined for the various distributions are:

\begin{align*}
\text{Tukey lambda} & \quad \text{LAMBDA} \\
\text{t (Student's } t) & \quad \text{NU} \\
\text{Chi-squared} & \quad \text{NU} \\
F & \quad \text{NU1 AND NU2} \\
\text{Gamma} & \quad \text{GAMMA} \\
\text{Beta} & \quad \text{ALPHA AND BETA} \\
\text{Weibull} & \quad \text{GAMMA} \\
\text{Extreme Value Type 2} & \quad \text{GAMMA} \\
\text{Pareto} & \quad \text{GAMMA} \\
\text{Binomial} & \quad \text{N AND P} \\
\text{Geometric} & \quad \text{P} \\
\text{Poisson} & \quad \text{LAMBDA} \\
\text{Negative Binomial} & \quad \text{K AND P}
\end{align*}

As an aside, note that DATAPLOT allows probability plots to be generated for the same extensive set of distributions and distributional families as enumerated above for random numbers. For distributions, only a single command is needed, as in

\[ \text{NORMAL PROBABILITY PLOT } Y \]

which generates a normal probability plot of the data in the variable \( Y \); but for distributional families, an additional command is needed to specify the desired member of the family, as in

\[ \text{LET } \nu = 20 \\
\text{T PROBABILITY PLOT } Y \]

which generates a t probability plot (with \( \nu = 20 \)) of the data in \( Y \).

For category 4 (Miscellaneous), the input depends on the operation, but the output is a variable. For example,

\[ \text{LET } Y = \text{SEQUENCE } 70 \text{ .1 } 80 \]

generates a sequence of numbers starting with 70 at increments of .1 and stopping with 80.

A total of 101 numbers are generated—70, 70.1, 70.2, 70.3, ..., 79.8, 79.9, 80. These 101 values are placed in the first 101 elements of the variable \( Y \).
**Built-in Library Functions**

**LET, LET FUNCTION, PLOT, 3D- PLOT, FIT, and PRE-FIT**

General FORTRAN-like expressions are allowable in the LET, LET FUNCTION, PLOT, 3D-PLOT, FIT, and PRE-FIT commands. These expressions may include any general FORTRAN-like combination of +, -, *, /, and ** operations, as well as any mixture of the following built-in library functions. In all of the following expressions, the X, Y, etc. in the arguments may be DATAPLOT parameters, variables, or functions.

Examples of usage of such functions are

\[
\text{LET } Y = \text{EXP}(-X)/(2*6*X)
\]

which if \(X\) is a variable (parameter) will form a corresponding variable (parameter) as given by the function expression.

\[
\text{LET FUNCTION } F = \text{LOG}(2*X)/\text{EXP}((\text{SQRT}(X)))
\]

defines a function \(F\) as given by the expression.

\[
\text{PLOT } \text{SQRT}(1-0.5*X**2) \text{ FOR } X = 0 \text{ TO } 1
\]
generates a plot of the function at the points 0, at increments of .1, and stopping at 1 (that is, at the points 0, .1, .2, ..., .8, .9, 1).

\[
\text{3D-PLOT } \text{EXP}(-X**2-Y**2) \text{ FOR } X = -2 \text{ TO } 2 \text{ FOR } Y = -2 \text{ TO } 2
\]
generates a 3-dimensional plot of the bivariate surface as given by the function and evaluated at the \(X\) points -2, at increments of .2, and stopping at 2; and at the \(Y\) points -2, at increments of .2, and stopping at 2. The plot will be a cross-hatched surface. The analyst should have previously entered (via the EYE COORDINATES command) the position of the analyst's eye in viewing the surface, as in EYE COORDINATES 10 11 5.

\[
\text{FIT } Y = \text{EXP}(-\text{ALPHA}*X)/(\text{A+B*X})
\]
carries out a non-linear fit according to the function.

\[
\text{PRE-FIT } Y = \text{LOG}(\text{A+B*X}) \text{ FOR } A = 1 \text{ TO } 10 \text{ FOR } B = 1 \text{ TO } 2
\]
carries out a pre-fit according to the function, with the pre-fit carried out over the lattice of points specified by \(A = 1, 2, 3, \ldots, 9, 10\) and \(B = 1, 1.1, 1.2, \ldots, 1.8, 1.9, 2\).

The built-in library functions include

1) General Functions;
2) Trigonometric Functions;
3) Probability Functions.

The general mathematical functions include various FORTRAN-library entries, plus Chebychev, Bessel, octal-decimal conversions, etc.
<table>
<thead>
<tr>
<th>Description</th>
<th>Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute value</td>
<td>ABS(X)</td>
<td>LET Z = A + B*EXP(ABS(X) + C)</td>
</tr>
<tr>
<td>Square root</td>
<td>SQRT(X)</td>
<td>LET Z = A + B*EXP(SQR(X) + C)</td>
</tr>
<tr>
<td>Exponential</td>
<td>EXP(X)</td>
<td>LET Z = A + B*EXP(EXP(X) + C)</td>
</tr>
<tr>
<td>Logarithm (natural)</td>
<td>LN(X)</td>
<td>LET Z = A + B*EXP(LN(X) + C)</td>
</tr>
<tr>
<td>Logarithm (natural)</td>
<td>LOG(X)</td>
<td>LET Z = A + B*EXP(LOG(X) + C)</td>
</tr>
<tr>
<td>Logarithm (base 10)</td>
<td>LOG10(X)</td>
<td>LET Z = A + B*EXP(LOG10(X) + C)</td>
</tr>
<tr>
<td>Logarithm (base 2)</td>
<td>LOG2(X)</td>
<td>LET Z = A + B*EXP(LOG2(X) + C)</td>
</tr>
<tr>
<td>Sign</td>
<td>SGN(X)</td>
<td>LET Z = A + B*EXP(SIGN(X) + C)</td>
</tr>
<tr>
<td>Integer portion</td>
<td>INT(X)</td>
<td>LET Z = A + B*EXP(INT(X) + C)</td>
</tr>
<tr>
<td>Fractional portion</td>
<td>FRACT(X)</td>
<td>LET Z = A + B*EXP(FRACT(X) + C)</td>
</tr>
<tr>
<td>Modulo</td>
<td>MOD(X,Y)</td>
<td>LET Z = A + B*EXP(MOD(X,Y) + C)</td>
</tr>
<tr>
<td>Minimum</td>
<td>MIN(X,Y)</td>
<td>LET Z = A + B*EXP(MIN(X,Y) + C)</td>
</tr>
<tr>
<td>Maximum</td>
<td>MAX(X,Y)</td>
<td>LET Z = A + B*EXP(MAX(X,Y) + C)</td>
</tr>
<tr>
<td>Positive difference -- x-min(x,y)</td>
<td>DIM(X,Y)</td>
<td>LET Z = A + B*EXP(DIM(X,Y) + C)</td>
</tr>
<tr>
<td>Octal to decimal conversion</td>
<td>OCT2DEC(X)</td>
<td>LET Z = A + B*EXP(OCT2DEC(X) + C)</td>
</tr>
<tr>
<td>Decimal to octal conversion</td>
<td>DECOCT(X)</td>
<td>LET Z = A + B*EXP(DECOCT(X) + C)</td>
</tr>
<tr>
<td>Error function</td>
<td>ERF(X)</td>
<td>LET Z = A + B*EXP(ERF(X) + C)</td>
</tr>
<tr>
<td>Complementary error function</td>
<td>ERFC(X)</td>
<td>LET Z = A + B*EXP(ERFC(X) + C)</td>
</tr>
<tr>
<td>Gamma Function</td>
<td>GAMMA(X)</td>
<td>LET Z = A + B*EXP(GAMMA(X) + C)</td>
</tr>
<tr>
<td>Log (to the base e) Gamma function</td>
<td>LOGGAMMA(X)</td>
<td>LET Z = A + B*EXP(LOGGAMMA(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 0</td>
<td>CHEB0(X)</td>
<td>LET Z = A + B*EXP(CHEB0(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 1</td>
<td>CHEB1(X)</td>
<td>LET Z = A + B*EXP(CHEB1(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 2</td>
<td>CHEB2(X)</td>
<td>LET Z = A + B*EXP(CHEB2(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 3</td>
<td>CHEB3(X)</td>
<td>LET Z = A + B*EXP(CHEB3(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 4</td>
<td>CHEB4(X)</td>
<td>LET Z = A + B*EXP(CHEB4(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 5</td>
<td>CHEB5(X)</td>
<td>LET Z = A + B*EXP(CHEB5(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 6</td>
<td>CHEB6(X)</td>
<td>LET Z = A + B*EXP(CHEB6(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 7</td>
<td>CHEB7(X)</td>
<td>LET Z = A + B*EXP(CHEB7(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 8</td>
<td>CHEB8(X)</td>
<td>LET Z = A + B*EXP(CHEB8(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 9</td>
<td>CHEB9(X)</td>
<td>LET Z = A + B*EXP(CHEB9(X) + C)</td>
</tr>
<tr>
<td>Chebychev polynomial of the first kind and order 10</td>
<td>CHEB10(X)</td>
<td>LET Z = A + B*EXP(CHEB10(X) + C)</td>
</tr>
<tr>
<td>Bessel function of the first kind and order 0</td>
<td>BESS0(X)</td>
<td>LET Z = A + B*EXP(BESS0(X) + C)</td>
</tr>
<tr>
<td>Bessel function of the first kind and order 1</td>
<td>BESS1(X)</td>
<td>LET Z = A + B*EXP(BESS1(X) + C)</td>
</tr>
</tbody>
</table>
### Trigonometric Functions--

<table>
<thead>
<tr>
<th>Description</th>
<th>Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine</td>
<td>SIN(X)</td>
<td>LET Z = A + B*EXP(SIN(X) + C)</td>
</tr>
<tr>
<td>Cosine</td>
<td>COS(X)</td>
<td>LET Z = A + B*EXP(COS(X) + C)</td>
</tr>
<tr>
<td>Tangent</td>
<td>TAN(X)</td>
<td>LET Z = A + B*EXP(TAN(X) + C)</td>
</tr>
<tr>
<td>Cotangent</td>
<td>COT(X)</td>
<td>LET Z = A + B*EXP(COT(X) + C)</td>
</tr>
<tr>
<td>Secant</td>
<td>SEC(X)</td>
<td>LET Z = A + B*EXP(SEC(X) + C)</td>
</tr>
<tr>
<td>Cosecant</td>
<td>CSC(X)</td>
<td>LET Z = A + B*EXP(CSC(X) + C)</td>
</tr>
<tr>
<td>Arcsine</td>
<td>ARCSIN(X)</td>
<td>LET Z = A + B*EXP(ARCSIN(X) + C)</td>
</tr>
<tr>
<td>Arccosine</td>
<td>ARCCOS(X)</td>
<td>LET Z = A + B*EXP(ARCCOS(X) + C)</td>
</tr>
<tr>
<td>Arctangent</td>
<td>ARCTAN(X)</td>
<td>LET Z = A + B*EXP(ARCTAN(X) + C)</td>
</tr>
<tr>
<td>Arccotangent</td>
<td>ARCCOT(X)</td>
<td>LET Z = A + B*EXP(ARCCOT(X) + C)</td>
</tr>
<tr>
<td>Arcsecant</td>
<td>ARCSEC(X)</td>
<td>LET Z = A + B*EXP(ARCSEC(X) + C)</td>
</tr>
<tr>
<td>Arc cosecant</td>
<td>ARCCSC(X)</td>
<td>LET Z = A + B*EXP(ARCSEC(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Sine</td>
<td>SINH(X)</td>
<td>LET Z = A + B*EXP(SINH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Cosine</td>
<td>COSH(X)</td>
<td>LET Z = A + B*EXP(COSH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Tangent</td>
<td>TANH(X)</td>
<td>LET Z = A + B*EXP(TANH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Cotangent</td>
<td>COTH(X)</td>
<td>LET Z = A + B*EXP(COTH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Secant</td>
<td>SECH(X)</td>
<td>LET Z = A + B*EXP(SECH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Cosecant</td>
<td>CSCH(X)</td>
<td>LET Z = A + B*EXP(CSCH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Arcsine</td>
<td>ARCSINH(X)</td>
<td>LET Z = A + B*EXP(ARCSINH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Arccosine</td>
<td>ARCCOSH(X)</td>
<td>LET Z = A + B*EXP(ARCCOSH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Arctangent</td>
<td>ARCTANH(X)</td>
<td>LET Z = A + B*EXP(ARCTANH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Arccotangent</td>
<td>ARCCOTH(X)</td>
<td>LET Z = A + B*EXP(ARCCOTH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Arcsecant</td>
<td>ARCCSECH(X)</td>
<td>LET Z = A + B*EXP(ARCCSECH(X) + C)</td>
</tr>
<tr>
<td>Hyperbolic Arc cosecant</td>
<td>ARCCSCH(X)</td>
<td>LET Z = A + B*EXP(ARCCSCH(X) + C)</td>
</tr>
</tbody>
</table>

### Probability Functions--

<table>
<thead>
<tr>
<th>Description</th>
<th>Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal N(0,1) cumulative distribution function</td>
<td>NORCDF(X)</td>
<td>LET Z = A + B*EXP(NORCDF(X) + C)</td>
</tr>
<tr>
<td>t</td>
<td>TCDF(X, NU)</td>
<td>LET Z = A + B*EXP(TCDF(X) + C)</td>
</tr>
<tr>
<td>F</td>
<td>CHPCDF(X, NU)</td>
<td>LET Z = A + B*EXP(CHPCDF(X) + C)</td>
</tr>
<tr>
<td>Normal N(0,1) probability density function</td>
<td>NORMPDF(X)</td>
<td>LET Z = A + B*EXP(NORMPDF(X) + C)</td>
</tr>
<tr>
<td>t</td>
<td>TDPPDF(X, NU)</td>
<td>LET Z = A + B*EXP(TDPPDF(X) + C)</td>
</tr>
<tr>
<td>F</td>
<td>CHSPPDF(X, NU)</td>
<td>LET Z = A + B*EXP(CHSPPDF(X) + C)</td>
</tr>
<tr>
<td>Normal N(0,1) percent point function</td>
<td>NORPFP(X)</td>
<td>LET Z = A + B*EXP(NORPPF(X) + C)</td>
</tr>
<tr>
<td>t</td>
<td>TPFPF(X, NU)</td>
<td>LET Z = A + B*EXP(TFFP(X) + C)</td>
</tr>
<tr>
<td>F</td>
<td>CHSPFPF(X, NU)</td>
<td>LET Z = A + B*EXP(CHSPFPF(X) + C)</td>
</tr>
<tr>
<td>Normal N(0,1) percent point function</td>
<td>NOTPPF(X)</td>
<td>LET Z = A + B*EXP(FTPFPF(X) + C)</td>
</tr>
<tr>
<td>t</td>
<td>TPFPF(X, NU)</td>
<td>LET Z = A + B*EXP(FTPFPF(X) + C)</td>
</tr>
<tr>
<td>F</td>
<td>CHSPFPF(X, NU)</td>
<td>LET Z = A + B*EXP(FTPFPF(X) + C)</td>
</tr>
</tbody>
</table>
Creating Parameters, Variables, and Functions

LET, LET FUNCTION, READ, and SERIAL READ

A parameter is a named scalar and may be defined via the LET command, as in

```
LET A = 17.26
LET B = 3.97
LET C = -5.38
LET D = 2.4*10**(-8)
LET E = 8**2-4*A*C
LET F = MEAN X
```

A variable is a named vector (a named single-dimension array), and may be defined via the LET, READ, and SERIAL READ commands, as in

```
LET Y = SEQUENCE 1 1 10
LET Z = PATTERN 1 2 3 FOR I = 1 1 9
LET U = NORMAL RANDOM NUMBERS FOR I = 1 1 100
LET X2 = X**2-LOG(Y)
LET Y2 = SORT(X+Y**3)
```

or

```
READ X Y
1 1
2 4
9 3
4 16
5 25
END OF DATA
```

or

```
SERIAL READ X Y
1 1 2 4 3 9 4 16 5 25
END OF DATA
```

A function is a named character string, and may be defined via the LET FUNCTION command, as in

```
LET FUNCTION P = EXP(-0.5*X**2)
LET FUNCTION G = SIN(2*PI*W*T)
LET FUNCTION H = F+LOG(G)
LET FUNCTION P2 = DERIVATIVE P WRT X
```

Functions may be concatenated and built-up piece-by-piece, as in

```
LET FUNCTION NUM = EXP(-ALPHA*X)
LET FUNCTION DENOM = A+B*X
LET FUNCTION RATIO = NUM/DENOM
```

which is equivalent to

```
LET FUNCTION RATIO = EXP(-ALPHA*X)/(A+B*X)
```

Copying Parameters, Variables, and Functions

LET

Copying parameters, variables, and functions is just a special case of more general operations involving the creation of parameters, variables, and functions.

Parameters may be copied via the LET command, as in

```
LET B = A
LET C = A
```

If A in the above examples is a parameter, then B and C will become parameters with the same value as A.

Variables may be copied with the LET command, as in

```
LET Y = X
LET Z = X
```

If X in the above examples is a variable, then Y and Z will become variables with the same values as X.

Functions may be copied via the LET FUNCTION command, as in

```
LET FUNCTION G = F
LET FUNCTION H = F
```

If F in the above examples is a function, then G and H will become functions with the same character string as F.
Deleting Parameters, Variables, and Functions

DELETE

The DELETE command will cause parameters, variables, and functions to be deleted. The form for the delete command is

    DELETE  list of parameters, variables, and/or function names

Thus, to delete parameters A and B, variables X and Y, and functions F and G, one could enter

    DELETE A
    DELETE B
    DELETE X
    DELETE Y
    DELETE F
    DELETE G

or simply

    DELETE A B X Y F G

Note that the DELETE command may be used in conjunction with SUBSET/EXCEPT/FOR qualifications to accomplish a partial or selected delete of elements within variables (partial deleting of parameters and functions may not be done). Whenever a partial delete of a variable is done, then the undeleted elements in the variable are automatically "packed" into the first available elements of the variable. For example,

    DELETE X Y SUBSET LAB 4

would delete all elements in X and Y corresponding to LAB 4 (and pack all remaining elements of X and Y). Note that X and Y would end up with the same number of elements, but the LAB variable would be unchanged and so would be longer and "out of alignment" with the new X and Y. To circumvent this, one could have entered

    DELETE X Y LAB SUBSET LAB 4

which would carry out the delete, and result in the variables X, Y, and LAB all having the same (shortened) length. The DELETE may be used with any general SUBSET/EXCEPT/FOR qualification. In particular,

    DELETE X Y FOR I = 1 1 20

would delete elements 1 through 20 of variables X and Y. The net result is that element 21 would shift up and become element 1, element 22 would become element 2, etc.

Specific elements of a variable may be deleted by explicit referencing of the element; for example,

    DELETE X(4) Y(10) Z(30)

which would delete the 4-th element of X, the 10-th element of Y, and the 30-th element of Z.

Immediately after the deletion is done, the remaining elements in the variable are packed. To delete successive elements in the same variable, the analyst should delete the higher elements first and then proceed to the lower elements. Thus to delete elements 2, 7, and 15 of variable X, one should enter

    DELETE X(15) X(7) X(2)

which is equivalent to

    DELETE X(15)
    DELETE X(7)
    DELETE X(2)

rather than

    DELETE X(2) X(7) X(15)

which is equivalent to

    DELETE X(2)
    DELETE X(7)
    DELETE X(15)

Because of the successive packing after each deletion, the latter code would not yield the desired results--the former code should be used. (Note that one need not heed this caution when one uses the SUBSET/EXCEPT/FOR qualification for deleting elements).

A practical example of when the deletion of specific elements would be useful is when the analyst has a variable Y, and suspects (perhaps via a lag-1 plot) that the i-th element of Y was related to the (i-1)st element of Y, in such a fashion that a first-order autoregressive model would be appropriate. To carry this out one could enter

    LET Y1 = Y
    LET N = NUMBER Y
    DELETE Y1(1)
    DELETE Y(N)
    FIT Y = A0 + A1*Y1

The above code would

1) Copy Y into Y1 (the elements of Y remain unaffected);

2) Determine the number of elements in Y (place this into the parameter N);

3) Delete the first element of Y1 (and shift all remaining elements up 1); note that Y1 now has N-1 elements;

4) Delete the last element of Y (so that Y also has N-1 elements);

5) Carry out a least squares linear fit of Y on Y1; thus yielding a first-order autoregressive fit.
Partial deletion of variables may also be done via the RETAIN command. The form is the same, but the RETAIN command will retain only those elements specified, and delete all others. As with the DELETE command, the remaining elements are "packed" into the first available elements after the deletion is done. Thus

\begin{verbatim}
RETAI N X Y FOR I = 10 1 20
\end{verbatim}

would delete all elements in X and Y except those from 10 to 20. These 11 elements would then be shifted up to become elements 1 to 11 of the new, shortened variables X and Y. The new length of X and Y would be 11.
Assigning Multiple Names to a Variable

NAME

The `NAME` command allows the analyst to attach more than one name to a variable. For example, suppose a variable `X` existed, to attach an additional name `X2`, say, to `X`, one would enter

```
NAME X2 X
```

The net effect is that the name `X2` would be attached to the name `X`, and either of the two names could be used at any time after that to refer to the same variable. Note that

```
NAME X2 X
```

is different than

```
LET X2 = X
```

The latter would in fact create a second variable `X2` and fill it with values of the variable `X`. This second variable `X2` is distinct from `X` (and takes up additional space in the internal DATAPLOT data storage area).

One may use the `NAME` command to attach as many different names as desired to an existing variable. To check what names are currently attached to a variable, the analyst should enter the `STATUS` command.

Renaming variables is not commonly done but does have application when used in conjunction with subprograms. Suppose, for example, that one has constructed a general subprogram that is residing in a file `ANALYSIS`, and which is written in general terms so as to operate on variables `Y` and `X`, such as

```
PLOT Y X
SUMMARY Y
FIT Y = A+B*X
CHARACTER X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
PLOT RES X
```

How can one use this subprogram if the variables in the main program are called something other than `X` and `Y`? This is done via the `NAME` command, for example,

```
READ DAY1. PRES1 VOL1
NAME X PRES1
NAME Y VOL1
TITLE CALIBRATION 1
CALL ANALYSIS.
```

```
READ DAY2. PRES2 VOL2
NAME X PRES2
NAME Y VOL2
TITLE CALIBRATION 2
CALL ANALYSIS.
```

```
READ DAY3. PRES3 VOL3
NAME X PRES3
NAME Y VOL3
TITLE CALIBRATION 3
CALL ANALYSIS.
```

The above code would

1) read data from file `DAY1` into the variables `PRES1` and `VOL1`;
2) assign the additional name `X` to `PRES1`;
3) assign the additional name `Y` to `VOL1`;
4) specify the title (of future plots) to be `CALIBRATION 1`;
5) invoke the subprogram residing in file `ANALYSIS` so as to carry out an analysis;
6) read data from file `DAY2` into the variables `PRES2` and `VOL2`;
7) assign the additional name `X` to `PRES2`;
8) assign the additional name `Y` to `VOL2`;
9) specify the title (of future plots) to be `CALIBRATION 2`;
10) invoke the subprogram residing in file `ANALYSIS` so as to carry out an analysis;
11) read data from file `DAY3` into the variables `PRES3` and `VOL3`;
12) assign the additional name `X` to `PRES3`;
13) assign the additional name `Y` to `VOL3`;
14) specify the title (of future plots) to be `CALIBRATION 3`;
15) invoke the subprogram residing in file `ANALYSIS` so as to carry out an analysis.
Creating Data Internally
LET, READ, and SERIAL READ

The most common way to create data in DATAPLOT is to simply read it in via the READ and SERIAL READ commands. Such data is usually read from an external mass storage file. Occasions arise, however, where it is convenient to create data internal to a DATAPLOT program. For example, in carrying out a spline fit, it is required to specify a variable containing the knot positions. The LET command may be used directly to create elements of a variable. For example,

```
LET X(1) = 20
LET X(2) = 55
LET X(3) = 23.66
```

will result in the first element of the variable X being set to the value 20, the second element of the variable X being set to 55, and the third element of the variable X being set to 23.66. Other related forms for this use of the LET command are shown in the following program--

```
LET A = 23
LET X(4) = A
LET X(5) = X(2)
LET X(6) = A+SQR(2.4)
LET J = 7
LET K = 3
LET X(J) = Y(K)
```

The above code will

1) define a parameter A with the value 23;
2) set the fourth element of the variable X equal to A (which currently has value 23);
3) set the fifth element of the variable X to the same value as the second element of the variable X;
4) set the sixth element of the variable X equal to A+SQR(2.4)--note that A has the value 23;
5) set the parameter J = 7;
6) set the parameter K = 3;
7) set the J-th element of the variable X to the same value as the K-th element of the variable Y--note that J and K currently have the values 7 and 3, respectively.

Note that if an element appears on the right side of the assignment statement, it must not be part of an arithmetic expression; thus

```
LET A = X(2)
LET B = X(5)
LET X(10) = A + B**2
```

are legal, but

```
LET X(10) = X(2) + X(5)**2
```

will result in an error message.

The SEQUENCE sub-form of the LET command allows the analyst to create a variable consisting of a sequence of numbers. The general form is

```
LET variable = SEQUENCE start increment stop
```

as in

```
LET X = SEQUENCE 1 1 10
```

which would result in the creation of a variable X with the 10 values--1, 2, ..., 9, 10. The command

```
LET Y = SEQUENCE 0 .1 20
```

creates the variable Y with 201 values--0, .1, .2, ..., 19.9, 20.

```
LET Z = SEQUENCE 10 -.1 -10
```

creates the variable Z with 201 values--10, 9.9, 9.8, ..., -9.9, -10.

```
LET START = 1
LET INC = .01
LET STOP = 2
LET U = SEQUENCE START INC STOP
```

creates the parameters START, INC, and STOP, and then creates the variable U with 101 values--1, 1.01, 1.02, ..., 1.99, 2.

The PATTERN sub-form of the LET command allows the analyst to create a variable consisting of a pattern of numbers. The general form is

```
LET variable = PATTERN sequence of values
```

as in

```
LET X = PATTERN 1 2 3 4 5 6 7 8 9 10
```

which would result in the creation of a variable X with the 10 values--1, 2, ..., 9, 10. The command

```
LET Y = PATTERN 1 2 3
```

creates the variable Y with 3 values--1, 2, and 3.

```
LET Z = PATTERN 1 1 1 2 2 2 3 3 3
```

which would result in the creation of a variable Z with the 9 values--1, 1, 1, 2, 2, 2, 3, 3, 3.
creates the variable \( Z \) with 9 values—1, 1, 1, 2, 2, 2, 3, 3, and 3. To repeat patterns, one simply augments the PATTERN sub-form with a \( \text{FOR} \) qualification; for example,

\[
\text{LET } X = \text{PATTERN 1 2 3 FOR } I = 1 1 12
\]

creates a variable \( X \) with 12 values—1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3. The use of the PATTERN command arises in connection with forming a variable (if not already available) for defining individual traces in a multi-trace plot, and in defining a variable for carrying out ANOVA and graphical ANOVA.

To generate random numbers internally, the analyst uses the RANDOM NUMBERS subform of the LET command. The general form is

\[
\text{LET variable} = \text{distribution RANDOM NUMBERS FOR } I = \text{start increment stop}
\]

The distribution name identifies the distribution from which random numbers are to be drawn. The \( \text{FOR} \) qualification at the end of the command tells DATAPLOT into which elements of the variable the random numbers are to be placed. Thus

\[
\text{LET } X = \text{NORMAL RANDOM NUMBERS FOR } I = 1 1 100
\]

will generate 100 normal random numbers and place them into elements 1 to 100 of variable \( X \).

\[
\text{LET } X = \text{NORMAL RANDOM NUMBERS FOR } I = 1 2 50
\]

\[
\text{LET } X = \text{CAUCHY RANDOM NUMBERS FOR } I = 2 2 50
\]

will generate 25 normal random numbers and place them into elements 1, 3, 5, ..., 49 of variable \( X \), and then generate 25 Cauchy random numbers and place them into elements 2, 4, 6, ..., 48, 50 of variable \( X \). To specify particular members of a distributional family, specify the distributional parameter value prior to use of the RANDOM NUMBERS sub-form, as in

\[
\text{LET } \text{nu} = 5
\]

\[
\text{LET } X = \text{t RANDOM NUMBERS FOR } I = 1 1 30
\]

which will generate 30 random numbers from the \( t \) distribution with \( \text{nu} = 5 \) degrees of freedom.

\[
\text{LET } \text{gamma} = 2.57
\]

\[
\text{LET } X = \text{weibull RANDOM NUMBERS FOR } I = 1 1 200
\]

will generate 200 random numbers from the Weibull distribution with \( \text{gamma} = 2.57 \).
Input/Output

Files and Subfiles 4-1
Input/Output 4-3
Reading in Columns of Data 4-4
READ
Reading Data in Sequentially 4-7
SERIAL READ
Reading in Parameters 4-8
READ PARAMETER
Reading in Functions 4-9
READ FUNCTION
Writing Out Parameters, Variables, and Functions 4-10
WRITE
Skipping Over Lines in a Read 4-11
SKIP
Restricting a Read to Certain Rows 4-11
ROW LIMITS
Restricting a Read to Certain Columns 4-12
COLUMN LIMITS
Terminating a Read 4-12
END OF DATA
Files and Subfiles

The DATAPLOT commands which may contain file/subfile specifications are:

- **READ** read data from a file;
- **SERIAL READ** read sequential data from a file;
- **WRITE** write data out to a file;
- **CREATE** dynamically create a subprogram;
- **CALL** execute a subprogram;
- **SAVE** save all internal DATAPLOT settings;
- **RESTORE** restore all internal DATAPLOT settings.

The DATAPLOT convention in designating files and subfiles is as follows:

1.Qualifier;
2.qualifier-file separator (an asterisk);
3.file name;
4.file name-subfile name separator (a period);
5.subfile name.

The qualifier is typically a user identifier, a project identifier, or simply an extended file name. The qualifier allows the computer to distinguish between 2 files which nominally have the same name. For example, if analyst Smith has a file called DATA, and analyst Jones has a file called DATA, then to distinguish between these 2 files called DATA, the analyst may be required to enter Smith*DATA or Jones*DATA, respectively, as in

```
READ Smith*DATA. X Y
```

```
READ Jones*DATA. X Y
```

Qualifier names may be up to 12 characters. On some computer systems, the qualifier name is optional; on others, it is required. If optional, and if omitted, then the computer system usually substitutes a default qualifier name that was defined by the user at the time of log-on. For example, on some computers, analyst Smith may be able to enter

```
READ DATA. X Y
```
as a shorter form of

```
READ Smith*DATA. X Y
```

The qualifier separator tells DATAPLOT that the qualifier name is finished, and the file name is starting. The separator is typically 1 character in length. At NBS, the separator character is * (an asterisk). Other computers may choose to some other separator besides *. For example, on VAX systems, the separator is usually 1.

The file name is up to 12 characters. The period at the end of the file name tells DATAPLOT that the file name is finished and the subfile name (if existent) is starting. The period must be appended to all file names in all relevant DATAPLOT commands regardless of whether the system allows subfiles or not. The period is important because it tells DATAPLOT that the entry is a file name as opposed to a variable name. For example, in the READ command, if DATAPLOT encounters

```
READ A, B C
```

then it expects to read from file A and to read the first 2 numbers on each line image of file A into variables B and C. On the other hand, if DATAPLOT encounters

```
READ A B C
```

then the absence of a file name is a signal to DATAPLOT that the data will be read in from the terminal where the first 3 numbers on each line image will be placed into the variables A, B, and C, as in

```
READ A B C
1 1 1
2 4 8
3 9 27
4 16 64
END OF DATA
```

Even though such ambiguity does not exist in other commands, such as

```
SAVE DATA.
```

(which will allow all internal DATAPLOT switch settings to be saved to the file DATA because the DATAPLOT run has to be interrupted for some reason) the DATAPLOT convention (for consistency's sake) is to have all file names appended with a period in all commands; thus

```
SAVE DATA.
```

is correct, while

```
SAVE DATA
```

is incorrect.

Some computer systems allow subfile structure—that is, allow files to be logically subdivided into smaller units which may be treated as logical entities unto themselves. For example, UNIVAC systems allow such subfiles, while some computers may not allow subfiles. If the computer system allows subfile structure, then DATAPLOT allows such names to be up to 12 characters, as in
READ DATA.CALIBRATION7 X Y

which would instruct DATAPLOT to read from subfile CALIBRATION7 of file DATA, and to read the first 2 values on each line image into variables X and Y.

The above allowances (12 characters for qualifiers, 12 characters for file names, and 12 characters for subfile names) are DATAPLOT allowances; these limits may be overridden by restrictions on name length imposed by the computer system itself (for example, some may allow only 8 characters for names). The user should check with local system personnel to determine if such restrictions exist.
Input/Output

DATAPLOT input/output is carried out via the following 3 commands--

READ
SERIAL READ
WRITE

All reading and writing is format-free. There are no format statements in DATAPLOT.

The READ and SERIAL READ have 4 auxiliary commands which may at times be of use to the analyst in reading data into DATAPLOT--

SKIP
ROW LIMITS
COLUMN LIMITS
END OF DATA
Reading in Columns of Data

READ

The most common method of inputting data into DATAPLOT is via the READ command. The general form for the READ command is

READ file name. (optional) list of variable names

In the following examples, data will be read from the terminal (as part of the program)--

READ X
READ X Y
READ X Y Z
READ Y1 Y2 Y3 Y4 Y5 Y6 X
READ RESPONSE TEMP VOLUME PRESSURE LAB

In the following examples, data will be read from file A. (the period at the end of the file name (A. as opposed to A) is important—it tells DATAPLOT that A is a file and not a variable). The period at the end of the file name is used in the NBS implementation—other implementations may have other punctuation to indicate a file (for example, a slash /). Some punctuation must be used, however, in order for DATAPLOT to distinguish file names from variable names. Check the local DATAPLOT service organization for file indicators at your installation.

READ A. X
READ A. X Y
READ A. X Y Z
READ A. Y1 Y2 Y3 Y4 Y5 Y6 X
READ A. RESPONSE TEMP VOLUME PRESSURE LAB

The input data file must be an editable file. In particular, such a file may not be a binary file that would (for example) result from a FORTRAN program which uses a formatless write, as in

WRITE(7)(X(I),Y(I),I=1,1000)

Such a FORTRAN write statement results in a packed, binary output file which is efficiently generated, but nevertheless uneditable. Such a file may not serve as input into DATAPLOT.

Some computers allow subfile structures. Thus in the following examples, data will be read from subfile B of file A--

READ A.B X
READ A.B X Y
READ A.B X Y Z
READ A.B Y1 Y2 Y3 Y4 Y5 Y6 X
READ A.B RESPONSE TEMP VOLUME PRESSURE LAB

A specific example of reading data in from the terminal itself (or the program itself) is as follows--

READ A.B X Y
DELETE X Y
SUBSET X < -9000
DELETE X Y
SUBSET Y < 9000
FIT Y = A+B*EXP(-C*X)
PLOT Y PRED VERSUS X
PLOT RES X
If there are more variables specified in the READ statement than there are data values on the line image, as in

READ X Y Z U
1 1 1
2 4 8
3 9 27
END OF DATA

then an error message will be generated. If there are fewer variables specified than there are data values, as in

READ X Y
1 1 1
2 4 8
3 9 27
END OF DATA

then only as many data values per line image will be read as there are variables specified (2 in this case), and remaining data values on each line image will be ignored. Thus in the example, the variable X will end up with values 1, 2, and 3; and the variable Y will end up with the values 1, 4, and 9. The values 1, 8, and 27 will be ignored.

Data may be in usual floating point representation, as in

1.234
-23.6489
26.00054

or exponential format, as in

1.234e+4
1.234E+4

Note that the E format is permitted for data on a file/subfile (because such a file is commonly the output from a previously-run FORTRAN program).
(On the other hand, it is recalled that E format is not permitted in any form of the LET command; thus LET A = 1.234E+4 is illegal).

Trailing zeros and the decimal point in integers may be omitted, as in

1.0
1
1

Note the following features of reading data into DATAPLOT:

1) data on succeeding line images do not have to be neatly lined up one under the other—the read is free-format;

2) the analyst does not need to be concerned with pre-specifying the number of line images—the READ will continue until an END OF DATA statement is encountered or until an end of file is encountered.

The READ command will generate a feedback message indicating the first line image that was read, the last line image that was read, the number of line images read, and the number of variables read.

Reading from multiple data files may be carried out. Suppose file A consisted of 5 line images—

11
12
13
14
15

and file B consisted of 3 line images—

21
22
23

then what will be the effect of the following 2 statements—

READ A X
READ B X

The first READ will fill the first 5 locations of X with 11 to 15. The second READ will fill the first 3 locations of X with 21 to 23. The net effect after both reads is that the variable X will contain the following 5 observations—

21
22
23
14
15

The rule is that all READs always start at the top of a variable unless otherwise directed. This allows variables to be partially overwritten which may (at times) be what the analyst intended. However, the more usual case is that the analyst would like to have data from various files concatenated one behind the other. This may be done by use of SUBSET/EXCEPT/FOR qualifications attached onto the end of the READ statement. For example,

READ A X
READ B X FOR I = 6 1 8

will have what effect? The first READ will (as before) fill the first 5 locations of X with 11 through 15. The second READ tells DATAPLOT to read from file B, but to place the read values internally in the variable X starting with location 6, at increments of 1, up to location 8—thus the 3 data values will be read into locations 6, 7, and 8 of variable X. After both reads, the variable X will contain the following 8 observations—
With the FOR qualification, we can direct data anywhere we like within a variable (or variables). It is a valuable tool for data entry.

The SUBSET/EXCEPT qualifications on the end of the READ statement behave similarly. They work with variables that have already been defined (via READ, LET, etc.). For example, suppose the analyst already had a variable called LAB which has 15 values (say)--

```
1
2
1
2
1
2
1
2
1
2
2
1
1
2
1
```

What would be the effect of the following statement?

```plaintext
READ A. X SUBSET LAB 1
```

The effect is that the 5 data values (11 through 15) of file A. would be read in, but these data values would not be placed into the first 5 locations of variable X, rather, they would be placed in the first available locations as defined in the SUBSET qualification (SUBSET LAB 1). Since the internal variable LAB has value 1 for locations 1, 3, 5, 7, 9, 11, 13, and 15, then the net effect is that the 5 data values that were read in would be placed in locations 1, 3, 5, 7, and 9 of variable X. What would be in locations 2, 4, 6, etc.? These locations will remain unaffected by the READ, and so they will contain whatever values they had before the READ. (Note that all values in DATALOT's internal data area have a default sign-on value of negative machine infinity.) Much more complicated SUBSET/EXCEPT/FOR constructs may be formed, although in practice these complicated constructs are rarely needed--especially for input.
Reading Data in Sequentially
SERIAL READ

All of the comments and conventions for the READ command also hold for the SERIAL READ command except for the following important difference—whereas the READ command only reads a fixed number of data values per line image, the SERIAL READ allows the analyst to read all the data values on a line image. The motivation for this is that some analysts prefer to place data on a file/subfile in a serial fashion (multiple values of a variable on each line image), as opposed to the more common parallel fashion (only 1 value of each variable on each line image). To amplify the difference between READ and SERIAL READ, consider the following—

Suppose the file A consisted of the following 4 line images—

11 12 13 14 15 16
21 22 23 24
31 32
41 42

The command

```
READ A. X
```

would result in X having 4 values—11, 21, 31, and 41. The command

```
SERIAL READ A. X
```

would result in X having 14 values—11, 12, 13, 14, 15, 16, 21, 22, 23, 24, 31, 32, 41, and 42.

Further, if instead of reading 1 variable we had read 2 variables, as in

```
READ A. X Y
```

then X would have 4 values—11, 21, 31, and 41; and Y would have 4 values—12, 22, 32, and 42; but the command

```
SERIAL READ A. X Y
```

would result in X having 7 values—11, 13, 15, 21, 23, 31, and 41; and Y having 7 values—12, 14, 16, 22, 24, 32, and 42.

And if instead, we had read in 3 variables, as in

```
READ A. X Y Z
```

then this would have resulted in an error message because lines 3 and 4 contain no Z value; whereas, the command

```
SERIAL READ A. X Y Z
```
Reading in Parameters

READ PARAMETER

The READ PARAMETER command allows the analyst to read in one or more parameters. The READ PARAMETER command will cause exactly one line image to be read and scanned. Thus

READ PARAMETER ZZZ. A B C

will cause the next line image of the file ZZZ to be read and scanned, and the first 3 numbers on this line image will be placed into the DATAPLOT parameters A, B, and C. If more than 3 numbers exist on the line, then the remaining numbers will be ignored. If less than 3 numbers exist on the line, then as many parameters as numbers will be formed and the remaining parameters will be ignored.

If no file name is entered, as in

READ PARAMETER A B C

then the next line image from the terminal (or program) will be used, and so

READ PARAMETER A B C
11 12 13

will result in creating 3 internal parameters A, B, C, and assigning the value 11 to parameter A, the value 12 to parameter B, and the value 13 to parameter C.

Note that only the next single line is scanned for parameters. This convention was imposed for simplicity, and is no constraint in practice because the analyst may enter multiple READ PARAMETER statements if multiple lines of parameters exist. Thus

READ PARAMETER ZZZ. A B C
READ PARAMETER ZZZ. D E F G H

with the next 2 lines of file ZZZ. consisting of

23.6  -55  212
3400  26.97  85.88  0.0056  10

will result in the prescribed 8 parameters being properly defined.
Reading in Functions
READ FUNCTION

The READ FUNCTION command allows the analyst to read in a function. The READ FUNCTION command will cause exactly one line image to be read and scanned. Thus

READ FUNCTION ZZZ. F

will cause the next line image of the file ZZZ to be read and scanned, and the text on that line to be placed into the DATAPLOT function F. Leading and internal blanks are preserved, but trailing blanks are ignored.

If no file name is entered, as in

READ FUNCTION F

then the next line image from the terminal (or program) will be used, and so

READ FUNCTION F
EXP(-X**2)

will result in the internal function F being created which has the 10-character contents EXP(-X**2).

Note that only 1 function may be read at a time, and only 1 line is scanned for such a function. Such conventions were imposed for simplicity, and they hardly serve as any constraint on the analyst since functions may be freely concatenated inside DATAPLOT via the LET FUNCTION command. Thus

READ FUNCTION ZZZ. F
READ FUNCTION ZZZ. G
LET FUNCTION H = F*G

with the next 2 lines of file ZZZ, consisting of

EXP(-X**2)
SIN(X)/SQRT(LOG(X))

will result in function F being defined as

EXP(-X**2)

the function G being defined as

SIN(X)/SQRT(LOG(X))

and the function H being defined as

EXP(-X**2)*SIN(X)/SQRT(LOG(X))
Writing Out Parameters, Variables, and Functions

WRITE

The WRITE command may be used to write out parameters, variables, or functions. The general form for the WRITE command is

```
WRITE  file/subfile name (optional)  list of names
```

To write information back to the terminal, one omits the file/subfile name and just enters the list of parameter/variable/function names to be printed, as in

```
WRITE X
WRITE A B
WRITE A B C
WRITE X Y Z U
WRITE Y X G PRED RES REPSD REPDF
```

The list of names may be in arbitrary order--parameter names, variable names, and function names may be mixed at will. To write to a file, one enters the file name immediately after WRITE, as in

```
WRITE A X
WRITE A A B
WRITE A A B C
WRITE A X Y Z U
WRITE A Y X G PRED RES REPSD REPDF
```

which would write out the designated parameter/variable/function names to file A. The period following the file name A. is important--it is that punctuation which tells DATAPLOT that A. is a file name as opposed to a parameter/variable/function name. The period at the end of the file name is the convention used with the NBS implementation--check locally for what the punctuation is that defines the end of a file name at your implementation. The WRITE command may also be used to write out strings, comments, and headings. The general form for this purpose is

```
WRITE "string to be written out"
```

as in

```
WRITE "LABORATORY 4 ANALYSIS"
```

which would print the line

```
LABORATORY 4 ANALYSIS
```

on the next available line. This is useful for headings as in

```
READ A X Y
ERASE
WRITE "STROTIUM ANALYSIS"
FIT Y = A+B/(C+X)
COPY
```

which would

1) read in data;
2) erase the screen;
3) print out the line STRONTIUM ANALYSIS ;
4) carry out a non-linear fit;
5) copy the screen contents to hardcopy.

The WRITE command may be used with a SUBSET/EXCEPT/FOR qualification to write out interesting subsets of the data. For example, if X and Y were variables,

```
WRITE X Y
```

would write out all values of X and Y; but

```
WRITE X Y FOR I = 11 20
```

would write out only elements 11 to 20 of variables X and Y; while

```
WRITE X Y SUBSET LAB 4
```

would write out only those elements of X and Y for which the LAB variable has the value 4; and

```
WRITE X Y SUBSET Y > 100
```

would write out all values of X and Y for which Y > 100 (which is useful for listing those data lines corresponding to outliers). Any general complicated SUBSET/EXCEPT/FOR qualification may be appended to the WRITE statement.
Skipping Over Lines in a Read

**SKIP**

The **SKIP** command sets an internal switch which indicates how many lines should be skipped in a file/subfile before the reading of data. The default setting is 0 (no lines to be skipped). To change the setting, enter

```
SKIP number
```

where the number indicates the desired number of lines at the beginning of the file which are to be skipped over (= ignored) for any subsequent reads (via READ, SERIAL READ, or READ FUNCTION). This command is usually used to skip over header and other non-numeric information at the beginning of a data file. Thus if the file A. consisted of the following 6 lines--

```
CALIBRATION STUDY
LABORATORY A
11
21
31
END OF DATA
```

and if one entered

```
SKIP 1
READ A. X
```

then an error message would result (because no numeric information is on line 2. If one entered

```
SKIP 2
READ A. X
```

then the variable X would end up with the 3 values 11, 21, and 31. If one entered

```
SKIP 3
READ A. X
```

then the variable X would end up with only 2 values-- 21 and 31.

Note that as with all underlying DATAPLOT settings, once the SKIP setting is made, it remains in effect for the duration of the run, or until overridden by another SKIP command.

Restricting a Read to Certain Rows

**ROW LIMITS**

The **SKIP** command allows one to skip over lines at the beginning of a file. The **ROW LIMITS** command is similar but it gives the analyst the ability to ignore lines at both the beginning and the end of a file. The general form for the command is

```
ROW LIMITS number number
```

where the first number is the first line number of the data file to be read, and the second number is the last number of the data file to be read. The default values for the row limits switch are 1 and infinity. Thus if the file A. consisted of

```
11
21
31
41
51
```

and if one entered

```
ROW LIMITS 2 4
READ A. X
```

then the variable X would end up with 3 values-- 21, 31, and 41.

The **ROW LIMITS** command adds a third rule to the list of those governing when a READ will stop; the 3 rules are

1) if an END OF DATA line is encountered;
2) if an end of file is encountered;
3) if the row limits maximum is encountered.
Restricting a Read to Certain Columns

COLUMN LIMITS

The COLUMN LIMITS command allows the analyst to focus on a certain region within a line image. The general form for the command is

```
COLUMN LIMITS number number number
```

where the column numbers indicate the start and stop columns for the data image scan. Thus if the data file consists of the following 3 lines

```
 11 12 13
 21 22 23
 31 32 33
```

where the first number on each line starts in column 1, the next number on each line starts in column 11, and the third number on each line starts in column 21, then

```
COLUMN LIMITS 10 15
READ X
```

would result in the variable X having 3 values—12, 22, and 32.

Terminating a Read

END OF DATA

The usual termination for a READ, SERIAL READ, or READ FUNCTION command is when a line image is encountered which has

```
END OF DATA
```

or

```
END DATA
```

on it. This is true regardless of whether or not the END OF DATA line image is in the data file (which is the usual practice) or is in the program itself, as in

```
READ X
11
21
31
END OF DATA
```

If the analyst is reading from a data file/subfile, then the usual procedure is for the analyst to append an extra line image to the file/subfile consisting of

```
END OF DATA
```

via the local editor.

In addition to an END OF DATA line image being encountered, there are 2 other ways in which a read may be terminated—

1) when the end of the file or subfile being read is encountered;

2) when the pre-specified row limits maximum has been attained.

These last 2 procedures may be used without using an END OF DATA statement.

The general rule of thumb is, therefore, that a READ or SERIAL READ will continue processing data from a file until one of the following 3 conditions is met—

1) an END OF DATA line image is encountered;

2) an end of file or subfile is reached;

3) a row limits maximum is attained.

The read will terminate as soon as any one of these 3 conditions is satisfied.
5

Program Control

Looping
LOOP 5-1

Conditional Statements
SUBSET, EXCEPT, FOR, and IF 5-3

Calling Subprograms
CALL 5-5

Creating Subprograms
CREATE 5-6

Listing Subprograms
LIST 5-6
Looping

LOOP

Most DATA PLOT programs do not need loops. In
FORTRAN (for example) the use of loops is commonly
used for intermediate calculations en route to the
final output. In DATA PLOT, however, the existence
of numerous high-level graphics and analysis
commands frequently obviates the need for such
intermediate looping.

To initiate a loop in DATA PLOT, use the LOOP
command. The general syntax is

LOOP FOR parameter = start increment name
    value
END OF LOOP

The loop parameter will be a parameter in the
usual DATA PLOT sense. As the loop is being
executed, the parameter value will be successively
changed (depending on start, increment, and stop
values). The

END OF LOOP

commands will terminate the loop. This parameter
value is global and may be used by any calculation
or specification within the loop. Thus

LOOP FOR I = 1 1 10
    WRITE I
END OF LOOP

will result in the sequence 1, 2, 3, ..., 9, 10
being printed out.

LOOP FOR L = 1 1 5
    PLOT Y X SUBSET LAB L
END OF LOOP

will result in 5 distinct plots being generated--
1) a plot of Y versus X but restricted to
   those values for which the LAB variable = 1;
2) a plot of Y versus X but restricted to
   those values for which the LAB variable = 2;
3) a plot of Y versus X but restricted to
   those values for which the LAB variable = 3;
4) a plot of Y versus X but restricted to
   those values for which the LAB variable = 4;
5) a plot of Y versus X but restricted to
   those values for which the LAB variable = 5.

Note that

LOOP FOR D = 1 1 10
    FIT Y = A+B*X SUBSET DAY D
    LET A2(D) = A
    LET B2(D) = B
    LET S(D) = RESSD
END OF LOOP

will carry out 10 linear fits of Y on X--each fit
being restricted to values of the DAY variable =
to 1, then 2, then 3, etc. Further, for each fit,
the value of the fit parameter A will be copied
into the D-th element of variable A2, the value of
the fit parameter B will be copied into the D-th
element of the variable B2, and the value of the
fit residual standard deviation parameter RESSD
(automatically provided by DATA PLOT) will be
copied into the D-th element of the variable S.

FEEDBACK OFF
ERASE
LOOP FOR IX1 = 10 10 80
    LET IY1=IX1
    LET IX2=IX1+10
    LET IY2=IX2
    BOX IX1 IY1 IX2 IY2
END OF LOOP

will generate a sequence of boxes along the lower
left to upper right diagonal of the screen.

As is usual, the start value, increment, and stop
value may be either numbers or parameters. Thus

LOOP FOR L = 1 1 5
    PLOT Y X SUBSET LAB L
END OF LOOP

and

LET A = 1
LET B = 1
LET C = 5
LOOP FOR L = A B C
    PLOT Y X SUBSET LAB L
END OF LOOP

are equivalent.

If one does use parameters for specifying the
loop, then these parameters may be altered
within the loop--this capability is at
times used in conjunction with the IF
qualification of the LET command so as to
prematurely terminate a loop when a certain
condition is arrived at. For example,
LET ALPHAMAX = 10
LOOP FOR ALPHA = 0.1 ALPHAMAX
LET INT = INTEGRAL X**ALPHA WRT X FOR X = 0 TO 1
LET ALPHAMAX = -INFINITY IF INT < 0.05
END OF LOOP

will terminate itself as soon as a value of ALPHA is arrived at in which the integral of X**ALPHA (over the domain 0 to 1) is less than 0.05.

The checking of the loop parameters is done at the beginning of the loop. It is possible for a loop not to be executed at all (if the increment is positive and the stop value is smaller than the start value; or if the increment is negative and the stop value is larger than the start value); thus

LET STOP = -1
LOOP FOR A = 1 1 STOP
PRINT A
END OF LOOP

would result in the loop not being executed.

Loops may be nested 7 deep. Thus to generate plots of Y versus X for each combination of variables OP (with 3 levels), LAB (with 5 levels), and DAY (5 levels), one could enter

LOOP FOR O = 1 1 3
LOOP FOR L = 1 1 4
LOOP FOR D = 1 1 5
PLOT Y X SUBSET OP O SUBSET LAB L SUBSET DAY D
END OF LOOP
END OF LOOP
END OF LOOP

To have the 60 plots automatically hardcopied as they are being generated, and to have them automatically numbered (1 to 60), one could enter

HARDCOPY
SEQUENCE
LOOP FOR O = 1 1 3
LOOP FOR L = 1 1 4
LOOP FOR D = 1 1 5
PLOT Y X SUBSET OP O SUBSET LAB L SUBSET DAY D
END OF LOOP
END OF LOOP
END OF LOOP

Any valid DATAPLOT command can be executed within the context of a loop. A particularly important command is the CALL command which allows a subprogram to be referenced. Thus for more complicated applications, it may be of interest to set up a loop in the main program and then specify a desired set of operations in a subprogram, as in

LOOP FOR O = 1 1 3
LOOP FOR L = 1 1 4
LOOP FOR D = 1 1 5
CALL A.
END OF LOOP
END OF LOOP
END OF LOOP

where subprogram A might consist of

XLABEL OPERATOR = VALU(JO)
X2LABEL LAB = VALU(JL)
X3LABEL DAY = VALU(JD)
PLOT Y X SUBSET OP O SUBSET LAB L SUBSET DAY D

To have them hardcopied, sequenced, and also to have them properly labeled with the current operator, lab, and day value, one could use the VALU() sub-command in conjunction with the ...

...LABEL command, as follows--
Conditionally Executing Statements

SUBSET, EXCEPT, FOR, and IF

Conditional execution of statements is handled in 2 different ways in DATAPLOT:

1) the augmentation of SUBSET/EXCEPT/FOR qualifications at the end of any graphics or analysis command (this is used when the conditionality is based on values within variables);

2) the use of the IF command in conjunction with the END OF IF (or END IF) command (this is used when the conditionality is based on values of parameters).

Method 1 is the most popular and is the most straightforward way of conditionally executing commands. It is a DATAPLOT feature that any graphics command or any analysis command may be conditionally executed (that is, have its execution restricted to a certain subset of data, or have its execution contingent on the value of certain variables) by simply appending the qualification to the end of the command line. As far as the analyst is concerned, the primary general objective is to generate a plot, carry out a fit, carry out an analysis of variance, etc., whereas the restriction of such an activity to a specific subset is, in a sense, an afterthought. This being the case, the DATAPLOT structure is to have the DATAPLOT command (PLOT, FIT, ANOVA, etc.) appear first and foremost on the line, and then to simply append the qualification to the end of the line. Thus it is easy and natural to extend

PLOT Y X
FIT Y = A+B*LOG(X)
ANOVA Y X1 X2 X3
to

PLOT Y X SUBSET LAB 4
FIT Y = A+B*LOG(X) SUBSET X . 50 EXCEPT X = 140
ANOVA Y X1 X2 FOR I = 1 1 40

The specified subset is extracted before the command is executed. If the subset is empty, then the command will not be executed.

Method 2 is a much less popular way of conditionally executing statements, but does have occasional application. To conditionally execute a "chunk" of DATAPLOT code, one may use the IF command in combination with the END OF IF (or END IF) command. The general form for the IF command is

IF parameter or relative parameter or number operator number

where the relational operators are

= equality
<> inequality
< less than
<= or =< less than or equal to
> greater than
>= or => greater than or equal to

The conditionality specification is done in terms of parameter values (as opposed to variable values in the SUBSET/EXCEPT qualification). For example, suppose a variable X has 100 values in it and it is desired to determine the maximum value in X. The easiest way to do this in DATAPLOT is to simply enter

LET MAX = MAXIMUM X

However, for purposes of illustration of the use of the IF command, one could achieve the same result via the following statements:

LET MAX = -INFINITY
LOOP FOR I = 1 1 100
LET XI=X(I)
IF XI > MAX
LET MAX = X(I)
END OF IF
END OF LOOP

It is clear that the second method is an extremely inefficient method for computing the parameter MAX.

As a second example, suppose it is desired to determine that combination of A and B (over a specified lattice of values) for which the R1 fit of A*LOG(B+X) is achieved. The easiest way to carry this out in DATAPLOT is

FIT POWER 1
PRE-FIT A*LOG(B+X) FOR A = 1 1 10 FOR B = 100 10 200

At the end of the PRE-FIT, the optimal values will reside in the parameters A and B. An alternate (and much longer) way of doing the same is
LET MIN = INFINITY
LOOP FOR A = 1 1 10
LOOP FOR B = 100 10 200
  LET PRED = A+LOG(B*X)
  LET RES = Y-PRED
  LET RES2 = ABS(RES)
  LET SUMA? = SUM RES2
  IF SUMAD < MIN
    LET A2 = A
    LET B2 = B
    END OF IF
  END OF LOOP
END OF LOOP

At the end of execution of this code, the optimal values of A and B will reside in the parameters A2 and B2.

IF statements may be nested up to 7 deep, as in

IF A > 10
  IF B = 7
    IF C <= 100
      WRITE A B C
    END OF IF
  END OF IF
END OF IF

The parameters A, B, and C will be scanned; only if the 3 specified conditions are satisfied will the WRITE occur.

Any valid DATAPLOT statement may occur within the context of the IF set of statements. A particularly powerful entry is the CALL statement which allows subprograms to be executed. Thus, for example, to conditionally execute (only if the residuals standard deviation RESSD from some fit is less than .1) a subprogram XYZ, one could enter

IF RESSD < .1
  CALL XYZ.
END OF IF

The END OF IF may optionally be written as END IF, thus the above code may also be written as

IF RESSD < .1
  CALL XYZ.
END IF

The IF capability in DATAPLOT is a low-level capability (no one does IF as an end objective in itself). Conditionally executing a chunk of code is an elementary capability of any computer language. In DATAPLOT, due to the existence of high-level capabilities, and due to the SUBSET/EXCEPT/FOR qualification capability, the IF capability is only lightly used.
Calling Subprograms

CALL

The typical DATAPLOT program consists only of a main program. The existence of higher-level capabilities (e.g., PLOT, FIT, LET, etc.) often precludes complicated nesting of subprograms. However, subprograms may be included in DATAPLOT as needed.

Subprograms are completely independent programs in their own right and must be placed in separate mass storage files (or subfiles). Such subprograms are accessed via the CALL command and are identified by the name of the mass storage file (or subfile) in which the subprogram resides, as in

CALL XY2.

(which would execute the subprogram residing in file XY2), or

CALL XY2.ABC

(which would execute the subprogram residing in the subfile ABC of the file XY2). As with all DATAPLOT commands which involve file/subfile usage, the trailing period after the file name is important—it tells DATAPLOT that the name is a file name. Further, for those computer systems which allow logical subfiles within files, the period also serves as a separator between file name and subfile name.

When the last line of the subprogram in the file is executed, it returns to the main program to the statement immediately following the CALL statement. The name of the subprogram is taken to be the name of the file where the subprogram resides. Modularity is strictly adhered to in that only 1 subprogram may exist in any given file/subfile. There is no necessity for internal identification of the subprogram (e.g., there is (as would be the case in FORTRAN, BASIC, and ALGOL) no need for a SUBPROGRAM, SUBROUTINE, or PROCEDURE statement as the first line of the subprogram). Nor is there any need to terminate the various subprograms with a terminator statement (e.g., RETURN). In fact, the existence of EXIT (or END, STOP, HALT, or BYE) will cause the DATAPLOT run to be terminated—thus EXIT (and its synonyms) is usually reserved for the last line of the main program. All parameter, variable, and function names are global—names defined in the main program may be used and modified in subprograms (and vice versa). Subprograms may in turn call other subprograms. The maximum number of nested subprograms is 7 (including the main program). Subprograms may not be recursive—that is, a subprogram may not call itself.

The 4 most important ways in which subprogram usage arises is as follows—

1) Unconditional calls

For example—

1) Unconditional calls
   In this case, a subprogram is called once and only once. For example,
   
   CALL SIGMA.
   
   would cause subprogram SIGMA to be unconditionally called once and only once.

2) Conditional calls
   This calls a subprogram exactly once and only under certain conditions. For example,
   
   IF A > 4
   CALL SIGMA.
   END OF IF
   
   would cause subprogram SIGMA to be called only if the current value of the parameter A exceeds 4, but not to call SIGMA otherwise.

3) Unconditional loops
   In this case, a subprogram is called multiple times. For example,
   
   LOOP FOR B = 1 1 10
   CALL SIGMA.
   END OF LOOP
   
   would cause subprogram SIGMA to be unconditionally called 10 times (for B = 1, 2, ..., 9, 10).

4) Conditional loops
   In this case, a subprogram is called for a number of times that depends on calculations that occur within the subprogram being called. For example,
   
   LOOP FOR B = 1 1 C
   CALL SIGMA.
   END OF LOOP
   
   (where the parameter C is pre-defined but may also be redefined within the subprogram being called)
   would cause subprogram SIGMA to be called a variable number of times depending on how C is redefined within the loop.

As one would expect, execution of a pre-stored main program may be initiated in an identical fashion as one would execute a subprogram, namely via the CALL command. Thus, for example, if the analyst has signed onto DATAPLOT, and if a pre-stored main program resides in a file called A, then execution of the main program could start with the entry of the statement

CALL A.
Creating Subprograms

CREATE

The most common and most efficient method for creating DATAPLOT subprograms is via the local editor. That is, before the analyst enters DATAPLOT, he/she enters into a file/subfile with the local editor and creates a DATAPLOT program and any needed DATAPLOT subprograms. There is no substitute for an excellent local editor--the analyst is encouraged to use such an editor for creating DATAPLOT programs and subprograms.

One the other hand, it is at times convenient to be able to create a DATAPLOT subprogram "on the fly" while one is executing a DATAPLOT program. A prime example of this is in the construction of diagrammatic graphics in which one makes an interactive pass at forming the diagram, stores the commands that were used en route, and then executes these commands as a block in a semi-interactive mode.

If one is running DATAPLOT interactively, one may automatically store the entered interactive command lines into a file by use of the

CREATE

command. For example, to instruct DATAPLOT that all subsequently keyed-in commands should be in a file XYZ as they are being entered, the analyst would enter

CREATE XYZ.

To terminate the creation of the file XYZ, the analyst enters

END OF CREATE

or

END CREATE

To execute the contents of XYZ, one would enter

CALL XYZ.

Listing Subprograms

LIST

The LIST command may be used to list the contents of any file or subfile. Usually such a capability is taken advantage of when the file/subfile contains a subprogram, but it may also be used when the file contains data. The usual convention about periods following the file name is followed. Thus to list the contents of the file XYZ, one enters

LIST XYZ.

If the local computer system may have subfiles of files, then to list subfile ABC of file XYZ, one enters

LIST XYZ.ABC

The LIST command only lists files/subfiles--it does not execute the contents of such files/subfiles.
Miscellaneous

Restricting Plots and Analyses to Subsets 6-1
SUBSET, EXCEPT, and FOR

In-line Text Sub-commands 6-4
TITLE, ...LABEL, LEGEND, and TEXT

Defaults, Restrictions, and Settings 6-8

Dumping Status Information 6-11
STATUS

Dumping On-Line Documentation 6-15
HELP

Redimensioning Internal Storage 6-17
DIMENSION

Specifying the Terminal 6-18
TERMINAL

Specifying the Host 6-18
HOST

Specifying the Communications Link 6-19
COMMUNICATIONS LINK and BAUD RATE

Specifying Trigonometric Units 6-20
TRIGONOMETRIC UNITS

Echoing Commands 6-21
ECHO

Suppressing Printed Output 6-22
FEEDBACK and PRINTING

Diverting Graphics Output to Offline Devices 6-23
PENPLOTTER, ZETA, CALCOMP, and VERSATEC

Redefining I/O Units and Saving DATAPLOT Output 6-25
SET

Controlling The Terminal--Erasing, Hardcopying, etc. 6-26
PRE-ERASE, HARDCOPY, ERASE, and COPY

Activating Local Settings 6-28
IMPLEMENT

continued...
Entering Comment Lines
COMMENT and

Recommended Programming Practices

Interrupting, Saving, and Resuming A Run
SAVE and RESTORE

Starting Over
RESET

Communicating with the Host Operator
OPERATOR

Communicating with DATAPLOT Service Group
MESSAGE, NEWS, MAIL, and BUGS

DATAPLOT File Information for Implementation

Default Output Device Settings

Color Graphics
TERMINAL, COLOR, and PICTURE POINTS

Multiple Commands Per Line
TERMINATOR CHARACTER

Specifying Post-Plot Cursor Position and Size
CURSOR SIZE
Restricting Plots and Analyses to Subsets
SUBSET, EXCEPT, and FOR

The DATAPLOT keywords

    SUBSET
    EXCEPT
    FOR

may be appended to the end of any DATAPLOT graphics or analysis command. They allow the analyst to specify precisely the desired subset (any subset) which the plot/analysis should be restricted to. The rules for using SUBSET/EXCEPT/FOR are straightforward; we mention them briefly here and then demonstrate them by example. When the SUBSET qualification is used, it is always followed by a variable name. Any variable whatsoever may be used to define the subset—one which is used in the plot/analysis statement, such as

    PLOT Y X SUBSET X 100 TO 200
    PLOT Y X SUBSET Y 10 TO 30

or one not explicitly involved in the plot/analysis statement, as in

    PLOT Y X SUBSET LAB 4 TO 6
    PLOT Y X SUBSET OPERATOR 7

After the variable specification, one may have one of the following 4 syntaxes—

1) a single number;
2) a series of numbers
3) a number followed by TO followed by another number;
4) a relational operator (\(=\), \(<\), \(\leq\), \(\geq\), \(<\), or \(>\)) followed by a single number.

A single number indicates that the subset is restricted to that specific value and no others, as in

    PLOT Y X SUBSET LAB 4

which will restrict the plot to values in Y and X corresponding to values in the LAB variable which equal 4.

A series of numbers indicates that the subset is restricted to those specific values and no others, as in

    PLOT Y X SUBSET LAB 4 5 9

which will restrict the plot to values of Y and X corresponding to LAB values of 4, 5, and 9.

A number followed by TO followed by another number indicates that the subset is restricted to all values in the interval specified by the 2 numbers, inclusively, as in

    PLOT Y X SUBSET LAB 4 TO 9

which will restrict the plot to values of Y and X corresponding to all LAB values between 4 and 9, inclusively.

A relational operator followed by a single number indicates that the subset is restricted to that specific interval, as in

    PLOT Y X SUBSET LAB >= 3

which will restrict the plot to values in Y and X corresponding to values in the LAB variable greater than or equal to 3.

    PLOT Y X SUBSET LAB = 3

will restrict the plot to values in Y and X corresponding to values in the LAB variable equal to 3 (and so will have the same effect as PLOT Y X SUBSET LAB 3 as discussed earlier).

    PLOT Y X SUBSET LAB < 3

will restrict the plot to values in Y and X corresponding to values in the LAB variable not equal to 3.

SUBSET qualifications may be strung together in a more complicated fashion. Each individual SUBSET specification may have any of the above 4 valid syntaxes. The final subset will be the intersection of the individual subsets. Thus

    PLOT Y X SUBSET LAB 4 SUBSET OPERATOR 7

will restrict the plot to values in Y and X corresponding to values in the LAB variable equal to 4 and (simultaneously) values in the OPERATOR variable equal to 7. The use of many SUBSET specifications strung together allows the analyst to easily focus on any particular subset desired. Thus

    PLOT Y X SUBSET LAB 4 5 9 SUBSET OPERATOR 7 TO 14 SUBSET Y < 1000

will restrict the plot to values in Y and X corresponding to values in the LAB variable equal to 4, 5, or 9, and (simultaneously) values in the OPERATOR variable equal to 7 to 14, and (simultaneously) values in the Y variable less than 1000.
In addition to direct subset specification via \texttt{SUBSET}, there is also a need at times for the subset to be defined via negation. \texttt{DATAPLOT} handles such a need via the \texttt{EXCEPT} qualification. For simplicity and consistency, the rules for using \texttt{EXCEPT} are identical to the rules for using \texttt{SUBSET}; thus

\texttt{PLOT Y X EXCEPT LAB 3}

will restrict the plot to values in \texttt{Y} and \texttt{X} corresponding to values in the \texttt{LAB} variable not equal to 3.

\texttt{PLOT Y X EXCEPT LAB 3 4 7 EXCEPT OPERATOR 10 TO 15 EXCEPT Y > 1000}

will restrict the plot to values in \texttt{Y} and \texttt{X} corresponding to values in the \texttt{LAB} variable not equal to 3, 4, or 7, and (simultaneously) values in the \texttt{OPERATOR} variable not equal to 10 to 15, and (simultaneously) values in the \texttt{Y} variable not greater than 1000.

At times the analyst may need to define subsets by a combination of specifying what is in the subset and what is not in the subset. \texttt{DATAPLOT} handles this by allowing the analyst to freely mix \texttt{SUBSET} and \texttt{EXCEPT} qualifications; thus

\texttt{PLOT Y X SUBSET LAB 3 TO 20 EXCEPT LAB 4}

will restrict the plot to values in \texttt{Y} and \texttt{X} corresponding to values in the \texttt{LAB} variable from 3 to 20 with the exception of \texttt{LAB} 4.

\texttt{PLOT Y X SUBSET LAB 3 4 7 EXCEPT OPERATOR 10 TO 15 EXCEPT Y > 1000}

will restrict the plot to values in \texttt{Y} and \texttt{X} corresponding to values in the \texttt{LAB} variable equal to 3, 4, or 7, and (simultaneously) values in the \texttt{OPERATOR} variable not equal to 10 to 15, and (simultaneously) values in the \texttt{Y} variable not greater than 1000.

At times the analyst finds it convenient to specify a subset based not on the magnitudes and values of numbers in variables but rather on the element position in the variable. For example, suppose the analyst wishes to \texttt{PLOT Y versus X} but to do so for only every other element in the variables \texttt{Y} and \texttt{X}. How may this be done? \texttt{DATAPLOT} accounts for this case via the use of the \texttt{FOR} qualification. The \texttt{FOR} qualification (which may be appended to any \texttt{DATAPLOT} graphics/analysis command) has the general form

\texttt{FOR dummy index = start increment stop}

And so to plot \texttt{Y versus X} for elements 1, 3, 5, \ldots, 499, one enters

\texttt{PLOT Y X FOR I = 1 2 500}

To plot \texttt{Y versus X} for elements 10, 20, 30, \ldots, 490, 500, one enters

\texttt{PLOT Y X FOR I = 10 10 500}

In the above example, we have used the dummy index \texttt{I}; this is the most common choice for the \texttt{FOR} qualification, although any name could have been entered. The effect of using \texttt{I} is that this name will become a \texttt{DATAPLOT} parameter. At the end of the execution of the statement

\texttt{PLOT Y X FOR I = 10 10 500}

the parameter \texttt{I} will have the value 500 in it. One may not string successive \texttt{FOR} qualifications together at the end of a statement—only one \texttt{FOR} qualification is permitted. One may not mix \texttt{FOR} qualifications together with \texttt{SUBSET} and \texttt{EXCEPT} qualifications.

Note that in the above description of \texttt{SUBSET}, \texttt{EXCEPT}, and \texttt{FOR}, we referred in many places to the use of numbers, for example one form for the \texttt{SUBSET} qualification is to have a number followed by \texttt{TO} followed by another number, and the form for the \texttt{FOR} qualification is to have a start number, an increment number, and a stop number. Note that all statements involving numbers are equally valid for \texttt{DATAPLOT} parameters; that is, any \texttt{SUBSET/EXCEPT/FOR} qualification having numbers in it may equally well have pre-defined parameters in it. Although this feature is only rarely used, it does give the analyst additional generality if needed. For example,

\texttt{LET A = 3}
\texttt{LET B = 7}
\texttt{PLOT Y X SUBSET X A B}

is exactly equivalent to

\texttt{PLOT Y X SUBSET X 3 7}

and

\texttt{LET START = 5}
\texttt{LET INCREMENT = 5}
\texttt{LET STOP = 300}
\texttt{PLOT Y X FOR I = START INCREMENT STOP}

is exactly equivalent to

\texttt{PLOT Y X FOR I = 5 5 300}
As mentioned earlier, one may use any variable as the subset variable; occasions arise in which it makes good sense to use one of the variables in the plot (e.g., X or Y) as a subset variable, thus

\[ \text{PLOT Y X SUBSET X > 100} \]

will restrict the plot to values in Y and X corresponding to values in the X variable greater than 100; while

\[ \text{PLOT Y X SUBSET Y < 1000} \]

will restrict the plot to values in Y and X corresponding to values in the Y variable less than 1000 (which, for example, may be of use in regenerating a plot with outliers removed).

In addition to the variables that the analyst has created, the subset specification variable may also be the automatically-generated variables PRED (= predicted values) and RES (= residuals) which result from any DATA PLOT fitting, spline fitting, smoothing, Anova, Median polish, etc. operation. Note how the use of the RES variable as the subset variable may be a valuable tool for the data analyst in carrying out residual analysis. For example, suppose that the analyst has carried out a fit, say, via

\[ \text{FIT Y = A*B*EXP(-C*X)} \]

and suppose the analyst has generated the usual superimposed plot of raw data and predicted values via

\[ \text{CHARACTERS X BLANK}\]
\[ \text{LINES BLANK SOLID}\]
\[ \text{PLOT Y PRED VERSUS X}\]

and suppose further that the analyst has generated the usual residual plot via

\[ \text{PLOT RES X}\]

Now suppose that the analyst is interested in generating a listing of X, Y, PRED, and RES values for all the data; this is easily done via

\[ \text{PRINT X Y PRED RES SUBSET RES > 10}\]

A final note to this section reinforces the fact that SUBSET/EXCEPT/FOR qualifications may be used with any graphics/analysis command. Of particular importance are the various forms of the PLOT command. We have already seen how

\[ \text{PLOT Y X}\]

may be restricted to become

\[ \text{PLOT Y X SUBSET LAB 4 TO 8}\]

Note also that the powerful 3-argument form

\[ \text{PLOT Y X TAG}\]

(which plots Y versus X but generates an individual trace for each distinct value of the TAG variable) may be also be restricted as in

\[ \text{PLOT Y X TAG SUBSET LAB 4 TO 8}\]

Further, the same is true for the VERSUS form--

\[ \text{PLOT Y1 Y2 Y3 Y4 Y5 Y6 VERSUS X}\]

(which is handy for generating multi-trace plots) may be similarly restricted, as in

\[ \text{PLOT Y1 Y2 Y3 Y4 Y5 Y6 VERSUS X SUBSET LAB 4 TO 8}\]

The DATA PLOT convention for subset extraction is a flexible, powerful, time-saving tool for the scientist/researcher. Its use allows the analyst to easily probe into interesting subsets of the data--so as to detect anomalous behavior in such subsets and so as to determine if conclusions and models which hold for the full set are equally valid for the individual subsets.

PRINT X Y PRED RES

Now finally suppose that the analyst noted on the residual plot that there was a patch of residuals which were larger than others (this was the region of poorest fit) and suppose the analyst was interested in printing out those X, Y, PRED, and RES values only for this region of poor fit. Such a listing would be of use in exactly specifying the X values which had the poorest fit. For sake of definiteness, suppose the analyst was interested in such a printing for all values corresponding to residuals in excess of 10 units. How may such a printing be generated? Very easily, namely by entering

PRINT X Y PRED RES SUBSET RES > 10
In-line Text Sub-commands
TITLE, . . . LABEL, LEGEND, and TEXT

The entries in this section are not commands in themselves; rather, they are sub-commands which may appear in-line in the text-oriented DATAPLOT commands--TITLE, . . . LABEL, LEGEND, and TEXT. The sub-commands are "flagged" for DATAPLOT by the existence of open and closed parentheses following each sub-command, as in

TEXT ALPH(BETA(GAMMA)

which would write out the Greek characters alpha, beta, and gamma. The most important of these sub-commands are LC, UC, SUB, SUP, INTE, VALU, and the Greek alphabet.

The sub-commands in this section are not applicable for the Tektronix font, but are applicable for all of the 7 "fancy script" fonts. Thus before use of any of the in-line text sub-commands, the analyst should inform DATAPLOT (via the FONT command) that the default Tektronix font is being overridden and another font is being used, as in

FONT TRIPLEX
TEXT ALPH(BETA(GAMMA

The VALU sub-command is especially important. It allows the analyst to substitute parameter values in titles, labels, legends, and text strings. For example,

LET LAB = 7
TEXT ANALYSIS FOR LABORATORY VALU(LAB

would result in the following line appearing--

ANALYSIS FOR LABORATORY 7

The VALU() sub-command will print out the value of the parameter which immediately follows the () of VALU(). In the example above, this parameter was LAB; this parameter had a value of 7.
### General Operations

<table>
<thead>
<tr>
<th>Sub-command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC</td>
<td>upper case</td>
<td>TEXT UC(ABC)</td>
</tr>
<tr>
<td>CAP</td>
<td>upper case</td>
<td>TEXT CAP(ABC)</td>
</tr>
<tr>
<td>CAPS</td>
<td>upper case</td>
<td>TEXT CAPS(ABC)</td>
</tr>
<tr>
<td>LC</td>
<td>lower case</td>
<td>TEXT LC(ABC)</td>
</tr>
<tr>
<td>SUB</td>
<td>subscript</td>
<td>TEXT ABCSUBDEF(DEF)</td>
</tr>
<tr>
<td>UNSB</td>
<td>un-subscript</td>
<td>TEXT ABCSUBDEFUNSUBDEF(DEF)</td>
</tr>
<tr>
<td>SUP</td>
<td>superscript</td>
<td>TEXT ABCSUPDEF(DEF)</td>
</tr>
<tr>
<td>UNSP</td>
<td>un-superscript</td>
<td>TEXT ABCSUPDEFUNSUPDEF(DEF)</td>
</tr>
<tr>
<td>VALU</td>
<td>substitute value of parameter</td>
<td>TEXT VALU(A)</td>
</tr>
</tbody>
</table>

### Greek Characters

<table>
<thead>
<tr>
<th>Sub-command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPH</td>
<td>alpha</td>
<td>TEXT ALPH( )</td>
</tr>
<tr>
<td>BETA</td>
<td>beta</td>
<td>TEXT BETA( )</td>
</tr>
<tr>
<td>GAMM</td>
<td>gamma</td>
<td>TEXT GAMM( )</td>
</tr>
<tr>
<td>DELT</td>
<td>delta</td>
<td>TEXT DELT( )</td>
</tr>
<tr>
<td>EPSI</td>
<td>epsilon</td>
<td>TEXT EPSI( )</td>
</tr>
<tr>
<td>ZETA</td>
<td>zeta</td>
<td>TEXT ZETA( )</td>
</tr>
<tr>
<td>ETA</td>
<td>eta</td>
<td>TEXT ETA( )</td>
</tr>
<tr>
<td>THET</td>
<td>theta</td>
<td>TEXT THET( )</td>
</tr>
<tr>
<td>IOTA</td>
<td>iota</td>
<td>TEXT IOTA( )</td>
</tr>
<tr>
<td>KAPP</td>
<td>kappa</td>
<td>TEXT KAPP( )</td>
</tr>
<tr>
<td>LAMB</td>
<td>lambda</td>
<td>TEXT LAMB( )</td>
</tr>
<tr>
<td>MU</td>
<td>mu</td>
<td>TEXT MU( )</td>
</tr>
<tr>
<td>NU</td>
<td>nu</td>
<td>TEXT NU( )</td>
</tr>
<tr>
<td>XI</td>
<td>xi</td>
<td>TEXT XI( )</td>
</tr>
<tr>
<td>OMIC</td>
<td>omicon</td>
<td>TEXT OMIC( )</td>
</tr>
<tr>
<td>PI</td>
<td>pi</td>
<td>TEXT PI( )</td>
</tr>
<tr>
<td>RHO</td>
<td>rho</td>
<td>TEXT RHO( )</td>
</tr>
<tr>
<td>SIGM</td>
<td>sigma</td>
<td>TEXT SIGM( )</td>
</tr>
<tr>
<td>TAU</td>
<td>tau</td>
<td>TEXT TAU( )</td>
</tr>
<tr>
<td>UPSI</td>
<td>upsilon</td>
<td>TEXT UPSI( )</td>
</tr>
<tr>
<td>PHI</td>
<td>phi</td>
<td>TEXT PHI( )</td>
</tr>
<tr>
<td>CHI</td>
<td>chi</td>
<td>TEXT CHI( )</td>
</tr>
<tr>
<td>PSI</td>
<td>psi</td>
<td>TEXT PSI( )</td>
</tr>
<tr>
<td>OMEG</td>
<td>omega</td>
<td>TEXT OMEG( )</td>
</tr>
</tbody>
</table>
### Mathematical Symbols

<table>
<thead>
<tr>
<th>Sub-command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PART</td>
<td>partial derivative</td>
<td>TEXT PART()</td>
</tr>
<tr>
<td>INTI</td>
<td>integral</td>
<td>TEXT INTI()</td>
</tr>
<tr>
<td>CINT</td>
<td>circular integral</td>
<td>TEXT CINT()</td>
</tr>
<tr>
<td>SUMM</td>
<td>summation</td>
<td>TEXT SUMM()</td>
</tr>
<tr>
<td>PROD</td>
<td>product</td>
<td>TEXT PROD()</td>
</tr>
<tr>
<td>INFI</td>
<td>infinity</td>
<td>TEXT INFI()</td>
</tr>
<tr>
<td>+=</td>
<td>+ or -</td>
<td>TEXT ++()</td>
</tr>
<tr>
<td>-=</td>
<td>- or +</td>
<td>TEXT --()</td>
</tr>
<tr>
<td>TIME</td>
<td>times</td>
<td>TEXT TIME()</td>
</tr>
<tr>
<td>DOTP</td>
<td>dot product</td>
<td>TEXT DOTP()</td>
</tr>
<tr>
<td>DEL</td>
<td>vector product</td>
<td>TEXT DEL()</td>
</tr>
<tr>
<td>DIVI</td>
<td>division</td>
<td>TEXT DIVI()</td>
</tr>
<tr>
<td>LT</td>
<td>less than</td>
<td>TEXT LT()</td>
</tr>
<tr>
<td>GT</td>
<td>greater than</td>
<td>TEXT GT()</td>
</tr>
<tr>
<td>LTEQ</td>
<td>less than or equal to</td>
<td>TEXT LTEQ()</td>
</tr>
<tr>
<td>GTEQ</td>
<td>greater than or equal to</td>
<td>TEXT GTEQ()</td>
</tr>
<tr>
<td>NOT=</td>
<td>not equal</td>
<td>TEXT NOT=()</td>
</tr>
<tr>
<td>APPR</td>
<td>approximately equal to</td>
<td>TEXT APPR()</td>
</tr>
<tr>
<td>EQUI</td>
<td>equivalence</td>
<td>TEXT EQUI()</td>
</tr>
<tr>
<td>VARI</td>
<td>varies</td>
<td>TEXT VARI()</td>
</tr>
<tr>
<td>TILD</td>
<td>tilde</td>
<td>TEXT TILD()</td>
</tr>
<tr>
<td>CARA</td>
<td>carat</td>
<td>TEXT CARA()</td>
</tr>
<tr>
<td>PRIM</td>
<td>prime</td>
<td>TEXT PRIM()</td>
</tr>
<tr>
<td>RADI</td>
<td>radical</td>
<td>TEXT RADI()</td>
</tr>
<tr>
<td>LRAD</td>
<td>large radicall</td>
<td>TEXT LRAD()</td>
</tr>
<tr>
<td>SUBS</td>
<td>subset</td>
<td>TEXT SUBS()</td>
</tr>
<tr>
<td>SUPB</td>
<td>superset</td>
<td>TEXT SUPB()</td>
</tr>
<tr>
<td>UNIO</td>
<td>union</td>
<td>TEXT UNIO()</td>
</tr>
<tr>
<td>INTR</td>
<td>intersection</td>
<td>TEXT INTR()</td>
</tr>
<tr>
<td>ELEM</td>
<td>is an element of</td>
<td>TEXT ELEM()</td>
</tr>
<tr>
<td>THEX</td>
<td>there exists</td>
<td>TEXT THEX()</td>
</tr>
<tr>
<td>THFO</td>
<td>therefore</td>
<td>TEXT THFO()</td>
</tr>
</tbody>
</table>
### Miscellaneous Symbols

<table>
<thead>
<tr>
<th>Sub-command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAC</td>
<td>Space</td>
<td>TEXT SPAC()</td>
</tr>
<tr>
<td>SP</td>
<td>Space</td>
<td>TEXT SP()</td>
</tr>
<tr>
<td>LAPO</td>
<td>left apostrophe</td>
<td>TEXT LAPO()</td>
</tr>
<tr>
<td>RAPO</td>
<td>right apostrophe</td>
<td>TEXT RAPO()</td>
</tr>
<tr>
<td>LBRA</td>
<td>left bracket</td>
<td>TEXT LBRA()</td>
</tr>
<tr>
<td>RBRA</td>
<td>right bracket</td>
<td>TEXT RBRA()</td>
</tr>
<tr>
<td>LCBR</td>
<td>left curly bracket</td>
<td>TEXT LCBR()</td>
</tr>
<tr>
<td>RCBR</td>
<td>right curly bracket</td>
<td>TEXT LCBR()</td>
</tr>
<tr>
<td>LELB</td>
<td>left elbow</td>
<td>TEXT LELB()</td>
</tr>
<tr>
<td>RELB</td>
<td>right elbow</td>
<td>TEXT RELB()</td>
</tr>
<tr>
<td>RACC</td>
<td>right accent</td>
<td>TEXT RACC()</td>
</tr>
<tr>
<td>LACC</td>
<td>left accent</td>
<td>TEXT LACC()</td>
</tr>
<tr>
<td>BREV</td>
<td>breve</td>
<td>TEXT BREV()</td>
</tr>
<tr>
<td>ROQU</td>
<td>right quote</td>
<td>TEXT ROQU()</td>
</tr>
<tr>
<td>LOQU</td>
<td>left quote</td>
<td>TEXT LOQU()</td>
</tr>
<tr>
<td>NASP</td>
<td>nasp</td>
<td>TEXT NASP()</td>
</tr>
<tr>
<td>IASP</td>
<td>inverted nasp</td>
<td>TEXT IASP()</td>
</tr>
<tr>
<td>RARR</td>
<td>right arrow</td>
<td>TEXT RARR()</td>
</tr>
<tr>
<td>LARR</td>
<td>left arrow</td>
<td>TEXT LARR()</td>
</tr>
<tr>
<td>UARR</td>
<td>up arrow</td>
<td>TEXT UARR()</td>
</tr>
<tr>
<td>DARR</td>
<td>down arrow</td>
<td>TEXT DARR()</td>
</tr>
<tr>
<td>PARA</td>
<td>paragraph</td>
<td>TEXT PARA()</td>
</tr>
<tr>
<td>DAGG</td>
<td>dagger</td>
<td>TEXT DAGG()</td>
</tr>
<tr>
<td>DDAG</td>
<td>double dagger</td>
<td>TEXT DDAG()</td>
</tr>
<tr>
<td>VBAR</td>
<td>vertical bar</td>
<td>TEXT VBAR()</td>
</tr>
<tr>
<td>DVBA</td>
<td>double vertical bar</td>
<td>TEXT DVBA()</td>
</tr>
<tr>
<td>LVBA</td>
<td>long vertical bar</td>
<td>TEXT LVBA()</td>
</tr>
<tr>
<td>HBAR</td>
<td>horizontal bar</td>
<td>TEXT HBAR()</td>
</tr>
<tr>
<td>LHBA</td>
<td>long horizontal bar</td>
<td>TEXT LHBA()</td>
</tr>
<tr>
<td>BAR</td>
<td>bar</td>
<td>TEXT BAR()</td>
</tr>
</tbody>
</table>
Defaults, Restrictions, and Settings

Plot Control—

<table>
<thead>
<tr>
<th>Operation</th>
<th>Default</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic pre-erase (before plots)</td>
<td>on</td>
<td>PPR-ERASE</td>
</tr>
<tr>
<td>Automatic bell (before plots)</td>
<td>on</td>
<td>BELL</td>
</tr>
<tr>
<td>Automatic hardcopy (after plots)</td>
<td>off</td>
<td>HARDCOPY</td>
</tr>
<tr>
<td>Automatic sequence numbers on plots</td>
<td>off</td>
<td>SEQUENCE</td>
</tr>
<tr>
<td>Plot characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title and labels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of characters in title</td>
<td>100</td>
<td>TITLE and ...LABEL</td>
</tr>
<tr>
<td>Maximum number of characters (each label)</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Plot scale</td>
<td>linear</td>
<td></td>
</tr>
<tr>
<td>Grid lines</td>
<td>off</td>
<td></td>
</tr>
<tr>
<td>Axis limits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame corner coordinates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tic marks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tic mark labels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tic mark position</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of legends</td>
<td>100</td>
<td>LEGEND ...</td>
</tr>
<tr>
<td>Maximum number of characters (all legends)</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>Arrows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of arrows</td>
<td>100</td>
<td>ARROW ...</td>
</tr>
<tr>
<td>Line segments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of line segments</td>
<td>100</td>
<td>SEGMENT ...</td>
</tr>
<tr>
<td>Boxes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum number of boxes</td>
<td>100</td>
<td>BOX ...</td>
</tr>
<tr>
<td>Font (for all titles, labels, legends, etc.)</td>
<td>Tektronix</td>
<td>PONT</td>
</tr>
<tr>
<td>Site for plot characters</td>
<td>3 (= 3k screen height)</td>
<td>CHARACTER SIZE</td>
</tr>
<tr>
<td>Site for plot lines</td>
<td>3 (= 3k screen height)</td>
<td>LINE SIZE</td>
</tr>
<tr>
<td>Site for plot tics</td>
<td>3 (= 3k screen height)</td>
<td>TIC SIZE</td>
</tr>
<tr>
<td>Site for plot titles</td>
<td>3 (= 3k screen height)</td>
<td>TITLE SIZE</td>
</tr>
<tr>
<td>Site for plot labels</td>
<td>3 (= 3k screen height)</td>
<td>...LABEL SIZE</td>
</tr>
<tr>
<td>Site for plot legends</td>
<td>3 (= 3k screen height)</td>
<td>LEGEND ...</td>
</tr>
<tr>
<td>Color for plot characters</td>
<td>red</td>
<td>CHARACTER COLOR</td>
</tr>
<tr>
<td>Color for plot lines</td>
<td>red</td>
<td>LINE COLOR</td>
</tr>
<tr>
<td>Color for plot tics</td>
<td>red</td>
<td>TIC COLOR</td>
</tr>
<tr>
<td>Color for plot tic labels</td>
<td>red</td>
<td>TIC LABEL COLOR</td>
</tr>
<tr>
<td>Color for plot tics</td>
<td>red</td>
<td></td>
</tr>
<tr>
<td>Color for plot labels</td>
<td>red</td>
<td>LABEL COLOR</td>
</tr>
<tr>
<td>Color for plot arrows</td>
<td>red</td>
<td></td>
</tr>
<tr>
<td>Color for plot segments</td>
<td>red</td>
<td>LEGEND ... COLOR</td>
</tr>
<tr>
<td>Color for plot boxes</td>
<td>red</td>
<td>SEGMENT ... COLOR</td>
</tr>
<tr>
<td>Color for background (= inside frame line)</td>
<td>blue</td>
<td>BOX ... COLOR</td>
</tr>
<tr>
<td>Color for margin (= outside frame line)</td>
<td>blue</td>
<td>BACKGROUND COLOR</td>
</tr>
<tr>
<td>3d origin</td>
<td></td>
<td>ORIGIN COORDINATES</td>
</tr>
<tr>
<td>3d frame lines</td>
<td>on</td>
<td>FRANE</td>
</tr>
<tr>
<td>3d eye coordinates</td>
<td>(xe,ye,ze)</td>
<td>EYE COORDINATES</td>
</tr>
<tr>
<td>3d pedestal color for 3d pedestal</td>
<td>red</td>
<td>PEDESTAL COLOR</td>
</tr>
<tr>
<td>Maximum number of plot points per plot</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>Maximum number of superimposed plots</td>
<td>no limit</td>
<td></td>
</tr>
</tbody>
</table>
Diagrammatic Graphics---

Operation

Default: Tektronix
Command: FONT

- Case: upper case
- Justification: left-justified
- Height: 3 (≈ 3% screen height)
- Width: 2 (≈ 2% screen width)
- Color of Background: blue
- Color of text and figures: red

Output Devices---

Operation

Default: Tektronix 4014
Command: TERMINAL MANUFACTURER

- Terminal: 1024
- Horizontal picture points: 781
- Vertical Picture points: terminal
- Output device #1: off (= non-color)
- Terminal color status: on
- Terminal continuity status: 5
- Maximum number of output devices: on
- Status of device 1: off
data devices 2 to 5: off
data status of penplotter: off

Data---

Operation

Default: 10
Command: DIMENSION

- Maximum number of variables: 10
- Maximum number of observations per variable: 1000
- Maximum total internal data storage: 10000
- Automatically provided variables: "PRE and RES"
- Maximum number of variable/parameter/function names (total): 200
- Maximum number of characters (total for all functions): 1000
- Maximum number of characters in FIT model: 200
- Maximum number of parameter names in FIT model: 100
- Maximum number independent variables in FIT: 5

Support---

Operation

Default: radians
Command: ANGLE UNITS

- Trigonometric units: radians
- Baud rate: 1200
- Test angle: 0 degrees (= horiz.)
- Echo: off
- Bell (at time of plot): on
- Host: Univac 1100 Exec 8
- Host cut-off standard deviation for FIT: 0.000005
- Cut-off number of iterations for FIT: 50
- Default fit criterion power for FIT: 2 (≈ least squares)
- Degree of internal calculations: single precision
- Feedback messages from commands: on
- Print output from analysis commands: on
- Number of lines to be skipped on READ/SEQREAD: 0
- Row limits for READ/SEQREAD: 1 to infinity
- Column limits for READ/SEQREAD: 1 to 132
- Command terminator character: ; or carriage return
General--

<table>
<thead>
<tr>
<th>Operation</th>
<th>Default</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of characters per terminal line</td>
<td>80</td>
<td>---</td>
</tr>
<tr>
<td>Maximum number of lines per command</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>Continuation indicator at end of line</td>
<td>...</td>
<td>---</td>
</tr>
<tr>
<td>Command terminator indicator</td>
<td>; or carriage return</td>
<td>TERMINATOR CHARACTER</td>
</tr>
</tbody>
</table>

Machine Constants (Example 1--Univac 1100/82)--

<table>
<thead>
<tr>
<th>Operation</th>
<th>Default</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard input unit</td>
<td>5</td>
<td>SET</td>
</tr>
<tr>
<td>Standard output unit</td>
<td>6</td>
<td>SET</td>
</tr>
<tr>
<td>Number of bits per character</td>
<td>9</td>
<td>---</td>
</tr>
<tr>
<td>Number of characters per word</td>
<td>4</td>
<td>---</td>
</tr>
<tr>
<td>Number of bits per word</td>
<td>36</td>
<td>---</td>
</tr>
<tr>
<td>Machine infinity</td>
<td>0.1701412**39</td>
<td>SET</td>
</tr>
</tbody>
</table>

Machine Constants (Example 2--Vax 11/780)--

<table>
<thead>
<tr>
<th>Operation</th>
<th>Default</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard input unit</td>
<td>5</td>
<td>SET</td>
</tr>
<tr>
<td>Standard output unit</td>
<td>6</td>
<td>SET</td>
</tr>
<tr>
<td>Number of bits per character</td>
<td>8</td>
<td>---</td>
</tr>
<tr>
<td>Number of characters per word</td>
<td>4</td>
<td>---</td>
</tr>
<tr>
<td>Number of bits per word</td>
<td>32</td>
<td>---</td>
</tr>
<tr>
<td>Machine infinity</td>
<td>zzz</td>
<td>SET</td>
</tr>
</tbody>
</table>

File/Subfile Nomenclature--

<table>
<thead>
<tr>
<th>Operation</th>
<th>Default</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualifier-file separator (Example 1--Univac 1100/82)</td>
<td>*</td>
<td>---</td>
</tr>
<tr>
<td>File-subfile separator (Example 1--Univac 1100/82)</td>
<td>.</td>
<td>---</td>
</tr>
<tr>
<td>Qualifier-file separator (Example 2--Vax 11/780)</td>
<td>}</td>
<td>---</td>
</tr>
<tr>
<td>File-subfile separator (Example 2--Vax 11/780)</td>
<td>}</td>
<td>---</td>
</tr>
</tbody>
</table>

File/Subfile Structure--

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Unit Number</th>
<th>Name</th>
<th>File or Subfile</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>21</td>
<td>DPMFESP</td>
<td>file only</td>
<td>MESSAGE</td>
</tr>
<tr>
<td>News</td>
<td>22</td>
<td>DPMWMP</td>
<td>file only</td>
<td>MEMZ</td>
</tr>
<tr>
<td>Help</td>
<td>23</td>
<td>DPMHELP</td>
<td>file only</td>
<td>MAIL</td>
</tr>
<tr>
<td>Bug</td>
<td>24</td>
<td>DPMBUG</td>
<td>file only</td>
<td>HELP</td>
</tr>
<tr>
<td>Query</td>
<td>26</td>
<td>DPMQUERY</td>
<td>file only</td>
<td>QUERY</td>
</tr>
<tr>
<td>Log</td>
<td>27</td>
<td>DPMLOG</td>
<td>off</td>
<td>---</td>
</tr>
<tr>
<td>Read</td>
<td>31</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>READ and SERIAL READ</td>
</tr>
<tr>
<td>Write</td>
<td>32</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>WRITE</td>
</tr>
<tr>
<td>Macro</td>
<td>33</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>CREATE and CALL</td>
</tr>
<tr>
<td>Save</td>
<td>34</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>SAVE and RESTORE</td>
</tr>
<tr>
<td>Scratch</td>
<td>41</td>
<td>DPPSRCH</td>
<td>file only</td>
<td>---</td>
</tr>
<tr>
<td>Data</td>
<td>42</td>
<td>DPPDATA</td>
<td>file only</td>
<td>---</td>
</tr>
<tr>
<td>Pipe</td>
<td>43</td>
<td>DPPLIP</td>
<td>file only</td>
<td>---</td>
</tr>
<tr>
<td>Plot 2</td>
<td>44</td>
<td>DPPPLIP</td>
<td>file only</td>
<td>---</td>
</tr>
</tbody>
</table>
Dumping Status Information

STATUS

The status of the most important internal DATAPLOT settings may be printed out via the STATUS command. This command has no arguments, one simply enters

```
STATUS
```

The following is an example of the type of output which would result from entering the STATUS command---
**************************************************************
* STORAGE INFORMATION *
**************************************************************
* NUMBER OF ... * USED * UNUSED * MAXIMUM *
**************************************************************
* VARIABLES (COLUMNS)* 6 * 4 * 10 *
* OBS PER VARIABLE * 98 * 902 * 1000 *
* OBS (TOTAL) * 6000 * 4000 * 10000 *
* FUNC CHAR (TOTAL) * 13 * 987 * 1000 *
* VAR/PAR/FUNC NAMES * 23 * 177 * 200 *
**************************************************************

<table>
<thead>
<tr>
<th>SET</th>
<th>PLOT</th>
<th>PLOT</th>
<th>PLOT</th>
<th>PLOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX</td>
<td>TYPE</td>
<td>LINE</td>
<td>INDEX</td>
<td>LINE</td>
</tr>
<tr>
<td>1</td>
<td>BLAN</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>2</td>
<td>SOLI</td>
<td>RED</td>
<td>BLAN</td>
<td>RED</td>
</tr>
<tr>
<td>3</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>4</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>5</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>6</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>7</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>8</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>9</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
<tr>
<td>10</td>
<td>SOLI</td>
<td>RED</td>
<td>X</td>
<td>RED</td>
</tr>
</tbody>
</table>

X-AXIS PLOT MINIMUM = -.17014118+039
X-AXIS PLOT MAXIMUM = .17014118+039
Y-AXIS PLOT MINIMUM = -.17014118+039
Y-AXIS PLOT MAXIMUM = .17014118+039

VARIABLE 1 (WITH 98 ELEMENTS) HAS THE FOLLOWING NAMES: TIME X
VARIABLE 2 (WITH 98 ELEMENTS) HAS THE FOLLOWING NAMES: STRAIN Y
VARIABLE 3 (WITH 98 ELEMENTS) HAS THE FOLLOWING NAMES: STRESS
VARIABLE 4 (WITH 3 ELEMENTS) HAS THE FOLLOWING NAMES: STRESS2
VARIABLE 5 (WITH 1 ELEMENTS) HAS THE FOLLOWING NAMES: S2
VARIABLE 6 (WITH 1 ELEMENTS) HAS THE FOLLOWING NAMES: C2

PARAMETER PI HAS THE FOLLOWING VALUE: .3141593+001
PARAMETER INFINITY HAS THE FOLLOWING VALUE: .1701412+039
PARAMETER NSTRESS2 HAS THE FOLLOWING VALUE: .3000000+001
PARAMETER K HAS THE FOLLOWING VALUE: .1000000+001
PARAMETER ST HAS THE FOLLOWING VALUE: .3000000+003
PARAMETER A HAS THE FOLLOWING VALUE: .1591170+000
PARAMETER B HAS THE FOLLOWING VALUE: -.6243119+001
PARAMETER C HAS THE FOLLOWING VALUE: .1450866+002
PARAMETER REPSD HAS THE FOLLOWING VALUE: .0000000
PARAMETER REPDP HAS THE FOLLOWING VALUE: .0000000
PARAMETER RESSD HAS THE FOLLOWING VALUE: .2687022+002
PARAMETER RESDP HAS THE FOLLOWING VALUE: .3400000+002

FUNCTION G = --A+B*EXP(-C*X)
The STATUS output has 3 sections:

1) storage;
2) plot control;
3) parameter/variable/function.

The storage section prints out information concerning the internal DATAPLOT arrays which have been used by the analyst. For the above example, the storage section indicates the following:

1) The maximum number of variables allowed is 10 (else use the DIMENSION command); 6 variables have been defined (TIME, STRAIN, STRESS, STRESS2, S2, and C2), and 10-6 = 4 variables remain to be defined;

2) The maximum number of observations per variable allowable is 1000 (else use the DIMENSION command); The longest (out of the 6 variables) has 98 observations; 1000-98 = 902 observations remain to be defined.

3) The total internal storage capacity is 10000 observations. The used storage is 6 variables x 1000 observations per variable = 6000 observations; the available storage available is 10000-6000 = 4000 observations.

4) The sum of the number of all characters in all DATAPLOT functions may not exceed 1000. For this run, 13 characters have been used (in the function C below); hence 1000-13 = 987 are available for further use by other functions.

5) The sum of the number of parameter names + variable names + function names may not exceed 200. For this run, 23 names have been defined:

1) 6 user-defined variable names (TIME, STRAIN, STRESS, STRESS2, S2, and C2);

2) 2 additional variable names (X and Y) that have been equated to TIME and STRAIN, respectively by use of the NAME command;

3) 2 DATAPLOT-provided variables PRED (for predicted values) and RES (for residuals); these variables are not listed in the STATUS output because their storage is separate from that of the 10 variables and 10000 total observation limit of the analyst. PRED and RES are used anytime a PRE-FIT, FIT, EXACT RATIONAL FIT, SPLINE FIT, SMOOTH, ANOVA, or MEDIAN POLISH command is executed.

4) 12 parameter names (INFINITY, PI, NSTRESS2, K, ST, A, B, C, REPSD, REPDF, RESSD, RESDF); of these 12 parameter names, PI and INFINITY are pre-defined by DATAPLOT and contain the value of the mathematical constant pi and the machine single-precision maximum. The 4 parameters REPSD, REPDF, RESSD, and RESDF are also internally-computed by DATAPLOT whenever any PRE-FIT, FIT, EXACT RATIONAL FIT, SPLINE FIT, SMOOTH, ANOVA, or MEDIAN POLISH is done— they contain the values

1) REPSD = replication standard deviation;
2) REPDF = replication degrees of freedom;
3) RESSD = residual standard deviation;
4) RESDF = residual degrees of freedom;

The remaining parameters (NSTRESS2, K, ST, A, B, and C) are user-defined parameters.

5) 1 function name (G).

The plot control section indicates information relevant to the generation of plots. For the example above, the STATUS output indicates:

1) The line types are all solid, except for the first line type which is blank (that is, no line.

2) All line colors are red (this is relevant only if one has a color terminal);

3) The character types are all X's, except for the second character which is blank (that is, no character);

4) All character colors are red (this is relevant only if one has a color terminal);

5) All character sizes are 1.0 (that is, 1% of total screen height);

6) the x-axis minimum is set at -infinity and the x-axis maximum is set at +infinity; these infinity values instruct DATAPLOT to have the plotted horizontal axis limits be neat and float with the data;

7) the y-axis minimum is set at -infinity and the y-axis maximum is set at +infinity; these infinity values instruct DATAPLOT to have the plotted vertical axis limits be neat and float with the data;
The parameter/variable/function section provides details about used parameters, variables, and functions. For the example above--

1) variable 1 has 98 observations and may be referred to by either the name TIME or X;
2) variable 2 has 98 observations and may be referred to by either the name STRAIN or Y;
3) variable 3 has 98 observations and has the name STRESS;
4) variable 4 has 3 observations and has the name STRESS2;
5) variable 5 has 1 observation and has the name S2;
6) variable 6 has 1 observation and has the name C2;
7) parameter PI has the value 3.141593;
8) parameter INFINITY has the value .1701412**39;
9) parameter NSTRESS2 has the value 3;
10) parameter K has the value 1;
11) parameter ST has the value 3;
12) parameter A has the value .1591170;
13) parameter B has the value -0.06243119;
14) parameter C has the value 0.001450866;
15) parameter RESSD has the value 0;
16) parameter RESDF has the value 0;
17) parameter REPSD has the value 0.002687022; thus the residual standard deviation from the last fit-type operation was 0.002687022;
18) parameter REPDF has the value 34; thus the residual degrees of freedom from the last fit-type operation was 34;
19) function G has the character string A*B*EXP(-C*X) .
Dumping On-Line Documentation
HELP

To print on-line documentation from within a DATAPLOT run, one uses the HELP command. The DATAPLOT HELP command may be followed by any one of the following 3 types of entries--

1) no entry at all, as in HELP

2) a DATAPLOT Command Category entry, as in HELP GRAPHICS

3) A DATAPLOT command name, as in HELP PLOT

These 3 cases will now be discussed individually.

If the analyst enters the HELP command alone with no trailing entry, as in HELP

then the following will be printed at the terminal--

DATAPLOT IS AN INTERACTIVE GRAPHICS LANGUAGE.
THERE ARE 7 DATAPLOT COMMAND CATEGORIES--

GRAPHICS
ANALYSIS
DIAGRAMMATIC GRAPHICS

PLOT CONTROL
SUPPORT
OUTPUT DEVICE

KEYWORDS

FOR A LISTING OF COMMANDS WITHIN A CATEGORY,
ENTER HELP FOLLOWED BY THE CATEGORY NAME,
AS IN--HELP GRAPHICS HELP ANALYSIS HELP PLOT CONTROL

The above is the list of the 7 general DATAPLOT command categories--

1) Graphics
2) Analysis
3) Diagrammatic Graphics
4) Plot Control
5) Output Device
6) Support
7) Keywords

If one enters HELP followed by any of the 7 command category names, as in

HELP GRAPHICS HELP ANALYSIS HELP DIAGRAMMATIC GRAPHICS

then a list of all DATAPLOT command names within that category will be printed. For example, if one enters HELP GRAPHICS then the following output will appear at the terminal--

THE DATAPLOT GRAPHICS COMMAND CATEGORY CONSISTS OF THE FOLLOWING 18 COMMANDS--

BOX PLOT
COMPLEX DEMODULATION ... PLOT
... CONTROL CHART
... CORRELATION PLOT
... FREQUENCY PLOT
... HISTOGRAM
... HOMOSCEDASTICITY PLOT
LAG ... PLOT
... NORMALITY PLOT
PERCENT POINT PLOT
... PERIODIC GRAPHS
PIE CHART
PLOT
... PPCC PLOT
... PROBABILITY PLOT
RUN SEQUENCE PLOT
... SPECTRAL PLOT
3-D PLOT

FOR ADDITIONAL INFORMATION ABOUT A PARTICULAR COMMAND,
ENTER HELP FOLLOWED BY THE COMMAND NAME, AS IN--HELP BOX PLOT
HELP COMPLEX DEMODULATION ... PLOT
HELP ... PROBABILITY PLOT

If one enters HELP followed by a command name, as in

HELP PLOT
HELP FIT
HELP LET

then summary, usage, default, and examples information will be printed out about that particular command. For example, if one entered HELP PLOT then the following output would result--

PLOT

PURPOSE--GENERATE A PLOT

EFFECT --A PLOT IS GENERATED
VISIBLE EFFECT--A PLOT IS GENERATED
DEFAULT --NOT APPLY
EXAMPLES --PLOT Y X

plot SIN(X) FOR X = 0 .1 5
PLOT Y PRED VERSUS X

Such on-line documentation is a useful feature which saves the analyst from a time-consuming look-up in a hardcopy manual. The HELP output
is not meant to be all-inclusive and comprehensive; rather, it is meant to summarize salient points pertaining to commands.

At times it is convenient to divert the output from the HELP command so that it does not come back to the terminal, but rather is directed out to some user-defined mass-storage file. For example, suppose the analyst wanted the HELP output for the following commands--

```
PLOT
TITLE
LABEL
CHARACTERS
LINES
PONT
HARDCOPY
```

to be printed out to a file (pre-existing) which has a numeric designation of, say, 12. This may be done by use of the HELP and SET commands in the following fashion--

```
SET IPR 12
HELP PLOT
HELP TITLE
HELP LABEL
HELP CHARACTERS
HELP LINES
HELP PONT
HELP HARDCOPY
SET IPR 6
```

The SET IPR 12 command redefines the standard output unit from default (usually 6) to 12. All subsequent output will thus be diverted to the file with numeric designation 12—no output will appear at unit 6 (the terminal). The HELP PLOT command will produce output to unit 12, as will the succeeding HELP commands. The last command (SET IPR 6) will redefine the standard output unit to be 6 again, and so control will return to the terminal. The net result is that the file with numeric designation 12 will contain all of the line images which DATAPLOT would normally send to the terminal. After exiting out of DATAPLOT, the analyst may then peruse this file at will and/or send it off to the printer via a local system print/list utility command.
Redimensioning Internal Storage

DIMENSION

Upon signing onto DATAPLOT, a message will be generated which states the limits (for your particular computer) of internal data storage in DATAPLOT. The usual limits are

1) maximum number of variables = 10
2) maximum number of observations per variable = 1,000
3) maximum total internal data storage = 10 x 1,000 = 10,000

The maximum total internal data storage size is typically fixed (here = 10,000) and unchangeable; however, the analyst does have some flexibility in terms of changing the number of variables (from 10 to something larger than 10) and in terms of changing the maximum number of observations per variable (from 1,000 to something smaller than 1,000). In any event, if the analyst changes either the number of variables, or the maximum number of observations per variable, the overriding constraint will exist that their product will = the total internal data storage size (= 10,000).

The DIMENSION command allows the analyst to change the number of variables, or the maximum number of observations per variable, (or both). To change the upper limit on the number of variables from 10 to, say, 200, enter

DIMENSION 200 VARIABLES

or (equivalently)

DIMENSION 200 COLUMNS

A message will result stating that the

1) maximum number of variables = 200
2) maximum number of observations per variable = 50
3) maximum total internal data storage = 200 x 50 = 10,000

Note that in addition to the number of variables being changed from 10 to 200, the maximum number of observations per variable has also been automatically changed from 1,000 to 10,000/200 = 50.

Similarly, to change the maximum number of observations per variable from 1000 to, say, 250, enter

DIMENSION 250 OBSERVATIONS

or (equivalently)

DIMENSION 250 ROWS

A message will result stating that the

1) maximum number of variables = 40
2) maximum number of observations per variable = 250
3) maximum total internal data storage = 40 x 250 = 10,000

Note that in addition to the maximum number of observations per variable being changed from 1,000 to 250, the number of variables has also been automatically changed from 10 to 10,000/250 = 40.

Current (1/83) configurations of DATAPLOT do not permit the DIMENSION statement to change the number of variables to more than 200, nor to change the maximum number of observations per variable to more than 1,000. However, with hardware advances in memory sizes occurring so rapidly, it is clear that future versions of DATAPLOT will have both of these limits raised considerably.
Specifying the Terminal

TERMINAL

The TERMINAL command allows the analyst to specify the particular terminal being used. The default terminal is a Tektronix 4014 or equivalent. The TERMINAL command should specify both the manufacturer and the model number, as in

TERMINAL TEKTRONIX 4027
TERMINAL TEKTRONIX 4010
TERMINAL RAMTEK 6011

When the TERMINAL command is processed by DATAPLOT, it will automatically adjust internal settings relating to

1) whether or not the terminal is continuous or discrete;

2) whether or not the terminal is color or non-color;

3) the number of screen picture points (horizontal and vertical) for the terminal.

Thus the use of the TERMINAL command obviates the need for the CONTINUOUS/DISCRETE command, the COLOR command, and the PICTURE POINTS command.

Specifying the Host

HOST

The HOST command is seldom used. This command specifies the current host. Usually such a host-specification is automatic upon signing onto DATAPLOT, hence its rare usage. The form for the HOST command includes both the computer manufacturer and model number, as in

HOST VAX 11/780
HOST UNIVAC 1100/82
HOST IBM 4341
HOST IBM PC
Specifying the Communications Link

COMMUNICATIONS LINK and BAUD RATE

The commands in this section are rarely used. The motivation for their inclusion stems more from completeness and the ability to address potential future problems than actual past problems.

When multiple terminals are running DATAPLOT from a common host, the terminals may be running over communication links that have differing baud rates. This can at times lead to timing and buffering problems in the terminal hardware as it interacts with the underlying continuous graphics software. Typical indications that such problems are occurring are that graphs will be generated which are clipped (incomplete)—parts of the plot which should be drawn (usually frame lines) remain undrawn.

A common local cause of this is that plot directives are being processed by the terminal while a screen erase is still in progress. It takes a typical display terminal about 1 full second to erase the screen. In that second, the terminal should be buffering commands and/or the graphics software should be sending null character strings. In order for the proper number of nulls to be sent, it is necessary that the underlying graphics software be correctly informed of the communication baud rate. If the actual baud rate is faster than the default baud rate, then an insufficient number of nulls will be generated in order for the screen erase to be completed. In such case, the terminal hardware buffering must take over. If the terminal hardware has such buffering, then the plot will be properly formed; however, if the terminal hardware does not have buffering, then the plot will be clipped.

The default internal DATAPLOT baud rate setting is 1200. Usually this baud rate does not have to be reset (well over 95% of all DATAPLOT runs at the NBS implementation are done under the default baud rate—even though the terminals are running anywhere from 300 baud to 19200 baud). However, if such clipping problems do arise, then the baud rate setting may be the problem. To change this rate, use the BAUD RATE command, as in

```
BAUD RATE 9600
```

The type of communication link may also affect the (continuous) graphics information being transmitted. For example, the phone line rarely "eats" or "adds" control characters to a transmission, but some specialized communication networks do "eat/add" certain character sequences. It is common for graphics software directives to involve "escape" sequences; it is also common for some intelligent communication networks to have command directives which themselves involve "escape" sequences. Thus when the graphics software sends such sequences, the network itself intercepts (and acts on) them, and the terminal never receives them. The net effect is missing parts on the graph, glitches on the graph, or improperly-formed components on the graph.

If the analyst is having problems in the formation of continuous graphs, and if the problem is traceable to the communications link, then it will be necessary to inform DATAPLOT of the type of communications link so that local systems personnel may adjust the internal code depending on the type of link. The default link is the standard phone lines (as would be used, e.g., with an acoustic coupler). To change the link specification, use the COMMUNICATIONS LINK command, as in

```
COMMUNICATIONS LINK NETWORK
COMMUNICATIONS LINK ARPANET
COMMUNICATIONS LINK ETHERNET
```

Specifying the baud rate is done occasionally; specifying the communications link is done even less frequently.
Specifying Trigonometric Units

TRIGONOMETRIC UNITS

DATAPLOT has an extensive set of trigonometric and inverse trigonometric library functions. Trigonometric calculations may be carried out in radians, degrees, or grads (as is common in Europe). The default unit is radians. If it is desired to have trigonometric calculations in degrees, then enter

```
TRIGONOMETRIC UNITS DEGREES
```
or simply

```
DEGREES
```

All subsequent trig calculations will thus be carried out in degrees. To change the angle units to grads, enter

```
TRIGONOMETRIC UNITS GRADS
```
or

```
GRADS
```

To change the angle units back to radians, enter

```
TRIGONOMETRIC UNITS RADIANS
```
or

```
RADIANS
```

Another instance in which the angle units come into play is in the scribing of text strings on a plot or diagram at an angle other than horizontal. For example, to write the text string abc starting in the middle of the screen in triplex font, one would enter

```
FONT TRIPLEX
MOVE 50 50
TEXT ABC
```

How would one write the same text string at a 45 degree angle? To do so, one uses both the TRIGONOMETRIC UNITS and the ANGLE command. The TRIGONOMETRIC UNITS command specifies the desired units (radians, degrees, or grads); the ANGLE command specifies the desired angle of the text string (in whatever angle units that were chosen). For example, to write out the text string abc in triplex font at an angle of 45 degrees, the easiest way is

```
TRIGONOMETRIC UNITS DEGREES
ANGLE 45
FONT TRIPLEX
MOVE 50 50
TEXT ABC
```
Echoing Commands

ECHO

When running pre-stored DATAPLOT programs, it is sometimes convenient to have the commands echoed back as they are being executed. To do this, one enters

    ECHO

or

    ECHO ON

down somewhere up near the beginning of the code.

The echoed commands will appear in an easily-noticeable box as they are being executed, the output from the execution of each command will appear immediately below the box. Thus the effect of

    LINES SOLID DOT DASH
    CHARACTERS A B C

is the usual

    LINE   1 HAS JUST BEEN SET TO SOLI
    LINE   2 HAS JUST BEEN SET TO DOT
    LINE   3 HAS JUST BEEN SET TO DASH

    CHARACTER  1 HAS JUST BEEN SET TO A
    CHARACTER  2 HAS JUST BEEN SET TO B
    CHARACTER  3 HAS JUST BEEN SET TO C

whereas the effect of

    ECHO ON
    LINES SOLID DOT DASH
    CHARACTERS A B C

is

    *******************************************
    ** LINES SOLID DOT DASH **
    *******************************************

    LINE   1 HAS JUST BEEN SET TO SOLI
    LINE   2 HAS JUST BEEN SET TO DOT
    LINE   3 HAS JUST BEEN SET TO DASH

    *******************************************
    ** CHARACTERS A B C **
    *******************************************

    CHARACTER  1 HAS JUST BEEN SET TO A
    CHARACTER  2 HAS JUST BEEN SET TO B
    CHARACTER  3 HAS JUST BEEN SET TO C
Suppressing Printed Output

FEEDBACK and PRINTING

All of DATAPLOT's commands result in some form of printed output. A broad partitioning of all DATAPLOT commands could consist of

1) Graphics commands;
2) Analysis commands;
3) All other commands (= secondary commands).

The commands in the GRAPHICS category (PLOT, HISTOGRAM, SPECTRUM, 3D- PLOT, NORMAL PROBABILITY PLOT, etc.) all result in immediate graphics output which is typically important in terms of carrying out the analysis. Such output is usually not suppressed.

The commands in the Analysis category (FIT, SPLINE FIT, ANOVA, SMOOTH, SUMMARY, LET, etc.) all have immediately-printed output which is of interest in terms of carrying out the goals of an analysis. With the possible exception of output from the LET command, the suppression of such output is rarely done.

The remainder of DATAPLOT's commands (such as CHARACTERS, LINES, READ, DIMENSION, DELETE, LEGEND, TITLE, LABEL, etc.) are not really end objectives in an analysis, but rather are necessary intermediate steps in carrying out an analysis precisely to one's specifications. These secondary commands also generate printed feedback information which assures the analyst that the command that the host received was identical to the command that was entered through the keyboard. This feedback information is comforting and reassuring to the analyst in that the specified commands are resulting in the desired action. Such feedback rarely "gets in the way" and is the default when one signs onto DATAPLOT. Occasions do arise, however, when it would be desirable to be able to suppress such feedback. To do so, enter

FEEDBACK OFF

When the FEEDBACK OFF command has been received, all feedback information from all secondary commands, (and all usual output from the LET command), will be suppressed. The most common occurrence of this is in the generation of diagrammatic graphics (diagrams, schematics, word charts, etc.) where one wishes to suppress feedback messages so that they will not clutter up the screen as the diagram is being formed. For example, the execution of the following pre-stored DATAPLOT program

FONT TRIPLEX ITALIC
HEIGHT 5
WIDTH 3
JUSTIFICATION CENTER
.
ERASE
MOVE 50 80
TEST GRAPHICS
HEIGHT 2
WIDTH 1

will (upon execution) result in the desired word chart being formed, but will also have the word chart "ruined" because the automatic feedback information from the

HEIGHT 2
WIDTH 1

calls will also appear superimposed on the word chart.

To circumvent this problem, the above code should be amended to

FONT TRIPLEX ITALIC
HEIGHT 5
WIDTH 3
JUSTIFICATION CENTER
.
FEEDBACK OFF
ERASE
MOVE 50 80
TEST GRAPHICS
HEIGHT 2
WIDTH 1
MOVE 50 70
TEXT ANALYSIS GRAPHICS
MOVE 50 60
TEXT PRESENTATION GRAPHICS
COPY

The desired word chart--uncluttered by feedback information--will be formed.

To counteract the FEEDBACK OFF command, enter

FEEDBACK ON

or

FEEDBACK

We have seen how the FEEDBACK OFF command will suppress output from all secondary commands and from the LET command. Is there any way to suppress output from any of the Analysis category commands other than LET (e.g., FIT, SPLINE FIT, ANOVA, SMOOTH, SUMMARY, etc.)? Although such suppression is rarely done, it may be carried out by entering

PRINTING OFF

To counteract this command, the analyst enters

PRINTING ON

or

PRINTING
Diverting Graphics Output to Offline Devices  
**PENPLOTTER, ZETA, CALCOMP, and VERSATEC**

When generating graphics on a terminal, it is sometimes convenient to have such graphics not only appear on the screen, but also to route them to an alternate graphics device. If the analyst enters the

```
PENPLOTTER
```

or

```
PENPLOTTER ON
```

command, then all subsequent plots will also appear on the local penplotter (e.g., Tektronix 4652). This penplotter must be connected in serial between the host and the terminal. When using a penplotter, the analyst must have the proper baud rate (typically quite slow due to the mechanical nature of the penplotter), and also must have the proper communication switch settings on the penplotter (which depends on the nature of the communications link and the host).

To turn the penplotter off, the analyst enters

```
PENPLOTTER OFF
```

For other offline devices (such as the Zeta, Calcomp, and Versatec plotters), the analyst should enter the commands

```
ZETA
CALCOMP
VERSATEC
```

or

```
ZETA ON
CALCOMP ON
VERSATEC ON
```

respectively. The net effect of any of these commands is the same in DATAPLOT, namely, to capture whatever graphics information is going to the screen, allow it to proceed to the screen, but also write that same information out to a plot file. The net result is that when the DATAPLOT session is finished, the analyst will have in this plot file a full record of all the character strings that (when sent to a Tektronix terminal) would cause the plots to reform. In fact, such plots could be reformatted by simply going into the plot file with the local editor and printing out some number of lines with an editor print. If the terminal being used is a Tektronix, then the plot will reform; if the terminal being used is a Tektronix-incompatible or an alphanumeric (non-graphics) terminal, then lines of unintelligible gibberish will appear.

The name of the plot file which DATAPLOT writes such information out to is DPPLIF. DATAPLOT automatically creates this file when these alternate plot devices are specified. The FORTRAN unit number that DATAPLOT assigns to this file is 43. The file is temporary (it will remain after the analyst signs off of DATAPLOT, but will disappear when the analyst logs off the computer). When the analyst signs off of DATAPLOT, a message will be generated which reminds the analyst what file name (DPPLIF) and file number (43) were given to the plot file, and will also give the analyst instructions on how to post-process the file so that the Tektronix characters may be converted to Zeta, Calcomp, Versatec, etc. plots.

To consider a specific example, note the following program:

```
ZETA
PLOT X**2 FOR X = 1 1 10
EXIT
```

When the

```
ZETA
```

command is encountered, the following feedback message will appear:

```
DEVICE  2  INFORMATION
        POWER --ON
        MANUFACTURER --ZETA
        CONTINUITY --ON
        COLOR --OFF
        PICTURE POINTS-- 1000 1000
```

Note that "device 2" is any alternate graphics output device, while "device 1" is the terminal itself.

When the

```
PLOT X**2 FOR X = 1 1 10
```

is encountered, the screen will erase, and the quadratic plot will appear. While this is happening, the same character strings which caused the plot to form on the screen are being copied into file DPPLIF (= file 43).

When the analyst enters

```
EXIT
```

DATAPLOT will remind the analyst that an alternate plotting device has been specified, and will provide the following feedback messages:
NOTE--THE ALTERNATE PLOT FILE
IS CLOSED.
THE NAME OF THE FILE IS DPPLIF.
TO EXAMINE THE FILE, USE ANY EDITOR,
(EDM, ED, CTX, ETC.), AS IN
EDM, DPPLIF.
P *
TO SEND THE FILE TO THE ZETA, ENTER
NBS*PLIBS.AUNIXPLLOT,I DPPLIF.,.,ZETA,FMT/TEK
WHERE THE I OPTION ALLOWS UP TO 3 LINES OF
OPERATOR DIRECTIVES (FEN, POINT, COLOR, PAPER)

THIS IS AN EXIT FROM DATAPLOT.

As many plots as desired may be written out to the
alternate plot file. Caution should be exercised
in multiple opening and closing of the alternate
plot file during the same DATAPLOT session--this
should not be done because subsequent openings of
the file will overwrite the existing plots on the
file. For example, consider the following 10-line
DATAPLOT program--

ZETA ON
PLOT X FOR X = 1 1 10
PLOT X**2 FOR X = 1 1 10
ZETA OFF

ZETA ON
PLOT X**3 FOR X = 1 1 10
PLOT X**4 FOR X = 1 1 10
ZETA OFF

The net effect is that the linear and quadratic
plot residing on file DPPLIF will be overwritten
by the cubic and quartic plots. If all 4 plots
were desired to be on the file, then the program
should be amended to

ZETA ON
PLOT X FOR X = 1 1 10
PLOT X**2 FOR X = 1 1 10
PLOT X**3 FOR X = 1 1 10
PLOT X**4 FOR X = 1 1 10
ZETA OFF

Note that if the analyst enters

ZETA OFF
CALCOMP OFF
VERSATEC OFF

then DATAPLOT will also close the alternate plot
file, and will provide the same feedback messages.
As in the example above, if the analyst chooses to
leave the file open, then upon exit from DATAPLOT,
it will automatically be closed.

Note that the contents of the exit message are
clearly site-dependent. The above message was
captured from output generated on the NBS Univac
1100/82. EDM, ED, CTX are local Univac
editors; and

NBS*PLIBS.AUNIXPLLOT,I DPPLIF.,.,ZETA,FMT/TEK

is a local Univac post-processor (completely
independent and separate from DATAPLOT) which
allows any file (in particular, file DPPLIF)
containing Tektronix graphics strings to be
converted and send to the NBS Zeta plotter.

Under current (1/83) configurations, DATAPLOT will
not write directly to the Zeta, Calcomp, or
Versatec); rather, DATAPLOT writes a Tektronix
graphics string file (the same file regardless of
whether the ultimate device is the Zeta, Calcomp,
Versatec, or some other device), and then the
analyst must apply a locally-provided,
system-dependent post-processor to this file to
send its contents to its ultimate destination.

DATAPLOT does not provide the Tektronix-to-...
post-processor; it must be locally provided. In
this regard, 2 items are noted--

1) future versions of DATAPLOT will have
direct writing to alternate graphics
devices;

2) it has been reported (though not yet
substantiated) that commercially-available
post-processors are being developed
for a variety of computers and a variety of
output devices.
Redefining I/O Units and Saving DATAPLOT Output

SET

For most computers,

1) the default numeric designation
   for the input unit is 5;

2) the default numeric designation
   for the input unit is 6.

The underlying code in which DATAPLOT is written
is FORTRAN (ANSI 77). The underlying FORTRAN
variable names which specify the above 2
default I/O unit numbers are

1) IRD for the input unit;
2) IPR for the output unit.

Whenever DATAPLOT is reading commands from the
terminal, it is reading from unit 5; whenever
DATAPLOT is sending information back to the
terminal (both non-graphics and graphics
information), it is writing to unit 6.

We have seen in other sections how the use of the
ZETA, CALCUMP, and VERSATEC commands allows us to
capture graphics information and copy it out to a
file (file DPPLLIP = file 43). This leads us to
the question of how one might capture all
information (graphical and non-graphical) and
place it in a file (perhaps for future perusal, or
perhaps to increase execution speed). Although
this is rarely done, it is nevertheless
possible—via the SET command. The SET command
allows us to change a variety of underlying
FORTRAN settings; we focus here on altering the
settings of the IRD and IPR variables. The form
for the SET command is

    SET   FORTRAN variable name   Desired setting

Thus to change the standard input unit from 5 to
11, one could enter

    SET IRD 11

and to change the standard output unit from 6 to
21, one could enter

    SET IPR 21

Note that if the analyst changes the standard
output unit from the terminal to some file, then
all output will go to that file (and no output
will go to the terminal). Such a file may, of
course, be subsequently perused and edited. In
practice, the “average” analyst will rarely (if
ever) find the need to change IRD and IPR
settings.
Controlling The Terminal—Erasing, Hardcopying, etc.

PRE-ERASE, HARDCOPY, ERASE, and COPY

The following 4 commands are useful to control elementary erasing and copying operations at the terminal:

PRE-ERASE
HARDCOPY
ERASE
COPY

The PRE-ERASE command, as in

PRE-ERASE ON

or simply

PRE-ERASE

instructs DATAPLOT that every subsequent plot command (that is, every command from the Graphics category--PLOT, BOX PLOT, 3D- PLOT, CONTROL CHART, LAG ... PLOT, HISTOGRAM, SPECTRUM, etc.) should automatically erase its screen prior to the plotting operation. Since graphics is a core DATAPLOT operation, this automatic pre-erasure is reserved for all of the commands in the DATAPLOT Graphics category. Only commands in this category have this automatic pre-erasure option. Note that the default setting for plot pre-erasure is ON, and so the analyst only rarely needs to explicitly enter the PRE-ERASE command. To turn the pre-erasure off, the analyst enters

PRE-ERASE OFF

This capability is only occasionally used. Its main use arises in the superposition of plots atop one another, as in

XLIMITS 10 20
YLIMITS 100 150
PRE-ERASE OFF
ERASE
PLOT Y1 X
PLOT Y2 X
PLOT Y3 X
etc.

This program will

1) Set the x-axis limits to 10 and 20 for all subsequent plots (so that all superimposed plots have the identical scale);

2) Set the y-axis limits to 100 and 150 for all subsequent plots (for the same reason);

3) Turn off the automatic plot pre-erasure (so that the second, third, etc. plots do not erase the screen when they are being superimposed);

4) Manually erase the screen prior to the first plot;

5) Generate one plot;

6) Superimpose a second plot;

7) Superimpose a third plot;

8) etc.

The superposition of plots in this fashion allows us to generate plots with an indefinitely large number of plot points—thus circumventing any DATAPLOT restrictions that might exist as to maximum number of points per plot.

The automatic, "hands-off" copying of all generated plots to the local hardcopy units is a convenient feature. To activate this feature, the analyst enters

HARDCOPY ON

or simply

HARDCOPY

Every plot command subsequently entered will result in not only the plot being generated on the terminal screen, but also the plot being automatically copied to the local hardcopy unit as soon as the plot finishes constructing itself on the screen.

To specify multiple hardcopies per plot, the analyst appends the desired number to the end of the command, as in

HARDCOPY ON 3

or

HARDCOPY 3

which will make 3 local hardcopies of every plot which appears on the screen.

To terminate the automatic copying of plots, the analyst enters

HARDCOPY OFF

The default setting of the plot hardcopy switch is off.

The PRE-ERASE and HARDCOPY commands are passive commands—they set internal switches which are later scanned when plot commands are entered. The last 2 commands (ERASE and COPY) are active commands in the sense that when these commands are encountered, something happens immediately—the terminal erases its screen, or the terminal makes a local hardcopy of the current contents of the screen.
To erase the screen under program control, use the ERASE command. To copy the contents of the screen under program control, use the COPY command. Anytime DATAPLOT encounters the ERASE and COPY commands, the screen will be immediately erased and copied, respectively. In addition, if the analyst is running interactively from a terminal, the terminal itself usually has erase and copy keys which may be manually depressed. The ERASE and COPY commands are, however, more convenient to use because they may be entered in pre-stored DATAPLOT programs. A typical example of when the ERASE and COPY commands are useful is

```
ERASE
FIT Y = some model
COPY
```

This program will

1) erase the screen;

2) generate the usual FIT output;

3) copy the FIT output to the hardcopy unit.

Another example is in diagrammatic graphics, as in

```
FONT TRIPLEX ITALIC
JUSTIFICATION CENTER
.
ERASE
MOVE 50 50
TEXT ABC
COPY
```

This program will

1) Set the font style to triplex italic;

2) Set the justification to center justified;

3) erase the screen;

4) move to the middle of the screen;

5) write out the text string ABC;

6) copy the screen to the hardcopy unit.
Activating Local Settings

IMPLEMENT

The IMPLEMENT command allows the analyst to automatically activate local settings, conventions, and preferences. The form of the command is

IMPLEMENT number

as in

IMPLEMENT 1

AND

IMPLEMENT 2

Different installations have different features—depending on local preferences. For example, at NBS, we have only 2 local settings which may be accessed via the IMPLEMENT command. If the analyst here enters

IMPLEMENT 1

then this would be equivalent to the following 5 lines of code—

X2TIC MARKS OFF
X2TIC MARK LABELS OFF
Y2TIC MARKS OFF
Y2TIC MARK LABELS OFF
Y2LABEL

and so the IMPLEMENT 1 command would have the net effect of causing all subsequent plots to be produced with neither tic marks nor tic mark labels on the upper and right frame lines, and with no vertical label on the right side of the frame. If the NBS analyst entered

IMPLEMENT 2

then this would be equivalent to entering

FRAME CORNER COORDINATES 15 20 75 85

that is, this would cause all subsequent plots to have a square shape rather than the default rectangular shape. Such a square shape is, for example, useful in Youden plots (an interlaboratory graphical data analysis technique). The IMPLEMENT 1 and IMPLEMENT 2 commands at other installations will typically cause these same 2 features to be activated. In addition, the local DATAPLOT service group may have added other convenient features which could be accessed via IMPLEMENT 3, IMPLEMENT 4, etc. Check the local DATAPLOT service group for such augmentations.
Entering Comment Lines

COMMENT and .

It is good programming practice to enter non-executable comments in stored DATAPLOT programs. There are 2 ways to enter a comment line--

1) via the COMMENT command;
2) via the . command.

The 2 commands (COMMENT and .) are equivalent. Any DATAPLOT statement that starts with COMMENT or . will become a null (or "no-op") statement--such statements are non-executable. The following programs are identical--

```
COMMENT CALIBRATION ANALYSIS
COMMENT JANUARY 1983
READ XYZ, X Y
TITLE CALIBRATION
PLOT Y X
LET A = 10
LET B = 5
LET C = .5
FIT Y = A+B*EXP(-C*X)
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
PLOT RES X
```

and

```
.. CALIBRATION ANALYSIS
.. JANUARY 1983
READ XYZ, X Y
TITLE CALIBRATION
PLOT Y X
LET A = 10
LET B = 5
LET C = .5
FIT Y = A+B*EXP(-C*X)
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
PLOT RES X
```

Note that the . command is often useful for visually segregating chunks of DATAPLOT code. Note the difference between the above code and

```
.. CALIBRATION ANALYSIS
.. JANUARY 1983
.. READ XYZ, X Y
.. TITLE CALIBRATION
.. PLOT Y X
.. LET A = 10
.. LET B = 5
.. LET C = .5
.. FIT Y = A+B*EXP(-C*X)
.. CHARACTERS X BLANK
.. LINES BLANK SOLID
.. PLOT Y PRED VERSUS X
.. PLOT RES X
```

The latter is easier to read because it has been separated into functionally-modularized "chunks" of code. The programs would execute identically. Note that the COMMENT command (and the . command in particular) must be followed by at least 1 space so as to separate it from any remaining text on the line; thus

```
.. CALIBRATION ANALYSIS
```

is correct, but

```
.. CALIBRATION ANALYSIS
```

is incorrect.

If COMMENT or . is not followed by any text, then the trailing space is, of course, not needed--COMMENT and . may be succeeded immediately by a carriage return.

The . command is also a convenient way of "commenting out" code (in stored DATAPLOT programs) that we have already executed in previous runs, that we no longer wish to execute, but still wish to retain in the code as a memory aid. For example, suppose we executed the above code which fitted the model \( Y = A+B\times\exp(-C\times X) \), but now wish to execute the same code with the model \( Y = A+B\times X^2 \). We could delete (via the local editor) the original FIT statement from the stored code, but alternatively we may wish to simply "comment out" the original FIT statement as a reminder of models that were already tried; as in

```
.. CALIBRATION ANALYSIS
.. JANUARY 1983
.. READ XYZ, X Y
.. TITLE CALIBRATION
.. PLOT Y X
.. LET A = 10
.. LET B = 5
.. LET C = .5
.. FIT Y = A+B*EXP(-C*X)
FIT Y = A+B*X**2
.. CHARACTERS X BLANK
.. LINES BLANK SOLID
.. PLOT Y PRED VERSUS X
.. PLOT RES X
```
Recommended Programming Practices

When constructing (via the editor) DATAPLOT programs (that are to be stored in a file/subfile) for later execution, it is recommended that liberal use of comments (especially at the beginning of the code) be used. The COMMENT command and the . command assist in this regard. Liberal use of the . command to visually separate chunks of DATAPLOT code also make for easier reading and understanding of the code after-the-fact. Contrast, for example, the following 2 DATAPLOT programs—both of which are identical in execution—

```
READ XYZ. X Y
TITLE CALIBRATION ANALYSIS
YLABEL PRESSURE
XLABEL TIME
PLOT Y X
LET A = 1
LET B = 10
LET C = .5
FIT Y = A+B*SQRT(C+X)
CHARACTERS LINES BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
YLABEL RESIDUALS
PLOT RES X
```

and

```
. CALIBRATION ANALYSIS
. SPECIMEN 247
. JANUARY 1983
.
READ XYZ. X Y
.
TITLE CALIBRATION ANALYSIS
YLABEL PRESSURE
XLABEL TIME
PLOT Y X
.
LET A = 1
LET B = 10
LET C = .5
FIT Y = A+B*SQRT(C+X)
.
CHARACTERS LINES BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
.
YLABEL RESIDUALS
PLOT RES X
```

A common set of commands which often appears at the beginning of many DATAPLOT runs (whether running interactively, or running stored DATAPLOT programs) is

```
ECHO ON
HARDCOPY ON
SEQUENCE ON
```

The ECHO ON command tells DATAPLOT to echo back all entered commands (in a noticeable box) as the commands are being executed. If one is running a stored DATAPLOT program in a rapid-fire fashion, then the echoing helps to keep track of exactly what command is being executed—because one sees not only the produced output (which may "fly by" and fill the screen quickly), but also one sees the specific input command that produced the output. This is useful for monitoring the execution of "canned" programs.

The HARDCOPY ON command causes every plot which appears on the terminal screen to be automatically copied to the local hardcopy unit. This "hands off" feature is very helpful in producing a record of how the analysis proceeded.

The SEQUENCE ON command causes an automatically-incremented sequence number to appear (inconspicuously) in the upper right corner of plots as they are being generated. When used in conjunction with the HARDCOPY ON command, it gives the analyst an ordered set of graphics output which provides a valuable record of how a data analysis proceeded.

For doing analysis graphics (= graphics for extracting structure from data), the analyst should use the default Tektronix font for the titles, labels, legends, etc. This font is not as "pretty" as the fancier-script fonts, but is quite adequate, very readable, and is quickly generated. For such graphical data analysis, the default Tektronix is more than adequate and will not slow down the analysis. One should wait until the end of the analysis—when structure has been extracted and when conclusions have been formulated—before one uses the fancier fonts. As the analysis graphics stage is being finished, and the presentation graphics (= graphics for communicating conclusions via journals, seminars, etc.) stage is being started, then the analyst should consider using the *fancier fonts. To switch to such fonts is a one-command operation in DATAPLOT, namely the FONT command, as in

```
TITLE LINE STANDARDS ANALYSIS
YLABEL WIDTH
XLABEL POSITION
FIT Y = A+B*X
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
.
FONT TRIPLEX ITALIC
PLOT Y PRED VERSUS X
```
The titles and labels from the first plot will be default Tektronix font—they will be generated very fast; the titles and labels from the second plot will be in triplex italic font—they will take more time to generate, but will be of much higher quality.

Of all the plot control commands available in DATAPLOT, the 2 commands which the analyst will find him/herself using most heavily are the CHARACTERS and LINES commands. The CHARACTERS command specifies the desired plot characters to appear at the plot points of a trace; the LINES command specifies the desired line type which connects the various plot points of the trace. The default character type is blank, and the default line type is solid. These defaults allow for the rapid continuous-trace plotting of data sets and functions. Display terminals typically take longer to generate traces which have non-blank plot characters than those with blank plot characters—such terminals draw solid traces very rapidly, but draw individual characters much more slowly. If the trace has 100 points or so, then the time difference will be negligible; if the trace has 1000 points, then the time difference will be noticeable. As the electronics of display terminals improve, this time-of-plot consideration should disappear.

A particular case of the CHARACTERS and LINES command which arises time and time again whenever a fitting operation has been carried out is

```
CHARACTERS X BLANK
LINES BLANK SOLID
```

The usual context in which this appears is

```
FIT Y = some model
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
PLOT RES X
```

The above code will

1) carry out a fit according to some model, and automatically store the resulting predicted values (in a variable named PRED) and residuals (in a variable named RES).

2) specify that the characters on succeeding plots have X's for the first trace, and blanks for the second trace (if there is a second trace);

3) specify that the lines on succeeding plots have no connecting lines for the first trace, and solid connecting lines for the second trace (if there is a second trace);

4) plot the raw data \( Y \) versus \( X \) and superimpose the predicted values \( (PRED) \) from the fit versus \( X \). Note that the \( Y \) versus \( X \) trace will have X's as plot characters and no connecting lines, and the \( PRED \) versus \( X \) trace will have blank plot characters and a solid connecting line. The usual form for the superimposed plot of raw data and predicted values is to have the raw data appear as discrete X's and the predicted values to appear as a solid trace. The above CHARACTERS and LINES commands in conjunction with the PLOT Y PRED VERSUS X command allow the analyst to do this easily.

5) plot the residuals \( (RES) \) from the fit versus \( X \). Note that since only 1 trace is being outputted here, then the plot will follow the first specification of the above CHARACTERS and LINES commands, namely to have X's as plot characters, and to have no connecting lines.
Interrupting, Saving, and Resuming A Run
SAVE and RESTORE

Suppose one is in the middle of an interactive session with DATAPLOT, and one is forced to interrupt a run (to go to a seminar, say). In such case it would be extremely convenient to be able to save all of DATAPLOT's internal settings so as to be able to continue the run at some later time. To do so, one uses the SAVE and RESTORE commands. The form for the SAVE command is

SAVE XYZ.

where XYZ is a pre-existing user file to which all of the internal DATAPLOT settings will be dumped. Upon signing onto DATAPLOT at a later time, the analyst would enter

RESTORE XYZ.

and then proceed with the analysis at precisely the point of interruption. As with all DATAPLOT commands which involve file/subfile usage, the file name should be followed by a period in the command statement; thus

SAVE XYZ.

is correct, but

SAVE XYZ

is incorrect.

Note that the RESTORE command requires the save file to pre-exist. For many computers, such a requirement may be circumvented by local operating system directives. For example, on UNIVAC computers, if file XYZ does not exist and yet is needed in the middle of a DATAPLOT run, then the analyst may enter

@BASC,UP XYZ.

and that will create a file XYZ while still leaving the analyst in DATAPLOT. For other computers, the analyst should check locally to see if a similar system capability exists.

Starting Over
RESET

It is at times convenient to simply reset all of DATAPLOT's internal settings to their sign-on initial values. One way to do this is, of course, to simply exit out of DATAPLOT (via EXIT) and then sign back onto DATAPLOT (via the local sign-on procedure). An alternate way to do the same is to use the RESET command, as in

RESET

This will reinitialize all internal data settings (including, of course, all variable, parameter, and function settings). A synonym for RESET is CLEAR.
Communicating with the Host Operator

OPERATOR

The OPERATOR command allows the analyst to send a message (while in the middle of a DATAPLOT run) to the host operator. The form of the command is

OPERATOR message

Check locally with the DATAPLOT Service Group to determine if this feature has been implemented, and to determine if there are any restrictions on this feature (for example, the NBS Univac 1100/82 restricts the number of characters per line for a given message to be 40 at most). If it is necessary to send a multi-line message, then each line should start with the OPERATOR command, as in

OPERATOR Please set Zeta plotter with
OPERATOR wet ink #1 pen (red) and 30 inch paper.
OPERATOR Please send output to room 707.
The user may communicate with the DATAPLOT Service Group via the MESSAGE command. The form for the MESSAGE command is

```
MESSAGE message
```

If a message takes more than one line, then each successive line should start with the same MESSAGE command, as in

```
MESSAGE What is the easiest way to do
MESSAGE dynamic, color, 3d rotation?
MESSAGE Bill Martingale, extension 3365
```

Note that the message should contain the name of the user, along with other contact information such as phone and room number. This message will be inserted into a message file which the DATAPLOT Service Group will peruse on a regular basis, and respond accordingly.

There are 4 ways in which the DATAPLOT Service Group can communicate with a user--

1) via the sign-on message;
2) via the NEWS command;
3) via the MAIL command;
4) via the BUGS command.

There is 1 way in which the user can communicate to the Dataplot Service Group--

5) via the MESSAGE command.

The sign-on message appears whenever any user signs onto DATAPLOT. It contains important current information, and is usually short and to the point.

Details regarding current DATAPLOT developments may be evoked by the analyst entering the NEWS command, as in

```
NEWS
```

This will print out more detailed information regarding topics of current interest.

The DATAPLOT Service Group communicates directly to an individual user by use of the MAIL command. For example, suppose the Service Group wished to inform user John Smith that the chunk of code that he desired has been written. This is done by using the sign-on message to inform John Smith that he has mail, and then allowing John Smith to enter the MAIL command, as in

```
MAIL SMITH
```

which will in turn print out the detailed mail message that the DATAPLOT Service Group had constructed.
The following information is of interest only to the local DATAPLOT Service Group and may be skipped over by the general reader.

The file which contains the desired sign-on message is named DPMSF and has unit number 21 attached to it. The first line image of the file consists of a floating point number which is the number of lines actually contained by the message. Such lines follow immediately after the first line. For example, DPMSF might consist of the following 3 lines--

```
2.
New developments--Looping, Conditionality, and Variable Printing. Use NEWS command for details.
```

If no message is to be printed out, then the file DPMSF should have the first line consisting of merely a 0. on it.

The file which contains the desired news is named DPNDFP and has the unit number 22 attached to it. The structure of this file is the same as the message file--namely, a first line consisting of a floating point number defining the number of lines of the news, and succeeding lines which actually consist of the news, for example,

```
4.
The general form of the LOOP command is LOOP FOR parameter name = start increment stop
To terminate the loop, use the END OF LOOP command.
```

The file which contains the desired bug information is named DPBUGF and has the unit number 25 attached to it. The structure is as before--a leading line image with the number of lines of bug information, and successive line images containing the actual bug information, as in

```
2.
TIC COORDINATES command not working.
DOUBLE PRECISION command not working.
```

The file which contains the desired mail information is named DPMAIL and has unit number 23 attached to it. Since mail can exist for several users simultaneously in the file, a different structure has been used--to enter the mail for Carter, say, the DATAPLOT Service Group enters (via the local editor) a line image in the file which consists only of the 6 characters CARTER. The following line images consist of the mail message. The message is terminated whenever a line image is encountered which consists of at least 5 hyphens in a row. Such a line image is deliberately entered into the file in order to terminate a mail message and to separate such a mail message from the mail message to another user. For example,
A summary of the DATAPLOT file/subfile structure is as follows—

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Unit Number</th>
<th>Name</th>
<th>File or Subfile</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>21</td>
<td>DPMESSF</td>
<td>file only</td>
<td>MESSAGE</td>
</tr>
<tr>
<td>News</td>
<td>22</td>
<td>DPNEWF</td>
<td>file only</td>
<td>NEWS</td>
</tr>
<tr>
<td>Mail</td>
<td>23</td>
<td>DPMAILF</td>
<td>file only</td>
<td>MAIL</td>
</tr>
<tr>
<td>Help</td>
<td>24</td>
<td>DPHHELP</td>
<td>file only</td>
<td>HELP</td>
</tr>
<tr>
<td>Bug</td>
<td>25</td>
<td>DPBUGF</td>
<td>file only</td>
<td>BUG</td>
</tr>
<tr>
<td>Query</td>
<td>26</td>
<td>DPQUEF</td>
<td>file only</td>
<td>QUERY</td>
</tr>
<tr>
<td>Log</td>
<td>27</td>
<td>DPLOGF</td>
<td>off</td>
<td>---</td>
</tr>
<tr>
<td>Read</td>
<td>31</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>READ and SERIAL READ</td>
</tr>
<tr>
<td>Write</td>
<td>32</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>WRITE</td>
</tr>
<tr>
<td>Macro</td>
<td>33, 35-40</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>CREATE and CALL</td>
</tr>
<tr>
<td>Save</td>
<td>34</td>
<td>user-defined</td>
<td>file or subfile</td>
<td>SAVE and RESTORE</td>
</tr>
<tr>
<td>Scratch</td>
<td>41</td>
<td>DPSCRF</td>
<td>file only</td>
<td>---</td>
</tr>
<tr>
<td>Data</td>
<td>42</td>
<td>DPDAFTF</td>
<td>file only</td>
<td>---</td>
</tr>
<tr>
<td>Plot</td>
<td>43</td>
<td>DPPL1F</td>
<td>file only</td>
<td>---</td>
</tr>
<tr>
<td>Plot 2</td>
<td>44</td>
<td>DPPL2F</td>
<td>file only</td>
<td>---</td>
</tr>
</tbody>
</table>
Default Output Device Settings

Only the primary output device (the terminal) is on. All 4 secondary output devices are off. The primary output device is a Tektronix 4014 (or equivalent). The primary output device is continuous, non-color, and has 1024 horizontal picture points and 781 vertical picture points. The penplotter is turned off. The automatic local hardcopy is off. The Zeta, Calcomp, and Versatec are off.

Color Graphics TERMINAL, COLOR, and PICTURE POINTS

The default color status for DATAPLOT is off—that is, DATAPLOT believes it is communicating with a non-color terminal for its graphics. If one wishes to generate color graphics, then the default device is the Tektronix 4027. To inform DATAPLOT of this, the analyst must use the TERMINAL command, as in

TERMINAL TEKTRONIX 4027

The message will come back that the primary output device has been changed to the Tektronix 4027, the color status has been changed to on, and the picture points have been changed to 640 (horizontally) by 480 (vertically).

A second color terminal which DATAPLOT supports is the RAMTEK 6211. To inform DATAPLOT that one is running on such a terminal, one enters the following—

TERMINAL RAMTEK 6211
Multiple Commands Per Line

TERMINATOR CHARACTER

The vast majority of DATAPLOT programs have only one command statement per line image. Such a style is conducive to readability. On occasions, however, the analyst may wish to have multiple command statements per line. This is permitted by use of the terminator character (= a semicolon). Anytime that DATAPLOT encounters a semicolon (or a carriage return) in a line image, it assumes that one command statement has terminated and the next command statement is beginning.

Although the default terminator character is the semicolon, other terminator characters are possible. The TERMINATOR CHARACTER command specifies such a character, as in

    TERMINATOR CHARACTER !

which would replace the ; with a ! as a terminator character.

The semicolon was chosen as the default terminator character because its use otherwise in the usual assortment of DATAPLOT command statements is rare. The only place where its inadvertent use might arise is in the TITLE and LABEL statements— one should be careful about the use of the semicolon in such cases. If one wants a semicolon as part of a title or label, then one should first change the terminator character to something else, otherwise the title or label will be clipped and a syntax error will result.

Given that the semicolon is the terminator character, then the following 2 programs are identical—

```
READ ABC. X Y
PLOT Y X
FIT Y = A+B*X+C*X**2
CHARACTERS X BLANK
LINES BLANK SOLID
PLOT Y PRED VERSUS X
PLOT RES X
NORMAL PROBABILITY PLOT RES
```

and

```
READ ABC. X Y; PLOT Y X; FIT Y = A+B*X+C*X**2;
CHARACTERS X BLANK; LINES BLANK SOLID; PLOT Y PRED VERSUS X;
PLOT RES X; NORMAL PROBABILITY PLOT RES
```

The terminator character may be either included or excluded when it is the last character of a line image. In lines 1 and 2 above, it was included; in line 3 above, it was omitted.

Specifying Post-Plot Cursor Position and Size

CURSOR SIZE

This section refers to the size of the cursor (and the size of the characters) when the terminal is not generating graphics, but rather is simply receiving commands from the keyboard or is simply printing out results from various non-graphics DATAPLOT commands (such as FIT, ANOVA, SMOOTH, LET, etc.) In such case, it is at times convenient to be able to control the size of the characters being printed. It is also convenient to have the cursor/character size at one setting for the non-graphics, and perhaps at other settings for the graphics. Thus after a plot is formed (regardless of the size of the characters used in the plot), it is preferable for the cursor/characters to return to the non-graphics size.

In DATAPLOT this is controlled via the CURSOR SIZE command. The default cursor/character size is 3 (= 3% of total screen height). To set the cursor size to 1.5, say, the analyst enters

    CURSOR SIZE 1.5

The above comments apply only for those terminals which have multiple cursor sizes. For those terminals with only 1 size (e.g., Tektronix 4010, and Tektronix 4025), that 1 size will, of course, appear regardless of the cursor size setting.
1. General
2. Command Categories
3. Parameters, Variables, and Functions
4. Input/Output
5. Program Control
6. Miscellaneous
Index

A

Absolute Value Function ABS 3-13
Alphanumeric Graphics (DISCRETE Command) 2-8
Analysis Commands 2-3
Analysis of Variance (ANOVA) Command 1-9, 2-3, 5-3
Annotating Plots
  TITLE Command 1-3, 2-5, 2-8, 3-18
  LABEL Command 1-3, 2-5
  LEGEND Command 2-5
  TEXT Command 6-4, 2-4

Arithmetic Operations 3-1
AUTOCORRELATION Sub-command (under LET) 3-7
AUTOCORRELATION PLOT Command 2-2
AUTOCOVARIANCE Sub-command (under LET) 3-7

B

BACKGROUND COLOR Command 2-5, 6-9
BATCH Command 2-8
Bessel Functions BESSO and BESSL 3-13
BOX PLOT Command 2-2
BUGS Command 2-7, 6-34
Built-in Functions 3-12 to 3-14

General Functions 3-13
Probability Functions 3-14
Trigonometric Functions 3-14
CALCOMP Plotter Command 2-8, 6-23, 6-24
Calculating Statistics
   Via SUMMARY Command 2-3
   Via Sub-commands under LET 3-7, 3-10, 5-3
   For Subsets 5-3
CALL Command 3-18, 4-1, 5-4, 5-5
CHARACTERS Command 1-3, 2-5
Chebychev Polynomial Functions CHEBO to CHEB10 3-13
COLOR Graphics Commands 2-5, 2-8, 6-37
COLUMN LIMITS Command 2-6, 4-3, 4-12
   See also ROW LIMITS Command 2-6, 4-3, 4-11
   See also SKIP Command 2-6, 4-3, 4-11
Command Syntax 1-12 to 1-14
Command Categories 2-1 to 2-9
   Graphics Commands 2-2
   Analysis Commands 2-3
   Diagrammatic Graphics Commands 2-4
   Plot Control Commands 2-5
   Support Commands 2-6
   Output Device Commands 2-8
   Keywords 2-9
Commas (Never Used In Punctuation) 1-14
COMMENT (Synonym for .) Command 2-6, 6-29
Communications Commands 6-33 to 6-36
   With OPERATOR 6-33
   With Users 6-34 to 6-36
COMMUNICATIONS LINK Command 2-7, 6-19
Conditional Clauses
   SUBSET Clause 1-7, 3-1, 3-16, 4-4, 4-10, 5-1 to 5-3,
      6-1 to 6-3
   EXCEPT Clause 5-3, 6-2
   FOR Clause 1-10, 2-9, 3-5, 3-10, 4-10, 5-1 to 5-5
   IF Clause 2-6, 5-3 to 5-5
CONVOLUTION Sub-command (under LET) 3-8
COPY Command 2-4, 4-10, 6-27
CORRELATION Sub-command (under LET) 3-7
COVARIANCE Sub-command (under LET) 3-7
CREATE Command 5-6, 4-1, 2-6
   See also CREATE Command 5-6, 4-1, 2-6
CURSOR Settings 2-7, 6-38
Data Analysis Commands

PLOT Command 1-3, 1-7 to 1-9, 2-2, 3-1, 3-4 to 3-5, 3-12, 3-18, 4-4, 5-1 to 5-3, 6-1 to 6-3
FIT Command 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12, 3-18, 4-4
LET Command (To Calculate Statistics) 3-6 to 3-7, 3-10, 2-3
LET Command (To Transform Variables) 3-5, 3-12 to 3-15

HISTOGRAM Command 2-2
PROBABILITY PLOT Command 2-2
PPCC PLOT Command 2-2
SUMMARY Command (To Calculate Statistics) 2-3
STEM AND LEAF Command 2-2
BOX PLOT Command 2-2
ANOVA Command 2-3
MEDIAN POLISH Command 2-3
SPLINE FIT Command 2-3
SMOOTH Command 2-3
NORMALITY PLOT Command 2-2
LAG PLOT Command 2-2
CORRELATION PLOT Command 2-2
SPECTRAL PLOT Command 2-2
COMPLEX DEMODULATION PLOT Command 2-2
I PLOT Command 2-2
CONTROL CHART Command 2-2
FREQUENCY PLOT Command 2-2
PIE CHART Command 2-2
FREQUENCY PLOT Command 2-2
PERCENT POINT PLOT Command 2-2
3-D PLOT Command 2-2

Data Manipulation Sub-commands (under LET) 3-8, 3-9
Default Settings 6-8 to 6-10, 6-37
DEGREES Command 6-20, 2-6
DELETE Command 2-6, 3-16, 4-4
DERIVATIVE Sub-command (under LET) 3-8
DEVICE Commands 2-8, 6-37
Diagrammatic Graphics Commands 2-4

Default Settings 6-9

DIMENSION (of Internal Data Area) Command 2-8
DISCRETE Command 2-8

Distributional Modeling

FREQUENCY PLOT Command 2-2
HISTOGRAM Command 2-2
KURTOSIS Sub-command (under LET) 3-6 to 3-7
HOMENTS Sub-command (under LET) 3-6 to 3-7
PERCENT POINT PLOT Command 2-2
PIE CHART Command 2-2
PPCC PLOT Command 2-2
PROBABILITY PLOT Command 2-2
SKEWNESS Sub-command (under LET) 3-6 to 3-7
SUMMARY Command 2-3

Documentation (HELP Command) 2-7, 6-15 to 6-16
DRAW Command 2-4
ECHO Command 2-6, 6-21
END (Synonym for EXIT, STOP, and QUIT) Command 1-2
END OF CREATE Command 5-6, 2-6
END OF DATA Command 2-6, 3-4 to 3-5, 3-15, 3-20, 4-3, 4-12
END OF IF Clause 2-6, 5-3 to 5-5
END OF LOOP Command 2-6, 5-1 to 5-5
ERASE.Command 2-4, 5-1, 6-26 to 6-27
Error Functions ERF and ERFC 3-13
Examples 1-5 to 1-10

Graphics Problem 1-5, 1-7
Fitting Problem 1-5, 1-8
Data Analysis Problem 1-5, 1-9
Mathematics Problem 1-5, 1-10

EXCEPT Clause 5-3, 6-2
EXIT (Synonym for END, STOP, and QUIT) Command 1-2
Exponential Function EXP 3-13

FEEDBACK Command 2-6, 5-1, 6-22
Files 1-7, 4-1, 6-36
Default File Settings 6-10

Fitting
FIT Command 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12, 3-18, 4-4
SPLINE Command 2-3, 2-6
EXACT RATIONAL FIT Command
PRE-FIT Command 2-3, 3-12, 5-3
Fitting Subsets of Data 5-3

FONT Command

(8 Available Fonts—Tektronix, Simplex, Duplex,
Triplex, Triplex Italic, Complex,
Simplex Script, Complex Script) 6-4, 6-8

FOR Clause 1-10, 2-9, 3-5, 3-10, 4-10, 5-1 to 5-5
Fraction Function FRACT 3-13
Functions 1-13

Built-in Functions 3-12 to 3-14
Copying Functions 3-5
Creating Functions 3-5, 3-15
Deleting Functions 3-16 to 3-17
Derivative of Function
Evaluating Functions 3-5
Fitting Functions 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12,
3-18, 4-4
General Form 3-12
Integral of Functions 3-8
Plotting Functions 1-5, 1-10, 2-2, 3-1, 3-5, 3-12
Roots of Functions 1-10, 3-8
Concatenation of Functions 3-5
Gamma Function GAMMA and LOGGAMMA 3-13
General Functions 3-13
Getting Started 1-3 to 1-4
GRADS Command 6-20, 2-6
Graphical ANOVA 1-9
Graphics Commands 2-2
Graphics Output Devices
Tektronix (e.g., DEVICE 1 TEKTRONIX 4105) 6-37, 6-9, 2-8
Hewlett-Packard (e.g., DEVICE 2 HP-GL) 6-37, 6-9, 2-8
Device-Independent (e.g., DEVICE 3 GENERAL) 6-37, 6-9, 2-8
Batch (e.g., BATCH ON) 2-8
Alphanumeric Terminal (e.g., DISCRETE ON) 2-8
Zeta (e.g., ZETA ON) 6-23, 6-37, 6-9, 2-8
Calcomp (e.g., CALCOMP ON) 6-23, 6-37, 6-9, 2-8
Versatec (e.g., VERSATEC ON) 6-23, 6-37, 6-9, 2-8
Greek Characters (Hershey Fonts Only) 6-5, 6-4
HARDCOPY Command 5-2, 6-26
See also COPY Command 2-4, 4-10, 6-27
HELP Command 2-7, 6-15 to 6-16
HINCE Sub-commands (under LET) 3-7
HISTOGRAM Command 2-2
HOST (Specification) Command 6-18
IF Clause 2-6, 5-3 to 5-5
IMPLEMENT Command 6-28
Implementation Information 6-35 to 6-36
Input/Output Commands 4-1 to 4-12, 2-6, 6-25
READ Command 1-3, 1-8 to 1-9, 2-6, 3-4 to 3-5, 3-15,
3-19 to 3-20, 4-1
SERIAL READ Command 3-4 to 3-5, 3-15, 3-19 to 3-20, 4-1,
4-3, 4-7
WRITE Command 2-6, 3-5, 4-1, 4-3, 4-10, 6-3, 6-22
Integer Function INT 3-13
INTEGRAL Sub-command (under LET) 3-8
Interactive Analysis 1-1, Preface
Interrupting, Saving, and Resuming a Run 6-32
JUSTIFICATION Command 6-9, 2-4
Labelling Plots

TITLE Command 1-3, 2-5, 2-8, 3-18
LABEL Command 1-3, 2-5
LEGEND Command 2-5
TEXT Command 6-4, 2-4

LAG PLOT Command 2-2
Language Components 1-14
LEGEND Command 2-5
LET Command 1-8 to 1-10, 2-3, 3-1 to 3-19

Defining Parameters 3-3
Defining Variables 3-4
Defining Functions 3-5
Evaluating Functions 3-5
Computing Statistics 3-7
Performing mathematical Operations 3-8
Generating random Numbers 3-9
Creating Data 3-9
Copying Data 3-15

LIMITS Command 6-8, 2-5
LINES (type) Command 1-3, 1-8 to 1-9, 2-5
LIST (files) command 5-6
Log Scale on Plots 6-8, 2-5
Logarithm Functions LN, LOG, LOG10, and LOG2 3-13
LOOP Command 2-6, 5-1 to 5-5

MACRO (Synonym of CALL) Command 2-6, 4-1, 5-6
MAIL Command 2-7, 6-34
Map Generation (Need PRE-SORT OFF Command) 2-5
Math Library 3-12 to 3-14
Mathematical Symbols (Hershey Fonts Only) 6-6, 6-4
Mathematical Sub-commands (under LET) 3-8
MAXIMUM Sub-command (under LET) 3-7
Maximum Functions MAX 3-13
MEAN Sub-command (under LET) 1-3, 3-7, 3-10
MEDIAN Sub-command (under LET) 3-7
MESSAGE Command 2-7, 6-34
MIDMEAN Sub-command (under LET) 3-7
MIDRANGE Sub-command (under LET) 3-7
MINIMUM Sub-command (under LET) 3-7
Minimum Functions MIN 3-13
Miscellaneous Symbols (Hershey Fonts Only) 6-7, 6-4
Modulo Function MOD 3-13
MOMENT Sub-commands (under LET) 3-7
MOVE Command 2-4

NAME Command 2-6, 3-18
NEWS Command 2-7, 6-34
Non-Linear Fit 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12, 3-18, 4-4
Numbers 3-2
Octal Conversion Functions OCTDEC and DECOCT 3-13
OPERATOR Command 2-7, 6-34
Output Device Commands 2-8

Default Settings 6-9

Parameters 3-3, 1-13
PIE CHART Command 2-2
PENPLOTTER Command 2-8, 6-23
PLOT Command 1-3, 1-7 to 1-9, 2-2, 3-1, 3-4 to 3-5, 3-12, 3-18, 4-4, 5-1 to 5-3, 6-1 to 6-3
Plot Control Commands 2-5 to 2-6, 6-8 to 6-9

Default Settings 6-8 to 6-9

Plotter Settings 6-23 to 6-24, 6-37

Plotting

Plotting Data 1-3, 1-7 to 1-9, 2-2, 3-1, 3-4 to 3-5, 3-12, 3-18, 4-4, 5-1 to 5-3, 6-1 to 6-3
Plotting Functions 1-5, 1-10, 2-2, 3-1, 3-5, 3-12
Plotting Subsets of Data 6-1 to 6-3

Predicted Values (From Regression) 2-9, 3-18, 4-4, 6-3

See also FIT Command 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12, 3-18, 4-4

Portability 1-1, Preface
PRE-FIT Command 2-3, 3-12, 5-3

See also FIT Command 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12, 3-18, 4-4

PRINT (Synonym for WRITE) Command 2-6, 3-5; 4-1, 4-3, 4-10, 6-3, 6-22

Probability Functions 3-14
PROBABILITY PLOT Commands 1-8, 2-2, 3-11
Program Control 5-1 to 5-6

Looping 5-1 to 5-5
Conditional Statements 5-3
Creating Subprograms 5-6
Calling Subprograms 5-5
Listing Subprograms and files 5-6

Programming

Conventions 1-11 to 1-14
Suggested Practices 6-30 to 6-31

Punctuation 1-14
QUARTILE Sub-commands (under LET) 3-7
QUIT (Synonym for END, EXIT, and STOP) Command 1-2

RADIANS Command 6-20, 2-6
RANDOM NUMBER Generator Sub-commands (under LET) 3-4, 3-9 to 3-11, 3-20
RANGE Sub-command (under LET) 3-7
READ Command 1-3, 1-8 to 1-9, 2-6, 3-4 to 3-5, 3-15, 3-19 to 3-20, 4-1
READ FUNCTION Command 4-8
READ PARAMETER Command 4-8
Redimensioning Internal Data Area 2-8
Regression
FIT Command 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12, 3-18, 4-4
SPLINE Command 2-3, 2-6
EXACT RATIONAL FIT Command
PRE-FIT Command 2-3, 3-12, 5-3
Fitting Subsets of Data 5-3

Relational Operators 3-1
RESET (Defaults) Command 2-6, 6-32
Residual Analysis 1-8, 2-9, 3-18, 4-4, 6-3

See also FIT Command 1-3, 1-8, 2-3, 2-6, 3-2 to 3-4, 3-12, 3-18, 4-4

RESTORE (Saved DATAPLOT Settings) Command 2-6, 4-1, 6-32
See also SAVE Command 2-6, 4-1, 6-32

ROOTS Sub-command (under LET) 3-8, 3-10
ROW LIMITS Command 2-6, 4-3, 4-11
See also COLUMN LIMITS Command 2-6, 4-3, 4-12
See also SKIP Command 2-6, 4-3, 4-11
SAVE (Internal DATAPLOT Settings) Command 2-6, 4-1, 6-32

See also RESTORE Command 2-6, 4-1, 6-32

Saving Internal Settings and Data to Interrupt a Run

SAVE Command 2-6, 4-1, 6-32
RESTORE Command 2-6, 4-1, 6-32

Screen Resolution (PICTURE POINTS Command) 6-9, 2-8
SERIAL READ Command 3-4 to 3-5, 3-15, 3-19 to 3-20, 4-1, 4-3, 4-7
Settings 6-8 to 6-10
SET (Trace Switches) Command 2-7, 6-25
Sign Function SIGN 3-13
SKEWNESS Sub-command (under LET) 3-7
SKIP (Lines) Command 2-6, 4-3, 4-11

See also COLUMN LIMITS Command 2-6, 4-3, 4-12
See also ROW LIMITS Command 2-6, 4-3, 4-11

SMOOTH Command 2-3
SORT Sub-command (under LET) 3-8
SPLINE Command 2-3, 2-6
Square Root Function SQRT 3-13
STANDARD DEVIATION Sub-command (under LET) 3-7
Statistical Calculations

Via SUMMARY Command 2-3
Via Sub-commands under LET 3-7, 3-10, 5-3
For Subsets 5-3

STATUS Command 6-11 to 6-14
STOP (Synonym for END, EXIT, and QUIT) Command 1-2
Subprograms

CREATE Command 5-6, 4-1, 2-6
CALL Command 3-18, 4-1, 5-4, 5-5
LIST Command 5-6

SUBSET Clause 1-7, 3-1, 3-16, 4-4, 4-10, 5-1 to 5-3, 6-1 to 6-3
SUMMARY Statistics Command 2-3
Support Commands 2-6

Default Settings 6-9

Suppressing Printed Output 6-22
Symbols (For Hershey Fonts)

Greek 6-5, 6-4
Mathematical 6-6, 6-4
Miscellaneous 6-7, 6-4

Syntax 1-12 to 1-14
Tektronix 6-37, 6-9, 2-8
Terminal Control Settings 6-18, 6-8 to 6-9

COPY Command 2-4, 4-10, 6-27
ERASE Command 2-4, 5-1, 6-26 to 6-27
HARDCOPY Command 5-2, 6-26

TERMINATOR CHARACTER Command 2-7, 6-38
TEXT Command 2-4, 6-4 to 6-8

Special Symbols 6-5 to 6-7

TIC MARK Command 2-5

Time Series Graphics

PLOT Command 1-3, 1-7 to 1-9, 2-2, 3-1, 3-4 to 3-5, 3-12,
3-18, 4-4, 5-1 to 5-3, 6-1 to 6-3
LAG PLOT Command 2-2
CORRELATION PLOT Command 2-2
SPECTRAL PLOT Command 2-2
COMPLEX DEMODULATION PLOT Command 2-2

TITLE Command 1-3, 2-5, 2-8, 3-18
Transformations

Via LET Command 3-5, 3-12 to 3-15
To Normality Via NORMALITY PLOT Command 2-2

Trigonometric Functions 3-14
TRIGONOMETRIC UNITS Command 2-6, 6-20

Variables 1-13

Creating Variables 3-4, 3-19 to 3-20
Copying Variables 3-15
Deleting Variables 3-16 to 3-17
Renaming Variables 3-18

VARIANCE Sub-command (under LET) 3-7
VERSATEC Plotter Command 2-8, 6-23 to 6-24
Word Charts 2-4, 6-4 to 6-9  
WRITE Command 2-6, 3-5, 4-1, 4-3, 4-10, 6-3, 6-22

ZETA Plotter Command 2-8, 6-23 to 6-24

3D-Plot Command 2-2, 3-12  
(Synonym for COMMENT) Command 2-6, 6-29
DATAPLOT--Introduction and Overview

Library of Congress Catalog Card Number: 83-600598

This manual provides DATAPLOT code solution to a variety of commonly occurring graphical problems. A line-by-line explanation of code is given, along with illustrations and general discussion.
NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $18; foreign $22.50. Single copy, $3.50 domestic; $6.90 foreign.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).

NOTE: The principal publication outlet for the foregoing data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St., NW, Washington, DC 20036.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15 of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace. Order the above NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.

Order the following NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.


NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.