

CERN Program Library Long Writeup Q121

PAW

Physics Analysis Workstation

The Complete Reference

Version 1.14 (July 1992)

Application Software Group

Computing and Networks Division

CERN Geneva, Switzerland

Copyright Notice

PAW – Physics Analysis Workstation

CERN Program Library entry Q121

Copyright and any other appropriate legal protection of these computer programs and associated documentation reserved in all countries of the world.

These programs or documentation may not be reproduced by any method without prior written consent of the Director-General of CERN or his delegate.

Permission for the usage of any programs described herein is granted apriori to those scientific institutes associated with the CERN experimental program or with whom CERN has concluded a scientific collaboration agreement.

CERN welcomes comments concerning this program but undertakes no obligation for its maintenance, nor responsibility for its correctness, and accepts no liability whatsoever resulting from the use of this program.

Requests for information should be addressed to:

CERN Program Library Office
CERN-CN Division
CH-1211 Geneva 23
Switzerland
Tel. +41 22 767 4951
Fax. +41 22 767 7155
Bitnet: CERNLIB@CERNVM
DECnet: VXCERN::CERNLIB (node 22.190)
Internet: CERNLIB@CERNVM.CERN.CH

Trademark notice: All trademarks appearing in this guide are acknowledged as such.

Contact Person: René Brun /CN (BRUN@CERNVM.CERN.CH)

Technical Realization: Michel Goossens /CN (GOOSSENS@CERNVM.CERN.CH)

Second edition - July 1992

About this guide

Preliminary remarks

This **Complete Reference** of PAW (for Physics Analysis Workstation), consists of three parts:

- 1 A **step by step** tutorial introduction to the system.
- 2 A **functional description** of the components.
- 3 A **reference guide**, describing each command in detail.

The PAW system is implemented on various mainframes and personal workstations. In particular versions exist for IBM VM/CMS and MVS/TSO, VAX/VMS and various Unix-like platforms, such as APOLLO, DEC Station 3100, Silicon Graphics and SUN.

In this manual examples are in `monotype face` and strings to be input by the user are underlined. In the index the page where a command is defined is in **bold**, page numbers where a routine is referenced are in normal type.

In the description of the commands parameters between square brackets [. . .] are optional.

Acknowledgements

The authors of PAW would like to thank all their colleagues who, by their continuous interest and encouragement, have given them the necessary input to provide a modern and easy to use data analysis and presentation system.

They are particularly grateful to Michel Goossens as main author of part one and as technical editor of the present document and to Michael Metcalf for his work on improving the index.

Vladimir Berezhnoi (IHEP, Serpukhov, USSR), the main author of the Fortran interpreter COMIS, provided one of the essential components of our system. Nicole Cremel has collaborated to the first versions of HPLOT. The PAW/HBOOK to MINUIT interface has been implemented in collaboration with Eliane Lessner (FNAL, USA) and Fred James. Jim Loken (Oxford, UK) has been our expert on VAX global sections. David Foster, Frederic Hemmer, Catherine Magnin and Ben Segal have contributed to the development of the PAW TCP/IP interface. Ben has also largely contributed to the TELNETG and 3270G systems. Per Scharff-Hansen and Johannes Raab from the OPAL collaboration have made possible the interface with the OS9 system. Harald Johnstad (FNAL, now SSC, USA) and Lee Roberts (FNAL, USA) have contributed to the debugging phases of PAW in the DI3000 and DECGKS environments. Initial implementations of PAW on MVS/TSO, the Sun and the DEC Station 3100 were made by Alain Michalon (Strasbourg, France), François Marabelle (Saclay, France) and Walter Bruckner (Heidelberg, FRG), respectively. Lionel Cons (now at ENSIMAG, Grenoble) has contributed to the implementation of the selection mechanisms for Ntuples. Isabelle Moulinier (Paris) has been working, as a summer student, on various improvements in the HIGZ/HPLOT packages. Federico Carminati, the main distributor of the CERN program library had to suffer from the many imperfections of our first releases. His collaboration for PAW consultancy is appreciated. Gudrun Benassi has always kindly organized the distribution of the various PAW manuals.

Related Manuals

This document can be complemented by the following manuals:

- COMIS, Compilation and Interpretation System [1]
- HBOOK User Guide — Version 4 [2]
- HIGZ — High level Interface to Graphics and ZEBRA [3]
- HPLOT User Guide — Version 5 [4]
- KUIP — Kit for a User Interface Package [5]
- MINUIT — Function Minimization and Error Analysis [6]
- ZEBRA — Data Structure Management System [7]

This document has been produced using LATEX [8] with the `cernman` style option, developed at CERN.
All pictures shown are produced with PAW and are included in PostScript [9] format in the manual.

A PostScript file `pawman.ps`, containing a complete printable version of this manual, can be obtained by anonymous ftp as follows (commands to be typed by the user are underlined):

```
ftp asis01.cern.ch
Trying 128.141.201.136...
Connected to asis01.cern.ch.
220 asis01 FTP server (Version 6.10 Mon Apr 13 15:59:17 MET DST 1992) ready.
Name (asis01:username): anonymous
331 Guest login ok, send e-mail address as password.
Password: your_mailaddress
ftp> cd doc/cernlib
ftp> get pawman.ps
ftp> quit
```

Table of Contents

I PAW – Step by step	1
1 A few words on PAW	3
1.1 A short history	3
1.2 What is PAW?	3
1.3 What Can You Do with PAW?	3
1.4 A User's View of PAW	5
1.5 Fundamental Objects of PAW	6
1.6 The Component Subsystems of PAW	9
1.6.1 KUIP - The user interface package	10
1.6.2 HBOOK and HPLOT - The histogramming and plotting packages	10
1.6.3 HIGZ - The graphics interface package	10
1.6.4 ZEBRA - The data structure management system	11
1.6.5 MINUIT - Function minimization and error analysis	11
1.6.6 COMIS - The FORTRAN interpreter	12
1.6.7 SIGMA - The array manipulation language	12
1.7 A PAW Glossary	12
2 General principles	15
2.1 Access to PAW	15
2.1.1 IBM/VM-CMS	15
2.1.2 VAX/VMS	15
2.1.3 Unix systems	15
2.1.4 Note on the X11 version	16
2.1.5 Important modes to run PAW	16
2.2 Initialising PAW	16
2.3 Command structure	17
2.4 Getting help	18
2.4.1 Usage	20
2.5 Special symbols for PAW	20
2.6 PAW entities and their related commands	20
3 PAW by examples	23
3.1 Vectors and elementary operations	25
3.2 One and two-dimensional functions	39
3.3 Using histograms	47
3.4 Examples with Ntuples	57
3.4.1 A first example - CERN personnel statistics	57
3.4.2 Creating Ntuples	58
3.5 The SIGMA application and more complex examples	73

II PAW - Commands and Concepts	89
4 User interface - KUIP	91
4.1 The PAW command structure	91
4.2 Multiple dialogue styles	92
4.2.1 Command line mode	92
4.2.2 An overview of KUIP menu modes	95
4.3 Macros	96
4.3.1 Special Parameters	98
4.3.2 Macro Flow Control	99
4.4 Aliases	101
4.5 System functions	103
4.5.1 The \$SIGMA system function in more detail	104
4.6 More on aliases, system functions and macro variables	105
4.7 Recalling previous commands	106
4.8 Exception condition handling	106
5 Vectors	107
5.1 Vector creation and filling	107
5.2 Vector addressing	108
5.3 Vector arithmetic operations	108
5.4 Vector arithmetic operations using SIGMA	108
5.5 Using KUIP vectors in a COMIS routine	109
5.6 Usage of vectors with other PAW objects	109
5.7 Graphical output of vectors	109
5.8 Fitting the contents of a vector	109
6 SIGMA	110
6.1 Access to SIGMA	110
6.2 Vector arithmetic operations using SIGMA	110
6.2.1 Basic operators	111
6.2.2 Logical operators	111
6.2.3 Control operators	111
6.3 SIGMA functions	112
6.3.1 SIGMA functions - A detailed description.	113
6.4 Available library functions	120

7 HBOOK	122
7.1 Introduction	122
7.1.1 The functionality of HBOOK	122
7.2 Basic ideas	123
7.2.1 RZ directories and HBOOK files	123
7.2.2 Changing directories	124
7.3 HBOOK batch as the first step of the analysis	125
7.3.1 Adding some data to the RZ file	127
7.4 Using PAW to analyse data	129
7.4.1 Plot histogram data	130
7.5 Ntuples: A closer look	131
7.5.1 Ntuple plotting	132
7.5.2 Ntuple variable and selection function specification	133
7.5.3 Ntuple selection mechanisms	134
7.5.4 Masks	134
7.5.5 Examples	138
7.6 Fitting with PAW/HBOOK/MINUIT	141
7.6.1 Basic concepts of MINUIT.	141
7.6.2 Basic concepts - The transformation for parameters with limits	141
7.6.3 How to get the right answer from MINUIT.	142
7.6.4 Interpretation of Parameter Errors:	142
7.6.5 Fitting histograms	144
7.6.6 A simple fit with a gaussian	145
7.7 Doing more with Minuit	151
8 Graphics (HIGZ and HPLOT)	155
8.1 HPLOT, HIGZ and local graphics package	155
8.2 The metafiles	156
8.3 The HIGZ pictures	156
8.3.1 Pictures in memory	157
8.3.2 Pictures on direct access files	158
8.4 HIGZ pictures generated in a HPLOT program	160
8.5 Setting attributes	161
8.6 More on labels	166
8.7 Colour, line width, and fill area in HPLOT	169
8.8 Information about histograms	171
8.9 Additional details on some IGSET commands	172
8.10 Text fonts	178
8.11 The HIGZ graphics editor	186

9	Distributed PAW	187
9.1	TELNETG and 3270G	187
9.2	ZFTP	190
9.3	Access to remote files from a PAW session	191
9.4	Using PAW as a presenter on VMS systems (global section)	192
9.5	Using PAW as a presenter on OS9 systems	193
 III PAW - Reference section		 195
10	KUIP	197
10.1	ALIAS	199
10.2	SET_SHOW	200
11	MACRO	206
12	VECTOR	208
12.1	OPERATIONS	212
13	HISTOGRAM	214
13.1	2D_PLOT	217
13.2	CREATE	219
13.3	HIO	221
13.4	OPERATIONS	223
13.5	GET_VECT	227
13.6	PUT_VECT	228
13.7	SET	228
14	FUNCTION	231
15	NTUPLE	234
16	GRAPHICS	242
16.1	MISC	243
16.2	VIEWING	244
16.3	PRIMITIVES	245
16.4	ATTRIBUTES	253
16.5	HPILOT	255
17	PICTURE	258

18 ZEBRA	261
18.1 RZ	261
18.2 FZ	262
18.3 DZ	263
19 FORTRAN	266
20 OBSOLETE	269
20.1 HISTOGRAM	269
20.1.1 FIT	269
21 NETWORK	272
A PAW tabular overview	273
Bibliography	284
Index	285

List of Figures

1.1 PAW and its components	9
2.1 PAW entities and their related commands	22
3.1 Simple vector commands	26
3.2 Further vector commands and writing vectors to disk	28
3.3 Various data representations	30
3.4 Difference between VECTOR/DRAW and VECTOR/PLOT	32
3.5 Vector operations	34
3.6 Simple macro with loop and vector fit	36
3.7 Plotting one-dimensional functions	40
3.8 One-dimensional functions and loops	42
3.9 Plotting two-dimensional functions	44
3.10 Plotting a two-dimensional function specified in a external file	46
3.11 Creation of one- and two-dimensional histograms	48
3.12 Reading histograms on an external file	50
3.13 One-dimensional plotting and histogram operations	52
3.14 Two-dimensional data representations	54
3.15 The use of sub-ranges in histogram specifiers	56
3.16 Ntuples - Creation and output to a file	62
3.17 Ntuples - Automatic and user binning	64
3.18 Ntuples - A first look at selection criteria	66
3.19 Ntuples - Masks and loop	68
3.20 Ntuples - Using cuts	70
3.21 Ntuples - Two dimensional data representation	72
3.22 Using the SIGMA processor - Trigonometric functions	74
3.23 Using the SIGMA processor - More complex examples	76
3.24 Histogram operations (Keep and Update)	78
3.25 Merging several pictures into one plot	80
3.26 Pie charts with hatch styles and PostScript colour simulation	82
3.27 A complex graph with PAW	85
3.28 Making slides with PAW using PostScript	88
4.1 Example of the PAW command tree structure	91
6.1 Using numerical integration with SIGMA	119
7.1 The layout of the /PAWC/ dynamic store	123
7.2 Schematic presentation of the various steps in the data analysis chain	125

7.3	Writing data to HBOOK with the creation of a HBOOK RZ file	126
7.4	Output generated by job HTEST	126
7.5	Adding data to a HBOOK RZ file	128
7.6	Reading a HBOOK direct access file	129
7.7	Plot of one- and two-dimensional histograms	130
7.8	Print and scan Ntuple elements	132
7.9	Graphical definition of cuts	136
7.10	Read and plot Ntuple elements	139
7.11	Selection functions and different data presentations	140
7.12	Example of a simple fit of a one-dimensional distribution	146
7.13	Example of a fit using sub-ranges bins	149
7.14	Example of a fit using a global double gaussian fit	150
8.1	HPLOT and HIGZ in PAW	155
8.2	Visualising a HIGZ picture produced in a batch HPLOT program	160
8.3	A graphical view of the SET parameters	165
8.4	Example of labelling for horizontal axes	166
8.5	Example of labelling for vertical axes	168
8.6	Example of fill area types in HPLOT	170
8.7	Examples of HIGZ portable hatch styles	175
8.8	GKSGRAL Device independent hatch styles	176
8.9	HIGZ portable marker types	177
8.10	HIGZ portable line types	177
8.11	PostScript grey level simulation of the basic colours	179
8.12	HIGZ portable software characters (Font 0, Precision 2)	180
8.13	PostScript fonts	181
8.14	Correspondence between ASCII and ZapfDingbats font (-14)	182
8.15	Octal codes for PostScript characters in Times font	183
8.16	Octal codes for PostScript characters in ZapfDingbats font (-14)	184
8.17	Octal codes for PostScript characters in Symbol font	185
8.18	The HIGZ graphics editor	186
9.1	The TELNETG program	188
9.2	Visualise histograms in global section	192
9.3	Visualising histograms on OS9 modules from PAW	193

List of Tables

2.1	Special symbols	21
3.1	Definition of the variables of the CERN staff Ntuple	57
4.1	List of statements possible inside KUIP macros	97
4.2	KUIP system functions	103
6.1	SIGMA functions	112
7.1	Syntax for specifying Ntuple variables	133
7.2	Syntax of a selection function used with a Ntuple	133
7.3	Functions callable from PAW	137
7.4	Comparison of results of fits for the double gaussian distribution	148
8.1	Parameters and default values for IGSET	162
8.2	Parameters and default values for OPTION	163
8.3	Parameters and default values in SET	164
8.4	Text alignment parameters	173
8.5	Codification for the HIGZ portable fill area interior styles	174
8.6	List of HPLOT escape sequences and their meaning	178
A.1	Alphabetical list of PAW commands	273
A.2	Overview of PAW commands by function	277

Part I

PAW – Step by step

Chapter 1: A few words on PAW

1.1 A short history

Personal workstations equipped with a 1 Mbit bitmap display, a speed of several tens of MIPS, with at least 20-30 Mbytes of main memory and 1 Gbyte of local disk space (e.g. DEC, HP-700, IBM RS6000, Sun Sparc and Silicon Graphics workstations) are now widely available at an affordable price for individual users. In order to exploit the full functionality of these workstations, at the beginning of 1986 the Physics Analysis Workstation project **PAW** was launched at CERN. The first public release of the system was made at the beginning of 1988. At present the system runs on most of the computer systems used in the High Energy Physics (HEP) community but its full functionality is best exploited on personal workstations. In addition to its powerful data analysis, particular emphasis has been put on the quality of the user interface and of the graphical presentation.

1.2 What is PAW?

PAW is an interactive utility for visualizing experimental data on a computer graphics display. It may be run in batch mode if desired for very large data analyses; typically, however, the user will decide on an analysis procedure interactively before running a batch job.

PAW combines a handful of CERN High Energy Physics Library systems that may also be used individually in software that processes and displays data. The purpose of PAW is to provide many common analysis and display procedures that would be duplicated needlessly by individual programmers, to supply a flexible way to invoke these common procedures, and yet also to allow user customization where necessary.

Thus, PAW's strong point is that it provides quick access to many facilities in the CERN library. One of its limitations is that these libraries were not designed from scratch to work together, so that a PAW user must eventually become somewhat familiar with many dissimilar subsystems in order to make effective use of PAW's more complex capabilities. As PAW evolves in the direction of more sophisticated interactive graphics interfaces and object-oriented interaction styles, the hope is that such limitations will gradually become less visible to the user.

PAW is most effective when it is run on a powerful computer workstation with substantial memory, rapid access to a large amount of disk storage, and graphics support such as a large color screen and a three-button mouse. If the network traffic can be tolerated, PAW can be run remotely over the network from a large, multiuser client machine to more economical servers such as an X-terminal. In case such facilities are unavailable, substantial effort has been made to ensure that PAW can be used also in noninteractive or batch mode from mainframes or minicomputers using text terminals.

1.3 What Can You Do with PAW?

PAW can do a wide variety of tasks relevant to analyzing and understanding physical data, which are typically statistical distributions of measured events. Below we list what are probably the most frequent and best-adapted applications of PAW; the list is not intended to be exhaustive, for it is obviously possible to use PAW's flexibility to do a huge number of things, some more difficult to achieve than others within the given structure.

Typical PAW Applications:

- **Plot a Vector of Data Fields for a List of Events.** A set of raw data is typically processed by the user's own software to give a set of physical quantities, such as momenta, energies, particle identities, and so on, for each event. When this digested data is saved on a file as an Ntuple, it may be read and manipulated directly from PAW. Options for plotting Ntuples include the following:
 - *One Variable.* If a plot of a one variable from the data set is requested, a histogram showing the statistical distribution of the values from all the events is automatically created. Individual events are not plotted, but appear only as a contribution to the corresponding histogram bin.
 - *Two or Three Variables.* If a plot of two or three variables from the data set is requested, no histogram is created, but a 2D or 3D scatter plot showing a point or marker for each distinct event is produced.
 - *Four Variables.* If a plot of four variables is requested, a 3D scatter plot of the first three variables is produced, and a color map is assigned to the fourth variable; the displayed color of the individual data points in the 3D scatter plot indicates the approximate value of the fourth variable.
 - *Vector Functions of Variables.* PAW allows the user to define arbitrary vector functions of the original variables in an Ntuple, and to plot those instead of the bare variables. Thus one can easily plot something like $\sqrt{(P_x^2 + P_y^2)}$ if P_x and P_y are original variables in the data without having to add a new data field to the Ntuple at the time of its creation.
 - *Selection Functions (Cuts).* PAW does not require you to use every event in your data set. Several methods are provided to define Boolean functions of the variables themselves that pick out subsets of the events to be included in a plot.
 - *Plot presentation options.* The PAW user can set a variety of options to customize the format and appearance of the plots.
- **Histogram of a Vector of Variables for a List of Events.** Often one is more interested in the statistical distribution of a vector of variables (or vector functions of the variables) than in the variables themselves. PAW provides utilities for defining the desired limits and bin characteristics of a histogram and accumulating the bin counts by scanning through a list of events. The following are some of the features available for the creation of histograms:
 - *One Dimensional Histograms.* Any single variable can be analyzed using a one-dimensional histogram that shows how many events lie in each bin. This is basically equivalent to the single-variable data plotting application except that it is easier to specify personalized features of the display format. A variety of features allow the user to slice and project a 2D scatter plot and make a 1D histogram from the resulting projection.
 - *Two-Dimensional Histograms.* The distribution of any pair of variables for a set of events can be accumulated into a 2D histogram and plotted in a various of ways to show the resulting surface.
 - *Three-Dimensional Histograms.* Will be supported soon.
 - *Vector Functions of Variables.* User-defined functions of variables in each event can be used to define the histogram, just as for an Ntuple plot.

- *Selection Functions (Cuts).* Events may also be included or excluded by invoking Boolean selection functions that are arbitrary functions of the variables of a given event.
- *Event Weights.* PAW allows the user to include a multiplicative statistical bias for each event which is a scalar function of the available variables. This permits the user to correct for known statistical biases in the data when making histograms of event distributions.
- *Histogram Presentation Options.* Virtually every aspect of the appearance of a histogram can be controlled by the user. Axis labels, tick marks, titles, colors, fonts, and so on, are specified by a large family of options. A particular set of options may be thought of as a “style” for presenting the data in a histogram; “styles” are in the process of becoming a formal part of PAW to aid the user in making graphics that have a standard pleasing appearance.
- **Fit a Function to a Histogram.** Once a histogram is defined, the user may fit the resulting shape with one of a family of standard functions, or with a custom-designed function. The parameters of the fit are returned in user-accessible form. Fitted functions of one variable may be attached to a 1D histogram and plotted with it. The capability of associating fits to higher dimensional histograms and overlaying their representations on the histogram is in the process of being added to PAW.
The fitting process in PAW is normally carried out by the MINUIT library. To use this package effectively, users must typically supply data with reasonable numerical ranges and give reasonable initial conditions for the fit before passing the task to the automated procedure.
- **Annotate and Print Graphics.** A typical objective of a PAW user is to examine, manipulate, and display the properties of a body of experimental data, and then to prepare a graph of the results for use in a report, presentation, or publication. PAW includes for convenience a family of graphics primitives and procedures that may be used to annotate and customize graphics for such purposes. In addition, any graphics display presented on the screen can be converted to a PostScript file for black-and-white or color printing, or for direct inclusion in a manuscript.

1.4 A User's View of PAW

In order to take advantage of PAW, the user must first have an understanding of its basic structure. Below we explain the fundamental ways in which PAW and the user interact.

Initialization. PAW may be invoked in a variety of ways, depending on the user's specific computer system; these are described in the following chapter. As PAW starts, it prompts the user to select an interaction mode (or non-interactive mode) and window size and type (if interactive). The available window sizes and positions are specified in the user file "higz_windows.dat". User-specific intializations are specified in the file "pawlogon.kumac".

Text Interface. The most basic interface is the **KUIP text interface**. KUIP provides a basic syntax for commands that are parsed and passed on to the PAW application routines to perform specific tasks. Among the basic features of KUIP with which the user interacts are the following:

- *Command Entry.* Any unique partially entered command is interpreted as a fully entered command. KUIP responds to an ambiguous command by listing the possible alternatives. On Unix systems, individual command lines can be edited in place using individual control keystrokes similar to those of the emacs editor, or the bash or tcsh Unix command shells. On other systems, a command line that is in error can only be revised after it is entered, using an “ed” style text line editing language.

- *Parameters.* Parameters are entered after the basic command on the same line and are separated by spaces, so algebraic expressions may not have embedded blanks. An exclamation point (!) can be used to keep the default parameters in a sequence when only a later parameter is being changed. If an underscore (_) is the last character on a line, the command may be continued on the next line; no spaces are allowed in the middle of continued parameter fields.
- *Command History.* A command history is kept both in memory for interactive inspection and on a disk file. The command history file can be recovered and used to reconstruct a set of actions carried out interactively.
- *On-Line Assistance.* The "usage" and "help" commands can be used to get a short or verbose description of parameters and features of any command.
- *Aliases.* Allow the abbreviation of partial or complete command sequences.
- *Macros.* A text file containing PAW commands and flow control statements.

KUIP/MOTIF Interface. If the user's workstation supports the X-window MOTIF graphics management system, PAW can be started in the KUIP/MOTIF mode. A small text panel and a command history panel keep track of individual actions and permit entry and recall of typed commands similar to the Text Interface mode. Other basic features of this interface include the following:

- *Pull-Down Menu Commands.* Each PAW command that can be typed as a text command has a corresponding item in a hierarchy of pull-down menus at the top of the MOTIF panel. Commands that require arguments cause a parameter-entry dialog box to appear; when the arguments are entered, the command is executed as though typed from the text interface.
- *Action Panel.* A user may have a family of frequently executed macros or commands assigned to specific buttons on the action panel.

Graphics Output Window. The graphics image produced by PAW commands, regardless of the command interface, appears on a separate graphics output window. The actual size and position of this window on the screen is controlled by a list of numbers of the form `x-upper-left y-upper-left x-width y-height` in the user file `higz_windows.dat`. The width and height of the drawing area within this window are subject to additional user control, and the user can specify "zones," which are essentially ways of dividing the window into panes to allow simultaneous display of more than one plot. Some facilities are available in the current version of PAW to use the mouse to retrieve data such as the height of a histogram bin. Applications currently under development will extend this style of interaction.

1.5 Fundamental Objects of PAW

PAW is implicitly based on a family of fundamental objects. Each PAW command performs an action that either produces another object or produces a "side-effect" such as a printed message or graphics display that is not saved anywhere as a data structure. Some commands do both, and some may or may not produce a PAW data structure depending on the settings of global PAW parameters. In this section, we describe the basic objects that the user needs to keep in mind when dealing with PAW. The reader should perhaps note that the PAW text commands themselves do not necessarily reflect the nature of PAW objects as clearly as they might, while the MOTIF interactive graphics interface currently in development in fact displays distinct icons for most of the object types listed below.

Objects:

- **Ntuples.** An Ntuple is the basic type of data used in PAW. It consists of a list of identical data structures, one for each event. Typically, an Ntuple is made available to PAW by opening a ZEBRA file; this file, as created by HBOOK, contains one or more Ntuples and possibly also ZEBRA logical directories, which may store a hierarchy of Ntuples. A storage area for an Ntuple may be created directly using `ntuple/create`; data may then be stored in the allocated space using the `ntuple/loop` or `ntuple/read` commands. Other commands merge Ntuples into larger Ntuples, project vector functions of the Ntuple variables into histograms, and plot selected subsets of events.
- **Cuts.** A cut is a Boolean function of Ntuple variables. Cuts are used to select subsets of events in an Ntuple when creating histograms and plotting variables.
- **Masks.** Masks are separate files that are logically identical to a set of boolean variables added on the end of an Ntuple's data structure. A mask is constructed using the Boolean result of applying a cut to an event set. A mask is useful only for efficiency; the effect of a mask is identical to that of the cut that produced it.
- **1D Histograms.** A histogram is the basic statistical analysis tool of PAW. Histograms are created (“booked”) by choosing the basic characteristics of their bins, variables, and perhaps customized display parameters; numbers are entered into the histogram bins from an Ntuple (the histogram is “filled”) by selecting the desired events, weights, and variable transformations to be used while counts are accumulated in the bins. Functional forms are frequently fit to the resulting histograms and stored with them. Thus a fit as an object is normally associated directly with a histogram, although it may be considered separately.
- **2D Histograms.** 2D (and higher-dimensional) histograms are logical generalizations of 1D histograms. 2D histograms, for example, are viewable as the result of counting the points in the sections of a rectangular grid overlaid on a scatter plot of two variables. Higher-dimensional histograms can also be fitted, and support for associating the results of a fit to a higher-dimensional histogram is currently being incorporated in PAW.
- **Styles.** A “style” is a set of variables that control the appearance of PAW plots. Commands of the form `igset parameter value` determine fundamental characteristics of lines, axis format, text, and so on. Commands of the form `option attribute` choose particular plotting options such as logarithmic/linear, bar-chart/scatter-plot, and statistics display. Commands of the form `set parameter value` control a vast set of numerical format parameters used to control plotting. While the “style” object will eventually become a formal part of PAW, a “style” can be constructed by the user in the form of a macro file that resets all parameters back to their defaults and then sets the desired customizations.
- **Metafile.** In normal interactive usage, images created on the screen correspond to no persistent data structure. If one wishes to create a savable graphics object, the user establishes a *metafile*; as a graphics image is being drawn, each command is then saved in a text file in coded form that allows the image to be duplicated by other systems. PostScript format metafiles are especially useful because they can be directly printed on most printers; furthermore, the printed quality of graphics objects such as fonts can be of much higher quality than the original screen image.

- **Pictures.** Metafiles describing very complex graphics objects can be extremely lengthy, and therefore inefficient in terms of storage and the time needed to redraw the image. A *picture* is an exact copy of the screen image, and so its storage and redisplay time are independent of complexity. On the other hand, a printed picture object will never be of higher quality than the original screen image.
- **ZEBRA(RZ) Logical Directories.** In a single PAW session, the user may work simultaneously with many Ntuples, histograms, and hierarchies of Ntuple and histograms. However, this is not accomplished using the native operating system's file handler. Instead, the user works with a set of objects that are *similar* to a file system, but are instead managed by the ZEBRA RZ package. This can be somewhat confusing because a single operating system file created by RZ can contain an entire hierarchy of ZEBRA logical directories; furthermore, sections of internal memory can also be organized as ZEBRA logical directories to receive newly-created PAW objects that are not written to files. A set of commands CDIR, LDIR, and MDIR are the basic utilities for walking through a set of ZEBRA logical directories of PAW objects; Each set of directories contained in an actual file corresponds to a logical unit number, and the root of the tree is usually of the form //LUNx; the PAW objects and logical directories stored in internal memory have the root //PAWC.
- **Operating System File Directories.** Many different ZEBRA files, some with logically equivalent Ntuples and histograms, can be arranged in the user's operating system file directories. Thus one must also keep clearly in mind the operating system file directories and their correspondence to the ZEBRA logical directories containing data that one wishes to work with. In many ways, the operating system file system is also a type of "object" that forms an essential part of the user's mental picture of the system.

1.6 The Component Subsystems of PAW

The PAW system combines different tools and packages, which can also be used independently and some of which have already a long history behind them (e.g. HBOOK and HPLOT, SIGMA, Minuit). Figure 1.1 shows the various components of PAW.

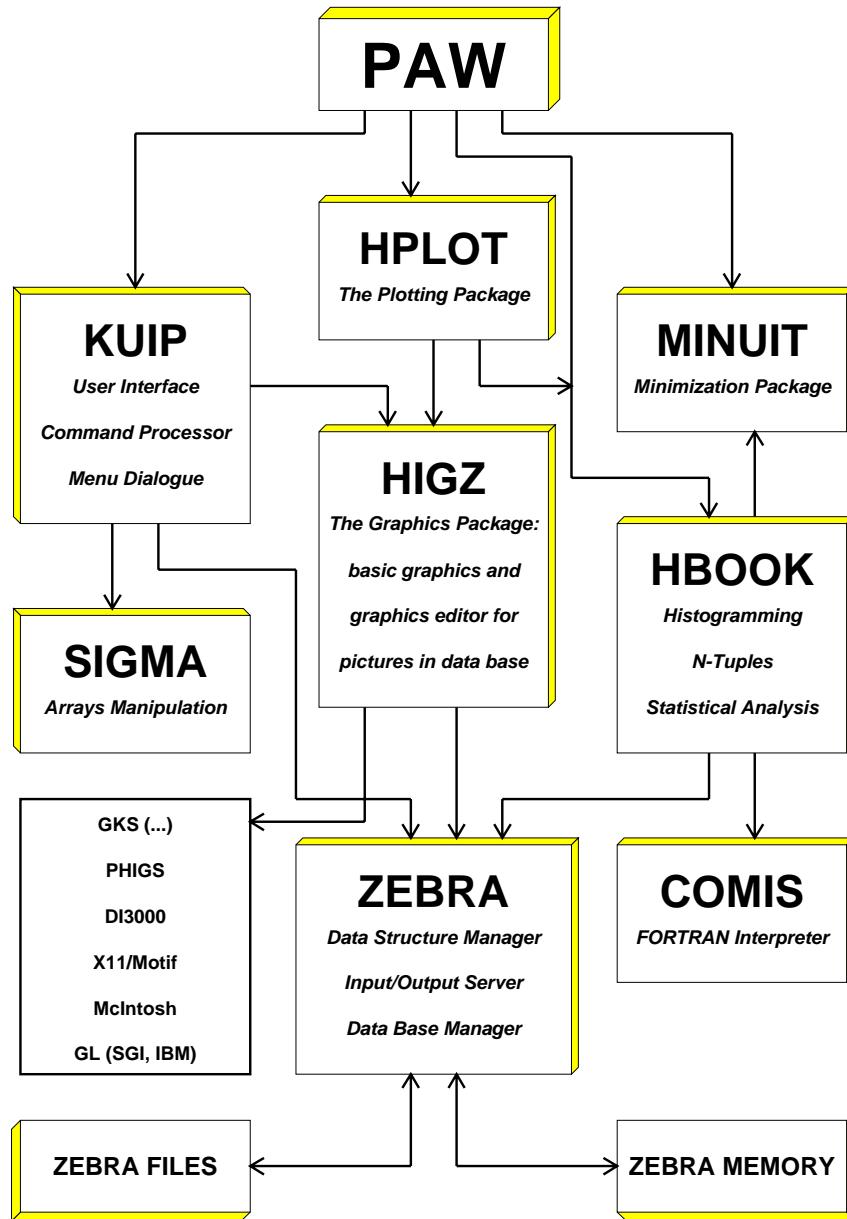


Figure 1.1: PAW and its components

1.6.1 KUIP - The user interface package

The purpose of KUIP (Kit for a User Interface Package) is to handle the dialogue between the user and the application program (PAW in our case). It parses the commands input into the system, verifies them for correctness and then hands over control to the relevant action routines.

The syntax for the commands accepted by KUIP is specified using a **Command Definition File** (CDF) and the information provided is stored in a ZEBRA data structure, which is accessed not only during the parsing stage of the command but also when the user invokes the **online help** command. Commands are grouped in a tree structure and they can be **abbreviated** to their shortest unambiguous form. If an ambiguous command is typed, then KUIP responds by showing all the possibilities. **Aliases** allow the user to abbreviate part or the whole of commonly used command and parameters. A sequence of PAW commands can be stored in a text file and, combined with flow control statements, form a powerful **macro** facility. With the help of **parameters**, whose values can be passed to the macros, general and adaptable task solving procedures can be developed.

Different **styles of dialogue** (command and various menu modes) are available and the user can switch between them at any time. In order to save typing, **default values**, providing reasonable settings, can be used for most parameters of a command. A **history file**, containing the n most recently entered commands, is automatically kept by KUIP and can be inspected, copied or re-entered at any time. The history file of the last PAW session is also kept on disk.

1.6.2 HBOOK and HPLOT - The histogramming and plotting packages

HBOOK and its graphics interface HPLOT are libraries of FORTRAN callable subroutines which have been in use for many years. They provide the following functionality:

- One- and two-dimensional histograms and Ntuples
- Projections and slices of two-dimensional histograms and Ntuples
- Complete control (input and output) of the histogram contents
- Operations and comparison of histograms
- Minimization and parameterization tools
- Random number generation
- Histograms and Ntuples structured in memory (directories)
- Histograms and Ntuples saved onto direct access ZEBRA files
- Wide range of graphics options:
 - Normal contour histograms, bar chart, shaded histograms, error bars, colour
 - Smoothed curves and surfaces
 - Scatter, lego, contour and surface plots
 - Automatic windowing
 - Graphics input

1.6.3 HIGZ - The graphics interface package

A **High level Interface to Graphics and ZEBRA** (HIGZ) has been developed within the PAW project. This package is a layer between the application program (e.g. PAW) and the basic graphics package (e.g. GKS) on a given system. Its basic aims are:

- Full transportability of the picture data base.
- Easy manipulation of the picture elements.
- Compactness of the data to be transported and accessibility of the pictures in direct access mode.
- Independence of the underlying basic graphics package. Presently HIGZ is interfaced with several GKS packages, X windows, GL (Silicon Graphics), GDDM (IBM), GPR and GMR3D (Apollo) as well as with the DI3000 system.

These requirements have been incorporated into HIGZ by exploiting the data management system ZEBRA.

The implementation of HIGZ was deliberately chosen to be close to GKS. HIGZ does not introduce new basic graphics features, but introduces some macroprimitives for frequently used functions (e.g. arcs, axes, boxes, pie-charts, tables). The system provides the following features:

- Basic graphics functions, interfaced to the local graphics package, but with calling sequences nearly identical to those of GKS.
- Higher-level macroprimitives.
- Data structure management using an interface to the ZEBRA system.
- Interactive picture editing.

These features, which are available simultaneously, are particularly useful during an interactive session, as the user is able to “replay” and edit previously created pictures, without the need to re-run the application program. A direct interface to PostScript is also available.

1.6.4 ZEBRA - The data structure management system

The data structure management package ZEBRA was developed at CERN in order to overcome the lack of dynamic data structure facilities in FORTRAN, the favourite computer language in high energy physics. It implements the **dynamic creation and modification** of data structures at execution time and their transport to and from external media on the same or different computers, memory to memory, to disk or over the network, at an **insignificant cost** in terms of execution-time overheads.

ZEBRA manages any type of structure, but specifically supports linear structures (lists) and trees. ZEBRA input/output is either of a sequential or direct access type. Two data representations, **native** (no data conversion when transferred to/from the external medium) and **exchange** (a conversion to an interchange format is made), allow data to be transported between computers of the same and of different architectures. The direct access package **RZ** can be used to manage hierarchical data bases. In PAW this facility is exploited to store histograms and pictures in a hierarchical direct access directory structure.

1.6.5 MINUIT - Function minimization and error analysis

MINUIT is a tool to find the **minima of a multi-parameter function** and analyse the **shape around the minimum**. It can be used for **statistical analysis** of curve fitting, working on a χ^2 or log-likelihood function, to compute the **best fit** parameter values, their uncertainties and correlations. **Guidance** can be provided in order to find the correct solution, parameters can be kept fixed and data points can be easily added or removed from the fit.

1.6.6 COMIS - The FORTRAN interpreter

The COMIS interpreter allows the user to execute interactively a set of FORTRAN routines in interpretive mode. The interpreter implements a large subset of the complete FORTRAN language. It is an extremely important tool because it allows the user to specify his own complex data analysis procedures, for example selection criteria or a minimisation function.

1.6.7 SIGMA - The array manipulation language

A scientific computing programming language SIGMA (System for Interactive Graphical Mathematical Applications), which was designed essentially for mathematicians and theoretical physicists and has been in use at CERN for over 10 years, has been integrated into PAW. Its main characteristics are:

- The basic data units are scalars and one or more dimensional rectangular arrays, which are automatically handled.
- The computational operators resemble those of FORTRAN.

1.7 A PAW Glossary

Data Analysis Terminology

DST	A “Data Summary Tape” is one basic form of output from a typical physics experiment. A DST is generally not used directly by PAW, but is analyzed by customized user programs to produce Ntuple files, which PAW can read directly.
Ntuple	A list of identical data structures, each typically corresponding to a single experimental event. The data structures themselves frequently consist of a row of numbers, so that many Ntuples may be viewed as two-dimensional arrays of data variables, with one index of the array describing the position of the data structure in the list (i.e., the row or event number), and the other index referring to the position of the data variable in the row (i.e., the column or variable number). A meaningful name is customarily assigned to each column that describes the variable contained in that column for each event. However, the underlying utilities dealing with Ntuples are currently being generalized to allow the name of an element of the data structure to refer not only to a single number, but also to more general data types such as arrays, strings, and so on. Thus it is more general to view an Ntuple as a sequence of tree-structured data, with names assigned to the top-level roots of the tree for each event.
Event	A single instance of a set of data or experimental measurements, usually consisting of a sequence of variables or structures of variables resulting from a partial analysis of the raw data. In PAW applications, one typically examines the statistical characteristics of large sequences of similar events.
Variable	One of a user-defined set of named values associated with a single event in an Ntuple. For example, the (x, y, z) values of a momentum vector could each be variables for a given event. Variables are typically useful experimental quantities that are stored in an Ntuple; they are used in algebraic formulas to define boolean cut criteria or other dependent variables that are relevant to the analysis.
Cut	A boolean-valued function of the variables of a given event. Such functions allow the user to specify that only events meeting certain criteria are to be included in a given distribution.

Mask	A set of columns of zeros and ones that is identical in form to a new set of Ntuple variables. A mask is typically used to save the results of applying a set of cuts to a large set of events so that time-consuming selection computations are not repeated needlessly.
Function	Sequence of one or more statements with a FORTRAN-like syntax entered on the command line or via an external file.

Statistical Analysis Terminology

Histogram	A one- or two-dimensional array of data, generated by HBOOK in batch or in a PAW session. Histograms are (implicitly or explicitly) declared (booked); they can be filled by explicit entry of data or can be derived from other histograms. The information stored with a histogram includes a title, binning and packing definitions, bin contents and errors, statistic values, possibly an associated function vector, and output attributes. Some of these items are optional. The ensemble of this information constitutes an histogram .
Booking	The operation of declaring (creating) an histogram.
Filling	The operation of entering data values into a given histogram.
Fitting	Least squares and maximum likelihood fits of parametric functions to histograms and vectors.
Projection	The operation of projecting two-dimensional distributions onto either or both axes.
Band	A band is a projection onto the X (or Y) axis restricted to an interval along the other Y (or X) axis.
Slice	A slice is a projection onto the X (or Y) axis restricted to one bin along the other Y (or X) axis. Hence a slice is a special case of a band, with the interval limited to one bin.
Weight	PAW allows the user to include a multiplicative statistical bias for each event which is a scalar function of the available variables. This permits the user to correct for known statistical biases in the data when making histograms of event distributions.

KUIP/ZEBRA User Environment Terminology

Macro	A text file containing PAW commands and logical constructs to control the flow of execution. Parameters can be supplied when calling a macro.
Vector	The equivalent of a FORTRAN array supporting up to three dimensions. The elements of a vector can be stored using a real or an integer representation; they can be entered interactively on a terminal or read from an external file.
Logical Directory	The ZEBRA data storage system resembles a file system organized as logical directories. PAW maintains a global variable corresponding to the “current directory” where PAW applications will look for PAW objects such as histograms. The ZEBRA directory structure is a tree, and user functions permit the “current directory” to be set anywhere in the current tree, as well as creating new “directories” where the results of PAW actions can be stored. A special directory called //PAWC corresponds to a memory-resident branch of this virtual file system. ZEBRA files may be written to the operating system file system, but entire hierarchies of ZEBRA directories typically are contained in a single binary operating system file.

Graphics Production Terminology

- GKS** The Graphical Kernel System is ISO standard document ISO 8805. It defines a common interface to interactive computer graphics for application programs.
- Metafile** A file containing graphical information stored in a device independent format, which can be replayed on various types of output devices. (e.g. the GKS Appendix E metafile and PostScript, both used at CERN).
- Picture** A graphics object composed of graphics primitives and attributes. Pictures are generated by the HIGZ graphics interface and they can be stored in a picture direct-access database, built with the RZ-package of the data structure manager ZEBRA.
- PostScript** A high level page description language permitting the description of complex text and graphics using only text commands. Using PostScript representations of graphics makes it possible to create graphics files that can be exchanged with other users and printed on a wide variety of printers without regard to the computer system upon which the graphics were produced. Any graphics display produced by PAW can be expressed in terms of PostScript, written to a file, and printed.

Chapter 2: General principles

2.1 Access to PAW

At CERN the PAW program is interfaced on all systems via a command procedure which gives access to the three release levels of the CERN Program Library (PROduction, OLD and the NEW areas) and sets the proper environment if necessary. Users who are not at CERN or who are using non-central computer systems should contact their system administrator for help on PAW.

2.1.1 IBM/VM-CMS

There are three versions available:

- GKS** For any ASCII graphic terminal capable of emulating Tektronix or PG.
- GDDM** For IBM 3192G graphic terminals or its emulators (e.g. tn3270 on a Mac-II)
- X11** For any X-window display connected to VM

You need a machine size of at least 7 Mb, that may be defined either temporarily for the current session (command DEFINE STORAGE 7M followed by an IPL CMS) or permanently for all subsequent sessions (command DIRM STOR 7M; you need to logoff once to make the definition effective).

An interface Rexx exec file PAW EXEC is located on the Q-disk and has the following interface:

PAW (ver driver

The first parameter **ver** can have the values PRO, NEW and OLD and the second parameter **driver** the values GKS, GDDM or X11. The defaults are: PRO GKS. Help is available via FIND CMS PAW.

2.1.2 VAX/VMS

There are two versions available on VXCERN: GKS and X11. A command file CERN_ROOT:[EXE]PAW.COM is defined system-wide via the logical symbol PAW; its interface is:

PAW/ver/driver

(default is PRO GKS). You may set the initialization of PAW either as a PAWLOGON.KUMAC located in your home directory, or through the logical symbol DEFINE PAW\$LOGON disk:[user subdir]file.kumac to be defined usually in your LOGIN.COM. Help is available via HELP @CERNLIB PAW.

2.1.3 Unix systems

There are three versions available: GKS, GPR and X11. The driver shell script is located in the file /cern/pro/bin/paw. In order to access it automatically you could add the directory /cern/pro/bin to your command search path. The command syntax is:

paw -v ver -d driver

(default is -v PRO -d GKS). In the GKS case this shell script sets the proper GKS environment.

2.1.4 Note on the X11 version

The X11 version needs to know the X-host where graphics must be displayed; this can be specified on each system on the command line:

VM/CMS :	<u>PAW (X11 HOST yourhost</u>
Vax/VMS :	<u>PAW/X11/host=yourhost</u>
Unix :	<u>paw -d X11 -h yourhost</u>

or at the “Workstation” prompt in PAW: Workstation type (?=HELP) [CR]=1 : 1.yourhost

On Vax/VMS the default X-window protocol is TCP/IP. If you want DECNET (e.g. when running from a Vaxstation) add the DECNET option to the command as follows:

PAW/X11/DECNET/host=yourhost

2.1.5 Important modes to run PAW

- A **batch** version of PAW is available (note that batch implies workstation type 0):

On Unix	do : <u>PAW -b macroname</u>
On VMS	do : <u>PAW/BATCH=macroname</u>
On VM	do : <u>PAW (BATCH=macroname</u>

- One can **disable** the automatic execution of the PAWLOGON macro:

On Apollo	do : <u>PAW -n</u>
On VMS	do : <u>PAW/NOLOG</u>
On VM	do : <u>PAW (NOLOG</u>

2.2 Initialising PAW

When PAW is started, a **system** startup procedure is initiated, which indicates the current version of PAW and requests the **workstation type** of the terminal or workstation which you are using.

```
$ PAW
*****
*          *          *
*      W E L C O M E      to      P A W      *
*          *          *
*      Version 1.13/00  9 March 1992      *
*          *          *
*****  
Workstation type (?=HELP) <CR>=10 : ?  
  
List of valid workstation types:  
 0: Alphanumeric terminal  
101: Tektronix 4010, 4014  
102: Tektronix 4012  
103: Tektronix 4014 with enhanced graphics option  
121: Tektronix 4107, 4207, Pericom MX2000  
122: Tektronix 4109  
123: Tektronix 4111  
125: Tektronix 4113  
127: Tektronix 4115, Pericom MX8000  
7800: MG600, MG200
```

```

7878: Falco, Pericom Graph Pac (old Pericom)
1020: VT240
1030: VT340
8601-6: Vaxstation GPX
10002: Apollo DNXXXX monochrome (GPR)
10003-4: Apollo DNXXXX colour (GPR)
9701-8: Apollo DNXXXX (GSR)
32120-9: X-Window

```

```

Metafile workstation types:
-111: HIGZ/PostScript (Portrait)
-112: HIGZ/PostScript (Landscape)
-113: HIGZ/Encapsulated PostScript
-777/8: HIGZ/LaTex

```

Note that if you specify 0, PAW will not open a graphics workstation. This may be appropriate if one wants to use PAW on an alphanumeric terminal.

Before passing control to the user, the system looks for a user-supplied file `pawlogon.kumac` or `PAWLOGON KUMAC` (VM/CMS). The latter can contain commands which the user wants to be executed at PAW startup, e.g. declaration of files, creation of aliases, definition of HPLOT parameters. A simple version of this PAW initialisation file, displaying date and time, can be:

```

mess '*****'
mess '*          *'
mess '* Starting PAW session on '//$/date//' at '//$/time//'      *'
mess '*          *'
mess '*****'

```

In order to only have one version of this file on VAX/VMS the user should define a **logical name** `PAW$LOGON` in his `LOGIN.COM`, as explained on the previous page. On a Unix workstation the file `pawlogon.kumac`, should be put into the directory. On IBM/VM-CMS the minidisk file search rule takes care of finding the file.

2.3 Command structure

PAW is based on the KUIP[5] User Interface package, which can provide different types of dialogue styles:

- Command mode, where the user enters a command line via the terminal keyboard.
- Alphanumeric menu mode, where the command is selected from a list.
- Graphics menu modes:
 - Pull-down menus, fixed layout reflecting the command structure;
 - Panels of function keys, interactive user definable multiple layouts.

It is possible to change interactively from one style to another.

The general format of a PAW command line is:

command parameters

The first part of the **command** has the format:

object/verb

where the **object** is the item on which the action is performed (e.g. HISTOGRAM, VECTOR, NTUPLE) and the **verb** is the action to be performed (e.g. CREATE, DELETE, PLOT). In some cases the object needs to be specified further (e.g. GRAPHICS/PRIMITIVE), while in other cases the verb's action needs to be clarified further (e.g. CREATE/1D). All components can be **abbreviated** to their shortest unambiguous form. For example the two following lines will have the same effect of creating a vector A with nine components:

VECTOR/CREATE A(9)
or
VE/CR A(9)

In the case that the form is ambiguous all possible interpretations for the given abbreviation are displayed. The second part of a command are its **parameters** and their meaning is determined by their **position**. Some of these can be **mandatory** with the remaining ones **optional**. If all mandatory parameters are not provided on the command line, PAW will prompt the user to specify them, indicating the default values if defined. If the user wants to assign the default value to a parameter from the command line he can use the **place-holder** character **exclamation mark** (!) to signify this to PAW. In the case of optional parameters, the user **must** provide them in the correct sequence if he wants to **change** their values, otherwise the corresponding defaults are taken. Parameters containing blanks must be enclosed within single quotes.

In the example below we create a one-dimensional histogram, providing the parameters one by one answering the PAW query:

```
PAW > histogram/create/1dhisto
Histogram Identifier (<CR>= ): 10
Histogram title (<CR>= ): title1
Number of channels (<CR>=100): <CR>
Low edge (<CR>=0): 10.
Upper edge (<CR>=100): 20.
```

On the command below we provide all parameters on the command line, including an optional one (1000.), which by default has the value 0. Note that this parameter **must** be specified explicitly, since PAW **does not** prompt for it, as seen in the previous example. Note also the use of the exclamation mark to take the default for the number of channels (100).

```
PAW > hi/cr/1d 20 title2 ! 10. 20. 1000.
```

2.4 Getting help

Once inside PAW, one can start entering commands. An interesting first try would be the HELP command, which displays a list of items, preceded by a number and followed by one line of explanation. In the next example we search for a command to create a one-dimensional histogram.

```
PAW > help
From / ...
1: KUIP Command Processor commands.
2: MACRO Macro Processor commands.
3: VECTOR Vector Processor commands.
4: HISTOGRAM Manipulation of histograms, Ntuples.
```

```

5: FUNCTION      Operations with Functions. Creation and plotting.
6: NTUPLE        Ntuple creation and related operations.
7: GRAPHICS     Interface to the graphics packages HPLOT and HIGZ.
8: PICTURE       Creation and manipulation of HIGZ pictures.
9: ZEBRA         Interfaces to the ZEBRA RZ, FZ and DZ packages.
10: FORTRAN      Interface to the COMIS FORTRAN interpreter.
11: NETWORK       To access files on remote computers.

```

Enter a number ('\'=one level back, 'Q'=command mode): 4

/HISTOGRAM

Manipulation of histograms, Ntuples.
Interface to the HBOOK package.

From /HISTOGRAM/...

```

1: * FILE          Open an HBOOK direct access file.
2: * LIST          List histograms and Ntuples in the current directory.
3: * DELETE        Delete histogram/Ntuple ID in Current Directory (memory).
4: * PLOT          Plot a single histogram or a 2-Dim projection.
5: * ZOOM          Plot a single histogram between channels ICMIN and ICMAX.
6: * MANY_PLOTS    Plot one or several histograms into the same plot.
7: * PROJECT       Fill all booked projections of a 2-Dim histogram.
8: * COPY          Copy a histogram (not Ntuple) onto another one.
9: * FIT           Fit a user defined (and parameter dependent) function
10: 2D_PLOT        Plotting of 2-Dim histograms in various formats.
11: CREATE         Creation ("booking") of HBOOK objects in memory.
12: HIO            Input/Output operations of histograms.
13: OPERATIONS    Histogram operations and comparisons.
14: GET_VECT       Fill a vector from values stored in HBOOK objects.
15: PUT_VECT       Replace histogram contents with values in a vector.
16: SET            Set histogram attributes.

```

Enter a number ('\'=one level back, 'Q'=command mode): 11

/HISTOGRAM/CREATE

Creation ("booking") of new HBOOK objects.

From /HISTOGRAM/CREATE/...

```

1: * 1DHISTO       Create a one dimensional histogram.
2: * PROFILE        Create a profile histogram.
3: * BINS          Create a histogram with variable size bins.
4: * 2DHISTO        Create a two dimensional histogram.
5: * PROX          Create the projection onto the x axis.
6: * PROY          Create the projection onto the y axis.
7: * SLIX          Create projections onto the x axis, in y-slices.
8: * SLIY          Create projections onto the y axis, in x-slices.
9: * BANY          Create a projection onto the x axis, in a band of y.
10: * BANY          Create a projection onto the y axis, in a band of x.
11: * TITLE_GLOBAL  Set the global title.

```

Enter a number ('\'=one level back, 'Q'=command mode): 1

* /HISTOGRAM/CREATE/1DHISTO ID TITLE NCX XMIN XMAX [VALMAX]

ID	C 'Histogram Identifier'
TITLE	C 'Histogram title' D=' '
NCX	I 'Number of channels' D=100
XMIN	R 'Low edge' D=0
XMAX	R 'Upper edge' D=100
VALMAX	R 'Valmax' D=0

Creates a one dimensional histogram. The contents are set to zero.

If VALMAX=0, a full word is allocated per channel, else VALMAX is used as the maximum bin content allowing several channels to be stored into the same machine word.

The meaning of the notation used in the text displayed by the HELP command is explained on page III. Moreover an item preceded by a **star** indicates a **terminal leaf** in the command tree, i.e. an **executable** command (see on Page 91 for more details).

One can also inquire about **creating a one-dimensional histogram** by typing simply:

HELP histogram/create/1dhisto
or
HELP his/cre/1d
or even
HELP 1

The system will then display the following information:

```
* /HISTOGRAM/CREATE/1DHISTO ID TITLE NCX XMIN XMAX [ VALMAX ]  
  
ID      C 'Histogram Identifier'  
TITLE   C 'Histogram title' D=' '  
NCX    I 'Number of channels' D=100  
XMIN   R 'Low edge' D=0  
XMAX   R 'Upper edge' D=100  
VALMAX  R 'Valmax' D=0  
  
Creates a one dimensional histogram. The contents are set to zero.  
If VALMAX=0, a full word is allocated per channel, else VALMAX is used as the maximum  
bin content allowing several channels to be stored into the same machine word.
```

2.4.1 Usage

Very often a single line description of the usage of a command is sufficient as a reminder. This can be obtained by the USAGE command, e.g.:

```
PAW > USAGE 1d  
  
* /HISTOGRAM/CREATE/1DHISTO ID TITLE NCX XMIN XMAX [ VALMAX ]
```

2.5 Special symbols for PAW

One should pay attention to the fact that, in addition to their common arithmetic meaning, the symbols in table 2.1 have a special connotation when working with PAW .

2.6 PAW entities and their related commands

Relations which exist between various PAW entities as described in section 1.6 on page 9 and the operations which can be performed upon them have been schematically represented in figure 2.1. All commands shown in the picture next to the lines connecting the objects have been abbreviated in a way that they are unambiguous and can be typed to PAW, which will then detail the various parameters to be supplied.

There are three main input/output formats, namely a simple text file (e.g. with data points or commands), a direct access ZEBRA RZ file (used by HBOOK and HIGZ for storing histograms and pictures on a

Symbol	Meaning
blank	Separator between command and parameter and between different parameters
/	Separator between command elements
	Comment line (if first character of the command line)
	Inline comments
,	String delimiter
-	Line continuation in KUIP commands
@	Escape character to be put in front of and ' to interpret them as literal
!	Place-holder for command parameter (i.e. default value is taken)
	At beginning of command line: Unix C shell-like history (e.g. !!, !number, !-number, !string)
[]	Macro argument delimiters
#	Separator between macro file and macro member
()	Vector subscript delimiters
:	Vector subscript range
,	Multi-dimensional vector subscript dimensions delimiter

Note: These special characters loose their effect when imbedded in single quotes.

Table 2.1: Special symbols

given machine) and a ZEBRA FZ sequential file, which can be used to transfer structured ZEBRA data between various computers. The RZ and FZ representations can be transformed into each other using the TOALFA and FRALFA commands.

The three main PAW objects, Ntuples, histograms and vectors, can be **printed** on an alphanumeric screen (PRINT commands) or they can be plotted on a graphics screen (PLOT commands). The picture can be transformed into a ZEBRA data structure and stored in a HIGZ database for later reference (e.g. editing by the HIGZ editor), or an external presentation can be obtained via the creation of a **metafile**. This “metafile” can for instance consist of GKS or PostScript commands, which can then be interpreted by the relative drivers and printed on an output device, if so desired.

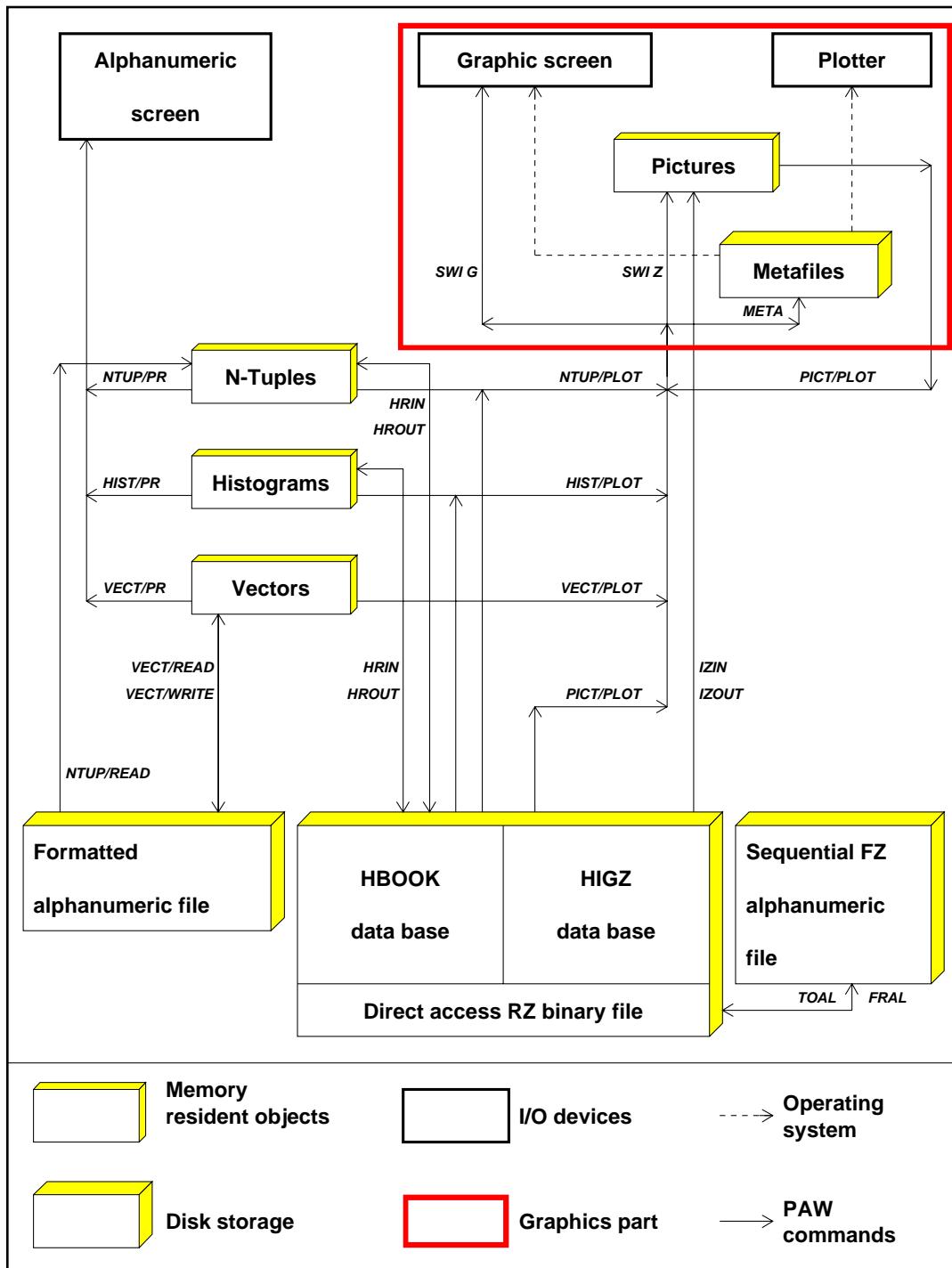


Figure 2.1: PAW entities and their related commands

Chapter 3: PAW by examples

This chapter shows how to use the basic functionality of the PAW system with the help of a series of simple examples. It is advisable to step through the complete series since most entries use information from examples upstream.

The so called **command mode**, available on all systems, is used throughout this chapter. For more details on other possible dialog modes, see Part 2.

List of examples in this chapter

- 1** Simple vector commands
- 2** Further vector commands and writing vectors to disk
- 3** Various data representations
- 4** Difference between VECTOR/DRAW and VECTOR/PLOT
- 5** Vector operations
- 6** Simple macro with loop and vector fit
- 7** Plotting one-dimensional functions
- 8** One-dimensional functions and loops
- 9** Plotting two-dimensional functions
- 10** Plotting a two-dimensional function specified in a external file
- 11** Creation of one- and two-dimensional histograms
- 12** Reading histograms on an external file
- 13** One-dimensional plotting and histogram operations
- 14** Two-dimensional data representations
- 15** The use of sub-ranges in histogram specifiers
- 16** Ntuples - Creation and output to a file
- 17** Ntuples - Automatic and user binning
- 18** Ntuples - A first look at selection criteria
- 19** Ntuples - Masks and loop
- 20** Ntuples - Using cuts
- 21** Ntuples - Two dimensional data representation
- 22** Using the SIGMA processor - Trigonometric functions
- 23** Using the SIGMA processor - More complex examples
- 24** Histogram operations (Keep and Update)
- 25** Merging several pictures into one plot
- 26** Pie charts with hatch styles and PostScript colour simulation
- 27** A complex graph with PAW
- 28** Making slides with PAW using PostScript

Notes

1. The files needed for these examples are available in the PAW account on the various machines (see also section 2.1).

VM/CMS PAWEX* KUMAC on the PAW 201 minidisk.
All text files must be created with RECFM=F,LRECL=80.

VAX/VMS DISK\$DL:[PAW.PAWMANUAL]PAWEX*.KUMAC

Apollo /user/paw/pawmanual/pawex*.kumac

2. The pictures shown in the present chapter have been produced using the HIGZ/PostScript metafile -113. What is actually displayed on the screen when running a given example, might be slightly different, depending on the workstation type specified at PAW initialisation. For example the fill area style index -3 is frequently used (SET HTYP -3). It displays a grey shadowing on the pictures, but will look different on the screen. The same remark applies to line-widths and -styles.

3.1 Vectors and elementary operations

The aim of the present section is to introduce the basic syntax of PAW. It assumes that the user has already succeeded to login to the PAW system and that PAW is waiting for input.

Simple vector commands

```

PAW > ****
PAW > *          TUTORIAL EXAMPLE PAWEX1      *
PAW > * Some simple vector commands           *
PAW > * Lines starting with a * are comments and are ignored by PAW      *
PAW > ****
PAW > *
PAW >       | similarly everything following a VERTICAL BAR is ignored
PAW >       | so this character can be used to provide INLINE COMMENTS
PAW > *
PAW > vector/create VECT1(10)           | Create vector VECT1 with 10 elements
PAW > vector/input VECT1 10 8 6 4 2 3 5 7 9 11 | Input values of the 10 elements
PAW >       | **** PAW commands are NOT CASE SENSITIVE
PAW > * The underscore _ is the continuation character
PAW > VECTOR/CREATE VX(20) R 1. 2. 3. 4. 5. 6. _
PAW > 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20.
PAW > VECTOR/create VY(20) R 1.1 3.2 5.3 7.4 7.5 _
PAW > 6.6 4.3 2.1 6.6 11.1 16.2 18.3 19.0 17.8 16.0 12.1 9.1 6.1 3.1 6.6
PAW > zone 1 2                         | 2 pictures on the same page
PAW > set HTYP -3                      | set hatch style for histogram
PAW > vector/draw VECT1                | Draw contents of vector VECT1
PAW > GRAPH 20 VX VY                  | Graph VX and VY (defaults)
PAW > igset mtyp 21                    | Set HIGZ polymarker
PAW > GRAPH 20 VX VY P                | Graph VX and VY with polymarker
PAW > set DMOD 2                      | Change line style
PAW > GRAPH 20 VX VY C                | Graph VX and VY (using splines)
PAW > set DMOD 0                      | Reset line style
PAW > vector/delete *                 | Delete vectors
PAW > zone                           | Reset picture layout

```

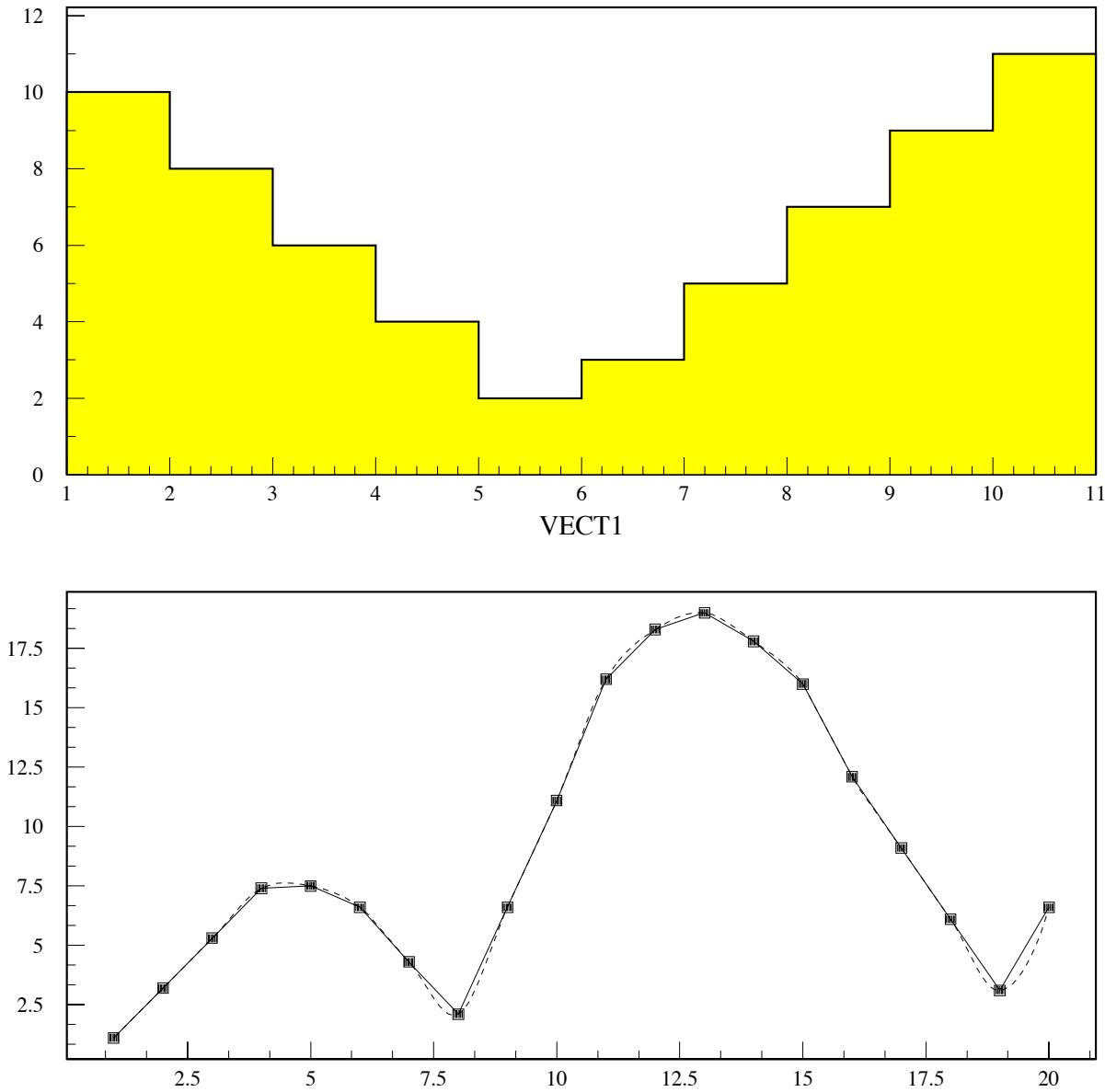


Figure 3.1: Simple vector commands

Further vector commands

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX2 *
PAW > * Example showing further vector commands *
PAW > ****
PAW > size 20 18 _ | Set picture size
PAW > vector/create VECT(10,3) R _ | Create a 2 dimensional vector VECT
PAW > 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. _
PAW > 9.1 8.1 7.1 6.1 5.1 4.1 3.1 2.1 1.1 0.1 _
PAW > 6.2 4.2 3.2 2.2 1.2 1.2 2.2 3.2 4.2 5.2
PAW > VECTOR/CREATE VECT1(10) R _ | Create a one-dimensional vector VECT1
PAW > 1.1 2.2 3.3 4.4 5.5 6.6 5.5 4.4 3.3 2.2 | Input the values of the 10 elements
PAW > *
PAW > * PAW commands can be ABBREVIATED to their shortest non-ambiguous form
PAW > *
PAW > set htyp -3 | Define hatch style
PAW > ve/dr VECT(1:10,3) | Draw contents of third row on VECT
PAW > set htyp 0 | Reset hatch style
PAW > set hwid 12 | Define histogram line width
PAW > *
PAW > * Draw third row of VECT once more, now as a continuous curve
PAW > *
PAW > ve/dr VECT(1:10,3) ! SC | Set line style
PAW > set dmod 13 | Define marker type
PAW > igset MTYP 29 | Marker scale factor
PAW > igset MSCF 3. |
PAW > set HWID 6 | Redefine histogram line width
PAW > *
PAW > * Draw contents of VECT1 as line with marker
PAW > *
PAW > vector/draw VECT1 ! LPS
PAW > *
PAW > * Notice the use of the EXCLAMATION MARK ! as a placeholder.
PAW > * It indicates that the default should be taken for the omitted parameter.
PAW > *
PAW > ve/list | List vectors in order of creation

      Vector Name          Type    Length   Dim-1   Dim-2   Dim-3
      VECT                  R       30       10       3
      VECT1                 R       10       10

Total of 2 Vector(s)

PAW > ve/write VECT 'vector.data' '(3(10f5.0, /))' | Write VECT to text output file
PAW > ve/delete * | Delete all vectors in memory

```

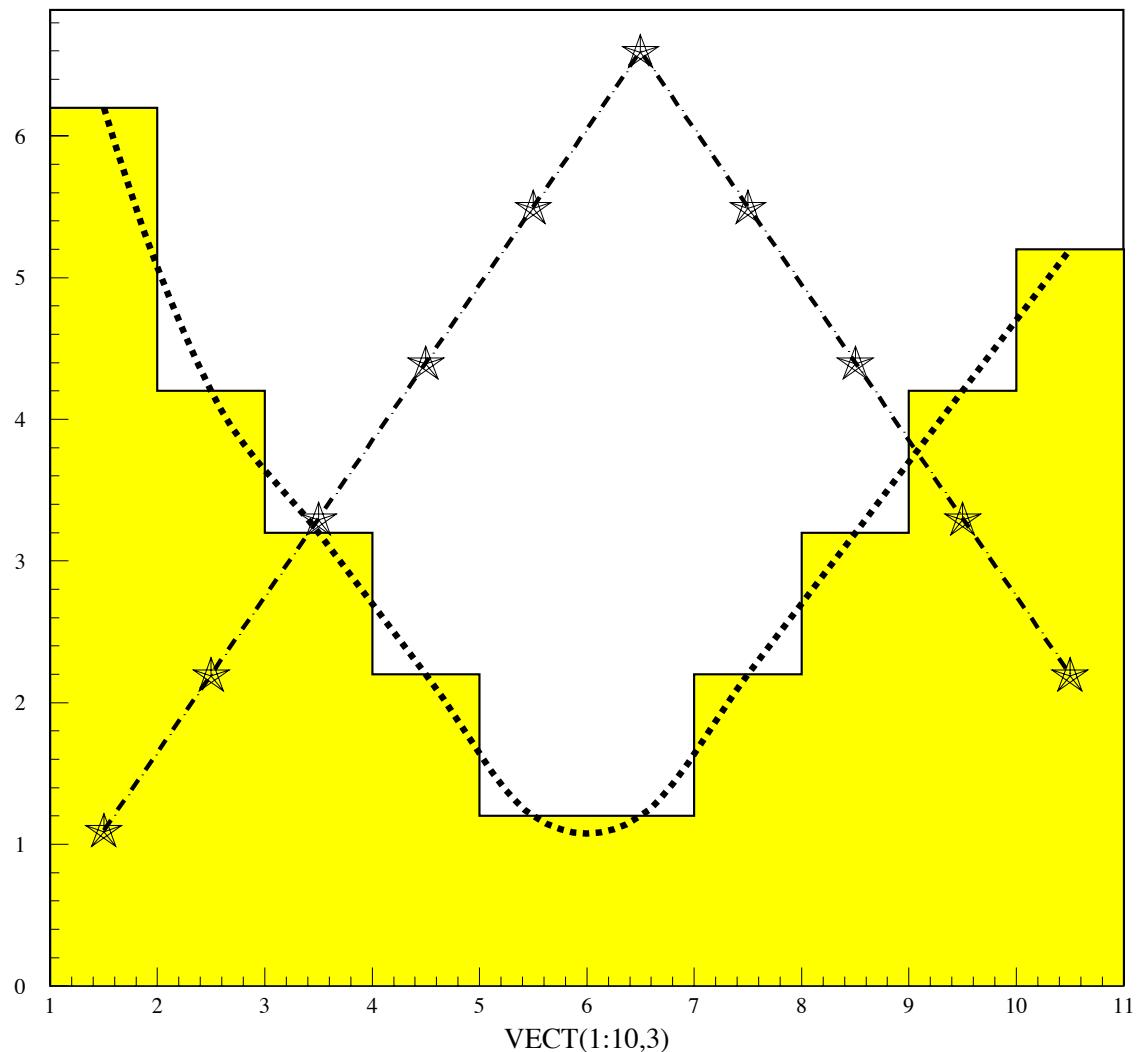


Figure 3.2: Further vector commands and writing vectors to disk

Vector draw data presentations

```
PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX3 *
PAW > * Example showing various possible data representation with VECTOR/DRAW *
PAW > ****
PAW > *
PAW > * Divide picture page in 2 by 3 images
PAW > *
PAW > zone 2 3
PAW > ve/create v(10) R 5 1 3 2 4 1 3 1 8 6 | Create vector V
PAW > set htyp -3 | Define hatch style
PAW > ve/draw v | Default plot
PAW > ve/draw v ! b | Plot as bar chart
PAW > ve/draw v ! l | Plot as lines
PAW > ve/draw v ! l* | Lines and a star
PAW > ve/draw v ! bl* | and bar option
PAW > igset mtyp 21 | Choose HIGZ marker
PAW > ve/draw v ! e | Plot error bars
PAW > ve/de V | Delete vector V
PAW > zone | Reset zone to default
```

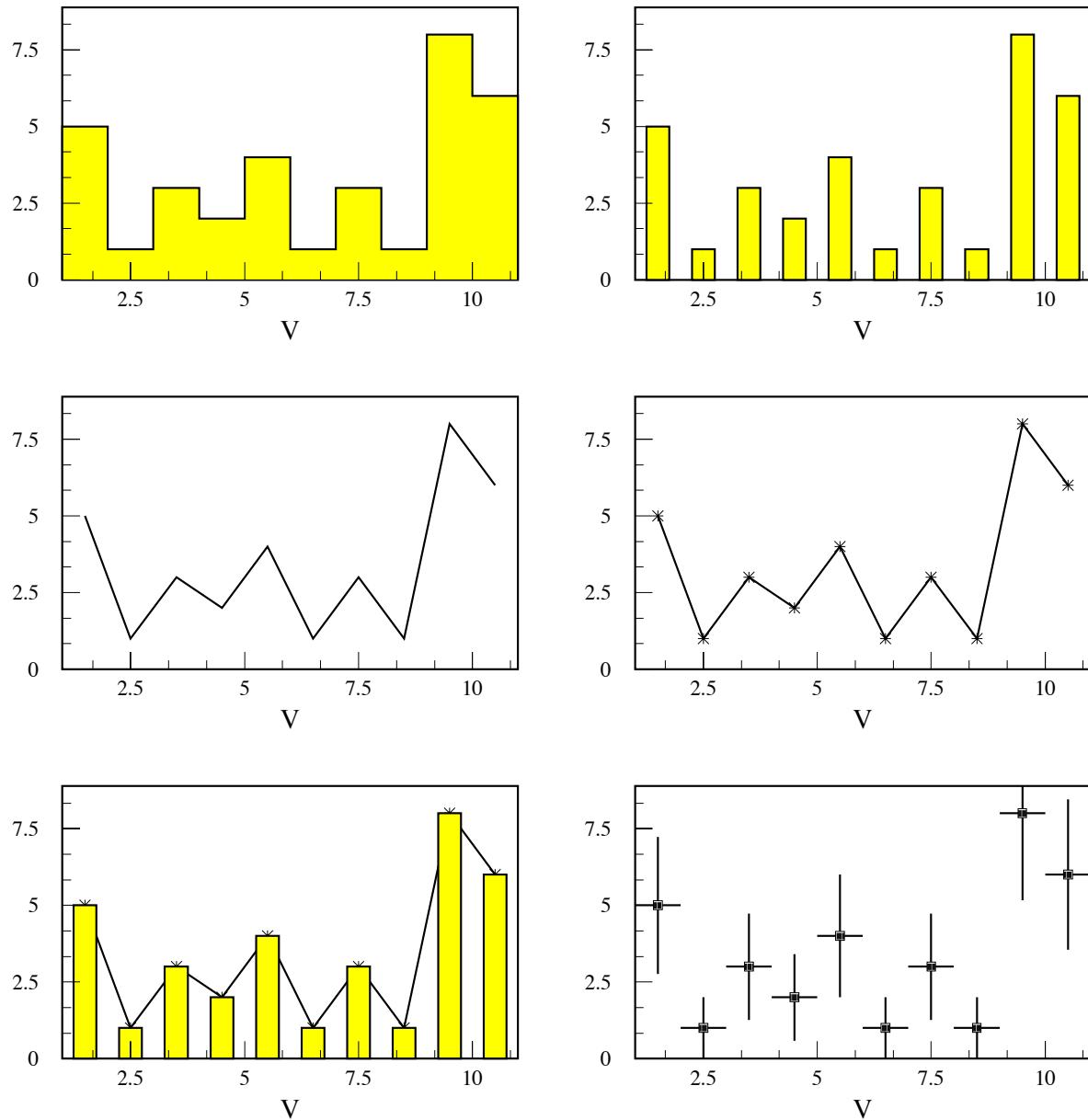


Figure 3.3: Various data representations

Difference between VECTOR/DRAW and VECTOR/PLOT

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX4 *
PAW > * Example which shows the difference between VECTOR/DRAW and VECTOR/PLOT *
PAW > * It also shows the functionality of VECTOR/HFILL and PUT/CONTENTS *
PAW > ****
PAW > zone 2 2
PAW > ve/create VECT1(10) R 1 2 3 4 5 5 4 3 2 1 | Create vector VECT1
PAW > set HTYP -3 | Set hatch type dotted (PostScript)
PAW > ve/draw VECT1 | Draw the contents of VECT1
PAW > *
PAW > * PLOT interprets contents of vector elements as values to be histogrammed
PAW > *
PAW > set HCOL 1001 | Black bars
PAW > ve/plot VECT1 | PLOT contents into hist with 100 channels
PAW > set HTYP -3 | Set hatch type dotted (PostScript)
PAW > create/1dhisto 100 'test vector/hfill' 5 1. 6. | Create a 1-dimensional histogram
PAW > MAX 100 2.5 | Define maximum for hist 100
PAW > ve/hfill VECT1 100 | Fill hist 100 with vector values
PAW > histo/plot 100 B | Plot hist 100 (Bar option)
PAW > hi/de 100 | Delete hist 100
PAW > create/1dhisto 100 'test put/contents' 10 1. 11. | Create a 1-dimensional histogram
PAW > MAX 100 5.5 | Define maximum for hist 100
PAW > MIN 100 0.5 | Define minimum for hist 100
PAW > put/contents 100 VECT1 | Fill contents of hist 100 with VECT1
PAW > histo/plot 100 | Plot hist 100 (identical to VECTOR/DRAW)
PAW > ve/de VECT1 | Delete VECT1
PAW > hi/de 100 | Delete hist 100
PAW > zone | Reset zone

```

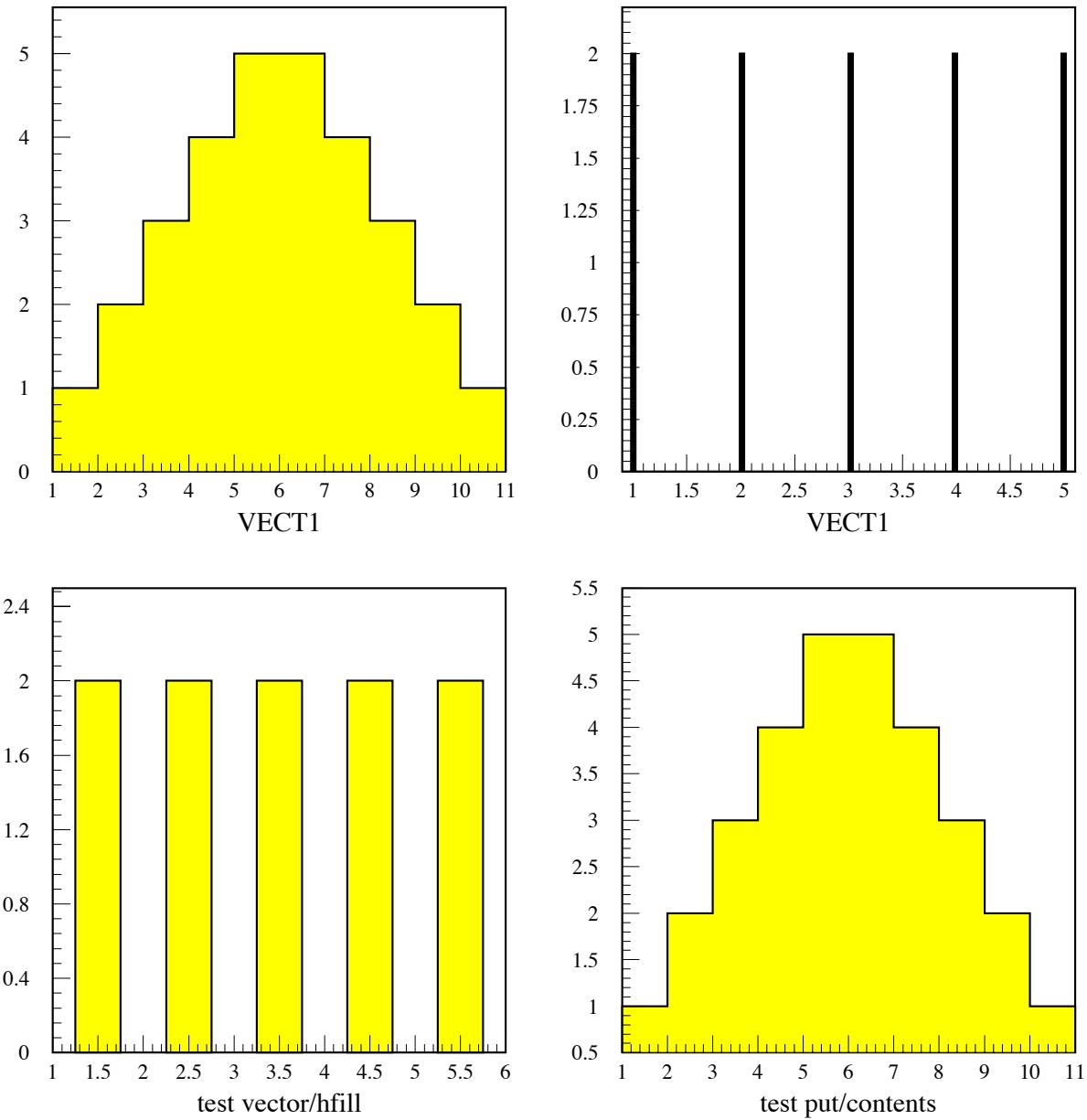


Figure 3.4: Difference between `VECTOR/DRAW` and `VECTOR/PL0T`

Operations on vectors

```
PAW > ****
PAW > *          TUTORIAL EXAMPLE PAWEX5      *
PAW > * Example showing vector operations. The resulting vectors are created  *
PAW > * automatically.                                *
PAW > ****
PAW > zone 1 2
PAW > ve/create V1(10) R 1 2 3 4 5 5 4 3 2 1           | Define vector V1 of 10 elements
PAW > vector/operations/vscale V1 0.5    V12           | Divide V1 by 2 into V12
PAW > ve/op/vscale           V1 0.25   V14           | Divide V1 by 4 into V14
PAW > set htyp 0                                     | No hatch
PAW > *
PAW > * Differences between various plots are shown using different line styles
PAW > *
PAW > ve/dr V1                                     | Draw V1
PAW > ve/dr V12 ! S                                | Draw V12 onto same graph
PAW > ve/dr V14 ! S                                | ditto for V14
PAW > vsub           V1 V14 V14M                  | Subtract V14 from V1 into V14M
PAW > *
PAW > * Differences between various plots are shown using different hatch styles
PAW > *
PAW > ve/dr V1                                     | Draw V1
PAW > set htyp 344                                | Change hatch style
PAW > ve/dr V14M ! S                            | Draw V14M on same graph
PAW > set htyp 144                                | Change hatch style once more
PAW > ve/dr V12 ! S                                | Draw V12 on same graph
PAW > ve/de *                                    | Delete all vectors in memory
PAW > zone                                      | Reset zone
```

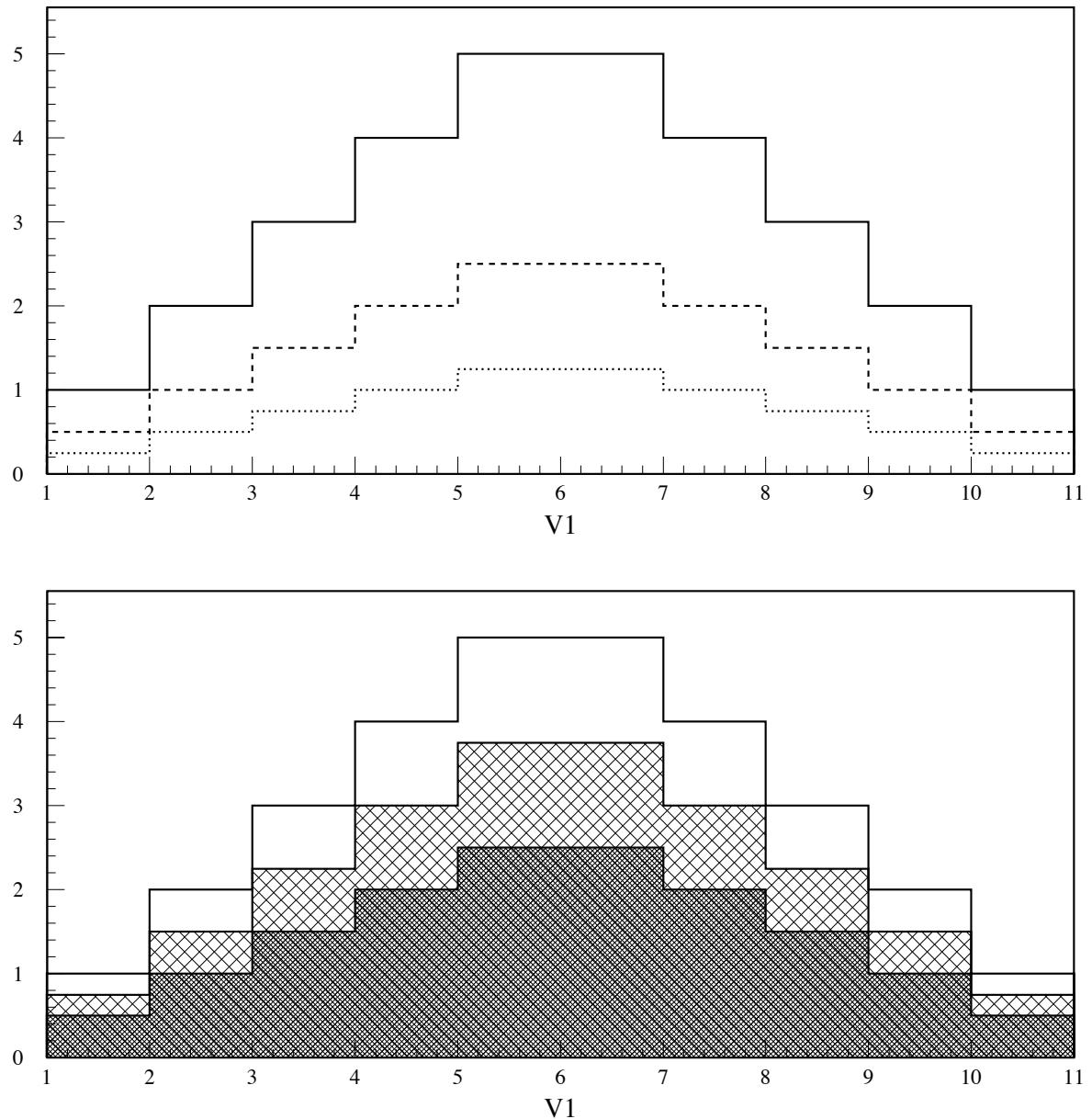


Figure 3.5: Vector operations

Using loops and fitting

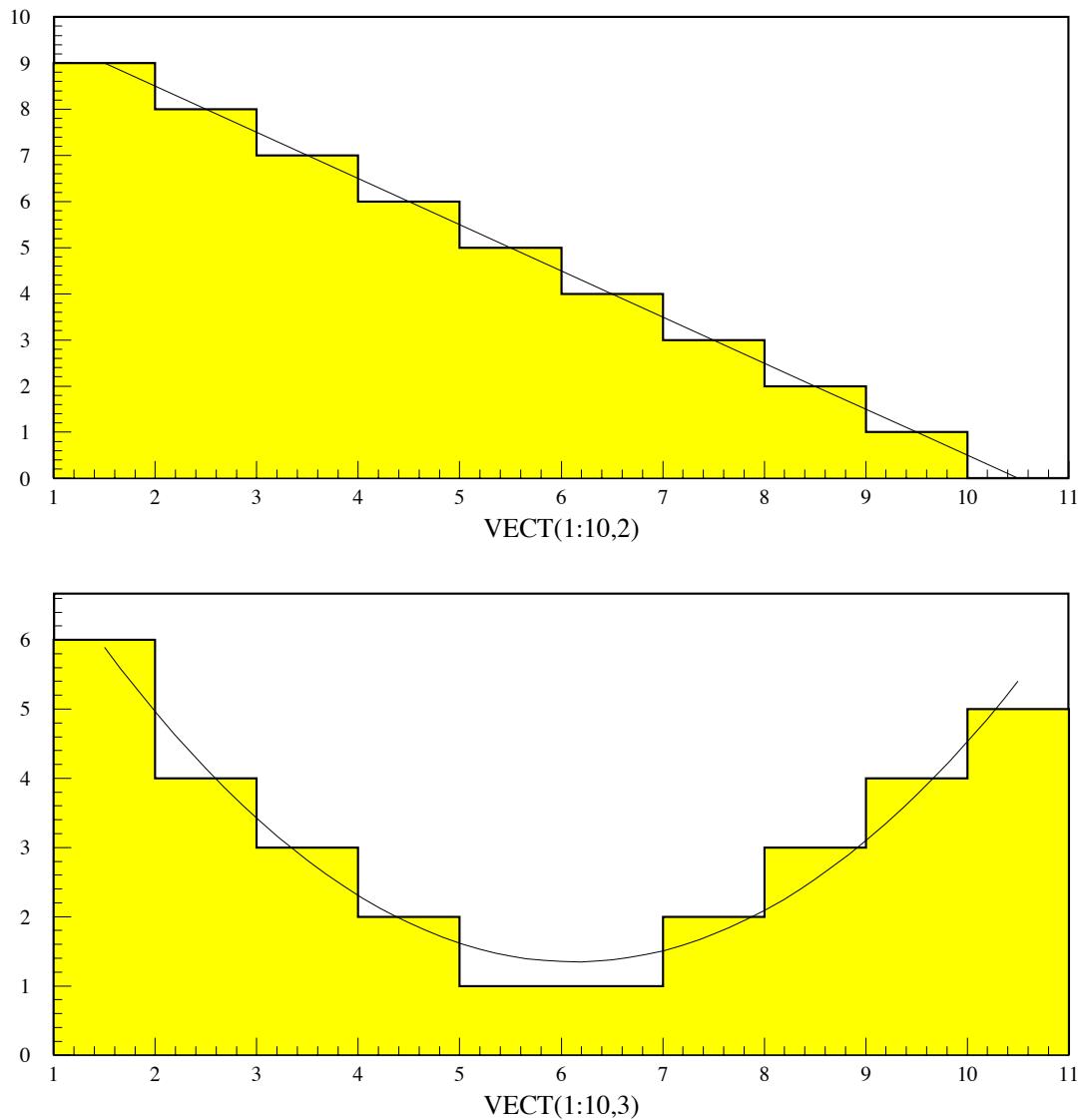


Figure 3.6: Simple macro with loop and vector fit

Output generated by previous macro

```

PAW > TRACE                                | Trace the macro execution
PAW > EXEC PAWEX6                         | Execute Macro PAWEX6.KUMAC
>>>> ve/create VECT(10,3)
>>>> ve/read VECT 'vector.data'
>>>> ve/print VECT(1:10,3)
VECT ( 1 ) = 6.000000
VECT ( 2 ) = 4.000000
VECT ( 3 ) = 3.000000
VECT ( 4 ) = 2.000000
VECT ( 5 ) = 1.000000
VECT ( 6 ) = 1.000000
VECT ( 7 ) = 2.000000
VECT ( 8 ) = 3.000000
VECT ( 9 ) = 4.000000
VECT ( 10 ) = 5.000000
>>>> vbias vect(1:10,1) 0.5 vect(1:10,1)
>>>> zone 1 2
>>>> IP=2
>>>> set htyp -3
>>>> ve/draw VECT(1:10,2)
>>>> ORDER=1
>>>> ve/fit VECT(1:10,1) VECT(1:10,2) ! P1 WS
MINUIT RELEASE 89.05b INITIALIZED.  DIMENSIONS 100/ 35  EPSMAC= 0.11E-18
*****
*** 1 ***SET EPS      0.10000E-05
*****
FLOATING-POINT NUMBERS ASSUMED ACCURATE TO 0.100E-05
*****
*
* Function minimization by SUBROUTINE HFITV *
* Variable-metric method *
* ID =          0  CHOPT = WWS *
*
*****
Convergence when estimated distance to minimum (EDM) .LT. 0.10E-03

FCN= 0.0000000    FROM MIGRAD     STATUS=CONVERGED  CALLS= 18 EDM= 0.88E-11
           STRATEGY= 1    ERROR DEF= 1.0000

INT EXT  PARAMETER                      STEP          FIRST
NO. NO.   NAME      VALUE        ERROR      SIZE      DERIVATIVE
 1  1  P1       10.500     0.73236    0.10000    0.00000
 2  2  P2      -1.0000    0.11010   0.15033E-01 -0.38147E-04

CHISQUARE = 0.0000E+00  NPFIT = 10

>>>> IP=3
>>>> ve/draw VECT(1:10,3)
>>>> ORDER=2
>>>> ve/fit VECT(1:10,1) VECT(1:10,3) ! P2 WS
*****
*
* Function minimization by SUBROUTINE HFITV *
* Variable-metric method *
* ID =          0  CHOPT = WWS *
*
```

```
*****
Convergence when estimated distance to minimum (EDM) .LT. 0.10E-03

FCN= 0.8969682      FROM MIGRAD      STATUS=CONVERGED  CALLS= 47 EDM= 0.36E-09
          STRATEGY= 1    ERROR DEF= 1.0000

INT EXT  PARAMETER                      STEP      FIRST
NO. NO.   NAME      VALUE      ERROR      SIZE      DERIVATIVE
 1 1  P1        9.3136    1.4106    0.10000  0.95368E-05
 2 2  P2       -2.6000    0.53292  0.15033E-01  0.19392E-03
 3 3  P3        0.21212   0.43457E-01  0.17681E-02  0.15594E-02

CHISQUARE = 0.1281E+00  NPFIT = 10

>>>> IP=4
>>>> ve/delete VECT
```

3.2 One and two-dimensional functions

PAW allows the plotting of **functions**. A function can be specified on the command line, or in an external file, which is read by the FORTRAN interpreter **COMIS**.

One-dimensional trigonometric function

```
PAW > ****
PAW > *          TUTORIAL EXAMPLE PAWEX7      *
PAW > * Plot a few one-dimensional trigonometric functions      *
PAW > ****
PAW > opt utit           | No title on histogram
PAW > opt grid           | Put a grid on the plot
PAW > func/plot x*sin(x)*exp(-0.1*x) -10. 10.    | Plot first function
PAW > set dmod 2         | Set new line style
PAW > func/plot (sin(x)+cos(x))**5 -10. 10. s     | Plot second function on same plot
PAW > set dmod 3         | Set new line style
PAW > func/plot (sin(x)/(x)-x*cos(x)) -10. 10. s | Plot third function on same plot
PAW > set dmod 1         | Reset line style
PAW > opt htit           | Reset to histogram title
PAW > opt ngri           | No grid
```

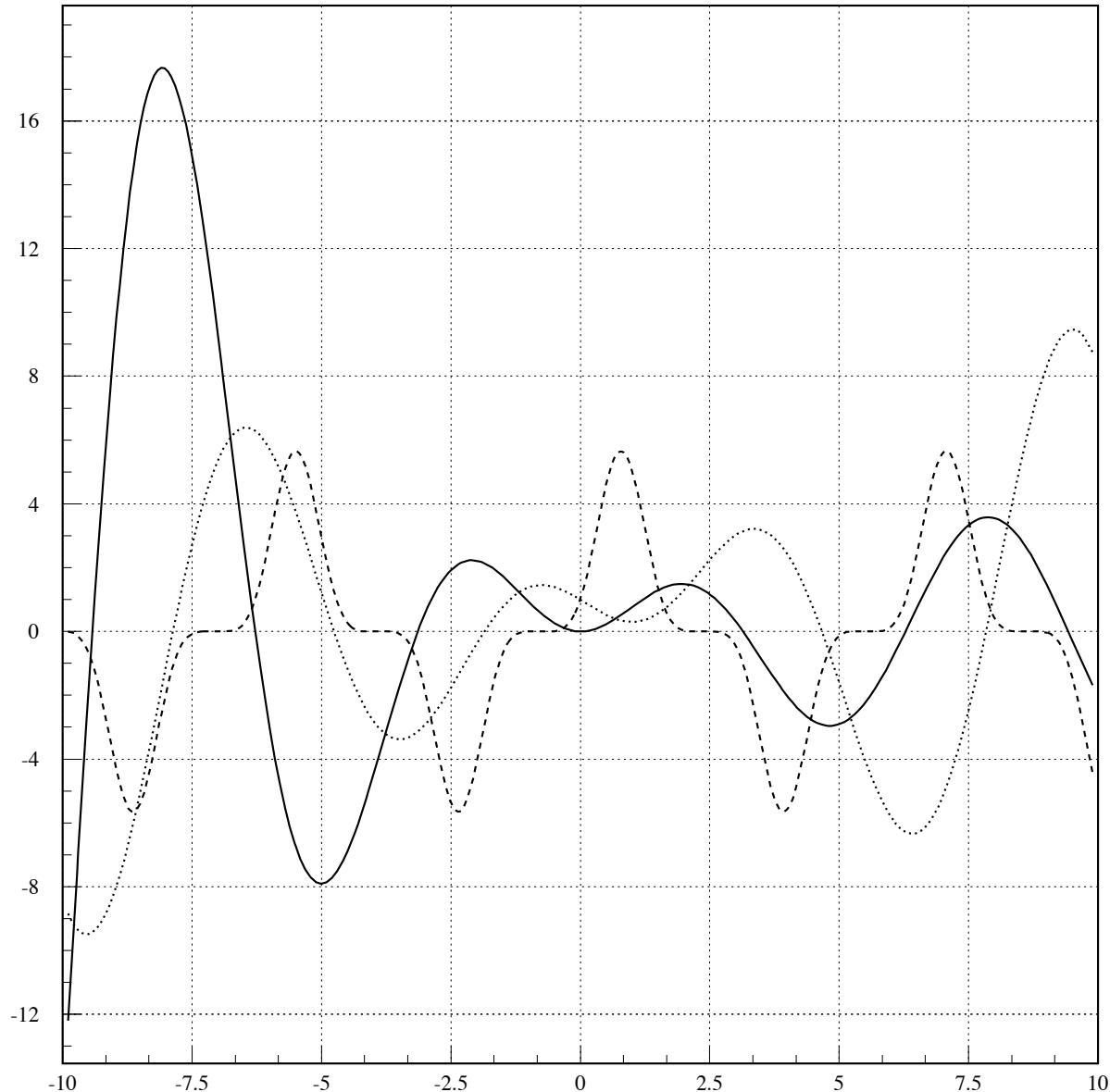


Figure 3.7: Plotting one-dimensional functions

One-dimensional functions and loops

```
PAW > edit pawex8 | Look at the Macro file
MACRO PAWEX8 1=8
*****
* TUTORIAL EXAMPLE PAWEX8 *
* Plot a one-dimensional function and loop *
* The Macro parameter is the number of plot to draw on the picture *
* The defaults is 8 *
*****
set DMOD 1 | Fix line style to 1
set XTIC 0.0001 | No tickmarks in X
set YTIC 0.0001 | No tickmarks in Y
set XVAL 100. | No values on Y axis
set YVAL 100. | No values on X axis
OPT utit | No title on histogram
FUN/PLOT X*SIN(X) -10 10 | Plot first function
FUN/PLOT X*COS(X)*SIN(X) -10 10 S | Plot second function
A=[1]-1 | Initialise loop variable
LOOP: | Start of loop <<<<<<
    FUN/PLOT X*SIN(X)*[A]/[1] -10 10 S | Next plot |
    FUN/PLOT X*COS(X)*SIN(X)*[A]/[1] -10 10 S | Next plot |
    A=[A]-1 | Update loop variable |
    IF [A]>0 GOTO LOOP | End of loop >>>>>>>
RETURN

PAW > *
PAW > * Execute Macro file PAWEX8 with as first parameter a value of 12
PAW > *
PAW > EXEC pawex8 12
PAW > opt htit | Reset to histogram title
```

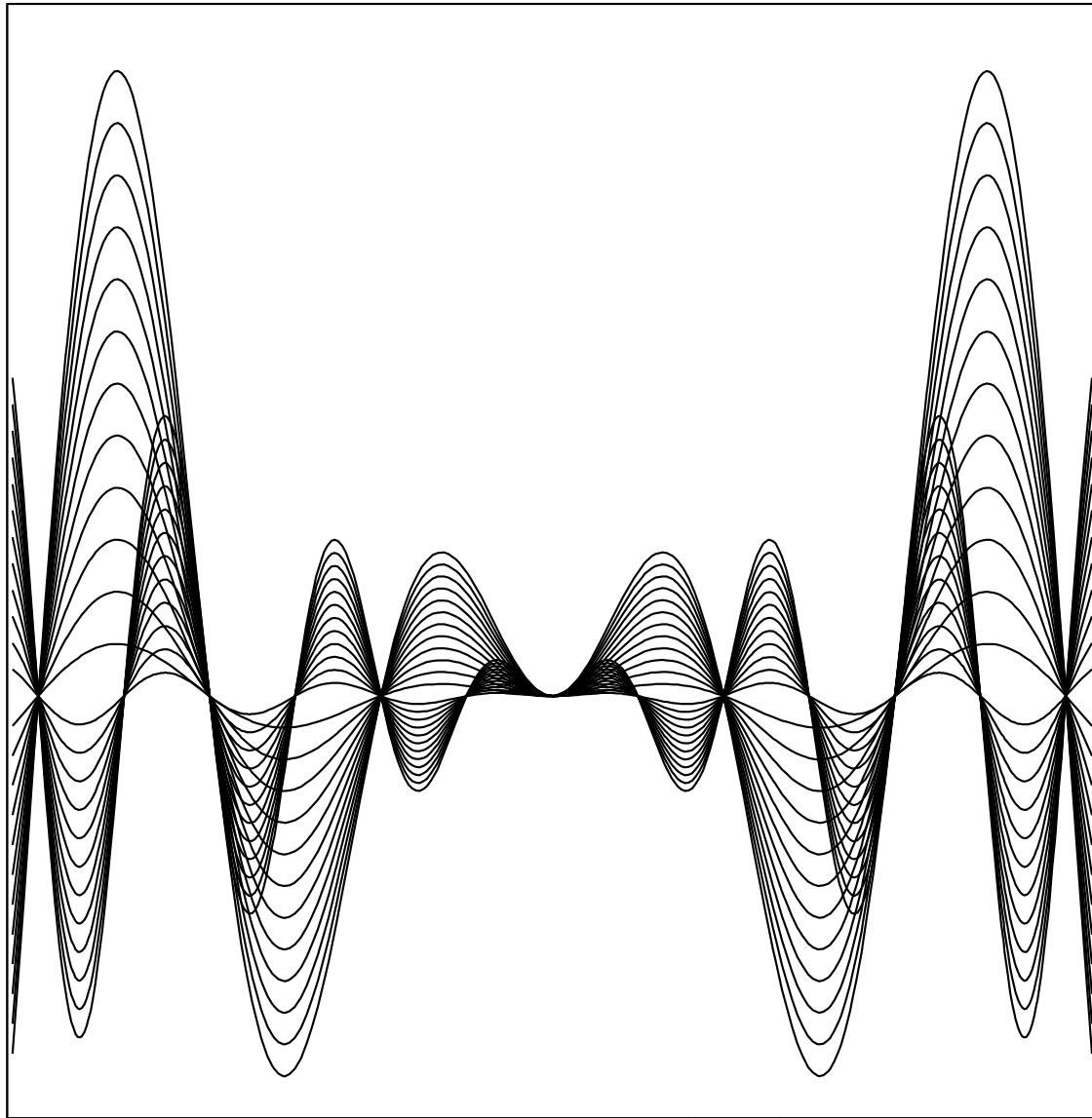


Figure 3.8: One-dimensional functions and loops

Two-dimensional representations

```
PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX9 *
PAW > * Plot two-dimensional functions *
PAW > ****
PAW > zone 2 2 | Divide picture in two by two
PAW > FUN2 10 abs(sin(x)/x)*(cos(y)*y) - | First 2-dim function using
PAW > 40 -6.0 6.0 40 -6.0 6.0 |
PAW > contour 10 40 0 | Contour plot representation
PAW > hi/de 10 | Delete histogram 10
PAW > FUN2 10 x*sin(x)*y*sin(y) 40 -10. 10. 40 -10. 10. C | Second 2-dim function
PAW > surf 10 | Surface plot representation
PAW > hi/de 10 | Delete histogram 10
```

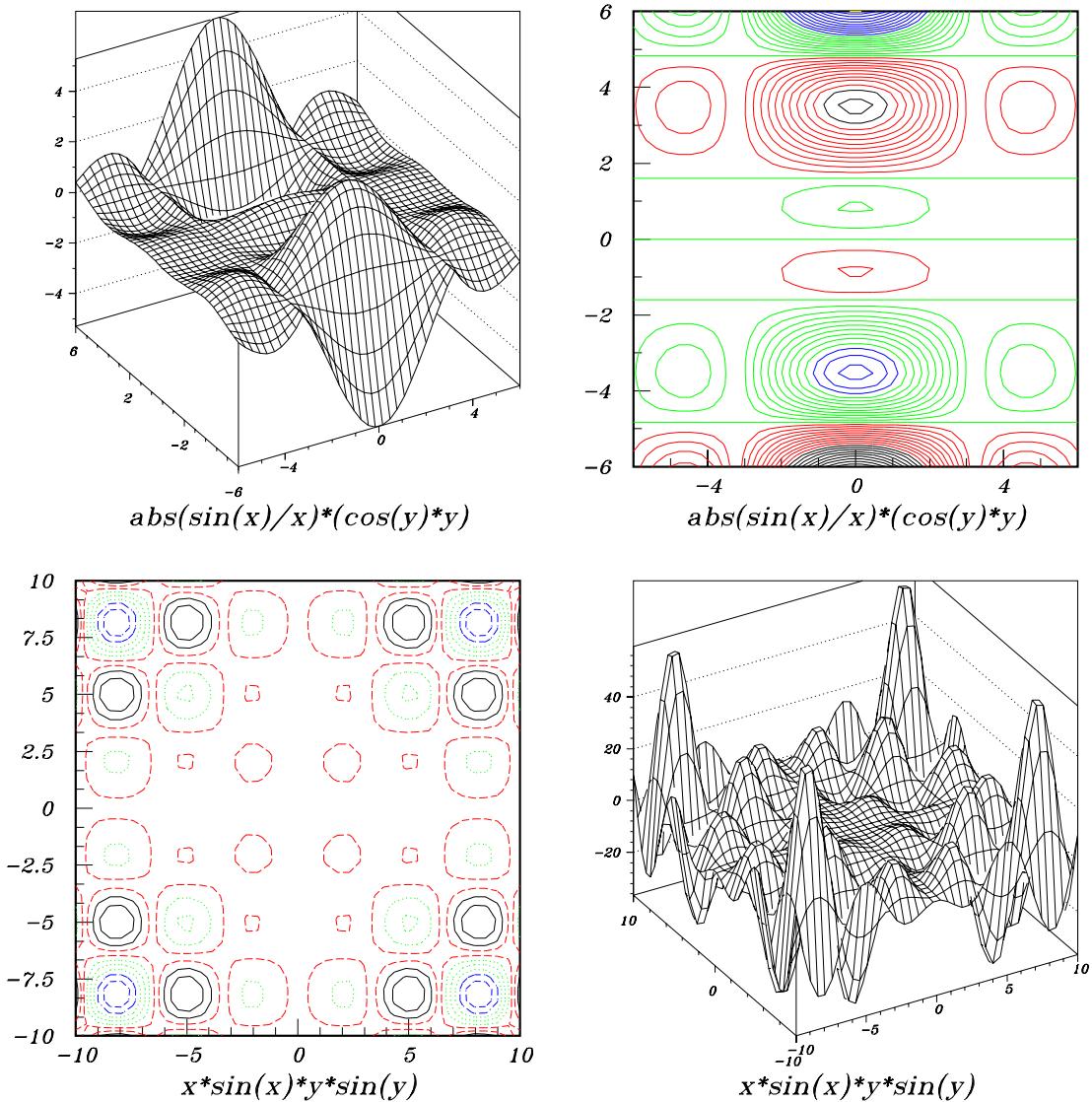


Figure 3.9: Plotting two-dimensional functions

Plotting a file from an external file

```
PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX10 *
PAW > * Calculate and plot (BOX option) Mandelbrot distribution *
PAW > ****
PAW > edit mandel.ftr | Look at FORTRAN function
      REAL FUNCTION MANDEL(XP)
      DIMENSION XP(2)
      DATA NMAX/30/
      X=XP(1)
      Y=XP(2)
      XX=0.
      YY=0.
      DO 30 N=1,NMAX
          TT=XX*XX-YY*YY+X
          YY=2.*XX*YY+Y
          XX=TT
          IF (4..LT.XX*XX+YY*YY) GO TO 1
30      CONTINUE
1      MANDEL=FLOAT(N)/FLOAT(NMAX)
      END
PAW > *
PAW > * Calculate the mandel function defining the limits of the plot
PAW > *
PAW > fun2 10 mandel.ftr 100 -2.4 .8 100 -1.2 1.2 , ,
PAW > *
PAW > * Open "metafile" for PostScript image
PAW > *
PAW > fortran/file 66 mandel.ps
PAW > metafile 66 -113 | HIGZ/PostScript portrait mode
PAW > hi/pl 10 BOX | Plot histogram with BOX option
PAW > close 66 | Close metafile
```

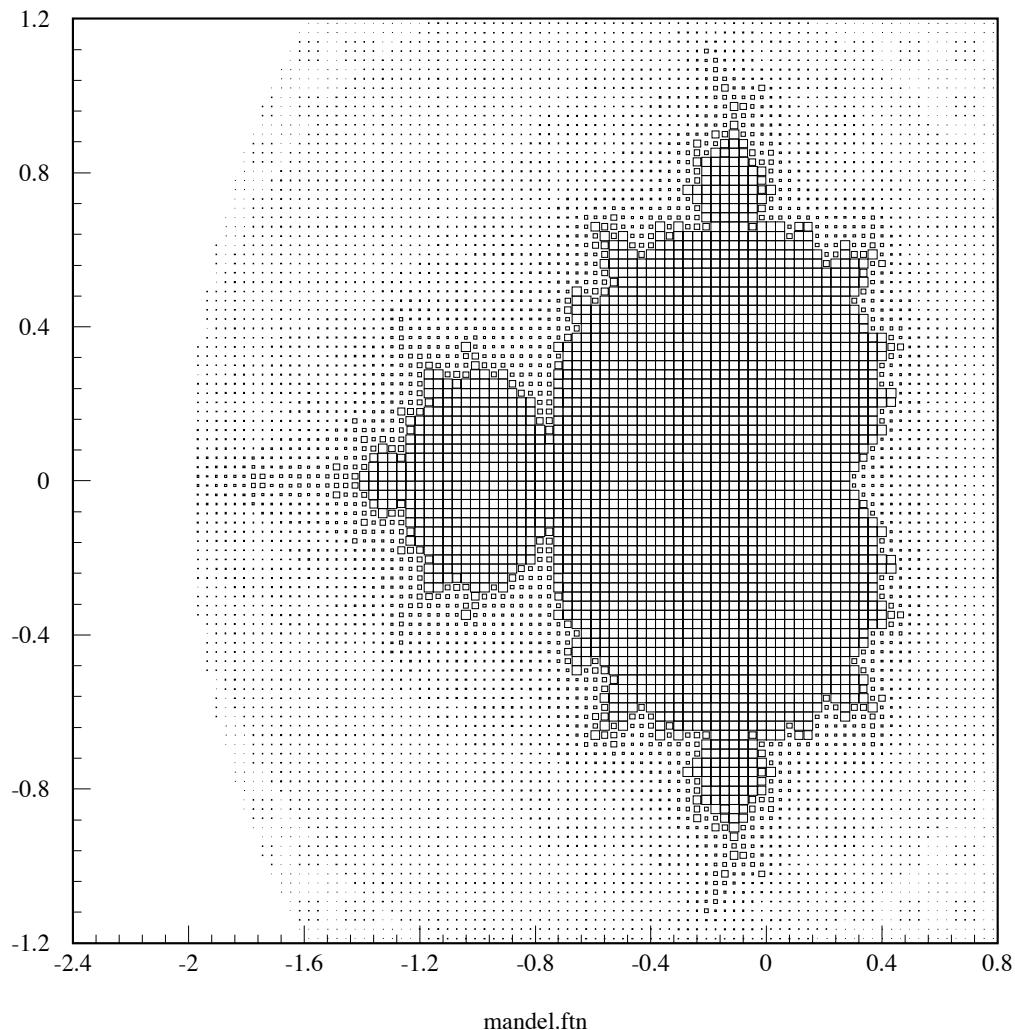


Figure 3.10: Plotting a two-dimensional function specified in a external file

3.3 Using histograms

Creating histograms

```

PAW > ****
PAW > *
PAW > * Creation of one and two-dimensional histograms
PAW > ****
PAW > edit htfun1.ftn | Look at FORTRAN function
      FUNCTION HTFUN1(X)
      DATA C1,C2,XM1,XM2,XS1,XS2/1.,0.5,0.3,0.7,0.07,0.12/
      A1=-0.5*((X-XM1)/XS1)**2
      A2=-0.5*((X-XM2)/XS2)**2
      X1=C1
      X2=C2
      IF(ABS(A1).GT.1.E-4)X1=C1*EXP(A1)
      IF(ABS(A2).GT.1.E-4)X2=C2*EXP(A2)
      HTFUN1=X1+X2
      END
      FUNCTION HTFUN2(X,Y)
      HTFUN2=HTFUN1(X)*HTFUN1(Y)
      END
PAW > edit urout.ftn | Edit COMIS Fortran routine
      SUBROUTINE UROUT(NEV)
      DO 10 I=1,NEV
          X=HRNDM1(100)
          CALL HFILL(110,X,0.,1.)
10 CONTINUE
      END
PAW > opt GRID | Add grid to plot
PAW > zone 1 2 | Divide plot in two
PAW > * -- Read functions from file htfun1.ftn into memory
PAW > * -- Fill 1-dimensional histogram 100 according to 1-dim function htfun1 and plot
PAW > fun1 100 htfun1.ftn 100. 0. 1.
PAW > 1dhisto 110 'Test 1-dim Histo' 100 0. 1. 1000. | Create a 1-dimensional histogram
PAW > * -- Fill histogram 110 according to function 100 (5000 events)
PAW > call urout.ftn(5000)
PAW > * -- Fill and plot as a contour 2-dimensional histogram 200 according to the
PAW > * -- 2-dim function htfun2 in memory
PAW > fun2 200 htfun2 25. 0. 1. 25. 0. 1. C
PAW > hi/li | List known histograms
      ===> Directory : //PAWC
          100 (1)    htfun1.ftn
          110 (1)    Test 1-dim Histo
          200 (2)    htfun2
PAW > histogram/file 1 pawhists.rzdat ! N | Define output unit 1
PAW > hrouut 0 | Write all histograms
PAW > close 1 | Close the output unit
PAW > hi/de 0 | Delete all histograms in memory

```

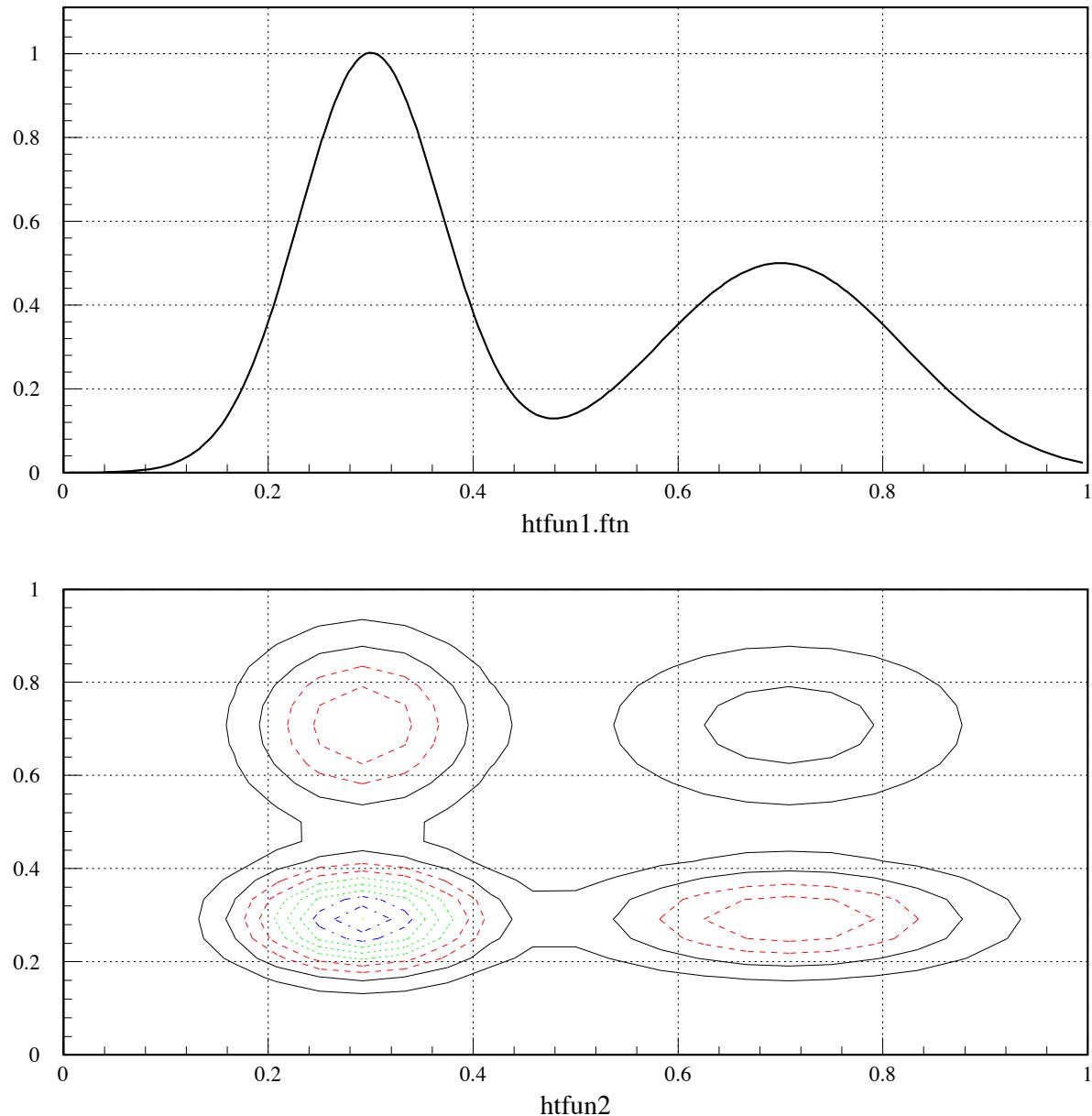


Figure 3.11: Creation of one- and two-dimensional histograms

Inputting histograms from an external file
--

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX12 *
PAW > * Read histograms from file and plot *
PAW > ****
PAW > histogram/file 1 pawhists.rzdat | Open histogram file created in PAWEX11
PAW > hrin * | Read all histograms from file
PAW > ldir | List current directory
***** Directory ==> //LUN1 <==

        Created 890915/1653 Modified 890917/1116

====> List of objects
    HBOOK-ID CYCLE DATE/TIME NDATA OFFSET REC1 REC2
      100      1 890915/1653   152      1     3
      110      1 890915/1653    85    153     3
      200      1 890915/1653   779    238     3

NUMBER OF RECORDS = 3 NUMBER OF MEGAWORDS = 0 + 3056 WORDS
PER CENT OF DIRECTORY QUOTA USED = 0.075
PER CENT OF FILE USED = 0.075
BLOCKING FACTOR = 66.146
PAW > hi/list | List histograms on file
====> Directory : //LUN1
      100 (1) htfun1.ftn
      110 (1) Test 1-dim Histo
      200 (2) htfun2.ftn

PAW > opt GRID | Put a grid on the picture
PAW > zone 2 2 | Divide picture in two by two parts
PAW > hi/pl 100 | Plot histogram 100
PAW > set htyp -3 | Choose hatch style
PAW > hi/pl 110 | Plot histogram 110 with option E
PAW > zone 1 2 2 'S' | Redefine zone
PAW > hi/plot 200 | Plot 2-dim histogram 2 (Box option)
PAW > h/del 0 | Delete histograms from memory
PAW > close 1 | Close input unit
PAW > zone | Reset zone

```

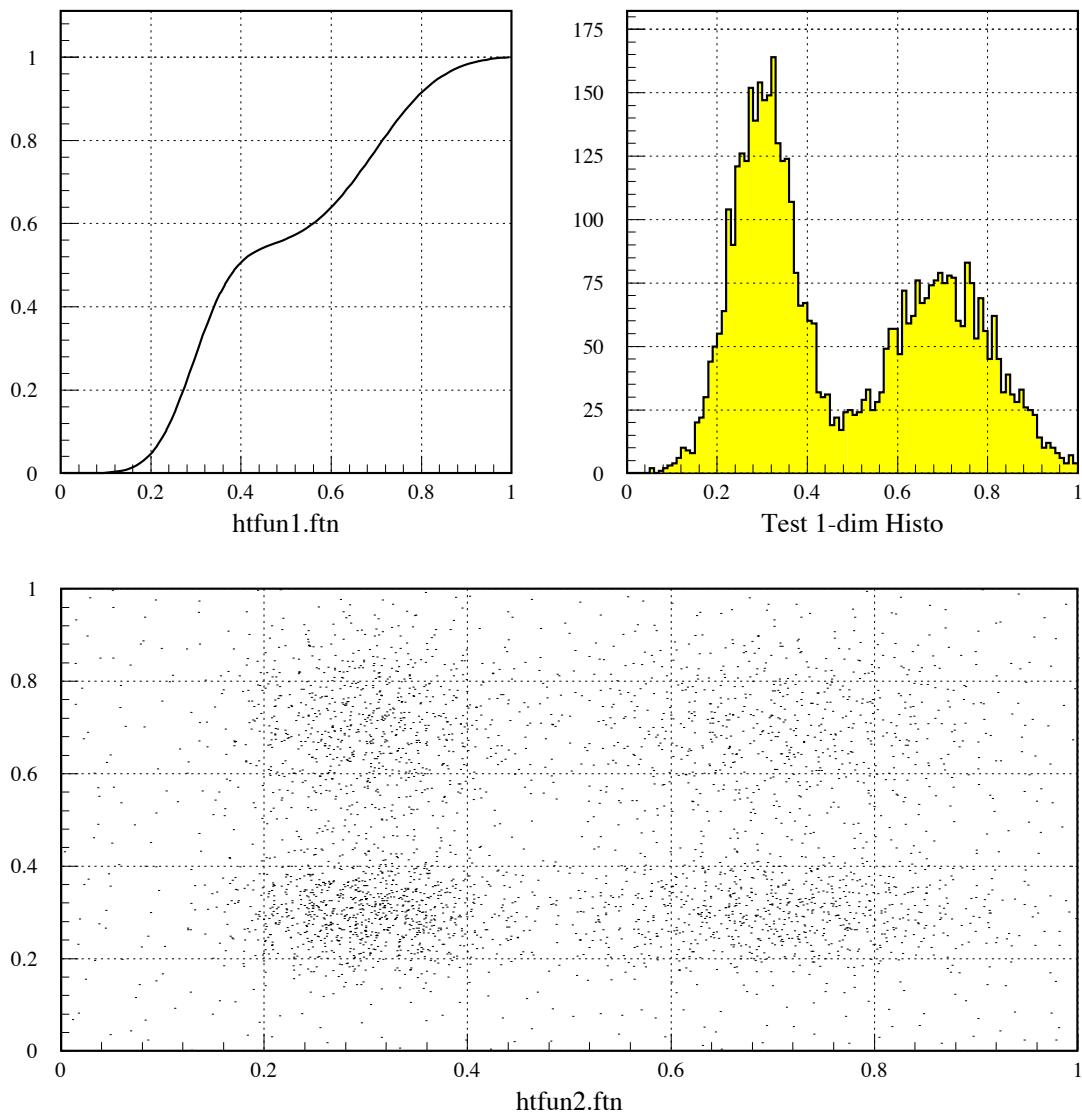


Figure 3.12: Reading histograms on an external file

Histogram operations

```
PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX13 *
PAW > * Perform operations on histograms read from file and save result *
PAW > ****
PAW > histogram/file 1 pawhists.rzdat ! U | Open histogram file
PAW > hrin 0 | Read all histograms on file
PAW > zon 2 2 | Divide picture in two by two parts
PAW > opt grid | Put a grid on the picture
PAW > igset mtyp 26 | Polymarker
PAW > set KSIZ 0.2 | Set size of polymarker (in cm)
PAW > hi/pl 110 e | Plot histo 110 with E option
PAW > hi/pl 110 pl | Plot histo 110 with polymarker
PAW > set CFON 2 | Set font for comments
PAW > set CSIZ 0.4 | Set size for comments (in cm)
PAW > key 0.5 130 26 '[p^+!,p^-!,m^+!,m^-]' | Example of HIGZ software fonts
PAW > zon 1 2 2 s | Redefine picture layout
PAW > hi/op/add 110 110 120 0.5 0. | Histogram 120 is half of 110
PAW > hi/op/add 110 110 130 0.25 0. | Histogram 130 is quarter of 110
PAW > set htyp -3 | Hatch style for histogram
PAW > hi/pl 110 | Plot histogram 110
PAW > set htyp 444 | Set hatch styles for histogram 120
PAW > hi/pl 120 s | Plot histogram 120 on same plot
PAW > set htyp 244 | Set hatch styles for histogram 130
PAW > hi/pl 130 s | Plot histogram 130 on same plot
PAW > text 0.55 95. 'LEP Very Preliminary' 0.35 25. | Add text to plot
PAW > hrouut 0 | Write histograms to file
PAW > close 1 | Close input/output unit
```

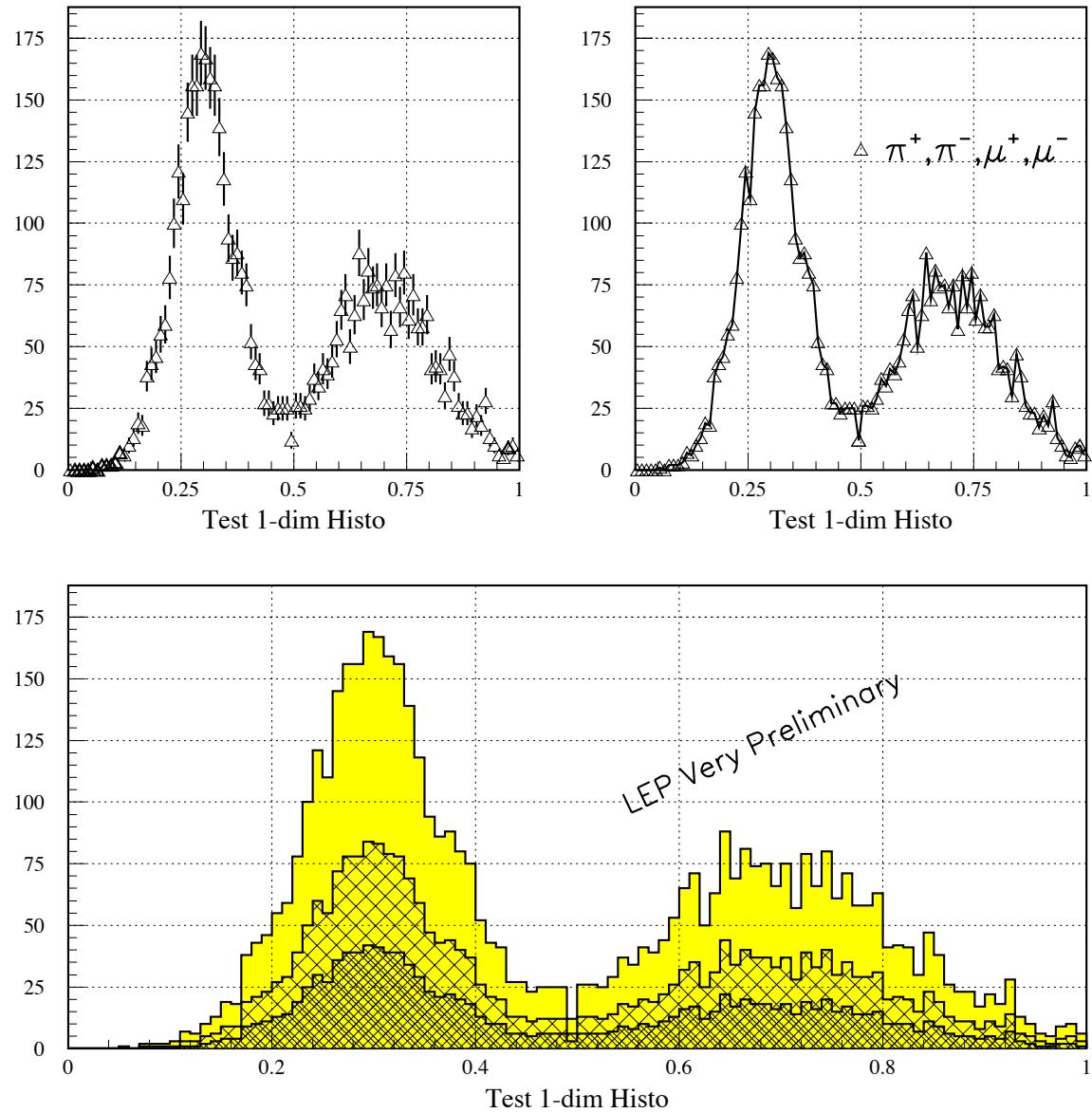


Figure 3.13: One-dimensional plotting and histogram operations

Two-dimensional data presentations

```
PAW > ****
PAW > *          TUTORIAL EXAMPLE PAWEX14      *
PAW > * Different representations of 2-dim histograms   *
PAW > ****
PAW > *
PAW > * Open histogram file created in PAWEX11, modified in PAWEX13
PAW > *
PAW > histogram/file 1 pawhists.rzdat
PAW > hrin 0                                | Read all histograms on file
PAW > hi/li                                | List histograms on file
      ===> Directory : //LUN1
          100 (1)  htfun1.ftn
          110 (1)  Test 1-dim Histo
          200 (2)  htfun2
          120 (1)  Test 1-dim Histo
          130 (1)  Test 1-dim Histo
PAW > close 1                                | Close input unit
PAW > opt UTIT                                | Turn off HBOOK titles
PAW > zone 2 2                                | Divide picture in two by two parts
PAW > hi/plot 200 BOX                          | Plot histogram 200 with BOX option
PAW > contour 200 20 0                         | Plot histogram 200 as contour plot
PAW > lego 200                                | Plot histogram 200 as lego plot
PAW > surf 200                                | Plot histogram 200 as surface plot
PAW > hi/del 0                                | Delete histograms in memory
PAW > zone                                    | Reset zone
```

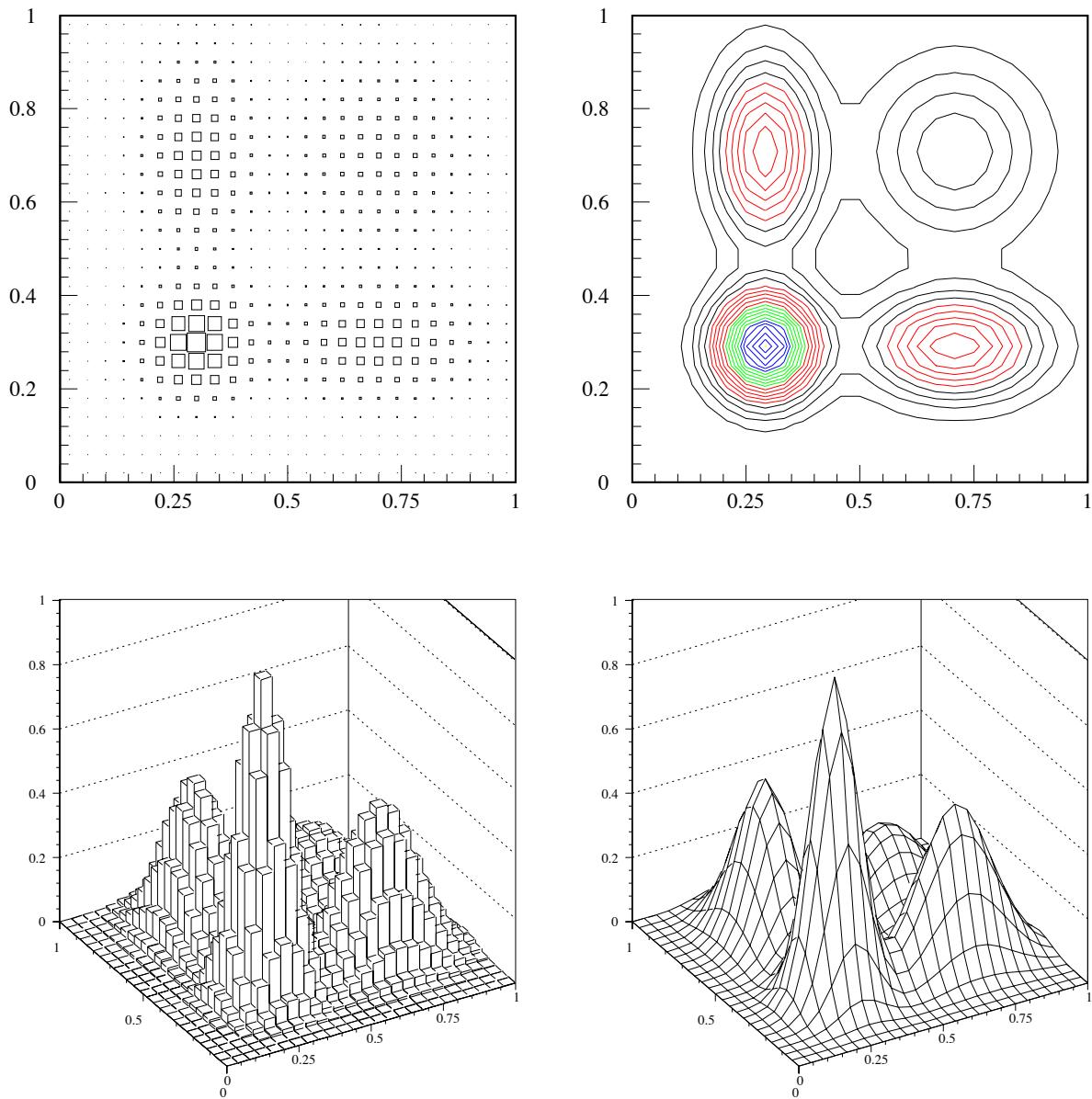


Figure 3.14: Two-dimensional data representations

Sub-ranges in histogram identifiers

```
PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX15 *
PAW > * Use of subranges in histogram specifiers *
PAW > ****
PAW > histogram/file 1 pawhists.rzdat | Open histogram file created in PAWEX11
PAW > hrin 0 | Read all histograms from file
PAW > close 1 | Close input unit
PAW > opt GRID | Draw grid on picture
PAW > opt UTIT | Turn off HBOOK titles
PAW > zone 2 2 | Divide picture in two by two parts
PAW > hi/pl 110(56:95) e | Plot subrange of histogram 110
PAW > opt NGRI | No grid on picture
PAW > set htyp -3 | Set hatch styles for histogram
PAW > hi/pl 200(8:8,) BOX | Plot subrange of 200 with BOX option
PAW > hi/pl 200(3:15,3:15) CONT | Plot subrange of 200 as a CONTOUR plot
PAW > hi/pl 200(4:12,4:12) LEGO | Plot subrange of 200 as a LEGO plot
PAW > hi/del 0 | Delete histograms in memory
PAW > zone | Reset zone
```

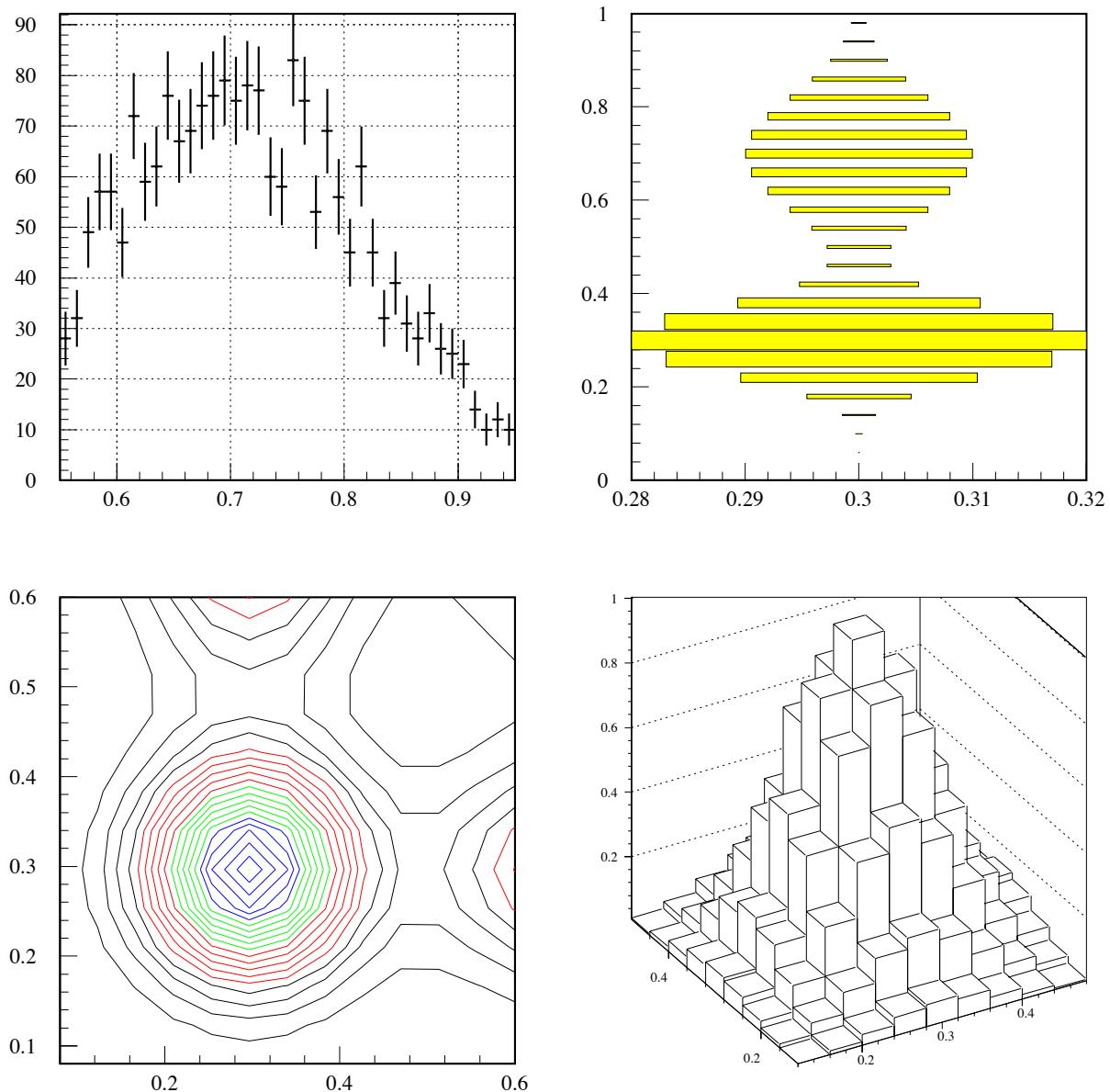


Figure 3.15: The use of sub-ranges in histogram specifiers

3.4 Examples with Ntuples

Ntuples are large named two-dimensional arrays. From the physicist's point of view they can be considered as **event** files. Ntuples can be accessed as a whole or single columns, or even single components. **Columns** can be identified by a name or by an index. A rather complete set of **operators** is available to deal with Ntuples - these include capabilities to apply **cuts** or **selection criteria** to the Ntuple data, using a notation where arithmetic and boolean operators and mathematical functions can be freely used. A powerful **mask** facility exists to enable the user to select a Ntuple subset which has particular characteristics, allowing in this way very fast access to a data subset (see section 7.5 on page 131).

3.4.1 A first example - CERN personnel statistics

In order to introduce and explain the main functionalities of the Ntuples, a simple data sample containing some characteristics of the CERN staff, will be used. For each member of the staff there exists one entry in the file. Each entry consists of 11 values, described in table 3.1.

Variable Name	Description and possible values
CATEGORY:	Professional category (integer between 100 and 600) 100-199: Scientific staff 200-299: Engineering staff 300-399: Technical support staff 400-499: Crafts and trade support staff 500-529: Supervisory administrative staff 530-559: Intermediate level administrative staff 560-599: Lower level administrative staff
DIVISION:	Code for each division (integer between 1 and 13) 1=AG 2=DD 3=DG 4=EF 5=EP 6=FI 7=LEP 8=PE 9=PS 10=SPS 11=ST 12=TH 13=TIS
FLAG:	A flag where the first four bits have the following significance Bit 1 = 0 means female otherwise male Bit 2 = 0 means resident otherwise non-resident Bit 3 = 0 means single otherwise head of family Bit 4 = 0 means fixed term contract otherwise indefinite duration contract
AGE:	Age (in years) of staff member
SERVICE:	Number of years of service that the staff member has at CERN
CHILDREN:	Number of dependent children
GRADE:	Staff member's position in Grade scale (integer between 3 and 14)
STEP:	Staff member's position (step) inside given grade (integer between 0 and 15)
NATION:	Code for staff member's nationality (integer between 1 and 15) 1=AT 2=BE 3=CH 4=DE 5=DK 6=ES 7=FR 8=GB 9=GR 10=IT 11=NL 12=NO 13=PT 14=SE 15=ZZ (non-member states)
HRWEEK:	Number of contractual hours worked per week (between 20 and 44)
COST:	Cost of the staff member to CERN (in CHF)

Table 3.1: Definition of the variables of the CERN staff Ntuple

3.4.2 Creating Ntuples

Ntuples can be created interactively, using PAW (see figure 3.16) or in an independent batch job, using HBOOK, using the following calling sequence [2]

```
CALL HBOOKN(IDN, TITLE, NCOLUMN, CHRZPA, NPRIME, CHTAGS)
```

Two kinds of Ntuples are supported: **memory-resident** Ntuples for small data samples, which can be completely contained in memory, or **disk-resident** Ntuples for large data samples which need the use of an extended store. For example the FORTRAN program equivalent to the code for the creation of the memory-resident Ntuple in figure 3.16 is as follows:

Creating a memory-resident Ntuple

```

PROGRAM APTUPLE
PARAMETER (NCOLUMN=11)
CHARACTER*8 CHTAGS(NCOLUMN)
COMMON/PAWC/H(70000)
REAL XTUPLE(NCOLUMN)
DATA NPRIME/1000/
*
DATA CHTAGS/'category','division','flag','age','service',
+ 'children','grade','step','nation','hrweek','cost'/
*
CALL HLIMIT(70000)
OPEN(UNIT=1,FILE='APTURE.DAT',STATUS='OLD')
CALL HBOOKN(10,'CERN Population',NCOLUMN, ,NPRIME,CHTAGS)
DO 100 I=1,10000
  READ(1,'(10F4.0,F7.0)',END=1000) XTUPLE
  CALL HFN(10,XTUPLE)
100 CONTINUE
*
1000 CALL HRPUT(0,'aptuple.rzdat','N')
END

```

Memory-resident Ntuple: CHRZPA= ' '

The storage required for a memory-resident Ntuple is approximately the product of the number of columns NCOLUMN multiplied by the number of events. With a call to HBOOKN an initial bank of size NPRIME words is created in memory. When the bank is full, a new bank with the same size is created and is linked to the previous bank, and so on up to the maximum memory allocation declared by the call to HLIMIT. A memory-resident Ntuple can be saved on a HBOOK file with a call to HROUT or HRPUT.

Disk-resident Ntuple: CHRZPA='RZ directory name to store Ntuple'

In this case a bank with a size of NPRIME words is also created in memory. However, when the bank is full, its contents is written to disk into the directory CHRZPA, and the **same** bank is used to store the data for the next events, thus overwriting the initial contents. As for a memory-resident Ntuple, a header is kept in memory. It contains among other things the number of events already treated as well as the number and the address of the disk extensions. This header **must** be saved (after filling the Ntuple) on the disk file after having set the current directory to CHRZPA. The number of extensions on disk can be interactively displayed using the command:

```
PAW > LDIR ' ' A
```

The example above could be modified in the following way to create a disk-resident Ntuple:

Creating a disk-resident Ntuple

```

OPEN(UNIT=1,FILE='aptuple.dat',STATUS='OLD')
LRECL=1024
CALL HROPEN(2,'APTUPLE','aptuple.rzdat','N',LRECL,ISTAT)
CALL HBOOKN(10,'CERN Population',NCOLUMN,'APTUPLE',NPRIME,CHTAGS)

. . .

1000 CALL HRROUT(0,ICYCLE,' ')
CALL HREND('APTUPLE')
END

```

When dealing with very large Ntuples, it is recommended to open the HBOOK file with a large block size (e.g. LRECL=8192 words and a large primary allocation (e.g. NPRIME=60000). Routine HROPEN allocates a maximum of 8000 records each LRECL words long. If more is required, put CHOPT='NQ' and IQUEST(10)=Number of records, where the vector IQUEST is defined in the ZEBRA common COMMON/QUEST/IQUEST(100).

Listing the directory of the memory-resident Ntuple

```

PAW > LDIR
SIGMA
***** Directory ==> //aptuple <==

Created 890627/1620 Modified 890627/1621

==> List of objects
HBOOK-ID CYCLE DATE/TIME NDATA OFFSET REC1 REC2
 10       1   890627/1621 38107      1       3       4 ==>    40

NUMBER OF RECORDS = 39 NUMBER OF MEGAWORDS = 0 + 39131 WORDS
PER CENT OF DIRECTORY QUOTA USED = 0.975
PER CENT OF FILE USED = 0.975
BLOCKING FACTOR = 95.420

```

It is seen that 38107 words were needed to store all the data associated to the Ntuple. The file aptuple.rzdat created in batch by the program or the one created interactively in figure 3.16 are completely interchangeable. Generally speaking, a Ntuple (or any HBOOK histogram RZ file) can be created in batch on a mainframe and then transferred to a machine running PAW. See chapter 9 on page 187 for more details.

In the listing below extensions are identified by 10000*Extension_Number+Ntuple_ID.

Listing the directory of the disk-resident Ntuple

```

PAW > LDIR , , A
***** Directory ==> //LUN2 <==

Created 891010/1038 Modified 891010/1038

==> List of objects
HBOOK-ID CYCLE DATE/TIME NDATA OFFSET REC1 REC2

```

10010	1	891010/1038	1005	1	3	
20010	1	891010/1038	1005	1006	3	4
30010	1	891010/1038	1005	987	4	5
40010	1	891010/1038	1005	968	5	6
50010	1	891010/1038	1005	949	6	7
60010	1	891010/1038	1005	930	7	8
70010	1	891010/1038	1005	911	8	9
80010	1	891010/1038	1005	892	9	10
90010	1	891010/1038	1005	873	10	11
100010	1	891010/1038	1005	854	11	12
110010	1	891010/1038	1005	835	12	13
120010	1	891010/1038	1005	816	13	14
130010	1	891010/1038	1005	797	14	15
140010	1	891010/1038	1005	778	15	16
150010	1	891010/1038	1005	759	16	17
160010	1	891010/1038	1005	740	17	18
170010	1	891010/1038	1005	721	18	19
180010	1	891010/1038	1005	702	19	20
190010	1	891010/1038	1005	683	20	21
200010	1	891010/1038	1005	664	21	22
210010	1	891010/1038	1005	645	22	23
220010	1	891010/1038	1005	626	23	24
230010	1	891010/1038	1005	607	24	25
240010	1	891010/1038	1005	588	25	26
250010	1	891010/1038	1005	569	26	27
260010	1	891010/1038	1005	550	27	28
270010	1	891010/1038	1005	531	28	29
280010	1	891010/1038	1005	512	29	30
290010	1	891010/1038	1005	493	30	31
300010	1	891010/1038	1005	474	31	32
310010	1	891010/1038	1005	455	32	33
320010	1	891010/1038	1005	436	33	34
330010	1	891010/1038	1005	417	34	35
340010	1	891010/1038	1005	398	35	36
350010	1	891010/1038	1005	379	36	37
360010	1	891010/1038	1005	360	37	38
370010	1	891010/1038	1005	341	38	39
10	1	891010/1038	1107	322	39	40

NUMBER OF RECORDS = 39 NUMBER OF MEGAWORDS = 0 + 39316 WORDS

PER CENT OF DIRECTORY QUOTA USED = 0.975

PER CENT OF FILE USED = 0.975

BLOCKING FACTOR = 95.883

Creating Ntuples

```

PAW > ****
PAW > *          TUTORIAL EXAMPLE PAWEX16      *
PAW > * Creation of a Ntuple and first look at its contents      *
PAW > ****
PAW > *
PAW > * Create Ntuple 10 with 11 variables
PAW > *
PAW > Ntuple/create 10 'CERN Population' 11 '3500
PAW > Category Division Flag Age Service Children Grade Step Nation Hrweek Cost
PAW > Ntuple/read 10 aptuple.dat           | Read the elements of the ntuple
PAW > Histo/file 1 aptuple.rzdat 1024 N    | Open Histogram file on unit 1
PAW > hroot 10                           | Write the ntuple to the file
PAW > ntuple/print 10                      | Look what Ntuple 10 contains
*****  

* NTUPLE ID= 10 ENTRIES= 3354 CERN Population *  

*****  

* Var numb * Name * Lower * Upper *  

*****  

*   1   * CATEGORY * 0.102000E+03 * 0.567000E+03 *  

*   2   * DIVISION * 0.100000E+01 * 0.130000E+02 *  

*   3   * FLAG   * 0.000000E+00 * 0.310000E+02 *  

*   4   * AGE    * 0.210000E+02 * 0.640000E+02 *  

*   5   * SERVICE * 0.000000E+00 * 0.350000E+02 *  

*   6   * CHILDREN * 0.000000E+00 * 0.600000E+01 *  

*   7   * GRADE   * 0.300000E+01 * 0.140000E+02 *  

*   8   * STEP    * 0.000000E+00 * 0.150000E+02 *  

*   9   * NATION   * 0.100000E+01 * 0.150000E+02 *  

*  10   * HRWEEK  * 0.200000E+01 * 0.440000E+02 *  

*  11   * COST    * 0.391000E+03 * 0.188530E+05 *  

*****  

PAW > zone 1 2                         | Divide the plot into two
PAW > opt stat                        | Print statistics on plot
PAW > set stat 110                     | Number entries and average
PAW > opt grid                        | Add grid to pictures
PAW > set htyp -3                      | Define hatch style for histograms
PAW > Ntuple/plot 10.Age                | Plot Age distribution
PAW > Ntuple/plot 10.Cost               | Cost distribution
PAW > Close 1                          | Close file 1
PAW > hi/delete 10                     | Delete Ntuple 10 from memory
PAW > zone                            | Reset zone

```

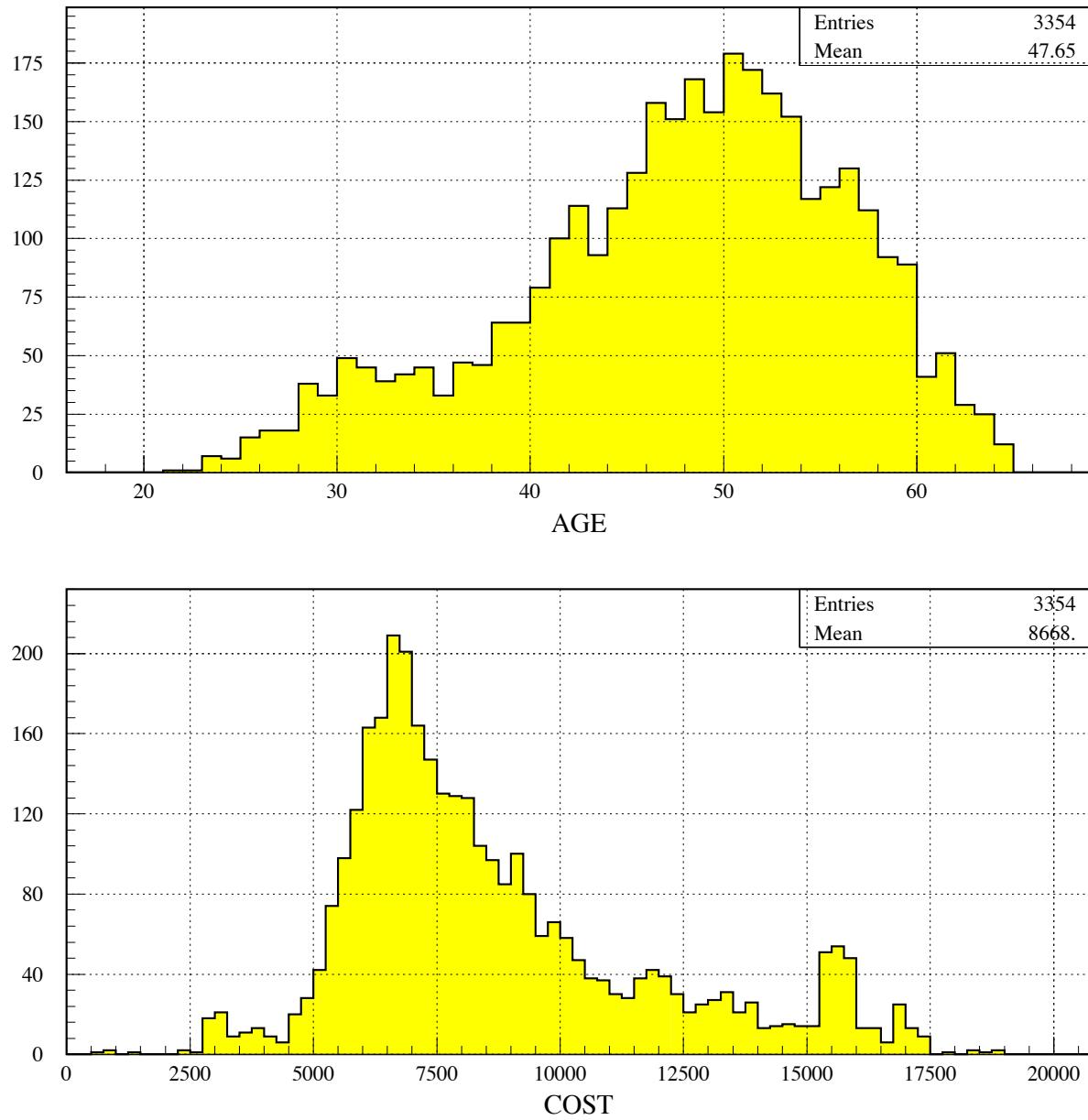


Figure 3.16: Ntuples - Creation and output to a file

Ntuples - automatic and user binning

```

PAW > **** TUTORIAL EXAMPLE PAWEX17 ****
PAW > * Read a Ntuple from a histogram file *
PAW > * Difference between automatic and user binning *
PAW > ****
PAW > hi/file 2 'aptuple.rzdat' | Open histogram file
PAW > * -- Divide plot into two by two squares
PAW > zone 2 2
PAW > opt nsta | No statistics on plot
PAW > opt grid | Add grid to pictures
PAW > set XTIC 0.15 | X-axis tick mark length (in cm)
PAW > set YTIC 0.15 | Y-axis tick mark length (in cm)
PAW > set HTYP -3 | Define hatch style for histograms
PAW > * -- Age distribution with automatic binning
PAW > Ntuple/pl 10.age
PAW > hi/create/1dhisto 11 'Age - User binning' 45 20. 65. | Book histogram 11
PAW > * -- Exactly 5 secondary and 9 primary divisions
PAW > set ndvx -509
PAW > * -- Age distribution with user binning
PAW > Ntuple/project 11 10.Age
PAW > hi/plot 11 | Plot histogram 11
PAW > hi/create/1dhisto 12 'Cost - User binning' 50 0. 20000. | Book histogram 12
PAW > set ndvx | Default divisions
PAW > * -- Cost distribution with automatic binning
PAW > Ntuple/plot 10.cost
PAW > * -- Exactly 5 secondary and 4 primary divisions
PAW > set ndvx -504
PAW > * -- Cost distribution with user binning
PAW > Ntuple/pl 10.Cost ! -12
PAW > close 2 | Close histogram file
PAW > zone | Reset zone
PAW > hi/de 0 | Delete histograms from memory

```

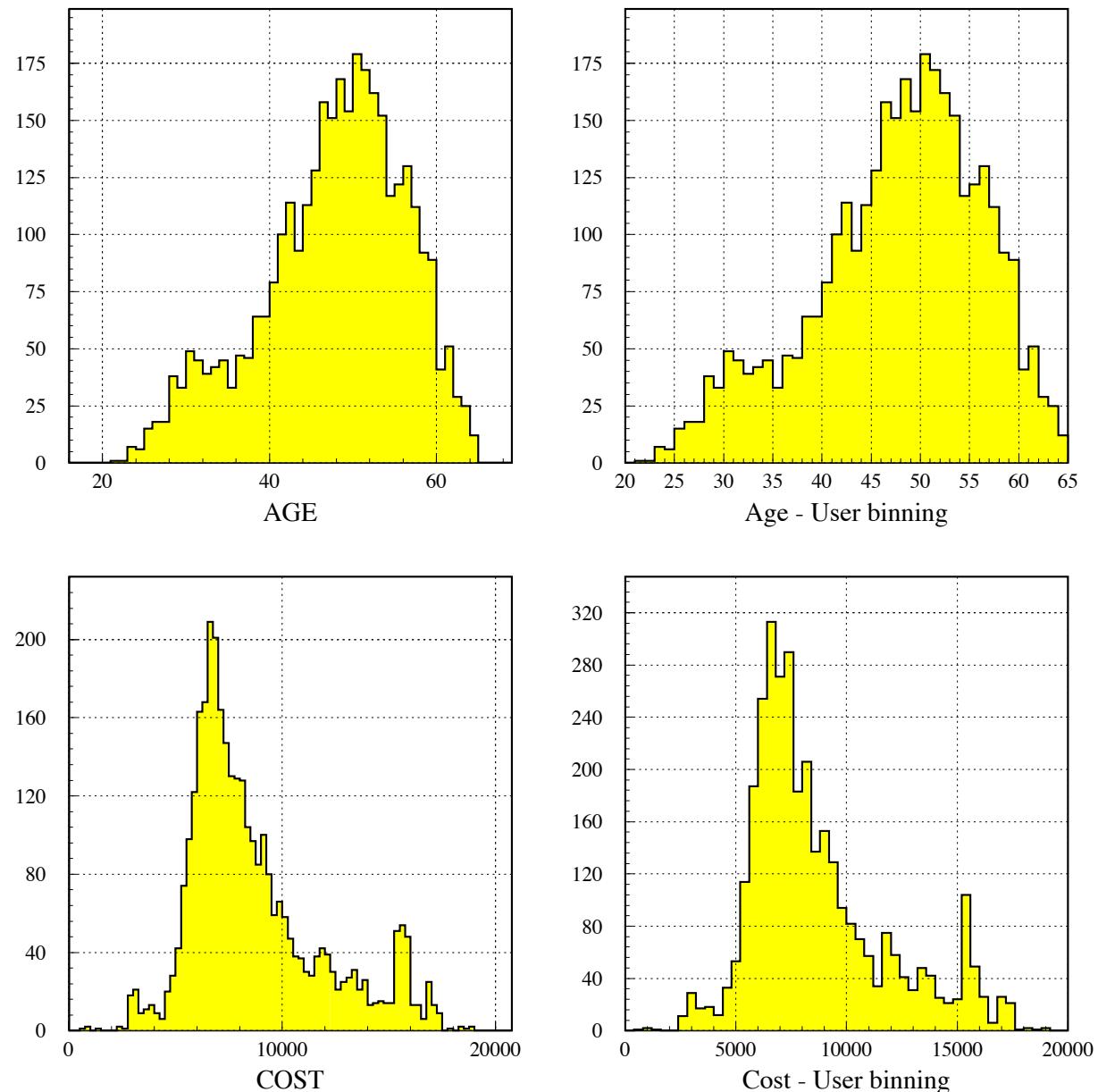


Figure 3.17: Ntuples - Automatic and user binning

Ntuples – selection criteria

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX18 *
PAW > * Ntuple SCAN and the use of simple selection criteria *
PAW > ****
PAW > hi/file 2 'aptuple.rzdat' | Open histogram file
PAW > opt grid | Add grid to pictures
PAW > opt stat | Print statistics on plot
PAW > set stat 110 | Indicate number of entries and average
PAW > ALIAS/CREATE DIVEP 5 | Create alias for EP Division
PAW > ALIAS/CREATE NATFR 7 | Create alias for French nationality
PAW > * -- Scan and look for French CERN staff in EP division printing the variables indicated
PAW > NT/SCAN 10 nation=NATFR.and.division=DIVEP ! ! 5 age service children grade step
*****
* ENTRY * AGE * SERVICE * CHILDREN * GRADE * STEP *
*****
! 48 ! 56.000 ! 34.000 ! 0.00000 ! 7.0000 ! 8.0000 !
! 194 ! 62.000 ! 27.000 ! 0.00000 ! 7.0000 ! 13.000 !
! 213 ! 56.000 ! 26.000 ! 0.00000 ! 6.0000 ! 13.000 !
! 214 ! 45.000 ! 26.000 ! 0.00000 ! 6.0000 ! 12.000 !
! 216 ! 56.000 ! 19.000 ! 0.00000 ! 5.0000 ! 13.000 !
! 266 ! 63.000 ! 26.000 ! 0.00000 ! 13.000 ! 10.000 !
! 267 ! 59.000 ! 32.000 ! 0.00000 ! 13.000 ! 10.000 !
! 273 ! 55.000 ! 26.000 ! 1.0000 ! 12.000 ! 13.000 !
! 275 ! 53.000 ! 26.000 ! 1.0000 ! 11.000 ! 13.000 !
! 279 ! 51.000 ! 30.000 ! 0.00000 ! 6.0000 ! 13.000 !
! 315 ! 56.000 ! 25.000 ! 0.00000 ! 8.0000 ! 6.0000 !
! 318 ! 64.000 ! 26.000 ! 0.00000 ! 6.0000 ! 13.000 !
! 320 ! 49.000 ! 26.000 ! 0.00000 ! 6.0000 ! 13.000 !
! 327 ! 59.000 ! 19.000 ! 0.00000 ! 5.0000 ! 13.000 !
! 328 ! 51.000 ! 25.000 ! 0.00000 ! 5.0000 ! 13.000 !
More...? ( <CR>/N ): n
==> 15 events have been scanned
PAW > hi/create/1d 200 'Number of years at CERN' 35 0 . 35. | Create histogram 200
PAW > max 200 250 | Maximum for histogram 200
PAW > * -- Maximum 5 secondary and 7 primary divisions
PAW > set ndvx 507
PAW > set htyp -3 | Define hatch style
PAW > Nt/pl 10.Service ! -200 | Plot years of service
PAW > ATITLE 'Years at CERN' 'Number of staff' | Define axis titles
PAW > set htyp 344 | Change hatch style
PAW > * -- Number of years at CERN for French staff members
PAW > Nt/pl 10.Service nation=NATFR -200 ! ! S | Change hatch style
PAW > set htyp 144 | Change hatch style
PAW > * -- Number of years at CERN for French Staff in EP
PAW > Nt/pl 10.Service division=DIVEP.and.nation=NATFR -200 ! ! S
PAW > close 2 | Close histogram file
PAW > set htyp 0 | Reset hatch style
PAW > hi/de 0 | Delete histograms from memory
PAW > zone | Reset zone

```

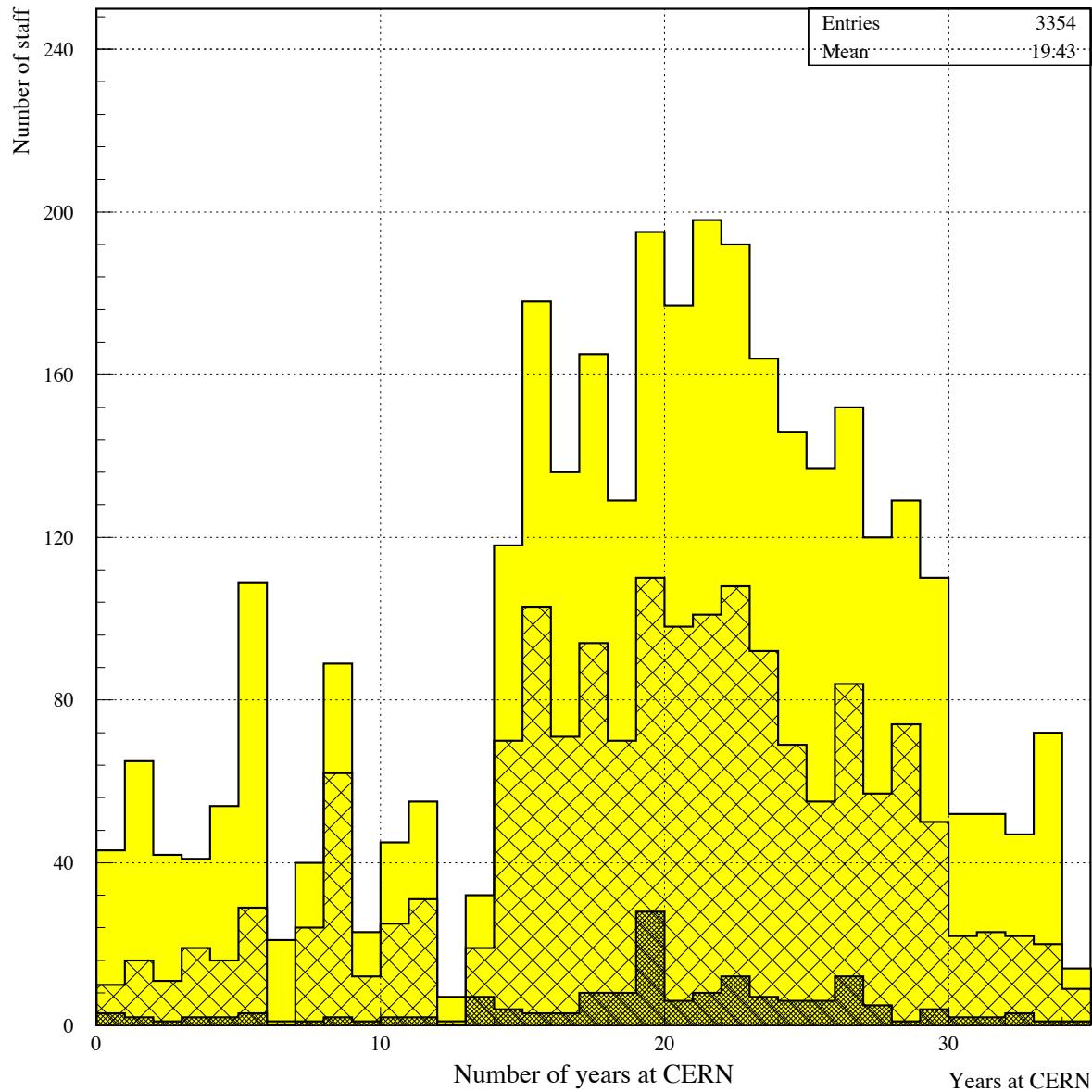


Figure 3.18: Ntuples - A first look at selection criteria

Ntuples – masks and loops

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX19 *
PAW > * Use of Ntuple masks and loop construct *
PAW > ****
PAW > hi/file 2 'aptuple.rzdat' | Open histogram file
PAW > hrin 0 | Read Ntuple into memory
PAW > close 2 | Close histogram file
PAW > set xtic 0.0001 | Make tick marks invisible
PAW > 1dhisto 20 'Distribution by grade' 12 3 15 | Create histogram 20
PAW > opt stat | Print statistics on plot
PAW > * -- Indicate number of entries and average
PAW > set stat 110
PAW > opt grid | Draw grid on picture
PAW > * -- Only horizontal grid lines with line type 3
PAW > set grid 1003
PAW > opt bar | Activate option bar
PAW > igset barw 0.8 | Width of bars
PAW > igset baro 0.1 | Origin of bar
PAW > max 20 700 | Maximum for histogram 20
PAW > * -- Exactly 12 divisions with text centered in X
PAW > set NDVX -12.05
PAW > * -- We want 5 secondary and 7 primary divisions in Y
PAW > set NDVY 507
PAW > set htyp -3 | Dotted hatch style (postscript)
PAW > * -- Plot grade distribution into histogram 10
PAW > Ntuple/plot 10.grade ! -20
PAW > box 9.5 10.5 610 640 | Draw a box on the histogram
PAW > igset TXAL 13 | Left adjust X and center Y
PAW > igset TXFP -130 | Roman font
PAW > igset CHHE 0.35 | Character height
PAW > itx 10.7 625 'All Staff' | Add some text to the histogram
PAW > set htyp 244 | Change the hatch style
PAW > * -- Define via a mask those people which are at the end of their grade
PAW > ntuple/mask stmask N 3500 | Define mask on file stmask.mask
PAW > ntuple/loop 10.grade step=15>>stmask(1)
PAW > ntuple/loop 10.grade grade>4.and.step=13>>stmask(2)
PAW > ntuple/loop 10.grade (grade=13.and.step=10).or.(grade=14.and.step=7)>>stmask(3)
PAW > Ntuple/plot 10.grade stmask(1).or.stmask(2).or.stmask(3)>>stmask(4) -20 ! ! S
PAW > ntuple/mask stmask C | Close mask file
PAW > box 9.5 10.5 560 590 | Draw second box on histogram
PAW > itx 10.7 575 'Staff at end of grade' | Add some more text to histogram
PAW > ATITLE 'Grade' 'Number of Staff' | Axis title
PAW > set htyp 0 | Reset hatch style
PAW > hi/de 0 | Delete histograms from memory

```

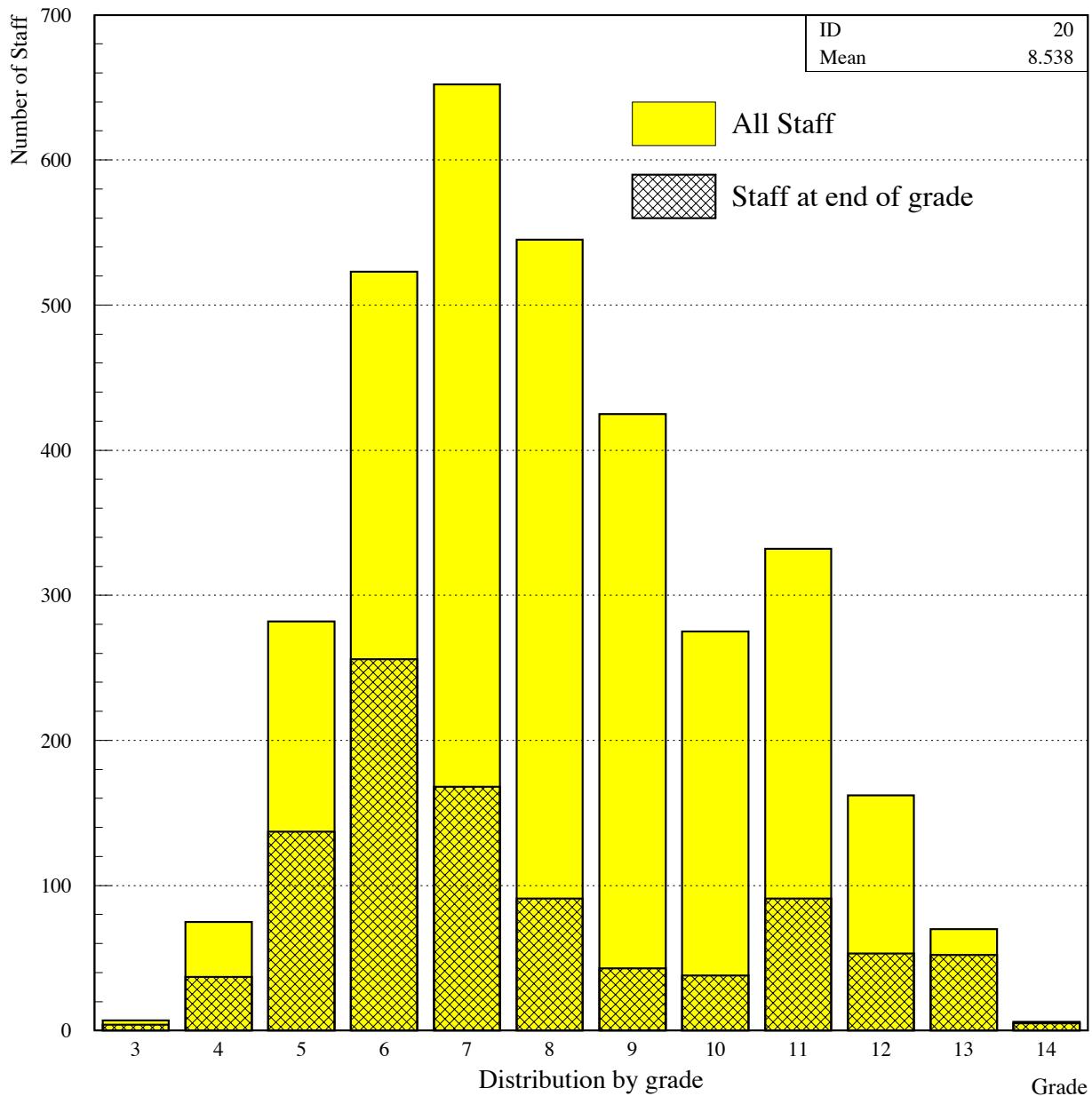


Figure 3.19: Ntuples - Masks and loop

Ntuple cuts

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX20 *
PAW > * The use of Ntuple Cuts *
PAW > ****
PAW > hi/file 2 'aptuple.rzdat' | Open histogram file
PAW > Ntuple/cut 1 MOD(FLAG,2).EQ.0 | CUT 1 : Female staff
PAW > Ntuple/cut 2 MOD(FLAG,4)>1 | CUT 2 : Non-resident staff
PAW > 1d 20 'Male/female and resident/non-resident Staff' 13 1 14
PAW > opt grid | Draw grid on picture
PAW > * -- Only horizontal grid lines with line type 3
PAW > set grid 1003
PAW > opt bar | Activate the bar chart option
PAW > opt stat | Print statistics on plot
PAW > set stat 10 | Only number of entries on plot
PAW > set xtic 0.0001 | Eliminate tick marks
PAW > igset barw 0.4 | Set bar origin
PAW > igset baro 0.1 | Set bar width
PAW > max 20 600 | Maximum for histogram 20
PAW > labels 1 13 AG DD DG EF EP FI LEP PE PS SPS ST TH TIS | Define labels for X axis
PAW > set NDVX 13.15 | X axis should centre the labels
PAW > * -- Plot all staff per division into histogram 20
PAW > Ntuple/plot 10.division ! -20
PAW > set htyp -3 | Choose hatch style
PAW > * -- Plot staff per division (satisfying cut 2)
PAW > Ntuple/plot 10.division 2 -20 ! ! s
PAW > igset baro 0.5 | Reset bar origin
PAW > set htyp 145 | Change hatch style
PAW > * -- Plot staff per division (satisfying cut 1)
PAW > Ntuple/plot 10.division 1 -20 ! ! s
PAW > set htyp 154 | Change hatch style
PAW > * -- Plot staff per division (satisfying cuts 1 and 2)
PAW > Ntuple/plot 10.division 1.and.2 -20 ! ! s
PAW > atitle 'Number of staff' | Axis title
PAW > close 2 | Close histogram file
PAW > set htyp 0 | Reset hatch style
PAW > hi/de 0 | Delete histograms from memory

```

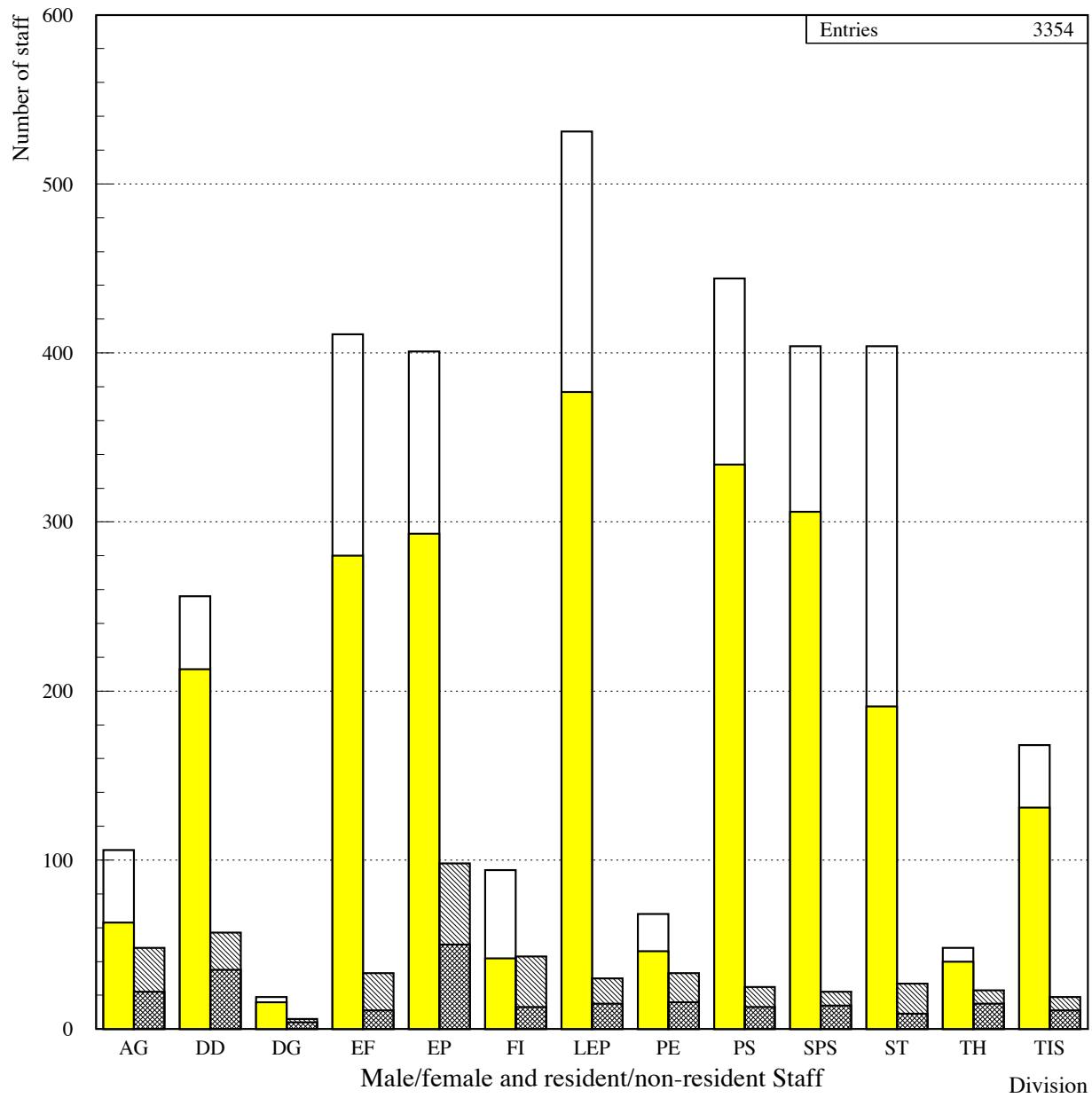


Figure 3.20: Ntuples - Using cuts

Ntuple – two-dimensional data presentation

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX21 *
PAW > * Two dimensional Ntuple distributions *
PAW > ****
PAW > hi/file 2 'aptuple.rzdat' | Open histogram file
PAW > clr | Clear screen
PAW > 2d 20 , , 12 3 15 16 0 16 0. | Book 2-dim histogram 20
PAW > Nt/project 20 10.step%grade | Project Ntuple onto histogram 20
PAW > *
PAW > * Lego plot of staff near end of grade (step &gt;7) and choose viewing angles
PAW > * Note that the 2-dim Ntuple projection and histogram sub-range specifications are reversed
PAW > *
PAW > lego 20(1:,7:) 20 40 | Plot resulting hist 20 as LEGO plot
PAW > close 2 | Close histogram file
PAW > hi/de 0 | Delete histograms from memory

```

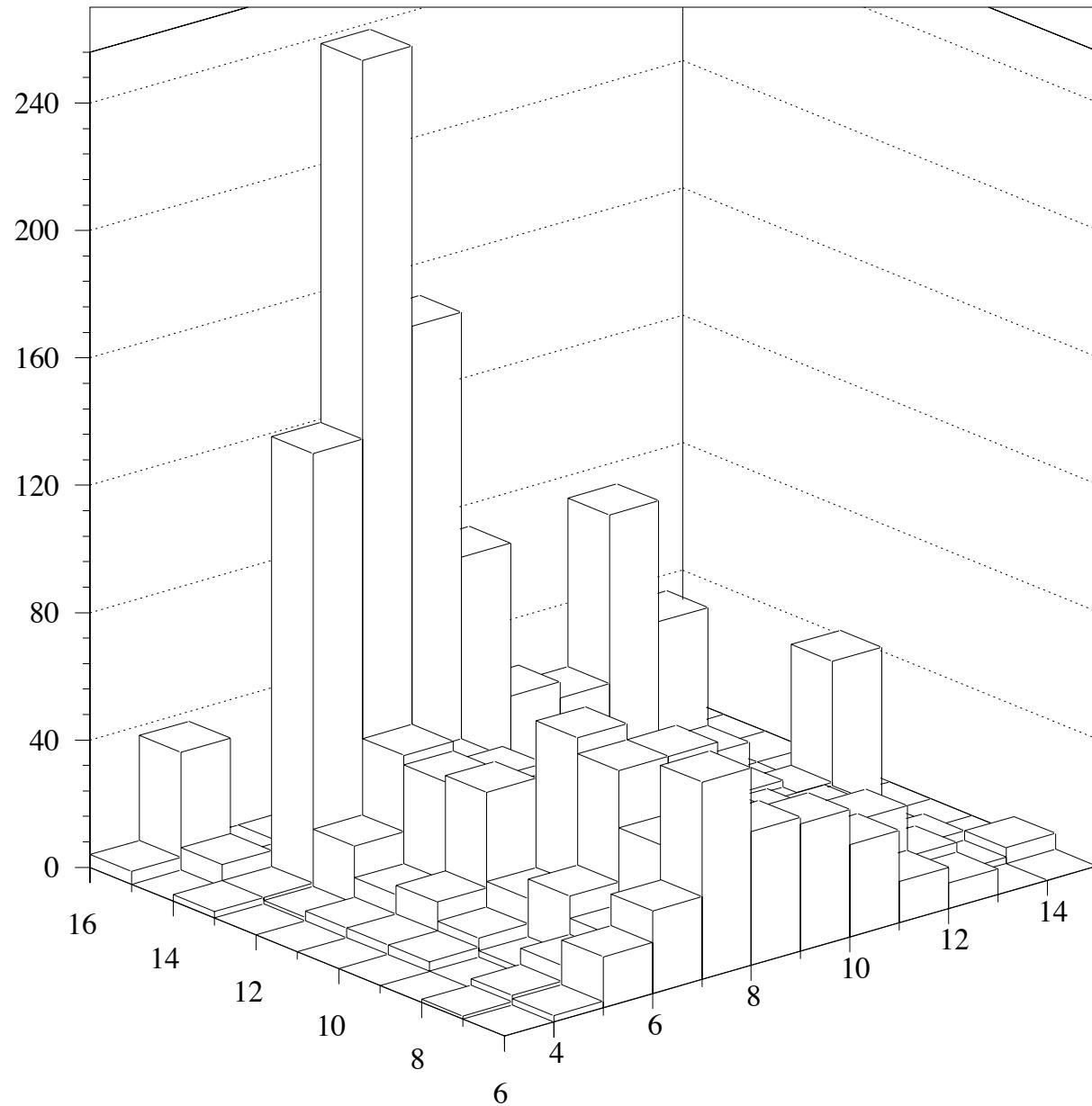


Figure 3.21: Ntuples - Two dimensional data representation

3.5 The SIGMA application and more complex examples

Using the SIGMA processor

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX22 *
PAW > * Examples of the SIGMA processor - 1 *
PAW > ****
PAW > zone 2_2 | Divide picture two by two
PAW > opt GRID | Draw grid on picture
PAW > SET NDVX 520 | Ask for 20 primary divisions
PAW > * -- Define array x with 200 elements values between 0 and 2 pi
PAW > SIGMA x=array(200,0#2*PI)
PAW > sigma sinus=sin(x) | sinus is array defined implicitly
PAW > sigma sinx=sin(x)/x | sinx ditto
PAW > gra 200 x sinus | Make graph of array sinus
PAW > set dmod 2 | New line style
PAW > gra 200 x sinx l | Make graph of array sinx
PAW > set dmod 0 | Default line style
PAW > * -- Define array x with 300 elements filled with values between 0 to 8
PAW > SIGMA x=array(300,0#8)
PAW > * -- Hyperbolic cosine oscillations are invisible
PAW > SIGMA g=cosh(x)+sin(1/(.1+X*X))
PAW > gra 300 X G | Plot the vectors x and g
PAW > * -- Define array x with 300 elements filled with values between 0 to 3
PAW > SIGMA x=array(300,0#3)
PAW > SIGMA g=cosh(x)+sin(1/(.1+X*X)) | Calculate the ordinate
PAW > gra 300 X G | Plot the vectors x and g
PAW > * -- Define array x with 300 elements filled with values between 0 to 1
PAW > SIGMA x=array(300,0#1)
PAW > SIGMA g=cosh(x)+sin(1/(.1+X*X)) | Calculate the ordinate
PAW > gra 300 X G | Plot vectors x against g

```

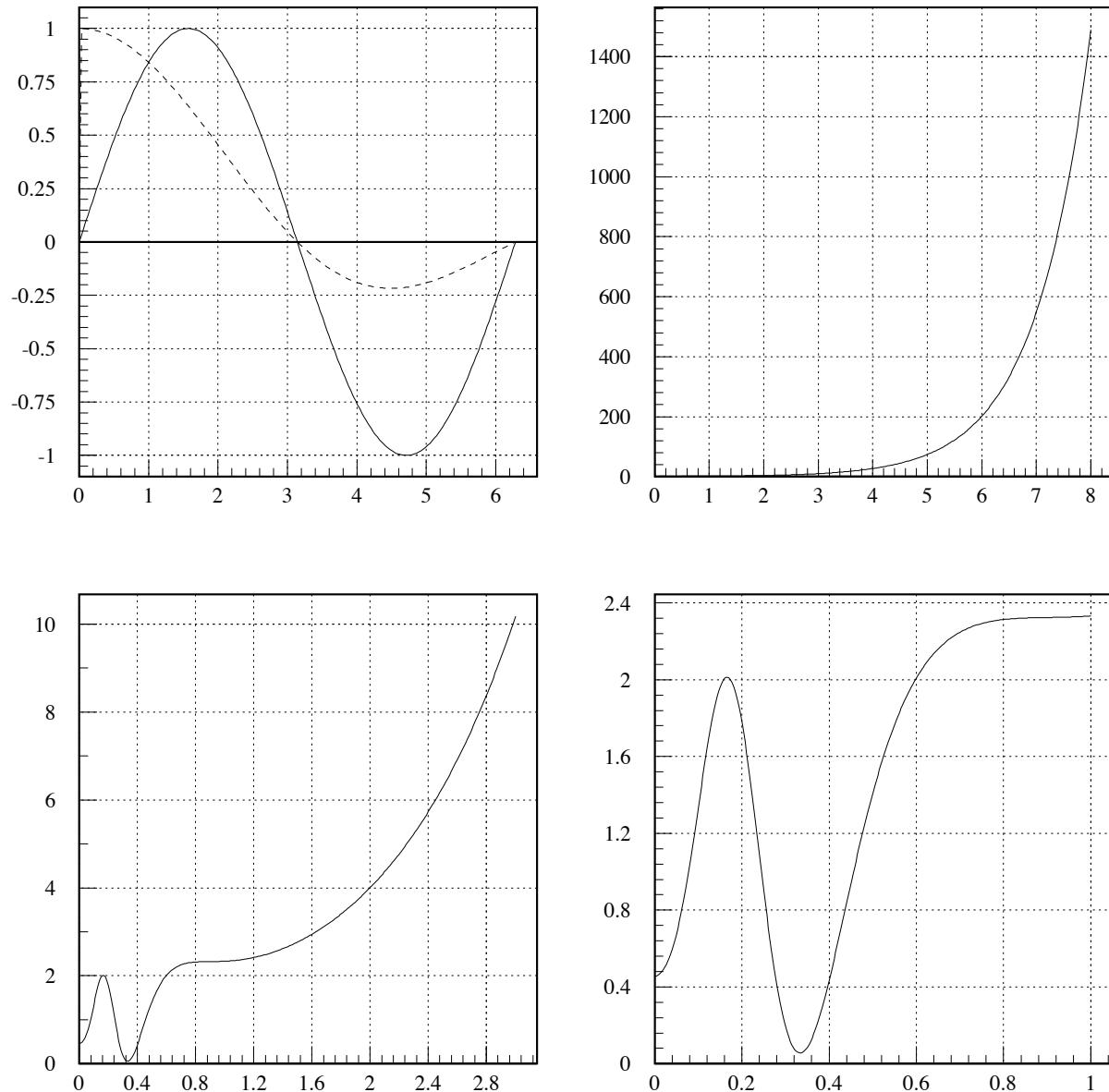


Figure 3.22: Using the SIGMA processor - Trigonometric functions

More examples of using the SIGMA processor

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX23 *
PAW > * Examples of the SIGMA processor - 2 *
PAW > ****
PAW > zone 2 2 | Divide picture two by two
PAW > * -- Define array x with 200 elements filled with values between 0 to 5
PAW > sigma x=array(200,0#5)
PAW > sigma A=8 | Assign scalar A
PAW > sigma B=.01 | Assign scalar B
PAW > sigma Y=EXP(-X)*SIN(A*X)+B*X*X | Y is an array function of A and B
PAW > opt GRID | Draw grid on picture
PAW > SET NDVX 520 | Ask for 20 primary divisions
PAW > gra 200 x y | 2-dim representation y versus x
PAW > opt NGR1 | No grid on picture
PAW > SIGMA x=array(200,0#2*pi) | Array between 0 and 2 pi
PAW > *
PAW > * -- Define some more arrays
PAW > *
PAW > SIGMA s=sin(x) | (automatic creation of array s)
PAW > SIGMA S2=S/2 | ditto s2
PAW > SIGMA c=cos(x) | ditto c
PAW > SIGMA c2=c/2 | ditto c2
PAW > SIGMA s4=s/4 | ditto s4
PAW > SIGMA c4=c/4 | ditto c4
PAW > *
PAW > * -- Plot all arrays on same plot
PAW > *
PAW > gra 200 s c
PAW > gra 200 s2 c 1
PAW > gra 200 s4 c 1
PAW > gra 200 s c2 1
PAW > gra 200 s2 c2 1
PAW > gra 200 s4 c2 1
PAW > gra 200 s c4 1
PAW > gra 200 s2 c4 1
PAW > gra 200 s4 c4 1
PAW > *
PAW > * -- An array of 100 elements with values between 0 and 59.77 (19*pi)
PAW > *
PAW > sigma a=array(100,0#59.77)
PAW > SIGMA NC=NCO(A) | interrogate SIGMA to know array length
PAW > SIGMA y=cos(a)*a | y becomes an array of length 100
PAW > SIGMA x=sin(a)*a | x becomes an array of length 100
PAW > GRA NC X Y | Plot x versus y
PAW > SIGMA a=a*2.55555 | Increase the amplitude
PAW > SIGMA y=cos(a)*a | y becomes an array of length 100
PAW > SIGMA x=sin(a)*a | x becomes an array of length 100
PAW > GRA NC X Y | Plot x versus y

```

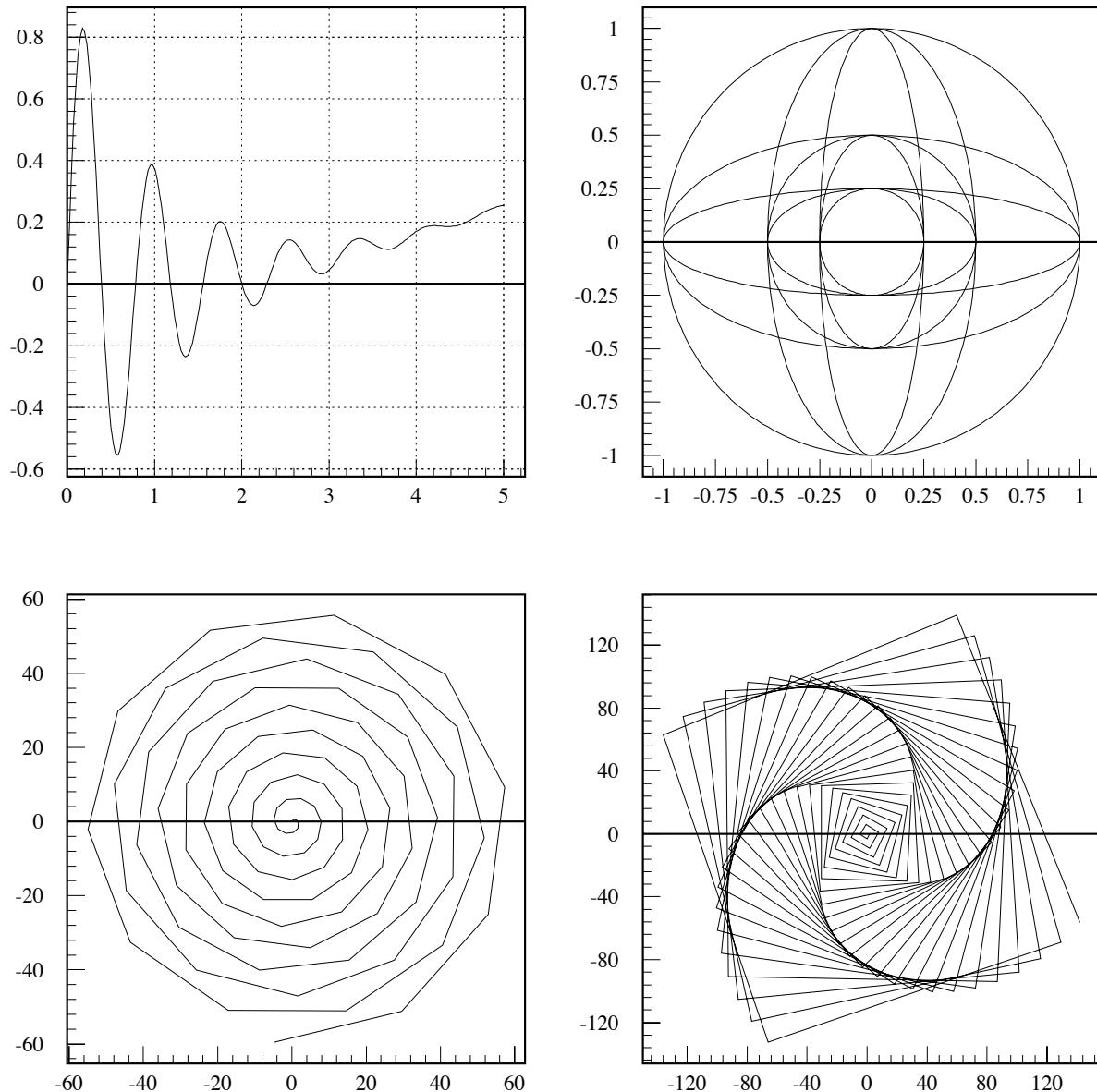


Figure 3.23: Using the SIGMA processor - More complex examples

Updating histogram contents

```
PAW > ****
PAW > *          TUTORIAL EXAMPLE PAWEX24      *
PAW > * Operations on histograms (Keep and Update)  *
PAW > ****
PAW > histogram/file 1 pawhists.rzdat           | Open histogram file
PAW > Zone 1 2                                | Divide plot in 2 vertically
PAW > opt grid                               | Draw grid on picture
PAW > set grid 1003                          | Horizontal grid only with line type 3
PAW > set hcol 1005                          | Histogram colour
PAW > h/pl 120 k                            | Plot hist 120 and keep in memory
PAW > set grid                               | Default grid (horizontal + vertical)
PAW > set htyp -3                           | Hatch style histogram colour
PAW > h/pl 110                             | Plot histogram 120
PAW > set hcol 1002                          | Histogram colour
PAW > h/pl 110 +                           | Add 110 on the last kept Histogram
PAW > set htyp 144                           | Histogram hatch style
PAW > h/pl 130 +                           | Add 130 on the last kept Histogram
PAW > zone                                 | Reset zone
PAW > close 1                            | Close input unit
```

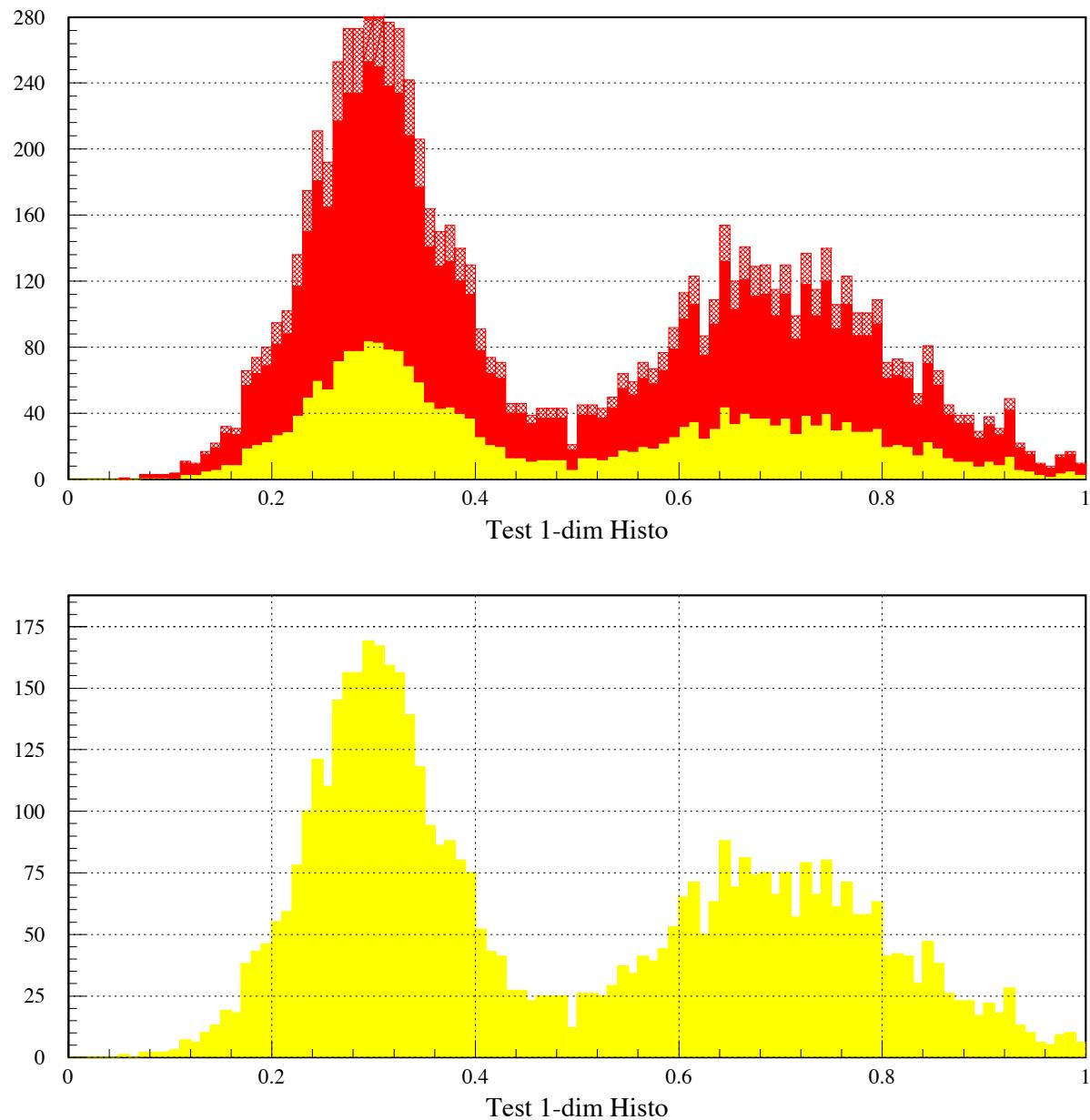


Figure 3.24: Histogram operations (Keep and Update)

Merging pictures

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX25 *
PAW > * Merge pictures onto one plot *
PAW > ****
PAW > edit PICTMERGE | Look at the macro for merging plots
Macro PICTMERGE HID=0 S1=1 S2=100
*****
* Macro Merge : Companion macro to PAW TUTORIAL EXAMPLE PAWEX25 *
*****
next | Initialise next picture
set * | Reset Graphics settings
opt * | Reset options
opt NBOX | No box around picture
set hwid 4 | Width of histogram lines
set bwid 4 | Width of box lines
set pwid 4 | Width of picture lines
igset lwid 4 | Width of lines
set vfon -1042 | Axes font (GKSGRAL font -104, precision 2)
set lfon -1042 | Axis labels font
swi Z | Z mode only
pic/cr MERGE2 | Create a picture with name MERGE2
set HTYP -3 | Set hatch type -3
hi/pl [hid] | Plot histo HISTID into picture MERGE2
set htyp 244 | Set histo hatch style
hi/pl [hid]([s1]:[s2]) s | Hatch sub-range of histo HISTID
pic/cr MERGE1 | Create next picture
set HTYP -3 | Reset hatch style
opt utit | No histogram title
opt grid | Draw grid on inset
hi/pl [hid]([s1]:[s2]) | Plot the subrange into picture MERGE1
opt ngri | Eliminate option grid
izpict MERGE2 c | Make MERGE2 current picture
pi/merge MERGE1 .45 .45 .47 D | Merge picture MERGE1 with current picture
swi G | G mode only
return
PAW > histogram/file 1 pawhists.rzdat | Open histogram file
PAW > EXEC PICTMERGE hid=110 s1=31 s2=40 | Plot histo 110 with subrange as inset
PAW > close 1 | Close input unit

```

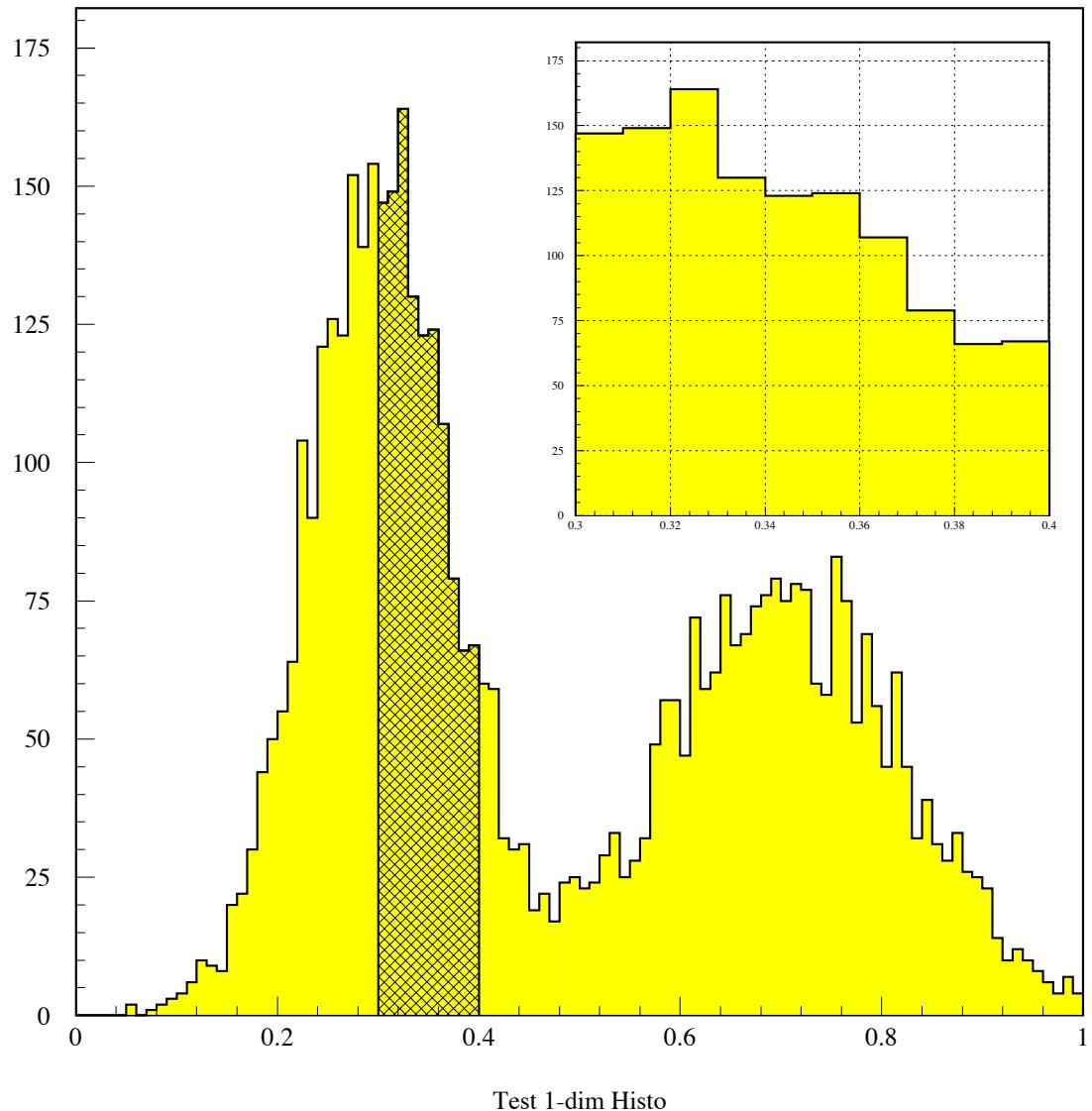


Figure 3.25: Merging several pictures into one plot

Pie chart and PostScript simulation

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX26 *
PAW > * Various forms of a PIE chart *
PAW > ****
PAW > edit PIE | Look at the macro for pie chart
MACRO PIE
*****
* Macro PIE : Companion macro to PAW TUTORIAL EXAMPLE PAWEX26 *
*****
alias/cre colbackg 0 | background colour
alias/cre colcompl 1 | complement of background
alias/cre colred 2 | red
alias/cre colgreen 3 | green
alias/cre colblue 4 | blue
alias/cre colyellow 5 | yellow
alias/cre colpurple 6 | purple
alias/cre colcyan 7 | cyan

v/cre values(5) R 28.3 18.6 16.9 13.5 22.7 | create vector with values
v/cre offset(5) R 2*0. 2*20. 0. | C
v/cre colour(5) R colred colgreen colblue colyellow colpurple
v/cre style(5) R 111 222 333 444 265
label 1 5 'Sun' 'DEC' 'HP' 'Apollo' 'Other'

igset bord 1 | Draw border on pie chart
zone 2 2 | Divide picture in 2 by 2
null 0 20 0 20 A | Initialize first picture part
igset fais 1
pie 10. 10. 7. 5 vws p offset ! colour
null 0 20 0 20 A
igset fais 1
pie 10. 10. 7. 5 vws l offset ! colour
null 0 20 0 20 A
pie 10. 10. 7. 5 vws n offset style
null 0 20 0 20 A
pie 10. 10. 7. 5 vws l offset style
al/de *
RETURN
PAW > EXEC PIE | Execute macro for pie and bar chart

```

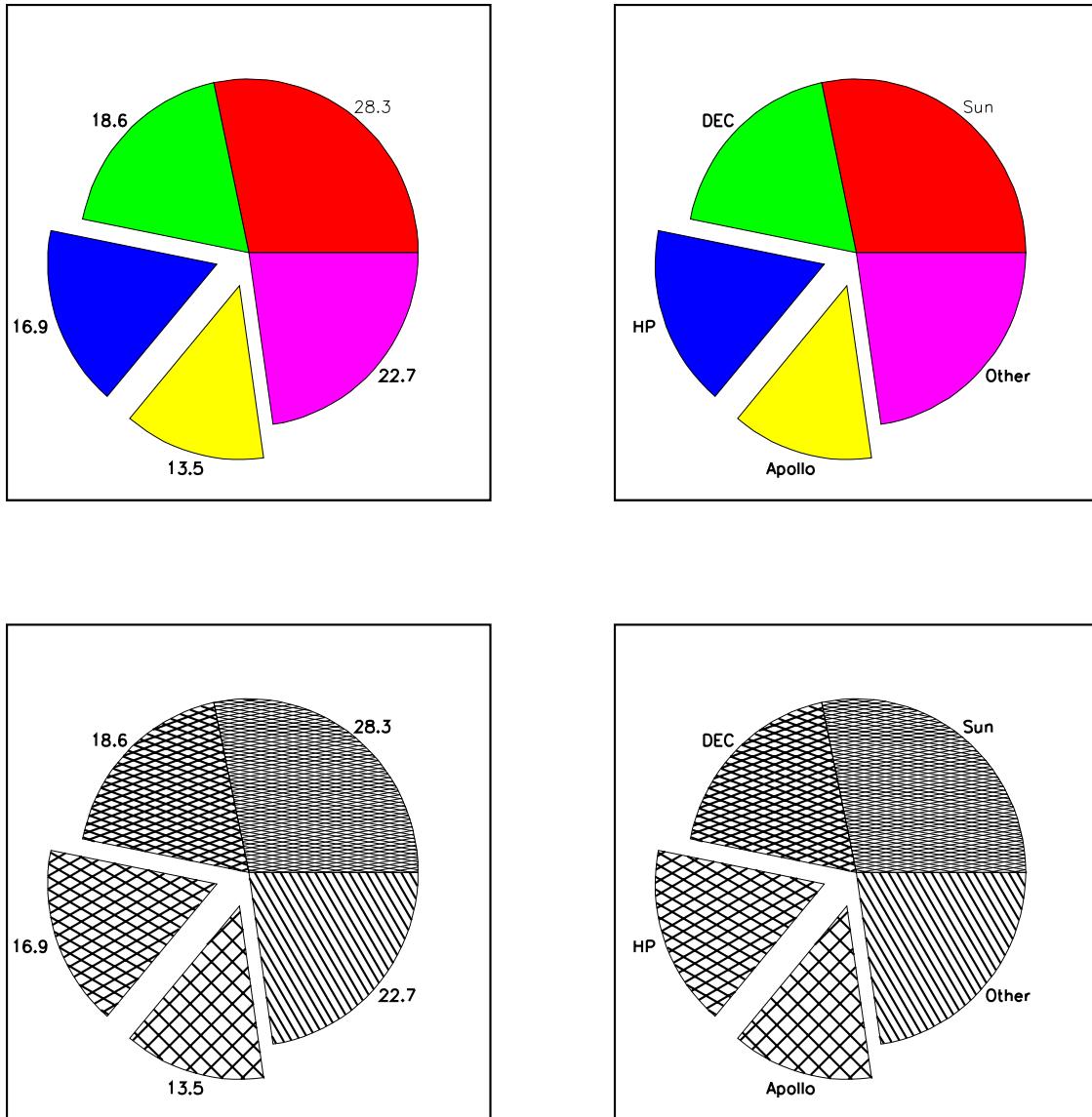


Figure 3.26: Pie charts with hatch styles and PostScript colour simulation

Making a complex graph with PAW

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX27 *
PAW > * Make a complex graph with PAW using PostScript fonts *
PAW > ****
PAW > edit COMPTIME | Look at the macro for computer time
MACRO COMPTIME
*****
* Companion macro to PAW TUTORIAL EXAMPLE 27 *
*****
OPT * | Reset options
SET * | Reset HPLOT settings
igset * | Reset HIGZ settings
ve/de * | Delete all vectors
OPT NBOX | No box on plot
OPT LOGY | Logarithmic plot in y
OPT TIC | Tic marks
OPT UTIT | User title
opt ZFL1 | Keep picture in memory
size 16 20 | Size of picture
set hwid 4 | Width of histogram lines
set fwid 4 | Width of fit lines
set bwid 4 | Width of box lines
set pwid 4 | Width of picture lines
igset lwid 4 | Width of lines
set VSIZ 0.20 | Size of axis values smaller
set YGTI 1.2 | Y position of global title
set XVAL 0.4 | Distance y axis to axis values
set YVAL 0.2 | Distance x axis to axis values
set XLAB 1.0 | Distance y axis to label
set YLAB 0.8 | Distance x axis to label
set XTIC 0.15 | Length x axis tick marks
set YTIC 0.15 | Length y axis tick marks
set ASIZ 0.26 | Size of axis text
set GSIZ 0.35 | Global title size
set gfon -60 | Global title font
set vfon -60 | Font of axis values
set lfon -10 | Font of axis labels
igset lwid 4 | Quadruple line width
title_gl 'CERN Central Computer Usage' | Global title of picture
* -- IBM168 equivalent hours used for each year since 1960
vector/create vy(30) R 9.2 11.8 34.9 60.7 87.1 217.8 360 1250 2500 4006 _
4478 5590 5910 6246 10879 12849 18429 19481 21171 25005 _
31219 33928 37057 45520 57000 75957 98806 118993 131800 151138
sigma vx=array(30,60#89) | Define vector of x values 60 to 89
ve/cre f1(2) r 2*0.0 | Create vector V1 for fit 1 result
ve/cre f2(2) r 2*0.0 | Create vector v2 for fit 2 result
* -- Exactly 30 divisions in X with labels centered on divisions
SET NDVX -30.05
NULL 60 90 5 250000 | Draw empty plot with given limits
igset MSCF 0.75 | Specify marker scale factor
igset mtyp 21 | Data points with polymarker
graph 30 vx vy p | Plot data points
ve/fi vx(:10) vy(:10) ! e ws ! f1 | Fit first part of data points
ve/fi vx(10:) vy(10:) ! e ws ! f2 | Fit second part of data points
arrow 64. 62. 10. 10. 0.15 | Draw arrow
igset txal 20 | Center text horizontally
igset chhe 0.18 | Character height
itx 63. 12. 'IBM 709' | Write text
arrow 65. 63. 35. 35. -0.11 | Draw arrow
itx 64. 40 'IBM 7090' | Write text
arrow 75. 65. 150. 150. -0.11 | Draw arrow
itx 70. 200. 'CDC 6600' | Write text
arrow 85. 72. 4000. 4000. -0.11 | Draw arrow
itx 78.5 4500. 'CDC 7600' | Write text
arrow 82. 78. 6500. 6500. -0.11 | Draw arrow

```

```

itx 80. 7500. 'IBM 168'
arrow 81. 79. 10000. 10000. -0.11
itx 80. 12000. 'IBM 3032'
arrow 85. 81. 18000. 18000. -0.11
itx 83. 20000. 'IBM 3081'
igset txal 10
arrow 84. 82. 27000. 27000. -0.11
itx 82. 30000. 'SIEMENS 7880'
igset txal 20
arrow 90. 84. 42000. 42000. 0.11
itx 87. 50000. 'SIEMENS 7890'
arrow 90. 85. 68000. 68000. 0.11
itx 87.5 72000. 'IBM 3090'
arrow 90. 88. 100000. 100000. 0.11
itx 89. 110000. 'CRAY'

arise=$sigma(int((exp(f1(2))-1)*100+0.5))//'% per Annum rise'
xhand=68.
yhand=$sigma(exp(f1(1)+f1(2)*[xhand]))
EXEC DRAW X=[xhand] Y=[yhand] TEXT=[arise]
arise=$sigma(int((exp(f2(2))-1)*100+0.5))//'% per Annum rise'
xhand=84.
yhand=$sigma(exp(f2(1)+f2(2)*[xhand]))
EXEC DRAW X=[xhand] Y=[yhand] TEXT=[arise]
atitle 'Year ' 'IBM 168 Units used '
RETURN

MACRO DRAW
igset TXAL 30
igset TANG -35.
igset TXFP -140
igset CHHE 0.50
itx $SIGMA([X]-0.9) [Y] +
igset TXAL 30
igset TANG 0.
igset TXFP -30
igset CHHE 0.22
y = y * 1.70
itx [X] [Y] [TEXT]
RETURN
PAW > EXEC COMPTIME
| Execute macro for computer time

```

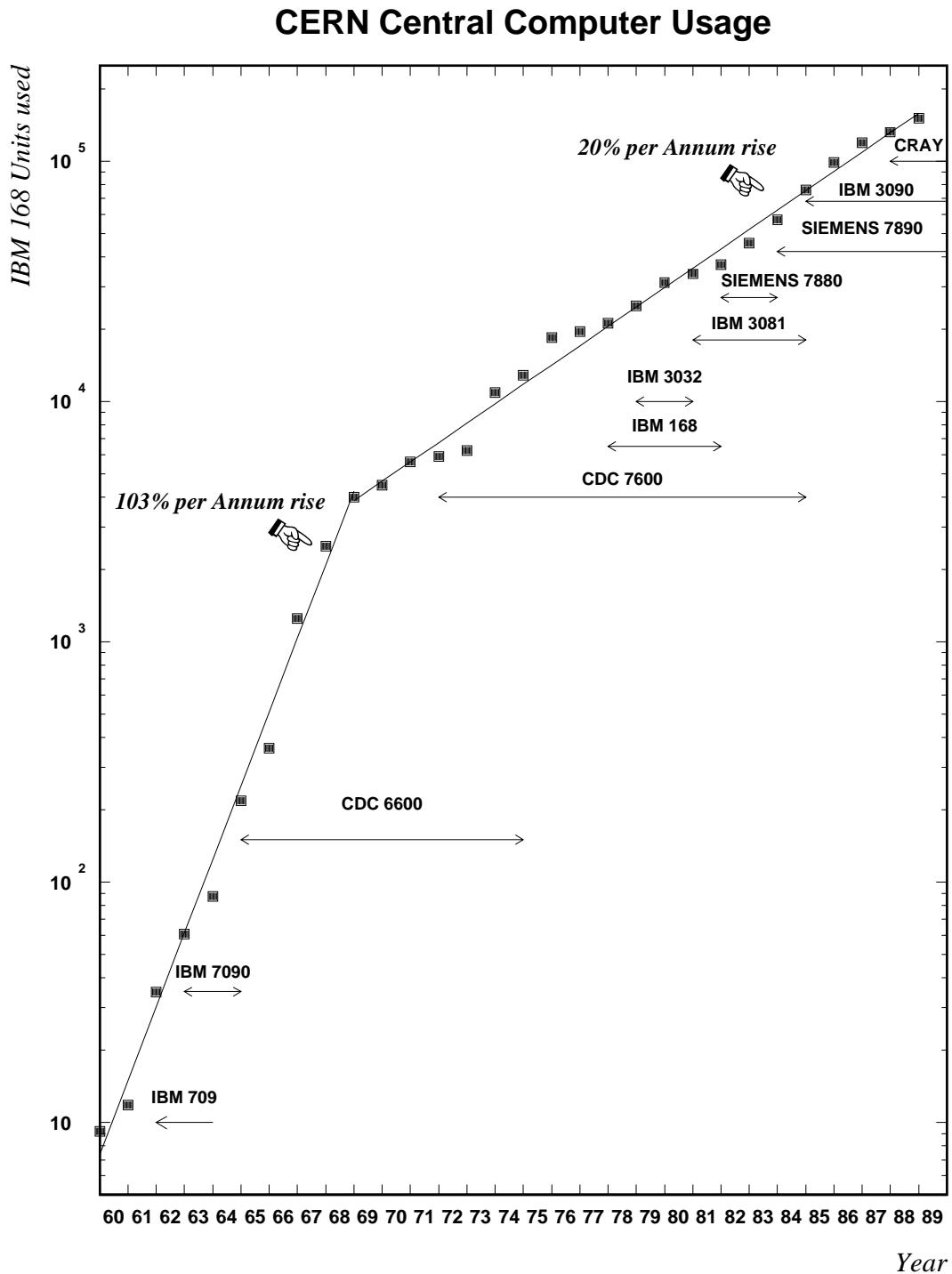


Figure 3.27: A complex graph with PAW

PAW, PostScript and making slides

```

PAW > ****
PAW > * TUTORIAL EXAMPLE PAWEX30 *
PAW > * Making slides with PAW and PostScript *
PAW > ****
PAW > edit slide | Look at master macro to produce slide
macro SLIDE xsize=18 ysize=22 width=0.4 name='Author/CERN CONF99' sn=' ' title=' '
*****
* PAW TUTORIAL EXAMPLE 30 - SLIDE *
* General macro to draw contours of SLIDE *
*****
xmax = [xsize]-[width] | X range for slide
ymax = [ysize]-[width] | Y range for slide
size [xsize] [ysize] | Total size for slide
next | Initialize next frame
igset lwid 2 | Double line width
pave 0 [xmax] 0 [ymax] [width] 0 1005 tr | Draw a shaded box around the picture
igset lwid 1 | Single line width
xtitle = $sigma(([xsize]-0.2)/2.) | X coordinate for title
ytitle = [ysize]-1.5 | Y coordinate for title
igset txfp -70 | Defined new font (Helvetica-Bold-Oblique)
igset txal 20 | Text alignment Hor-Centered / Vert-Base
igset chhe 0.6 | Set character height
itx [xtitle] [ytitle] [title] | Write text (title)
igset chhe 0.3 | Set character height
igset txal 10 | Text alignment Hor-Left / Vert-Base
xtext = [xmax]-0.2 | X coordinate for name
ytext = 0.1 | Y coordinate for name
igset txfp -1042 | Defined new font (Helvetica-Bold-Oblique)
igset chhe 0.2 | Set character height
igset txal 30 | Text alignment Hor-Right / Vert-Base
itx [xtext] [ytext] [name] | Write text (name)
igset txal 10 | Text alignment Hor-Left / Vert-Base
itx 0.1 0.1 [sn] | Write text (slide name)
igset chhe 0.3 | Set character height
igset lwid 2 | Double line width
return

PAW > edit discomp | Look at example macro showing use of slide
MACRO DISCOMP
*****
* PAW TUTORIAL EXAMPLE 30 - DISCOMP *
* Text for slide example DISCOMP *
*****
exec slide sn='DisComp' title='Distributed Computing' | Define the outline of the slide
igset txfp mainfont | Defined font via alias
igset chhe 0.5 | set character height
itx 2 17 'With a distributed operating system (not yet !)' | Write text
itx 2 15 'With tools on top (RPCs, NCS,... ?)' | Write text
igset chhe 0.4 | Set character height
itx 3 14 Tmess | Write text
itx 3 13 Tfork | Write text
itx 3 12 Tdata | Write text
itx 3 11 Tcomp | Write text
* -- Defined new font (Helvetica-Bold-Oblique)
igset txfp -70
itx 5 14 'Time to send message to remote process' | Write text
itx 5 13 'Time to fork a process' | Write text

```

```

itx 5 12 'Time to pass data (in and out)' | Write text
itx 5 11 'Time used for computation on remote process' | Write text
igset txfp mainfont | Defined font via alias
pave 2 16 2 9 0.3 0 1001 trs | Draw 'pave' (shaded box)
* -- Text alignment Hor-Right / Vert-Centered
igset txal 33
* -- Defined new font (Helvetica-Bold-Oblique)
itx 6 7 'Efficiency =' | Write text
* -- Text alignment Hor-Centered / Vert-Base
igset txal 20
line 6.1 7 14.1 7 | Draw line
itx 10 7.2 Tcomp | Write text
itx 10 6.3 'Tcomp + Tmess + Tfork + Tdata' | Write text
* -- Defined new font (ZapfDingbats-Bold)
igset txfp -240
igset chhe 0.6 | Set character height
* -- Text alignment Hor-Right / Vert-Base
igset txal 30
itx 1.5 17 P | Write text
itx 1.5 15 P | Write text
igset chhe 0.3 | Set character height
* -- Text alignment Hor-Centered / Vert-Base
igset txal 20
* -- Defined new font (Helvetica-Bold-Oblique)
igset txfp -70
itx 9 4 'Many time consuming applications today have:' | Write text
itx 9 3 'Efficiency > 0.9' | Write text
return

PAW > * Alias for main text font and precision (PostScript Helvetica-Bold) for slide text
PAW > alias/create mainfont -60
PAW > * Alias to print PostScript metafile (See page~158, where macro POST is described)
PAW > alias/create @ 'exec post'
PAW > opt zfl1 | Keep last picture in memory
PAW > exec discomp | Exec the slide macro containing the slide
PAW > @ | Print generated slide
PAW > alias/delete *

```

Distributed Computing

★ With a distributed operating system (not yet !)

★ With tools on top (RPCs, NCS,.. ?)

Tmess *Time to send message to remote process*

Tfork *Time to fork a process*

Tdata *Time to pass data (in and out)*

Tcomp *Time used for computation on remote process*

$$\text{Efficiency} = \frac{\text{Tcomp}}{\text{Tcomp} + \text{Tmess} + \text{Tfork} + \text{Tdata}}$$

Many time consuming applications today have:

Efficiency > 0.9

Figure 3.28: Making slides with PAW using PostScript

Part II

PAW - Commands and Concepts

Chapter 4: User interface - KUIP

4.1 The PAW command structure

All PAW commands may be seen as a path along the PAW tree structure:

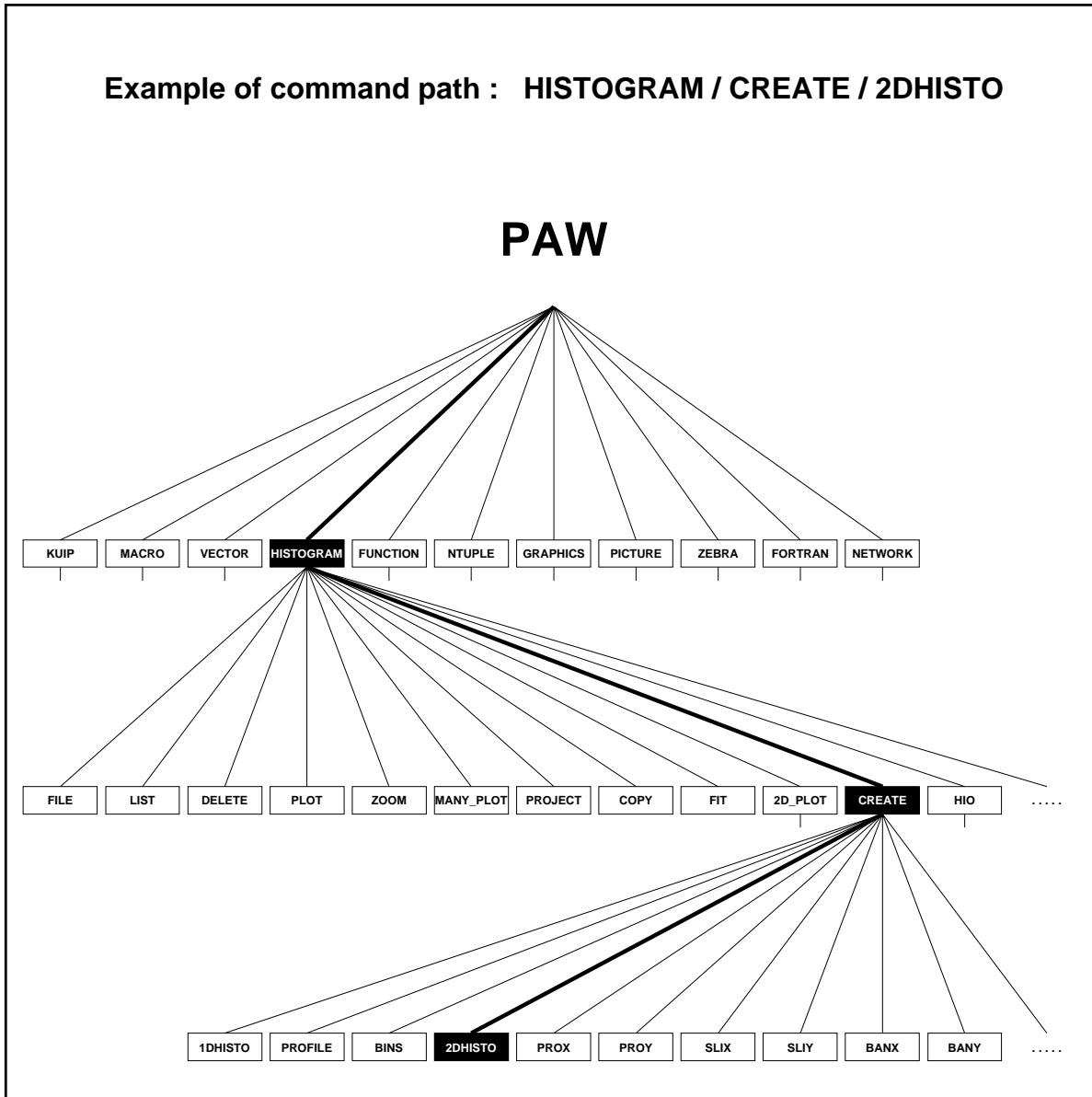


Figure 4.1: Example of the PAW command tree structure

4.2 Multiple dialogue styles

PAW is based on the KUIP [5] User Interface package, which can provide different types of dialogue styles. It is possible to change interactively from one style to another using the command `STYLE`.

4.2.1 Command line mode

In command mode the user enters a command line via the terminal keyboard.

The general syntax of a `command_line` is a `command_name` optionally followed by a `parameter_list`. The `command_name` and `parameter_list` are separated by one or more blanks (therefore, no blanks should appear within the `command_name`). Using regular expressions notation one can write:

```
command_line ::= command_name ( blank+ parameter_list )?
```

where the postfix unary operator `'+'` means “one or more instances of the postfixed item” and `'?'` means “zero or one instances of the postfixed item”. The parameters in the `parameter_list` are again separated by one or more blanks:

```
parameter_list ::= parameter ( blank+ parameter )*
```

where `'*'` means “zero or more instances of the postfixed item”. No blanks should then appear within a parameter, unless the whole parameter is enclosed in single quotes, like for example “This parameter has blanks” or the blank filled parameter `' '`.

The command name is a structured name representing the path along the inverted tree structure handled by KUIP. Each element of the path, called `command_element`, is separated from the others by one slash:

```
command_name ::= command_element ( / command_element )*
```

The rightmost `command_element` of a `command_name` must be a leaf of the tree, i.e. a terminal `command_element`, while the others are considered menus. The `command_name` can have up to 10 levels of command elements (i.e. 9 levels of menus).

Command abbreviations

A command can always be abbreviated, as long as it does not become ambiguous with other commands, by omitting:

- the leftmost command elements
- the rightmost characters of a command element

The shortest unambiguous abbreviation for any command is not fixed, but depends on the whole command tree structure: KUIP takes care to list all possible ambiguities should the user enter an ambiguous command.

The list of all executable commands can be obtained just by typing one slash. This is a command line having a null command element both to the left and to the right of the separator slash; by definition a null command element is ambiguous with every non-null command element, therefore all the available commands will be listed as possible ambiguities.

Examples of ambiguous commands

```
PAW > CONT
*** Ambiguous command. Possible commands are :
/HISTOGRAM/2D_PLOT/CONTOUR
/HISTOGRAM/GET_VECT/CONTENTS
/HISTOGRAM/PUT_VECT/CONTENTS
PAW > PUT_VECT/CONTENTS
Histogram Identifier (<CR>=110): 100
Vector name (<CR>=XYZ): VEC
```

```
PAW > P/C
*** Ambiguous command. Possible commands are :
/HISTOGRAM/PUT_VECT/CONTENTS
/PICTURE/CREATE
/PICTURE/COPY
```

```
PAW > P/CO
*** Ambiguous command. Possible commands are :
/HISTOGRAM/PUT_VECT/CONTENTS
/PICTURE/COPY
PAW > PU/C
Histogram Identifier (<CR>=100): 110
Vector name (<CR>=VEC): VV
```

The shortest unambiguous abbreviation for any command is not fixed, but depending on the whole command tree structure: KUIP takes care to list all possible ambiguities should the user have entered an ambiguous command.

The list of all executable commands can be obtained just by typing one slash. This is a command line having a null command element both to the left and to the right of the separator slash; by definition a null command element is ambiguous with every non-null command element, therefore all the available commands will be listed as possible ambiguities.

Parameters

As explained above, a command line consists of a **command** part optionally followed by a **parameter** part. For example, the PAW command NTUPLE/LIST has no parameters, while NTUPLE/PRINT has one parameter, i.e. the Ntuple identifier.

Using the USAGE command

```
PAW > USAGE NTUPLE/LIST
* NTUPLE/LIST

PAW > USAGE NTUPLE/PRINT
* NTUPLE/PRINT IDN
```

Parameters can be **mandatory** or **optional**. For example the command ZEBRA/DZ/STORE has one optional parameter, i.e. the “ZEBRA store number”. An optional parameter always has a **default value**, which is used when the user does not specify the parameter. In the example above the default value is 0, therefore entering just STORE is equivalent to STORE 0.

On the other hand the command ZEBRA/FZ/TOALPHA has one mandatory parameter, i.e. the name of a FZ text file. If the user enters just TOALPHA, he will be prompted also for the file name:

```
PAW > TOALPHA
Name of the FZ text file (<CR>=FF.DAT) : GG.DAT
```

The **order** of parameters in the command line is important and must match the semantic definition of the command. Mandatory parameters are always specified before any optional parameters.

An **exclamation mark** may be used as default value filler character. As an example consider the following PAW command:

```
PAW > USAGE NTUPLE/PLOT
* NTUPLE/PLOT IDN [ UWFUNC NEVENT IFIRST NUPD CHOPT ]
```

which has one mandatory and five optional parameters. If only the fourth parameter, IFIRST, needs to be specified (hence taking the default values for all other optional parameters), then one may enter:

```
PAW > NTUPLE/PLOT 30 ! ! 1000
```

Parameters can be entered in command lines also by their name, i.e. independently from their position. This is particularly useful when an optional parameter has to be specified for a command with several optional parameters. Values are assigned to parameters by indicating the name of the parameter, followed by an equal sign, followed by the value, with no blanks in between (see first line of the example below). When the parameter's name is CHOPT a shortcut is possible: a minus sign preceding a (non-numerical) value means 'CHOPT=' (see third and fourth line of the example below). If a parameter (with no NAME=) is specified after a named parameter, it will refer to the parameter following the named one (see second line of the example below).

For example, consider the following command:

```
NTUPLE/PLOT IDN [ UWFUNC NEVENT IFIRST NUPD CHOPT ]
```

One could then enter:

<u>N/PL 123.x NUPD=100</u>	instead of <u>N/PL 123.x ! ! ! 100</u>
<u>N/PL NEVENT=1000 500</u>	instead of <u>N/PL idn ! 1000 500</u> (idn must be given interactively)
<u>N/PL 123.x CHOPT=B</u>	instead of <u>N/PL 123.x ! ! ! ! B</u>
<u>N/PL 123.x -B</u>	same as above

Note that, unlike command elements, parameter names cannot be abbreviated.

4.2.2 An overview of KUIP menu modes

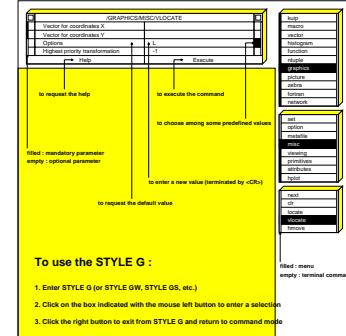
Only a short overview is given here. See the KUIP manual for more details.

Alphanumeric, entered by STYLE AN or STYLE AL.

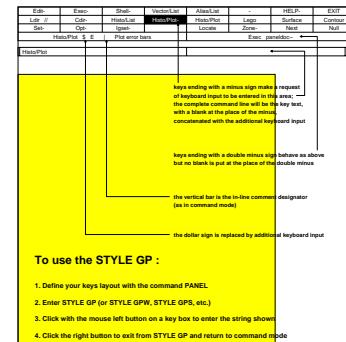
The desired command is selected from a list by number or by letter.

Graphics, entered by STYLE G or STYLE GP. This mode is particularly interesting for workstations. It should not be used with simple terminals.

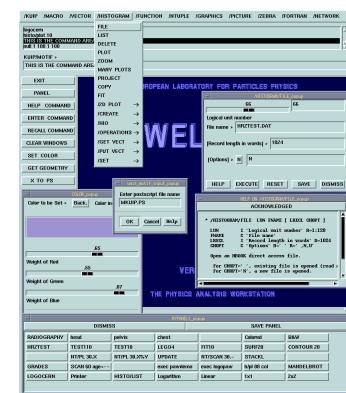
- **STYLE G:** Pull-down menus, fixed layout, reflecting the command structure;



- **STYLE GP:** Panels of function keys, allowing interactive user definable multiple layouts.



Motif This mode, to be released soon, is suited for X11 based WorkStations or X-terminals.



User:

This mode must be used in conjunction with routine KUSER (see KUIP manual).

4.3 Macros

A macro is a set of command lines stored in a file, which can be created/edited with a local editor and executed with the command EXEC. For example the command

```
PAW > EXEC MNAME
```

executes the command lines contained in the macro file MNAME. As a macro file can contain several macros, a dash sign (#) is used to select a particular macro inside a file:

- If MNAME does not contain the character '#', the file MNAME.KUMAC is searched and the first macro is executed (it may be an unnamed macro if a MACRO statement is not found as first command line in the file)
- If MNAME is of the form FILE#MACRO, the file named FILE.KUMAC is searched and the macro named MACRO is executed

Example of macro calls

```
PAW > EXEC ABC | Execute first (or unnamed) macro of file ABC.KUMAC
PAW > EXEC ABC#M | Execute macro M of file ABC.KUMAC
```

In addition to all available KUIP commands the special “macro statements” in table 4.1 are valid only inside macros (except for EXEC, which is valid both inside and outside). In the last line of the table par stands for either an argument passed with the command EXEC (in the command mode or from another macro) or a local variable of the macro.

`arithmetic_expression` and `logical_expression` are expressions with only two terms, which are defined as follows (“|” stands for “or” and juxtaposition stands for “and”):

<code>operand</code>	=	<code>par</code> <code>constant</code>
<code>arithmetic_expression</code>	=	<code>operand arithmetic_operator operand</code>
<code>arithmetic_operator</code>	=	<code>+</code> <code>-</code> <code>*</code> <code>/</code>
<code>logical_expression</code>	=	<code>operand logical_operator operand</code>
<code>logical_operator</code>	=	<code>=</code> <code><</code> <code><=</code> <code>></code> <code>>=</code> <code><></code>

A label is any string in a line that is terminated by a colon (therefore labels must stand alone on a line). A label definition is local to a macro, so that the same label can be re-used in different macros.

The ON ERROR GOTO statement is activated by error conditions of the system and by the application program¹. In executing a macro, the latest ON ERROR GOTO executed is the active one (i.e. the previous one is superseded).

¹More precisely, after execution of a line inside a macro, the variable IQUEST(1) (in COMMON/QUEST/IQUEST(100)) is checked. If it is different from 0, then the ON ERROR GOTO logic is triggered.

Macro Statements	
STATEMENT	DESCRIPTION
MACRO mname par1=val1 ...	begin macro mname
EXEC mname par1 par2=val2 ...	execute macro mname
RETURN	end of a macro
READ par	read macro parameter par from keyboard
SHIFT	control parameters list
label:	label (must terminate with a colon)
GOTO label	jump to label
ON ERROR GOTO label	resume at label on error condition
OF ERROR	temporarily deactivate the ON ERROR GOTO handling
ON ERROR	reactivate the latest ON ERROR GOTO handling
IF logical_expression GOTO label	conditional statement
IF-THEN, ELSEIF, ELSE, ENDIF	Macro flow control
CASE, ENDCASE	Macro flow control
WHILE-DO, ENDWHILE	Macro flow control
REPEAT, UNTIL	Macro flow control
DO, ENDDO	Macro flow control
FOR, ENDFOR	Macro flow control
BREAKL	Macro flow control
EXITM	Macro termination
par = arithmetic_expression	assignment statement

Table 4.1: List of statements possible inside KUIP macros

Positional parameters can be passed to a macro, separated by blanks. Inside a macro, positional parameters can be retrieved by including in brackets the number representing their order in the list.

Example of macro file	Example of macro execution
<pre>MACRO ABC MESSAGE [1] [3] [2] RETURN</pre>	<pre>PAW > EXEC ABC P1 123 'This is P3' P1 This is P3 123</pre>

Note that normal variables are not translated if they have not been assigned a value, whereas unassigned positional parameters are **always** replaced by the blank character ' '. Macro parameters can be concatenated to anything in the command line; whenever a parameter number (or name - see below), enclosed in brackets, is encountered in the command line, it will be substituted by its value before execution of the command line.

Example of macro file	Example of parameter substitution
<pre>MACRO OPEN HISTO/FILE 20 DST[1].DAT RETURN</pre>	<p>PAW > <u>EXEC OPEN 123TEST</u> will execute the command: HISTO/FILE 20 DST123TEST.DAT</p>

Non-positional (i.e. named) parameters can also be passed. This is useful when several parameters are associated to a macro. Initial values of parameters should be specified in the MACRO statement. For example, changing the macro OPEN above to:

Example of macro with lot of parameters	
<pre>MACRO OPEN LUN=20 NAME=JUNK EXT=DAT LRECL=1024 CHOPT=' ' HISTO/FILE [LUN] [NAME].[EXT] [LRECL] [CHOPT] RETURN</pre>	

Example of macro call	Output generated by macro call
PAW > <u>EXEC OPEN EXT=TEMP LUN=10</u>	HISTO/FILE 10 JUNK.TEMP 1024

Parameters can also be read in at macro run time. When a READ statement is executed the user will be asked to provide values for the given parameters. If just <CR> is entered, the values remain unchanged.

Example of macro reading parameters at run time
<pre>MACRO INP READ PPP READ 1 MESSAGE 'The value of the parameter PPP is ... ' [PPP] MESSAGE 'The value of the parameter 1 is ' [1] RETURN</pre>

The order of priority for macro parameters is such that the values given in the EXEC statement supersede those given in the MACRO statement.

4.3.1 Special Parameters

The following Three special parameters are always defined inside any macro:

- [#] number of arguments given to the macro in the EXEC command which called it.
- [*] String containing the arguments given to the macro in the EXEC command, separated by spaces.
- [@] Return code (see the description of EXITM) of the last macro called by the current one (0 if no macro has been called).

In addition, it is possible to use **indexed positional parameters** of the form:

- [%var] var is a variable with an integer value. This accesses the positional parameter corresponding to the value of var. If var does not have an integer value then parameters of this form will not be replaced by a value. This can be used in conjunction with the parameter [#] to loop through all of the parameters given to a macro.

Note that positional parameters may not be assigned values using this form.

4.3.2 Macro Flow Control

There are several constructs available for controlling the flow of macro execution, which include conditional statement blocks, several looping constructs and variable assignments.

Assignments

Assignments to a variable simply take the form

```
variable = expression
```

where **variable** is the name of the variable to be assigned, and **expression** is the expression which is evaluated to obtain the new value of **variable**.

Inside a macro, values may be assigned to variables without distinction of their type: an automatic mechanism is used to distinguish between integer, real or character type variables.

Example of macro	Output when executing
<pre>MACRO DOC1 A = 10 NN = 1.5 TOT = [A]+[NN] IF [TOT] > 11 THEN MESSAGE Sum of [A] and [NN] is [TOT] AOK = correct ELSE AOK = wrong ENDIF MESSAGE KUIP arithmetic is [AOK]. RETURN</pre>	<pre>PAW > EXEC DOC1 Sum of 10 and 1.5 is 11.5 KUIP arithmetic is correct.</pre>

Unassigned variables cannot be substituted by their values.

Example of macro	Output when executing
<pre>MACRO DOC2 A = 10 NN = 1.5 TOT = [A]+[XX] MESSAGE Result of sum is [TOT] RETURN</pre>	<pre>PAW > EXEC DOC2 Result of sum is 10+[XX] XX is unassigned, hence no value is substituted.</pre>

The right hand side of an assignment command may be a vector name with an optional subscript, as in the following.

Example of a macros containing subscripted vector

```
MACRO DOC3
A=10
IF $VEXIST(VV)>0 THEN
  VEC/DEL VV
ENDIF
VEC/CRE VV(5)
VEC/INP VV 10 20 30 2*0
VECVAR=VV
MESSAGE First component of vector VV is [VECVAR]
VECVAR=VV(2)
MESSAGE Second component of vector VV is [VECVAR]
VECVAR=VV($VLEN(VV,1))
MESSAGE Last non-zero component of vector VV is [VECVAR]
VECVAR=VV($VDIM(VV,1))
MESSAGE Last component of vector VV is [VECVAR]
RETURN
```

Output generated when running DOC3

```
PAW > EXEC DOC3
First component of vector VV is 10
Second component of vector VV is 20
Last non-zero component of vector VV is 30
Last component of vector VV is 0
```

Note that if no subscript is given, the first component of the vector is used.

4.4 Aliases

Aliases are defined to provide shortcut abbreviations for the input line (either in the command elements or in the parameter list) or for some part of it. An alias name can be any string of characters (except the single quote and the blank) and whenever encountered in an input line it will be replaced literally by its value (another string of characters). Alias substitution does not apply in quoted strings. Aliases are defined by using the command ALIAS/CREATE.

Example of a KUIP session

```
PAW > ALIAS/CREATE M7 'EXEC MACRO7'
PAW > ALIAS/CREATE PP '10 20 30'
PAW > ALIAS/LIST
M7 => EXEC MACRO7
PP => 10 20 30
PAW > M7PP
*** Unknown command
PAW > M7 PP
... Executing: MACRO7 10 20 30
PAW > MESSAGE M7
EXEC MACRO7
PAW > MESSAGE 'M7'
M7
```

Note that if CHOPT='C' then the alias is a command alias, i.e. an alias that will only be translated when it is the first token on a command line, e.g.

PAW > Alias/Create LS DIR C is equivalent to: PAW > DIR

Only when LS is the first token on a command line, i.e. in the case below LS will not be translated:

PAW > SHELL LS

Aliases need separators to be recognized in the input line, as evident from the M7PP line in the example above. Possible separators are blank / , = : . % ' ().

A double slash // can be used to concatenate aliases without any separator (i.e. to juxtapose them):

```
PAW > Alias/Create DIR disk$dl:[paw]
PAW > Alias/Create FIL mydatafile
PAW > HISTO/FILE 3 DIR//FIL
... Executing: HISTO/FILE 3 disk$dl:[paw]mydatafile
```

Note that aliases are **recursive**. Example:

```
PAW > a/cr aa bb
PAW > a/cr bb cc
PAW > mess aa
cc
PAW > a/cr doc3 'exec doc3'
PAW > doc3
*** Line is too long after alias expansion
```

Another way of legally omitting EXEC before the name of a macro, is using the command DEFAULTS -AUTO. After having typed this command, a macro is searched whenever a command is not found: when CMD fails, EXEC CMD is issued automatically. But this is valid only in command mode: this logic is not active within macros, for security and portability reasons.

A more complex example of the use of aliases

Consider the use of ALIAS on a macro file DOC9 (containing three macros):

Example of input macro	Output when executing
<pre>macro doc9 message '... Executing: DOC9' return macro m1 message '... Executing: DOC9#M1' exec m2 return macro m2 message '... Executing: DOC9#M2' return</pre>	<pre>PAW > a/cre m1 'exec doc9#m1' PAW > <u>m1</u> ... Executing: DOC9#M1 ... Executing: DOC9#M2 PAW > a/cre m2 'exec doc9#m2' PAW > <u>m2</u> *** Unknown file EXEC.kumac</pre>

This happens because when the string `m2` is substituted by its alias value `exec doc9#m2'`, the macro `m1` becomes:

```
macro m1
  message '... Executing: MACRO DOC9#M1'
  exec exec doc9#m2
return
```

To avoid this, one could simply add a character (for example an underscore) before the macro names, as:

Example of input macro	Output when executing
<pre>macro _doc9 message '... Executing: _DOC9' return macro _m1 message '... Executing: _DOC9#_M1' exec _m2 return macro _m2 message '... Executing: _DOC9#_M2' return</pre>	<pre>PAW > a/cr m1 'exec doc9new#_m1' PAW > <u>m1</u> ... Executing: _DOC9#_M1 ... Executing: _DOC9#_M2 PAW > a/cr m2 'exec doc9new#_m2' PAW > <u>m2</u> ... Executing: _DOC9#_M2</pre>

4.5 System functions

While aliases have a fixed value, system functions can be seen as aliases whose value is variable and dependent on the function name and its arguments (if any). Therefore, also system functions are literally replaced by their current value whenever encountered in a KUIP command line. System functions, unlike aliases, do not need separators because they are predefined and known by KUIP. Their names always start with a dollar sign, and some of them have a parameter list, enclosed in parentheses. System functions are mainly used inside macros.

Table 4.2: KUIP system functions

System Functions	
Function (arguments)	Returned value
\$STYLE	Current style as defined by command SET/STYLE
\$ANUM	Number of aliases
\$ANAM(I)	Name of I-th alias
\$AVAL(I)	Value of I-th alias
\$LAST	Latest command line executed
\$KEYNUM	Address of latest clicked key in STYLE GP
\$KEYVAL	Value of latest clicked key in STYLE GP
\$ARGS	Command line at program invocation
\$DATE	Current date in format DD/MM/YY
\$TIME	Current time in format HH.MM.SS
\$CPTIME	CP time elapsed since last call (sec)
\$RTIME	Real time elapsed since last call (sec)
\$VDIM(VNAME, IDIM)	Physical length of vector VNAME on dimension IDIM(1..3)
\$VLEN(VNAME, IDIM)	Logical length (stripping trailing '0') of vector VNAME
\$NUMVEC	Current number of vectors
\$VEXIST(VNAME)	Index of vector VNAME(1..\$NUMVEC (0 if nonexistent)
\$SUBSTRING(STRING, IX, NCH)	STRING(IX:IX+NCH-1)
\$UPPER(STRING)	STRING changed to upper case
\$LOWER(STRING)	STRING changed to lower case
\$LEN(STRING)	STRING length, stripping quotes and leading/trailing blanks
\$SIGMA(SIGMA_Expression)	Result of SIGMA_Expression, as computed by SIGMA
\$OS	Returns Operating System in capital letters (UNIX, VM,etc.)
\$MACHINE	Returns machine type in capital letters (HP/UX, VAX, etc.)
\$PID	Returns UNIX process PID id (1 for non-UNIX systems)

4.5.1 The \$SIGMA system function in more detail

A SIGMA expression can involve scalar or vector types of operands, and, according to the type of the result, the string \$SIGMA(Sigma_expression) will be substituted by either the numerical value of Sigma_expression, if the result is a scalar, or the name of a temporary vector (generated by SIGMA) containing the result of the evaluation of the Sigma_expression.

Example of the use of a SIGMA function

```
PAW > MESSAGE $SIGMA(SQRT(100)*PI)
31.41593

PAW > VEC/CR WWW R 10          | Create vector WWW
PAW > VEC/CR SSS R 10 20 30 40 50 | Create vector SSS
```

SIGMA uses temporary vectors (?SIGMA1, ?SIGMA2,...). They are deleted automatically after the execution of the command.

Example showing how SIGMA uses a temporary vector

```
PAW > VEC/PRINT WWW
WWW ( 1 ) = 10.00000
PAW > VEC/PRINT SSS
SSS ( 1 ) = 10.00000
SSS ( 2 ) = 20.00000
SSS ( 3 ) = 30.00000
SSS ( 4 ) = 40.00000
SSS ( 5 ) = 50.00000
PAW > VEC/PRINT $SIGMA(WWW*10+PI)
?SIGMA1 ( 1 ) = 103.1416
PAW > VEC/PRINT $SIGMA(SSS*10+PI)
?SIGMA1 ( 1 ) = 103.1416
?SIGMA1 ( 2 ) = 203.1416
?SIGMA1 ( 3 ) = 303.1416
?SIGMA1 ( 4 ) = 403.1416
?SIGMA1 ( 5 ) = 503.1416
```

Multiple vector references are possible in the same command line.

Example of the use of multiple vector references

```
PAW > GRAPH 100 $SIGMA(SIN(XVEC)) $SIGMA(COS(XVEC))
```

4.6 More on aliases, system functions and macro variables

Substitutions for aliases and system functions are performed “literally”, i.e. “as a character string” and regardless of the type of parameter. For example, a system function resulting in a Character string value can be inserted in place of a numeric type parameter (**Integer** or **Real**). KUIP will complain when executing that line only if the string cannot be interpreted as a numeric parameter.

As an example consider the PAW command /GRAPHICS/PRIMITIVES/ITX X Y TEXT which has the first two parameters of type **Real** and the third one of type **Character**:

```
PAW > ITX $SUBSTRING(ABC100,4,2) 15 'Test of ITX'
PAW > ITX $SUBSTRING(ABC100,1,2) 15 'Test of ITX'
*** Error in decoding 'real' parameter: AB
*** Default is taken
*** Default is not defined, unpredictable returned value !
```

the first call to ITX draws the text string Test of ITX at coordinates (10,15); the second call ends up with an error.

Macro variables, aliases and system functions resolution

The general rules governing the resolution of macro variables, aliases and system functions are:

- Any string within brackets, like [variable], is replaced **everywhere** in a macro by its “literal” value. It is left unchanged only when:
 - it is within a quoted string
 - the variable is undefined, i.e. neither variable=value was executed within the macro nor the calling sequence was of the type EXEC MACRO variable=value.
- At the left hand side of an assignment statement a variable appears always **without** square brackets, while at the right hand side of an assignment statement a variable appears always **between** square brackets.

```
ABC=15
E=[ABC]+10      | E is set to 25
```

- Variables and aliases are resolved **before** system functions, e.g.

```
ABC=15
D=$SUBSTRING([ABC],2,1) | D is set to 5
```

- System functions cannot be nested, e.g. D=\$LEN(\$SUBSTRING([ABC],2,1)) is invalid.
- Macro control statements cannot be aliased, e.g. the following code will produce an error:

```
ALIAS/CREATE JUMP GOTO
IF 1=1 JUMP 10
10:
```

- Arguments of EXEC and GOTO may be aliases or macro variables, e.g.

```
MACRO JUNK
  EXEC [1]
RETURN
```

where entering EXEC JUNK M1 is equivalent to enter EXEC M1.

4.7 Recalling previous commands

In addition to the host machine local facilities in recalling previous commands, KUIP allows the user to:

- Enter the command LAST which writes all (or some) commands typed in the session to a disk file (by default LAST.KUMAC) and invokes the local editor on the same file.

The history file is updated automatically every 25 commands (but the rate can be changed with the command RECORDING) and at the end of a session. At the beginning of another session the old history file LAST.KUMAC is renamed into LAST.KUMACOLD and a new LAST.KUMAC is opened. In this way the user always keeps track of all the commands entered in the previous and in the current sessions. The history files contain also heading and trailing comment lines, showing the date and time at which the sessions were started and stopped.

The command LAST 0 MYFILE may be put in the user's LOGON.KUMAC to define the name of the history files as MYFILE.KUMAC and MYFILE.KUMACOLD. This is useful to avoid sharing the same LAST.KUMAC file by several KUIP-based applications running on the same disk directory (e.g. PAW, GEANT [11], CMZ [12], etc.)

- Use a UNIX C-shell-like history mechanism, starting a command with an exclamation mark followed by:
 - An absolute number n, to re-execute the n-th command entered since the beginning of the session (e.g. !3)
 - A minus sign and a relative number, to re-execute the command identified by the current command number minus n (e.g. !-2)
 - Another exclamation mark, to re-execute the last command entered (i.e. !!)
 - Anything else (i.e. a non-numeric string), to re-execute the latest command entered which starts with the specified string (e.g. !EXE)
 - Nothing else, to show the list of recallable commands (i.e. just !)

To obtain the numbering of the command lines the prompt string must be defined as containing an open and closed brackets (e.g. by the command SET/PROMPT 'MYPROG []'), inside which KUIP will put the command line number.

4.8 Exception condition handling

User breaks (e.g. CTRL/C) are handled within PAW: when a break is issued the execution of the current command is aborted and PAW is waiting again for the next command.

This feature is not available in the IBM version.

Program exception conditions (for example, floating-point overflow, negative square root, etc.) are handled separately (also on IBM): when such an exception occurs, a warning message is issued and the program goes into a state waiting for the next command, as for user breaks.

Chapter 5: Vectors

Vectors are named arrays of numerical data, memory resident, which can be created during a session, loaded from HBOOK objects, typed in by hand, read from disk files, operated upon using the full functionality of SIGMA or COMIS. Vectors can be used to produce graphics output, and, if necessary, stored away on disk files for further usage. Vectors provide a very convenient mechanism to transport numerical information between different PAW objects, and to manipulate mathematically their content. At the end of an interactive session, they are lost, unless previously saved onto disk files.

Vectors can have up to 3 dimensions (in fact they are “arrays”, called “vectors” for historical reasons). They can be handled in PAW either interactively, by using `VECTOR/...` commands, or by means of KUIP routines which return the addresses of a given vector.

Simple arithmetic operations can be applied to vectors. In addition, as SIGMA is part of PAW, powerful array manipulation operations are available, through the SIGMA, \$SIGMA and APPLICATION SIGMA commands (see section 6.1 on page 110).

An “invisible” vector named ?, mono-dimensional and of length 100, is always present. It is used for communication between arrays in the user code (for instance in a COMIS[1] routine) and KUIP vectors, being equivalenced with the real array `VECTOR(100)` in the labelled common block /KCWORK/.

5.1 Vector creation and filling

A vector is **created** either by the **PAW command** `VECTOR/CREATE`, by the **SIGMA function** `ARRAY`. or by the **COMIS statement** `VECTOR`.

Example of vector creation

<code>VECTOR/CREATE X(100)</code>	will create a 100-components vector, values = 0.
<code>SIGMA X=ARRAY(100,1#100)</code>	will create a 100-components vector and assign to each element the values 1,2,...100
<code>VECTOR X(100)</code>	in a COMIS routine creates a 100-components vector and initialises each element to zero

Once the vector is created, it can be manipulated using the following PAW commands:

<code>VECTOR/INPUT vlist</code>	Input from the terminal values into the vector elements specified by the list <code>vlist</code> .
<code>VECTOR/READ vlist</code>	Values can be read in from a file into the vector elements specified by the list <code>vlist</code> .
<code>VECTOR/COPY v1 v2</code>	Values in <code>v1</code> are copied into <code>v2</code> .
<code>VECTOR/WRITE vlist</code>	Values in the vector elements specified by the list <code>vlist</code> can be saved on a file.
<code>VECTOR/PRINT vlist</code>	Values of the vector elements specified in <code>vlist</code> will be printed on the terminal.
<code>VECTOR/LIST</code>	A list of existing vectors and their characteristics is printed on the terminal.
<code>VECTOR/DELETE</code>	Allows global or selective deletion of vectors.

5.2 Vector addressing

Indexing of vectors is possible¹.

Example of vector indices

<code>Vec</code>	for all elements
<code>Vec(13)</code>	for element 13
<code>Vec(12:)</code>	for elements 12 up to the last
<code>Vec(:10)</code>	for elements 1 to 10
<code>Vec(5:8)</code>	for elements 5 to 8

Sub-elements of the two-dimensional vector `Vec(3,100)` (3 columns by 100 rows) may be addressed by:

Using two-dimensional vectors

<code>Vec(2,5:8)</code>	for elements 5 to 8 in column 2
<code>Vec(2:3,5:8)</code>	for elements 5 to 8 columns 2 to 3
<code>Vec(2,5)</code>	for element 5 in column 2
<code>Vec(:,3)</code>	for all elements in row 3
<code>Vec(2)</code>	for all elements in the 2-nd column (SPECIAL CASE)

5.3 Vector arithmetic operations

A number of basic vector arithmetic operations is available:

<code>VBIAS v1 bias v2</code>	<code>v2(I) = bias + v1(I)</code>
<code>VSCALE v1 scale v2</code>	<code>v2(I) = scale * v1(I)</code>
<code>VADD v1 v2 v3</code>	<code>v3(I) = v1(I) + v2(I)</code>
<code>VMULTIPLY v1 v2 v3</code>	<code>v3(I) = v1(I) * v2(I)</code>
<code>VSUBTRACT v1 v2 v3</code>	<code>v3(I) = v1(I) - v2(I)</code>
<code>VDIVIDE v1 v2 v3</code>	<code>v3(I) = v1(I) / v2(I), if v2(I)<>0</code>

In all operations only the minimum vector length is considered, i.e. an operation between a vector A of dimension 10 and a vector B of dimension 5 will involve the first 5 elements for both vectors. If the destination vector does not exist, it is created with the same length as specified in the source vector.

5.4 Vector arithmetic operations using SIGMA

A more complete and convenient mechanism for the mathematical manipulation of entire vectors is provided by SIGMA. SIGMA-generated arrays are stored as PAW vectors and therefore are accessible to PAW commands, and PAW vectors are accessible to SIGMA. The facilities available via SIGMA are described in the next chapter.

¹Note that the indexing permitted in PAW can be considered as a superset of that permitted by FORTRAN. This feature cannot be used from within SIGMA.

5.5 Using KUIP vectors in a COMIS routine

The declaration `VECTOR vector_name` may be used inside a COMIS routine to address a KUIP vector. If the vector does not exist, it is created with the specifications provided by the declared dimension.

5.6 Usage of vectors with other PAW objects

Vectors can be used to transport numerical information between different PAW objects, and to manipulate mathematically their content.

<code>VECTOR/HFILL VNAME ID</code>	Each vector element of VNAME is used to fill the existing histogram ID.
<code>HISTOGRAM/GET_VECTOR/CONTENTS</code>	Provides an interface between vectors and histograms.
<code>HISTOGRAM/PUT_VECTOR/CONTENTS</code>	Provides an interface between histograms and vectors.

5.7 Graphical output of vectors

<code>VECTOR/DRAW VNAME</code>	Interprets the content of the vector VNAME as a histogram contents and draw a graph .
<code>VECTOR/PLOT VNAME</code>	Vector elements are considered as individual values to be entered into a histogram and a graph is produced. If VNAME is the name of a vector, then each vector element of VNAME is used to fill a histogram which is automatically booked with 100 channels and plotted. If VNAME has the form VNAME1%VNAME2 then a scatter-plot of vector VNAME1 versus VNAME2 is plotted.

See figure 3.4 in the tutorial section for an explanation of the difference between `VECTOR/DRAW` and `VECTOR/PLOT`.

A number of HIGZ [3] macro-primitives are available in PAW. Those directly related to the graphical output of vectors are:

<code>GRAPH N X Y</code>	Draw a curve through a set of points defined by arrays X and Y.
<code>HIST N X Y</code>	Draw an histogram defined by arrays X and Y.
<code>PIE X0 Y0 RAD N VAL</code>	Draw a pie chart, of N slices, with size of slices given in VAL, of a radius RAD, centered at X0, Y0.

5.8 Fitting the contents of a vector

A user defined (and parameter dependent) function can be fitted to the points defined by the two vectors X and Y and the vector of associated errors EY. The general syntax of the command to fit vectors is:

`VECTOR/FIT x y ey func [chopt np par step pmin pmax errpar]`

For more information the reader is referred to the reference part of the present manual.

Chapter 6: SIGMA

6.1 Access to SIGMA

The SIGMA array manipulation package can be accessed in three different ways in PAW:

Precede the statement by the prefix SIGMA

Example

```
PAW > SIGMA xvec=array(100,-pi#pi*2)
PAW > SIGMA y=sin(xvec)*xvec
```

Note the use of the predefined constant PI in SIGMA with the obvious value.

The PAW command: APPLication SIGMA

All commands typed in after this command will be directly processed by SIGMA. The command EXIT will return control to PAW, e.g.

```
PAW > APPLication SIGMA
SIGMA > xvec=array(100,-pi#pi*2)
SIGMA > sinus=sin(xvec)*xvec
SIGMA > cosinus=cos(xvec)*xvec
SIGMA > exit
PAW > vector/list
Vector Name          Type    Length   Dim-1   Dim-2   Dim-3
XVEC                 R       100      100
SINUS                R       100      100
COSINUS              R       100      100
Total of 3 Vector(s)
```

The PAW system function \$SIGMA

The expression to be evaluated must be enclosed in parentheses. The function will return the numerical value of the expression (if the result is a scalar) or the name of a temporary vector (if the result is a vector).

Assuming that the computation of the function $\sin(x)*x$ in the above example would be only for the purpose of producing a graph, (i.e. the result is not needed for further calculations), then one could just have typed the following commands:

```
PAW > SIGMA xvec=array(100,-pi#pi*2)
PAW > GRApH 100 xvec $SIGMA(SIN(XVEC)*XVEC)
```

6.2 Vector arithmetic operations using SIGMA

A complete and convenient mechanism for the mathematical manipulation of vectors is provided by SIGMA. In the following, we use the words “array” and “vector” as synonyms. In both cases, we refer to PAW vectors, in the sense that SIGMA offers an alternative way to generate and to manipulate PAW vectors (see section 5 on page 107). The notation of SIGMA is similar to that of FORTRAN, in the sense that is based upon formulae and assignment statements.

The special operator **ARRAY** is used to generate vectors:

```
vname = ARRAY (arg1,arg2)
```

- vname Name of the vector (array) being created.
- arg1 Defines the array **structure**, i.e. the Number of COnponents (NCO) of the array.
- arg2 Provides the **numerical values** filling the array row-wise.
If arg2 is absent (or does not provide enough values) the array is filled with 1.

6.2.1 Basic operators

- + Add
- Subtract
- * Multiply
- / Divide
- ** Exponentiation
- & Concatenation

Note that ill defined operations will give 0. as result. For instance: a division by zero gives zero as result.

6.2.2 Logical operators

Logical operators act on entities that have **Boolean** values 1 (true) or 0 (false). The result is Boolean.

- AND Logical operation AND
- NOT Logical operation NOT
- OR Logical operation OR
- EQ EQual to
- GE Greater or Equal to
- GT Greater Than
- LE Less or Equal to
- LT Less Than
- NE Not Equal

6.2.3 Control operators

- !PRINT Provides the automatic printing of every newly created array or scalar.
- !NOPRINT Suppresses the automatic printing of every newly created array or scalar.

Examples

<u>A=ARRAY (6,1#6)</u>	1 2 3 4 5 6
<u>A=ARRAY (4)</u>	1 1 1 1
<u>A=ARRAY (5,2&3&-1&2&1.2)</u>	2 3 -1 2 1.2
<u>A=ARRAY (3)*PI</u>	3.1415927 3.1415927 3.1415927
<u>A=ARRAY (1,123E4)</u>	1230000.0

6.3 SIGMA functions

SIGMA provides some functions which perform a task on a whole array. These functions have no analogues in FORTRAN because all FORTRAN functions operate on one or more single numbers. Presently available SIGMA functions are listed in table 6.1 below.

Name	Result	Explanation
ANY	Scalar	The result is a Boolean scalar of value 1 (true) if at least one component of the argument is true and 0 (false) otherwise.
DEL	Vector	Analog to the Dirac-DELta Function . $V1=DEL(V)$ sets each element of V1 to 0.0 (if corresponding element in V is non-zero) or to 1.0 (if corresponding element is zero).
DIFF	Vector	$V2=DIFF(V)$ forward difference of V. The rightmost value in V1 is obtained by quadratic extrapolation over the last three elements of V.
LS	Vector	$V1=LS(V,N)$ shifts index of V to the left by N steps (cyclic).
LVMAX	Scalar	$S1=LVMAX(V1)$ sets S1 equal to the index (location) of the maximum value in vector V1.
LVMIM	Scalar	$S1=LVMIN(V1)$ sets S1 equal to the index (location) of the minimum value in vector V1.
MAX	Vector	$V3=MAX(V1,V2)$ sets each element of V3 equal to the maximum of the corresponding elements in V1 and V2.
MAXV	Vector	$V1=MAXV(V)$ sets each element of V1 equal to the maximum value in V.
MIN	Vector	$V3=MIN(V1,V2)$ sets each element of V3 equal to the minimum of the corresponding elements in V1 and V2.
MINV	Vector	$V1=MINV(V)$ sets each element of V1 equal to the minimum value in V.
NCO	Scalar	$V1=NCO(V)$ Number of COnponents of vector of V.
ORDER	Vector	$V1=ORDER(V,V2)$ finds a permutation that brings V2 in a non-descending order and applies it to V to generate V1.
PROD	Vector	$V1=PROD(V)$ V1 is the running product of V.
QUAD	Vector	$V2=QUAD(V1,H)$ The quadrature function QUAD numerically integrates each row of V1 with respect to the scalar step size H.
SUMV	Vector	$V2=SUMV(V1)$ running sum of V.
VMAX	Scalar	$S1=VMAX(V1)$ sets S1 equal to the maximum value in vector V1.
VMIN	Scalar	$S1=VMIN(V1)$ sets S1 equal to the minimum value in vector V1.
VSUM	Scalar	$S1=VSUM(V)$ sum of all components of V.

Table 6.1: SIGMA functions

6.3.1 SIGMA functions - A detailed description.

In the following description of the SIGMA functions, the letter R always denotes the **result** and arg denotes one or more **arguments**. Any argument may itself be an expression. In that case arg means the result of this expression. Let OP denote any of the above array functions, then the statement:

```
R = OP (arg1, arg2, ...)
```

produces R without doing anything to the contents stored under the names appearing in arg1, arg2, Thus, although in the description we may say "...OP does such and such to arg ...", in reality it leaves arg intact and works on the argument to produce R.

```
R = ANY (arg)
```

The function ANY considers the result of the argument expression as a Boolean array. SIGMA represents "true" by 1 and "false" by 0. Thus the components of arg must be either 0 or 1, otherwise an error is generated.

If at least one component of the result of the argument expression is 1, than ANY returns the scalar 1. If all components of the result of the argument expression are 0 then ANY returns the scalar 0. If arg is a Boolean scalar, R = arg.

Example of the ANY command

```
PAW > APPL SIGMA
SIGMA > !PRINT                                | Print newly created vectors and scalars
SIGMA > W=(-2)**ARRAY(10, 1#10)
NCO(W)      )= 10
W           =
-2.000      4.000     -8.000      16.00     -32.00      64.00
-128.0      256.0     -512.0      1024.
SIGMA > X=W GT 0
NCO(X)      )= 10
X           =
0.0000      1.000     0.0000      1.000     0.0000      1.000
0.0000      1.000     0.0000      1.000
SIGMA > R=ANY(X)
NCO(R)      )= 1
R           1.000
```

```
R = DEL (arg)
```

DEL is a discrete analogue of a **Dirac delta function**. DEL works independently on each row of the argument array. If the elements of any row of the argument are denoted by $X_1, X_2, \dots, X_i, \dots, X_n$ then the corresponding row of the result of the delta function operation will be $Z_1, Z_2, \dots, Z_i, \dots, Z_n$ where all $Z_i = 0$ except in three cases, in which $Z_i = 1$, namely:

- 1 When the component X_i is itself zero.
- 2 When X_{i-1}, X_i are of opposite sign and $|X_i| < |X_{i-1}|$ If $i = 1$ then linear extrapolation to the left is used.
- 3 When X_i, X_{i+1} are of opposite sign and $|X_i| \leq |X_{i+1}|$ If $i = 1$ then linear extrapolation to the right is used.

If arg is a scalar, the value of DEL(arg) will be 1 if arg is zero, and 0 otherwise.

Example of the del command

```
SIGMA > W=array(11,-1#1)
NCO(W      )=   11
W      =
-1.000    -0.8000    -0.6000    -0.4000    -0.2000    -0.2980E-07
 0.2000     0.4000     0.6000     0.8000     1.000

SIGMA > X=(W+1.01)*W*(W-.35)*(W-.42)
NCO(X      )=   11
X      =
-0.1917E-01 -0.2357    -0.2384    -0.1501    -0.5524E-01-0.4425E-08
 0.7986E-02 -0.5640E-03  0.4347E-01  0.2476     0.7578

SIGMA > R=del(x)
NCO(R      )=   11
R      =
 1.000     0.0000     0.0000     0.0000     0.0000     1.000
 0.0000     1.000     0.0000     0.0000     0.0000
```

R = DIFF (arg)

The DIFF function generates the **forward difference** of each row of the argument array, say $X_1, X_2, \dots, X_i, \dots, X_n$ and creates an array with components equal to the forward difference of X : $X_2 - X_1, X_3 - X_2, \dots, X_n - X_{n-1}, X_0$ where the rightmost value X_0 is obtained by quadratic extrapolation over the last three elements of the result of arg. Applied to a scalar DIFF gives a zero result.

Example of the DIFF command

```
SIGMA > x=array(6,5#0)
NCO(X      )=   6
X      =
 5.000     4.000     3.000     2.000     1.000     0.0000
SIGMA > y=x*x
NCO(Y      )=   6
Y      =
 25.00     16.00     9.000     4.000     1.000     0.0000
SIGMA > Z=Diff(Y)
NCO(Z      )=   6
Z      =
 -9.000    -7.000    -5.000    -3.000    -1.000     1.000
```

R = LS (arg1,arg2)

The LS rearrangement function performs a **left shift**. arg1 is the array to be shifted; arg2 must be a scalar value (rounded if necessary by the system), interpreted as the number of places the array has to be shifted to the left. The scalar arg2 can be negative, in which case LS shifts to the right a number of places equal to the absolute value of arg2.

It should be noted the the shift is performed circularly modulo N, where N is the number of components in the rows of the array to be shifted. Hence, LS(X,N+1) shifts the N component rows of X by 1 to the left, and LS(X,-1) shifts the rows by N-1 to the left (or by 1 to the right). If arg1 is a scalar, R = arg1.

Example of the left shift command

```
SIGMA > X=array(4&5,array(20,1#20))
NCO(X      )=   4   5
X      =
 1.000    2.000    3.000    4.000
 5.000    6.000    7.000    8.000
 9.000   10.000   11.000   12.000
13.000   14.000   15.000   16.000
17.000   18.000   19.000   20.000

SIGMA > y=ls(x,1)
NCO(Y      )=   4   5
Y      =
 2.000    3.000    4.000    1.000
 6.000    7.000    8.000    5.000
10.000   11.000   12.000   9.000
14.000   15.000   16.000   13.000
18.000   19.000   20.000   17.000

SIGMA > y=ls(x,-3)
NCO(Y      )=   4   5
Y      =
 2.000    3.000    4.000    1.000
 6.000    7.000    8.000    5.000
10.000   11.000   12.000   9.000
14.000   15.000   16.000   13.000
18.000   19.000   20.000   17.000

SIGMA > X=array(5,1#5)
NCO(X      )=   5
X      1.000    2.000    3.000    4.000    5.000
SIGMA > z=ls(x,3)
NCO(Z      )=   5
Z      4.000    5.000    1.000    2.000    3.000
SIGMA > z1=ls(x,-4)
NCO(Z1     )=   5
Z1     2.000    3.000    4.000    5.000    1.000
```

R = LVMAX(arg1) and R = LVMIN(arg1)

The functions LVMAX and LVMIN returns as a scalar result the index (position) of the largest or smallest element, respectively, in the argument array.

Example of using the LVMAX and LVMIN commands

```
SIGMA > x=sin(array(10,1#10))
NCO(X      )=   10
X      =
 0.841    0.909    0.141   -0.757   -0.959   -0.279    0.657
 0.989    0.412   -0.544

SIGMA > r=lvmax(x)
NCO(R      )=   1
R      8.00
```

R = MAX(arg1,arg2) and **R = MIN(arg1,arg2)**

The functions **MAX** and **MIN** work independently on each element of their arguments. **arg2** can be a scalar. The result has the same dimension as the argument array **arg1** and each element of the result is set equal to the largest or smallest element, respectively, of the corresponding element of the argument arrays.

Example of using the MAX and MIN commands

```
SIGMA > x=sin(array(10,1#10))
NCO(X )= 10
X =
0.841      0.909      0.141     -0.757     -0.959     -0.279      0.657
0.989      0.412     -0.544

SIGMA > y=cos(array(10,1#10))
NCO(Y )= 10
Y =
0.540     -0.416     -0.990     -0.654      0.284      0.960      0.754
-0.146     -0.911     -0.839

SIGMA > z=min(x,y)
NCO(Z )= 10
Z =
0.540     -0.416     -0.990     -0.757     -0.959     -0.279      0.657
-0.146     -0.911     -0.839
```

R = MAXV(arg) and **R = MINV(arg)**

The extrema functions **MAXV** and **MINV** work on each element of their argument and the result has the same dimension as the argument array **arg1**. Each element of the result is set equal to the largest or smallest element, respectively, of the corresponding row of the argument array.

All these functions, if applied to a scalar argument, yield **R=arg**.

Example of using the MAX and MIN commands

```
SIGMA > x=array(10,0#10)
NCO(X )= 10
X =
0.0000      1.111      2.222      3.333      4.444      5.556
6.667       7.778      8.889     10.00

SIGMA > s=sin(x)*x
NCO(S )= 10
S =
0.0000      0.9958     1.767     -0.6352     -4.286     -3.695
2.494       7.755      4.539     -5.440

SIGMA > x=minv(s)
NCO(X )= 10
X =
-5.440     -5.440     -5.440     -5.440     -5.440     -5.440
-5.440     -5.440     -5.440     -5.440
```

R = NCO (arg)

The “Number of COMponents” (NCO) control function obtains the NCO vector of the arg. The NCO vector of a scalar is the scalar 1. For any argument the NCO (NCO(arg)) gives the number of dimensions of the arg.

Using the NCO command

```
SIGMA > x=array(4&3&2,array(24,2#48))
NCO(X      )=   4   3   2
X      =
2.000    4.000    6.000    8.000
10.00    12.00    14.00    16.00
18.00    20.00    22.00    24.00

26.00    28.00    30.00    32.00
34.00    36.00    38.00    40.00
42.00    44.00    46.00    48.00

SIGMA > r=nco(x)
NCO(R      )=   3
R      4.000    3.000    2.000
SIGMA > ndim=nco(nco(x))
NCO(NDIM  )=   1
NDIM   3.000
```

R = ORDER (arg1,arg2)

The ordering function ORDER acts independently on each row of arg1. arg2 must have the same row length as arg1.

ORDER finds the permutation that brings arg2 into a non-descending sequence (row-wise) and constructs the result by applying this permutation to arg1. It may in some cases be expanded to that structure by using the techniques of the topological arithmetic. This is particularly useful if arg2 is a single vector with the length of the rows of arg1.

Using the ORDER command

```
SIGMA > X = 1&1&2&4&-3&1&3
NCO(X      )=   7
X      =
1.00    1.00    2.00    4.00    -3.00    1.00    3.00
SIGMA > P = ORDER(X,X)
NCO(P      )=   7
P      =
-3.00    1.00    1.00    1.00    2.00    3.00    4.00
SIGMA > P = ORDER(X,-X)
NCO(P      )=   7
P      =
4.00    3.00    2.00    1.00    1.00    1.00    -3.00
SIGMA > Y = ARRAY(7,1# 7)
NCO(Y      )=   7
Y      =
1.00    2.00    3.00    4.00    5.00    6.00    7.00
SIGMA > P = ORDER(Y,X)
NCO(P      )=   7
P      =
5.00    1.00    2.00    6.00    3.00    7.00    4.00
```

R = PROD (arg)

The PROD function generates the **running product** of each row of the argument array, say X_1, X_2, \dots, X_n and creates an array with components equal to the running product of the component of the argument: $X_1, X_2, \dots, X_n, X_1 \times X_2, \dots, X_1 \times X_2 \times \dots \times X_n$

Using the TIMES command

```
SIGMA > x=array(6&4,array(24,1#24))
NCO(X      )=   6    4
X      =
 1.000    2.000    3.000    4.000    5.000    6.000
 7.000    8.000    9.000   10.00    11.00   12.00
 13.00   14.00   15.00   16.00   17.00   18.00
 19.00   20.00   21.00   22.00   23.00   24.00

SIGMA > y=prod(x)
NCO(Y      )=   6    4
Y      =
 1.000    2.000    6.000   24.00   120.0   720.0
 7.000   56.00   504.0   5040.  0.5544E+05 0.6653E+06
 13.00  182.0   2730.  0.4368E+05 0.7426E+06 0.1337E+08
 19.00  380.0   7980.  0.1756E+06 0.4038E+07 0.9691E+08
```

R = QUAD (arg1,arg2)

The **quadrature function** QUAD numerically integrates each row of arg1 with respect to the scalar step size h defined by arg2.

The result R has the same dimension as arg1 and the integration constant is fixed by choosing the first point of the result to be zero.

The method uses a four-point forward and backward one-strip-formula based on Lagrange interpolation. We have for the first point of the result:

$$R_1 = \int_{x_1}^{x_1} (arg1) dx = 0$$

for the second and third points

$$R_{i+1} = R_i + \frac{h}{24}(9f_i + 19f_{i+1} - 5f_{i+2} + f_{i+3})$$

and for all subsequent points

$$R_i = R_{i-1} + \frac{h}{24}(f_{i-3} - 5f_{i-2} + 19f_{i-1} + 9f_i)$$

where the f_i are elements of arg1 and are assumed to be values of some functions evaluated at equidistant intervals with interval width equal to h (h being equal to the value of arg2).

```

SIGMA > *****
SIGMA > * SIGMA application *
SIGMA > * showing use of *
SIGMA > * QUAD numeric *
SIGMA > * integration *
SIGMA > *****
SIGMA > x=array(101,0#2*pi)
SIGMA > * Function value array
SIGMA > y=sin(x)
SIGMA > * Step size
SIGMA > dx=0.6283186E-01
SIGMA > print dx
NCO(DX) = 1
DX 0.6283186E-01
SIGMA > * Integration of SIN(X)
SIGMA > * in interval 0<=X<+2*PI
SIGMA > f=quad(y,dx)
SIGMA > * Analytical result
SIGMA > * is 1-COS(X)
SIGMA > g=1-cos(x)
SIGMA > * Compute the difference
SIGMA > erro=(g-f)*10**6
SIGMA > * Plot the difference
SIGMA > * in units of 10-6
SIGMA > exit
PAW > opt GRID
PAW > gra 101 x erro

```

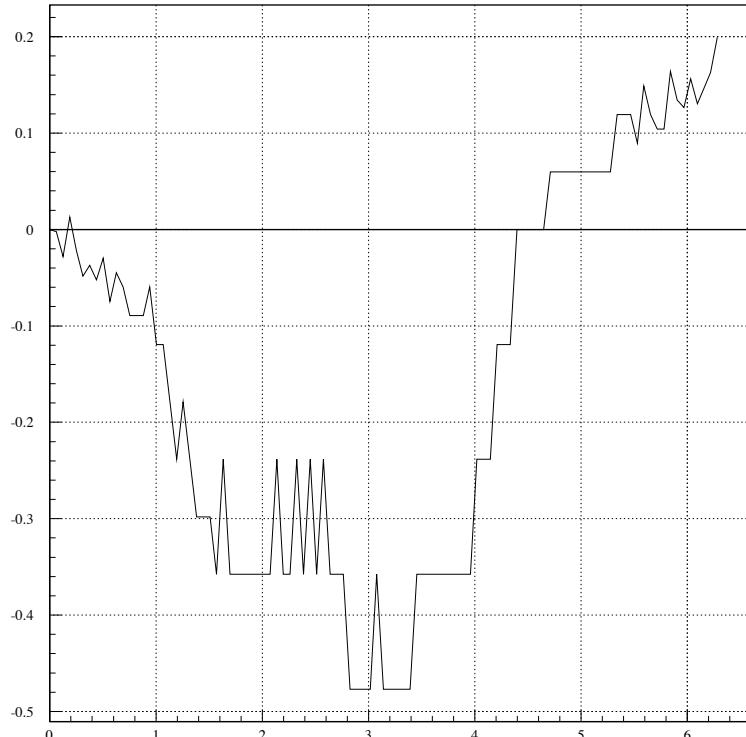


Figure 6.1: Using numerical integration with SIGMA

R = SUMV (arg)

The SUMV function generates the **running summation** of each row of the argument array, say $X_1, X_2, \dots, X_i, \dots, X_n$ and creates an array with components equal to the running sum of the X_i namely: $X_1, X_1 + X_2, \dots, X_1 + X_2 + \dots + X_i, \dots, X_1 + X_2 + \dots + X_n$.

Using the SUM function

```

SIGMA > x=array(6&4,array(24,1#24))
NCO(X) = 6 4
X =
 1.000 2.000 3.000 4.000 5.000 6.000
 7.000 8.000 9.000 10.00 11.00 12.00
 13.00 14.00 15.00 16.00 17.00 18.00
 19.00 20.00 21.00 22.00 23.00 24.00

SIGMA > y=sumv(x)
NCO(Y) = 6 4
Y =
 1.000 3.000 6.000 10.00 15.00 21.00
 7.000 15.00 24.00 34.00 45.00 57.00
 13.00 27.00 42.00 58.00 75.00 93.00
 19.00 39.00 60.00 82.00 105.0 129.0

```

```
R = VMAX(arg) and R = VMIN(arg)
```

The functions VMAX and VMIN return a scalar equal to the largest or smallest element of the array arg.

```
R = VSUM (arg1)
```

The VSUM function generates the **sum** of each element of the argument array, say $X_1, X_2, \dots, X_i, \dots, X_n$ and creates a scalar whose value is equal to the sum of all the components of X namely: $X_1 + X_2 + X_3, \dots, X_n$

Using the VSUM function

```
SIGMA > x=array(10)
NCO(X      )=   10
X      =
 1.00      1.00      1.00      1.00      1.00      1.00      1.00
 1.00      1.00      1.00

SIGMA > r=vsun(x)
NCO(R      )=    1
R      10.0
```

6.4 Available library functions

The library functions available under SIGMA are listed below. All these functions have a single argument, unless otherwise indicated. The number indicated between parentheses corresponds to the number of the same function in the CERN program library.

ABS	ABSolute value
ACOS	ArCOSine
ALOGAM	LOGarithm of the GAMma Function (C341)
ASIN	ArcSINE
ATAN	ArcTANgent
ATAN2	ArcTANgent2 (2 arguments)
BESI0	Mod. Bessel Function I0 (C313)
BESI1	Mod. Bessel Function I1 (C313)
BESJ0	Bessel Function J0 (C312)
BESJ1	Bessel Function J1 (C312)
BESK0	Mod. Bessel Function K0 (C313)
BESK1	Mod. Bessel Function K1 (C313)
BESY0	Bessel Function Y0 (C312)
BESY1	Bessel Function Y1 (C312)
COS	COSine
COSH	Hyperbolic COSine
COSINT	COSine INTegral (C336)
Dilog	DILOGarithm Function (C304)
EBESI0	$\exp(- x)I_0(x)$ (C313)
EBESI1	$\exp(- x)I_1(x)$ (C313)

EBESK0	$\exp(x)K_0(x)$ (C313)
EBESK1	$\exp(x)K_1(x)$ (C313)
ELLICK	Complete Elliptic Integral K (C308)
ELLICE	Complete Elliptic Integral E (C308)
ERF	Error Function ERF (C300)
ERFC	Error Function ERFC (C300)
EXP	EXPonential
EXPINT	EXPonential INTegral (C337)
FREQ	Normal Frequency Function FREQ (C300)
GAMMA	GAMMA Function (C305)
INT	Takes INTegral part of decimal number
LOG	Natural LOGarithm
LOG10	Common LOGarithm
MOD	Remaindering
RNDM	Random Number Generator: V1=RNDM(V), with NCO(V1)=NCO(V) generates random numbers between 0 and 1.
SIGN	Transfer of SIGN: V2=SIGN(V,V1), V2= V *V1/ V1
SIN	SINe Function
SINH	Hyperbolic SINe
SININT	SINe INTegral (C336)
SQRT	SQuare Root
TAN	TANgent
TANH	Hyperbolic Tangent

Ill defined functions will return 0. as result. (e.g. SQRT of a negative number is taken as 0).

Chapter 7: HBOOK

7.1 Introduction

Many of the ideas and functionality in the area of data presentation, manipulation and management in PAW find their origin in the HBOOK subroutine package [2], which handles statistical distributions (histograms and Ntuples). HBOOK is normally run in a batch environment, and it produces generally graphics output on the line printer or, optionally, via the HPLOT [4] package on a high resolution graphic output device.

The HBOOK system consists of a few hundred FORTRAN subroutines which enable the user to symbolically define, fill and output one- and two-dimensional density estimators, under the form of **histograms**, **scatter-plots** and **tables**.

Furthermore the analysis of large data samples is eased by the use of **Ntuples**, which are two-dimensional arrays, characterised by a **fixed** number N, specifying the number of entries per element, and by a **length**, giving the total number of elements. An element of a Ntuple can be thought of as a physics “event” on e.g. a Data Summary Tape (micro-DST). **Selection criteria** can be applied to each “event” or element and a complete Ntuple can be statistically analysed in a fast, efficient and interactive way.

7.1.1 The functionality of HBOOK

The various user routines of HBOOK can be subdivided by functionality as follows:

Booking	Declare a one- or two-dimensional histogram or a Ntuple
Projections	Project two-dimensional distributions onto both axes
Ntuples	Way of writing micro data-summary-files for further processing. This allows to make later projections of individual variables or correlation plots. Selection mechanisms may be defined
Function representation	Associates a real function of 1 or 2 variables to a histogram
Filling	Enter a data value into a given histogram, table or Ntuple
Access to information	Transfer of numerical values from HBOOK-managed memory to Fortran variables and back
Arithmetic operations	On histograms and Ntuples
Fitting	Least squares and maximum likelihood fits of parametric functions to histogrammed data
Smoothing	Splines or other algorithms
Random number generation	Based on experimental distributions
Archiving	Information is stored on mass storage for further reference in subsequent programs
Editing	Choice of the form of presentation of the histogrammed data

7.2 Basic ideas

The basic data elements of HBOOK are the **histogram** (one- and two-dimensional) and the **Ntuple**. The user identifies his data elements using a **single integer**. Each of the elements has a number of **attributes** associated with it.

The HBOOK system uses the ZEBRA [7] data manager to store its data elements in a COMMON block /PAWC/, shared with the KUIP [5] and HIGZ [3] packages, when the latter are also used (as is the case in PAW). In fact the first task of a HBOOK user is to declare the length of this common to ZEBRA by a call to HLIMIT, as is seen in figures 7.3 and 7.5¹.

In the /PAWC/ data store, the HBOOK, HIGZ and KUIP packages have all their own **division** (see [7] for more details on the notion of divisions) as follows (figure 7.1):

- LINKS Some locations at the beginning of /PAWC/ for ZEBRA pointers.
- WORKS Working space (or division 1) used by the various packages storing information in /PAWC/
- HBOOK Division 2 of the store. Reserved to HBOOK
- HIGZ A division reserved for the HIGZ graphics package.
- KUIP A division reserved for the KUIP user interface package.
- SYSTEM The ZEBRA system division. It contains some tables, as well as the Input/Output buffers for HRIN and HROUT.

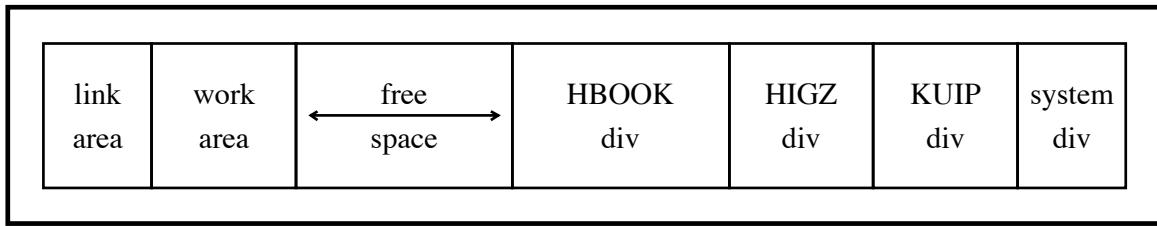


Figure 7.1: The layout of the /PAWC/ dynamic store

7.2.1 RZ directories and HBOOK files

An advantage of using ZEBRA in HBOOK is that ZEBRA's direct access **RZ package** is available. The latter allows data structures to be uniquely addressed via **pathnames**, carrying a mnemonic meaning and showing the relations between data structures. Related data structures are addressed from a **directory**. Each time a RZ file is opened via a call to HRFIL_E a supplementary top directory is created with a name specified in the calling sequence. This means that the user can more easily keep track of his data and also the **same** histogram identifiers can be used in various files, what makes life easier if one wants to study various data samples with the same program, since they can be addressed by changing to the relevant file by a call to HCDIR first.

¹This is of course not necessary in PAW, which is already precompiled when it is run. However when treating very large data samples or in other special applications, it might be necessary to specify a different value for the length of the dynamic store, which is defined by a call to PAWINT from the main initialisation routine PAMAIN. The “default” value for the length of /PAWC/ is 500000 (Apollo), 200000 (IBM) or 300000 (other systems), with respectively 10000 and 68000 words initially reserved for HIGZ and KUIP.

Example of using directories

```

CALL HRFILE(1,'HIST01',' ')          ! Open first HBOOK RZ file (read only)
CALL HRFILE(2,'HIST02','U')          ! Open second HBOOK RZ file (update)
CALL HCDIR('//HIST01',' ')           ! Make HIST01 current directory
CALL HRIN(20,9999,0)                 ! Read ID 20 on file 1
.
.
.
CALL HCDIR('//HIST02',' ')           ! Make HIST02 current directory
CALL HRIN(10,9999,0)                 ! Read ID 10 on file 2
.
.
.
CALL HROUT(20,ICYCLE,' ')            ! Write ID 20 to file 2
CALL HREND('HIST01')                 ! Close file 1
CALL HREND('HIST02')                 ! Close file 2

```

In the previous example (and also in figures 7.3 and 7.5) it is shown how an external file is available via a directory name inside HBOOK (and PAW), and that one can change from one to the other file by merely **changing directory**, via the PAW command CDIR, which calls the HBOOK routine HCDIR.

7.2.2 Changing directories

One must pay attention to the fact that **newly** created histograms go to **memory** in the //PAWC directory (i.e. the /PAWC/ common). As an example suppose that the current directory is //LUN1, and an operation is performed on two histograms. These histograms are first copied to memory //PAWC, the operation is performed and the result is **only** available in //PAWC,

PAW > <u>CDIR //LUN1</u>	Set current directory to //LUN1
PAW > <u>ADD 10 20 30</u>	Add histograms 10 and 20 into 30
	Histogram 30 is created in //PAWC
PAW > <u>Histo/Plot //PAWC/30</u>	Show the result of the sum
PAW > <u>CD //PAWC</u>	Set the current directory to memory
PAW > <u>Histo/plot 30</u>	Show the result once more

Similarly when histograms or Ntuples are plotted (e.g. by the HISTO/PLOT command), they are copied to memory possibly replacing an old copy of the same ID. As long as the copy in memory is not changed, each time the ID is read from the **external** file. This is because in a **real time** environment, e.g. using **global sections** on VMS or **modules** with OS9, the data base on the external medium can be changed by concurrent processes. However if the HBOOK data structure, associated with the histogram or Ntuple in memory is **altered** (e.g. by a MAX, IDOPT, FIT command), then it becomes the **default** for subsequent operations. If one wants the **original copy** one first must delete the copy from memory or **explicitly** use the pathname for the external file.

PAW > <u>Histo/file 1 his.dat</u>	The file contains ID=10
PAW > <u>Histo/Plot 10</u>	ID=10 read from file and plotted
PAW > <u>H/plot 10</u>	ID=10 read again from file and plotted
PAW > <u>H/fit 10 ! G</u>	Read from file, make a Gaussian fit on //PAWC/10
PAW > <u>H/plot 10</u>	ID=10 read from memory since it changed
PAW > <u>H/del 10</u>	Delete histogram 10 from memory
PAW > <u>H/plot 10</u>	ID=10 read again from file and plotted

7.3 HBOOK batch as the first step of the analysis

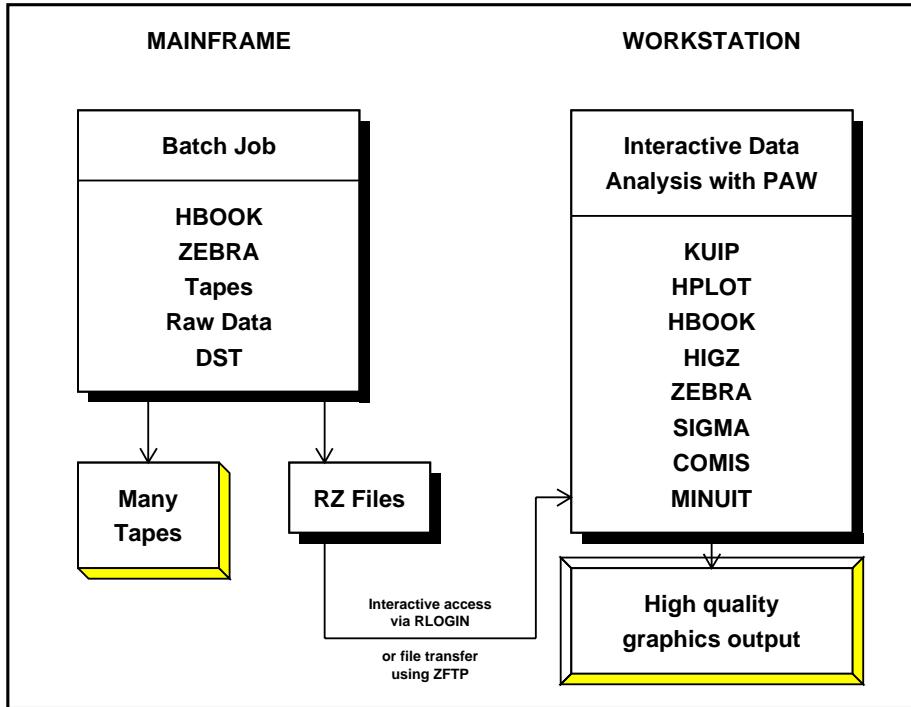


Figure 7.2: Schematic presentation of the various steps in the data analysis chain

Although it is possible to define histograms interactively in a PAW session, and then read the (many thousands of) events, in general for large data samples the relevant variables are extracted from the **Data Summary Files or DSTs** and stored in **histograms** or an **Ntuple**. The histogram needs already that a certain choice has to be made as to the range of values for the plotted parameter, because the **binning**, or the coarseness, of the distribution has to be specified when the histogram is defined (**booked**). Also only one- and two-dimensional histograms are possible, hence the correlations between various parameters can be difficult to study. Hence it seems in many cases more appropriate to store the value of the important parameters for each event in an **Ntuple**. This approach preserves the correlation between the parameters and allows selection criteria to be applied on the (reduced) data sample at a later stage.

In general, the time consuming job of analysing all events available on tape is run on a mainframe or CPU server, and the important event parameters are stored in a Ntuple to allow further detailed study. For convenience the Ntuple can be output to disk for each run, and then at a later stage the Ntuples can be **merged** in order to allow a global interactive analysis of the complete data sample.

A typical batch job in which data are analysed offline and some characteristics are stored in HBOOK is given in 7.3. After opening the RZ HBOOK file, HBOOK is initialised by a call to **HLIMIT**, which declares a length of 20000 words for the length of the **/PAWC/** dynamic store. Then the one- and two- dimensional histograms 110 and 210 are filled respectively according to the functions **HTFUN1** and **HTFUN2**. The output generated by the program is shown in Figure 7.4.

```

PROGRAM HTEST
PARAMETER (NMPAWC=20000)
COMMON/PAWC/H(NMPAWC)
EXTERNAL HTFUN1,HTFUN2
*-----.
*      CALL HLIMIT(NMPAWC)
*          Book histograms and declare functions
CALL HBFUN1(100,'Test of HRNDM1',100,0.,1.,HTFUN1)
CALL HBOOK1(110,'Filled according to HTFUN1',100,0.,1.,1000.)
CALL HBFUN2(200,'Test of HRNDM2',100,0.,1.,40,0.,1.,HTFUN2)
CALL HSCALE(200,0.)
CALL HBOOK2(210,'Fill according to HTFUN2',100,0.,1.,40,0.,1.,30.)
*          Fill histograms
DO 10 I=1,10000
   X=HRNDM1(100)
   CALL HFILL(110,X,0.,1.)
   CALL HRNDM2(200,X,Y)
   CALL HFILL(210,X,Y,1.)
10 CONTINUE
*          Save all histograms on file HTEST.DAT
CALL HPUT(0,'HTEST.DAT','I')
CALL HDELET(100)
CALL HDELET(200)
CALL HPRINT(0)
END
FUNCTION HTFUN2(X,Y)
   Two-dimensional gaussian
HTFUN2=HTFUN1(X)*HTFUN1(Y)
RETURN
END
FUNCTION HTFUN1(X)
   Constants for gaussians
DATA C1,C2/1.,0.5/
DATA XM1,XM2/0.3,0.7/
DATA XS1,XS2/0.07,0.12/
*          Calculate the gaussians
A1=-0.5*((X-XM1)/XS1)**2
A2=-0.5*((X-XM2)/XS2)**2
X1=C1
X2=C2
IF(ABS(A1).GT.0.0001)X1=C1*EXP(A1)
IF(ABS(A2).GT.0.0001)X2=C2*EXP(A2)
*          Return function value
HTFUN1=X1+X2
RETURN
END

```

Figure 7.3: Writing data to HBOOK with the creation of a HBOOK RZ file

Figure 7.4: Output segments 11 and 11' LTEST

7.3.1 Adding some data to the RZ file

The second run using program HTEST1 shows how to add some data to the HBOOK RZ file created in the job HTEST. After opening the file in question in update mode ('U' option) with the name EXAM2, a new directory NTUPLE is created, known as //EXAM2/NTUPLE as seen in the output of HLDIR command at the end of the output. A one- and a two-dimensional histogram and a Ntuple with identifiers of respectively 10, 20 and 30 are booked. Each Ntuple element or "event" is characterised by three **variables** (labelled 'X', 'Y' and 'Z'). The Ntuple data, when the initial size of 1000 words is exhausted, will be written to the directory specified in the call to HBOOKN, i.e. //EXAM2/NTUPLE, and the data in memory are replaced with those newly read. A one- and a two-dimensional projection of X and XY are then made onto histograms 10 and 20 respectively, before they are printed and written on the HBOOK RZ file. At the end the **current** and **parent** directories are listed. The contents of the latter shows that the data written in the first job (HTEST) are indeed still present in the file under the top directory //EXAM2. The call to RZSTAT shows usage statistics about the RZ file.

Example of adding data to a HBOOK RZ file

```

PROGRAM HTEST1
PARAMETER (NWPWC=20000)
COMMON/PAWC/H(NWPWC)
DIMENSION X(3)
CHARACTER*8 CHTAGS(3)
DATA CHTAGS/'     X    , , Y    , , Z    , /
```

```

* .-----.
   CALL HLIMIT(NWPWC)
*      Reopen data base
   CALL HOPEN(1,'EXAM2','HTEST.DAT',0,'U')
   CALL HMDIR('NTUPLE','S')
   CALL HBOOK1(10,'TEST1',100,-3.,3.,0.)
   CALL HBOOK2(20,'TEST2',30,-3.,3.,30,-3.,3.,250.)
   CALL HBOOKN(30,'N-TUPLE',3,'//EXAM2/NTUPLE',
+           1000,CHTAGS)
```

```

*      DO 10 I=1,10000
         CALL RANNOR(A,B)
         X(1)=A
         X(2)=B
         X(3)=A*A+B*B
         CALL HFN(30,X)
10    CONTINUE
```

```

*      CALL HPROJ1(10,30,0,0,1,999999,1)
      CALL HPROJ2(20,30,0,0,1,999999,1,2)
      CALL HPRINT(0)
      CALL HROUT(0,ICYCLE,' ')
      CALL HLDIR(' ',' ',' ')
      CALL HCDIR(' ',' ')
      CALL HLDIR(' ',' ',' ')
      CALL RZSTAT(' ',999,' ')
      CALL HREND('EXAM2')
END
```

Figure 7.5: Adding data to a HBOOK RZ file

7.4 Using PAW to analyse data

After transferring the HBOOK RZ file, which was created in the batch job as explained in the previous section, we start a PAW session to analyse the data which were generated². The PAW session below shows that the file HTEST.DAT is first opened via a call to HIST0/FILE. The data on the file are now accessible as the top directory //LUN1. When listing with the LDIR command the contents of the top directory //LUN1 and its NTUPLE subdirectory, the same information (histograms and Ntuples) is found as in the batch job (figure 7.5)

```

Reading a HBOOK direct access file
PAW > histo/file 1 htest.dat | open the HBOOK RZ file
PAW > ldir | list current directory

***** Directory ==> //LUN1 <==

Created 890902/1955 Modified 890902/1958

==> List of subdirectories
NTUPLE           Created 890902/1958 at record      9

==> List of objects
  HBOOK-ID CYCLE   DATE/TIME    NDATA   OFFSET    REC1    REC2
    100       1  890902/1955     153        1        3
    110       1  890902/1955     88       154        3
    200       1  890902/1955   4335      242        3        4 ==>      7
    210       1  890902/1955    767      481        7        8

NUMBER OF RECORDS =    7 NUMBER OF MEGAWORDS =  0 +  6367 WORDS
PER CENT OF DIRECTORY QUOTA USED =    0.175
PER CENT OF FILE USED          =    0.175
BLOCKING FACTOR              =  74.540
PAW > ldir ntuple           | list directory in NTUPLE

***** Directory ==> //LUN1/NTUPLE <==

Created 890902/1958 Modified 890902/1958

==> List of objects
  HBOOK-ID CYCLE   DATE/TIME    NDATA   OFFSET    REC1    REC2
    30        2  890902/1958   1082      215       41       42
                  1  890902/1958   1082      725       39       40
    10        1  890902/1958    151      783       40
    20        1  890902/1958    305      934       40       41

NUMBER OF RECORDS =   34 NUMBER OF MEGAWORDS =  0 + 34064 WORDS
PER CENT OF DIRECTORY QUOTA USED =    0.851
PER CENT OF FILE USED          =    0.850
BLOCKING FACTOR              =  94.899

```

Figure 7.6: Reading a HBOOK direct access file

²In fact it is possible to leave the data on the disk of the machine where they were written in the batch job, and connect with NETWORK/RLOGIN host to the machine in question, getting access to the file via TCP/IP. See page 187 for more details.

7.4.1 Plot histogram data

The analysis of the data can now start and we begin by looking at the histograms in the top directory. Figure 7.7 shows the commands entered and the corresponding output plot. They should be compared with the lineprinter output in figure 7.4.

```

Plotting histogram data
PAW > zon 1 2           | Divide picture into 2 vertically
PAW > set htyp -3        | Set hatch style for histogram
PAW > hi/pl 110          | Plot 1-dimensional histogram 110
PAW > hi/pl 210          | Plot 2-dimensional histogram 210

```

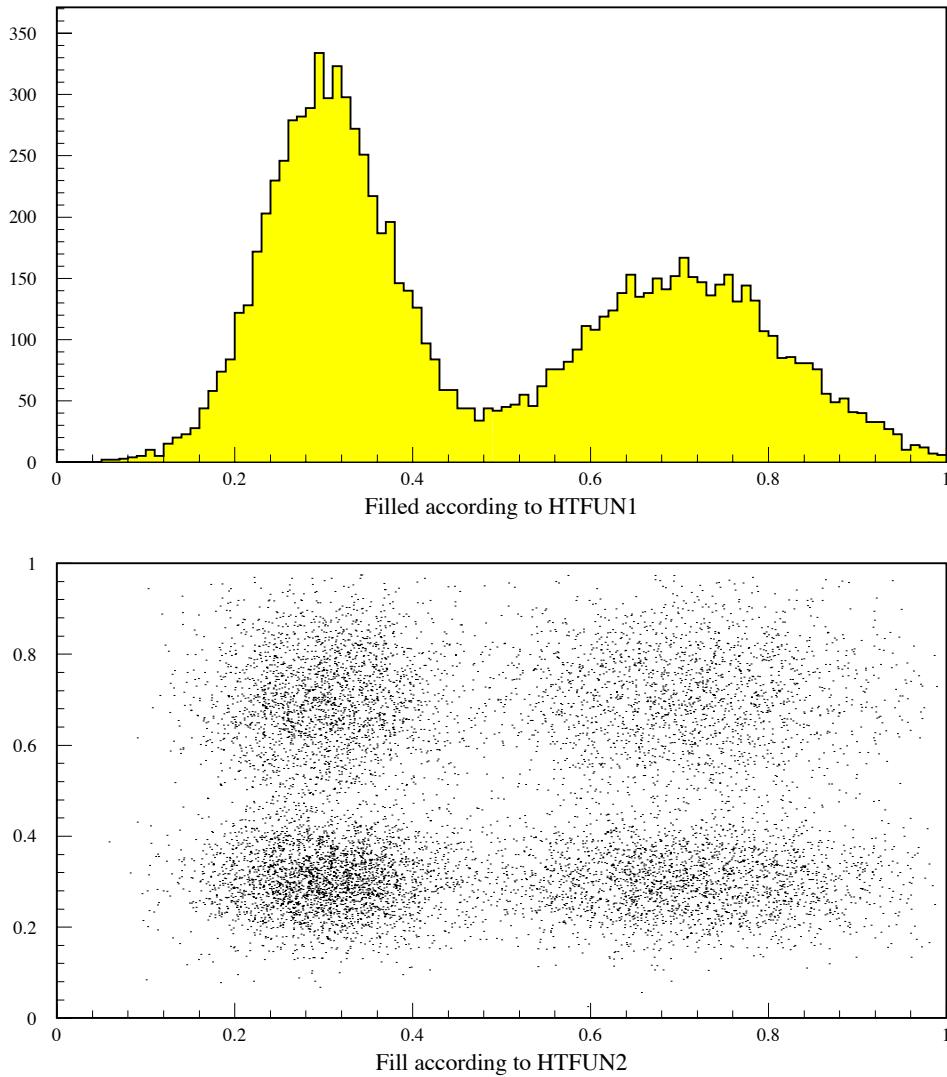


Figure 7.7: Plot of one- and two-dimensional histograms

7.5 Ntuples: A closer look

We now turn our attention to the NTUPLE directory to show the functionality and use of Ntuples. After making NTUPLE the **current** directory the available HBOOK objects are listed. The structure of the Ntuple with identifier 30 is PRINTed. The contents of the various Ntuple elements (“events”) can be viewed by the NTUPLE/SCAN command. As with most Ntuple commands a **selection criterion** can be given to treat only given “selected” subsamples of the Ntuple (two examples are seen with the further NTUPLE/SCAN commands (see figure 7.8).

Looking at Ntuple elements

```

PAW > cd ntuple                                | move to NTUPLE directory
PAW > hi/li                                  | list HBOOK objects

====> Directory : //LUN1/NTUPLE
      30 (N)   N-TUPLE
      10 (1)   TEST1
      20 (2)   TEST2

PAW > nt/print 30                            | print summary for Ntuple 30

*****
* NTUPLE ID= 30  ENTRIES= 10000  N-TUPLE          *
*****
* Var numb * Name    * Lower     * Upper      *
*****
*      1    * X      * -.422027E+01 * 0.386411E+01 *
*      2    * Y      * -.411077E+01 * 0.378365E+01 *
*      3    * Z      * 0.485188E-04 * 0.179518E+02 *
*****
PAW > nt/scan 30                            | scan the first elements
*****
* ENTRY *   X    *   Y    *   Z    *
*****
!      1 ! -1.0765 ! 1.4405 ! 3.2337 !
!      2 ! -1.2429 ! -1.6043 ! 4.1185 !
!      3 ! 0.54489 ! 1.7043 ! 3.2017 !
!      4 ! -0.81803 ! 0.66588 ! 1.1126 !
!      5 ! -1.8752 ! 0.38176 ! 3.6621 !
!      6 ! 0.37968 ! -1.0601 ! 1.2680 !
!      7 ! -0.52406 ! -0.68243E-01! 0.27930 !
!      8 ! 1.2175 ! 0.91701 ! 2.3231 !
!      9 ! -0.21487 ! -0.26670 ! 0.11730 !
!     10 ! 0.70368 ! 0.82514 ! 1.1760 !
!     11 ! 0.93648E-01! -2.0311 ! 4.1343 !
!     12 ! -0.48216 ! -2.5980 ! 6.9820 !
!     13 ! -0.45801 ! 0.71523 ! 0.72132 !
!     14 ! -0.60272 ! 0.98909E-01! 0.37306 !
!     15 ! 0.70454 ! -0.25562 ! 0.56172 !
More...? (<CR>/N): N
==>      15 events have been scanned

```

```

PAW > nt/sc 30 z>16                                | example of a condition on the Z variable
*****
* ENTRY *      X      *      Y      *      Z      *
*****
!   43 !  3.8641    ! -1.5822    ! 17.435    !
! 1964 ! -4.2203    ! -0.37562   ! 17.952    !
! 7480 !  0.94503   ! -4.1108    ! 17.791    !
! 9213 !  0.71434   ! -4.0068    ! 16.565    !
==>      4 events have been scanned

PAW > nt/sc 30 abs(x)>4.or.abs(y)>4          | example of a more complex selection criterium
*****
* ENTRY *      X      *      Y      *      Z      *
*****
! 1964 ! -4.2203   ! -0.37562   ! 17.952    !
! 7480 !  0.94503  ! -4.1108    ! 17.791    !
! 9213 !  0.71434  ! -4.0068    ! 16.565    !
==>      3 events have been scanned

```

Figure 7.8: Print and scan Ntuple elements

7.5.1 Ntuple plotting

The general format of the command NTUPLE/PLOT to project and plot a Ntuple as a (1-Dim or 2-Dim) histogram with automatic binning, possibly using a selection algorithm is:

```
NTUPLE/PLOT idn [ uwfunc nevent ifirst nupd chopt]
```

- IDN Ntuple Identifier and variable(s) (see table 7.1)
- UWFUNC Selection function (see table 7.2) - Default no function
- NEVENT Number of events to be processed (default is 999999)
- IFIRST First event to be procesed (default is 1)
- NUPD Frequency with which to update histogram (default is 1000000)
- CHOPT HPLOT options (C,S,+,B,L,P,*,U,E,A)

7.5.2 Ntuple variable and selection function specification

Format	Explanation	Example	
IDN.CHNAME	The variable named "CHNAME"	30.x	variable x
IDN.n	The Ntuple variable at position n	30.3	variable 3
IDN.expression	Expression is any numerical expression of Ntuple variables. It may include a call to a COMIS function.	30.X**2+Y**2 30.X*COMIS.FOR	
IDN.B%A	Scatter-plot of variable B versus A for each event	30.Y%X	Y versus X
IDN.2%1	Scatter-plot of variable nb. 2 versus variable nb. 1	30.1%3	1 versus 3
IDN.expr1%expr2	expr1 and expr2 can be any numerical expression of the Ntuple variables. They can be COMIS functions.	30.SQRT(X**2+Y**2)%SIN(Z) 30.COMIS1.FTN%COS(Z)	
	Any combination of the above	30.3%COMIS2.FTN*SIN(X)	

Table 7.1: Syntax for specifying Ntuple variables

Format	Explanation	Example	
0 or missing	No selection is applied (weight is 1).	<u>NT/PLOT 30.X</u>	
Combination of cuts	A CUT or combination of CUTs, each created by the command NTUPLE/CUTS	<u>NT/PLOT 30.X 1</u> (use cut 1) <u>NT/PLOT 30.X 1.AND.2</u> <u>NT/PLOT 30.X .NOT.(1.AND.3).OR.2</u>	
Combination of masks	A MASK or combination of MASKs, each created by the command NTUPLE/MASK	Assuming there exists a mask vector MSK: <u>NT/PLOT 30.X MSK(4)</u> (bit 4) <u>NT/PLOT 30.X MSK(1).OR.MSK(6)</u>	
Logical expression	Any logical combination of conditions between Ntuple variables, cuts and masks.	<u>NT/PLOT 30.X X>3.14.AND.(Y<Z+5.)</u> <u>NT/PLOT 30.X 1.AND.MASK(3).OR.Z<10</u>	
Numerical expression	Any numerical combination of constants and Ntuple variables. In this case the value of the expression will be applied as a weight to the element being plotted.	<u>NT/PLOT 30.X Y</u> weight X by Y <u>NT/PLOT 30.X X**2+Y**2</u> weight X by <u>X²+Y²</u>	
Selection function	Name of a selection function in a text file of the form fun.ftn (Unix), FUN FORTRAN (IBM) and FUN.FOR (VAX). The function value is applied as a weight	<u>NTUPLE/PLOT 30.X SELECT.FOR</u> For each event the plotted value of X will be multiplied by the value of the selection function SELECT calculated for that event.	
	Any combination of the above	<u>NT/PL 30.3%F1.FTN*SIN(X) 1.OR.F2.FTN</u>	

Table 7.2: Syntax of a selection function used with a Ntuple

7.5.3 Ntuple selection mechanisms

With most Ntuple operations a selection “function” UWFUNC of a form described in table 7.2 can be used, i.e. it can take the form of a simple or composed **expression** or an **external FORTRAN function**, executed by COMIS [1], a **cut** or a **mask**. When used together with the NTUPLE/PLOT command the selection function also acts as a **weighting factor**.

7.5.4 Masks

The mask facility allows the user to specify up to **32** selection criteria associated with a Ntuple. These criteria are defined like cuts, but their value for each event are written to an external direct access file, from which the information can be readily retrieved at a later stage, without recalculating the condition value in question. In the example session below first a **new** mask file MNAME.MASK is defined, which can contain information for up to 10000 Ntuple elements. Next we define event election criteria and store their result at various bit positions in the mask vector MNAME.

Defining cuts and masks

```

PAW > NT/CUT 4 Z>X**2                                | Define cut 4
PAW > NT/MASK MNAME N 10000
PAW > NT/PLOT 30.X X**2+Y**2>2>>MNAME(1)
PAW > NT/PLOT 30.X 4.AND.Y>1>>MNAME(2)
PAW > NT/PLOT 30.Y SIN(Z).GT.SIN(Y)>>MNAME(3)
PAW > NT/MASK MNAME P                                | Print mask definitions

=====> Current active selections in mask MNAME

      Bit   Nevents     Selection
      1      3723     X**2+Y**2>2
      2      1558     4.AND.Y>1
      3      7051     SIN(Z)>SIN(Y)

PAW > NT/MASK MNAME C                                | close MNAME.MASK file

```

Of course doing this kind of gymnastics makes sense only if a **time consuming** selection mechanism is used and only a few events are selected. In a subsequent run the mask file can then be read to display the information much more quickly.

Using a mask file of a previous run

```

PAW > NT/MASK MNAME                                | open the mask file for read
PAW > NT/PLOT 30.X MNAME(1)                        | plot using bit 1
PAW > NT/PLOT 30.X MNAME(2)                        | plot using bit 2
PAW > NT/PLOT 30.Y MNAME(3)                        | plot using bit 3
PAW > NT/MASK MNAME C                                | close MNAME.MASK file

```

Cuts

A **cut** is identified by an integer (between 0 and 100) and is a **logical expression** of Ntuple elements, other cuts, masks or functions.

Example of cuts

PAW > <u>NT/CUT 1</u> $4 < X$	variable
PAW > <u>NT/CUT 2</u> $0.4 < X < 0.8 \text{ AND } Y < \text{SQRT}(Z)$	ditto
PAW > <u>NT/CUT 3</u> <u>FUN.FOR</u>	external function
PAW > <u>NT/CUT 4</u> <u>FUN.FOR.AND.Z>X**2</u>	ditto plus variable
PAW > <u>NT/CUT 5</u> $(1 \text{ AND } 2) \text{ OR } 4$	combination of cuts
PAW > <u>NT/CUT 6</u> $1 \text{ AND } Z < 0$	cut and variable
PAW > <u>NT/CUT 7</u> <u>X</u>	event weight
PAW > <u>NT/CUT 8</u> <u>SQRT(Y)</u>	ditto
PAW > <u>NT/CUT 9</u> <u>MASK(23)</u> $\text{AND} .8$	mask and cut

Cut definitions can be written to a file and later re-read.

PAW > <u>NT/CUT 0</u> <u>W</u> <u>cuts.dat</u>	write all cuts to file
PAW > <u>NT/CUT 4</u> <u>R</u> <u>cuts.dat</u>	read cut 4 from file
PAW > <u>NT/CUT 4</u> <u>P</u>	print cut 4

CUT number= 4 Points= 1 Variable= 1
FUN.FOR.AND.Z>X**2

Graphical cut

One can also define a cut on the screen in a **graphical** way, by pointing out the upper and lower limits (1-dimensional case) or an area defined by up to 20 points (2-dimensional case) by using the mouse or arrow keys (see figure 7.9).

Note that graphical cuts are only valid for the **original** Ntuple variables and not for combinations of the latter.

Using graphical cuts

PAW > <u>nt/pl 30.x%y</u>	plot y versus x
PAW > <u>CUT 1 G</u>	graphical cut 1 for current plot
PAW > <u>zon 1 2</u>	define picture layout
PAW > <u>title 'Graphical cuts'</u>	title for picture
PAW > <u>2d 211 'X versus Y' 50 -2.5 2.5 50 -2.5 2.5 0.</u>	user binning
PAW > <u>1d 212 'X - Before and after cut' 60 -3. 3. 0.</u>	ditto
PAW > <u>1d 213 'Y - Before and after cut' 60 -3. 3. 0.</u>	ditto
PAW > <u>nt/pl 30.x%y ! -211</u>	plot y versus x in histogram 211
PAW > <u>cut 1 d</u>	draw graphical cut 1
PAW > <u>zon 2 2 3 s</u>	redefine the picture layout
PAW > <u>nt/pl 30.x ! -212</u>	plot x BEFORE cut in histogram 212
PAW > <u>set htyp -3</u>	use hatch for plot after cut
PAW > <u>nt/pl 30.x 1 -212 ! ! S</u>	plot x AFTER cut on same plot
PAW > <u>set htyp 0</u>	no hatch for plot without cut
PAW > <u>nt/pl 30.y ! -213</u>	plot y BEFORE cut in histogram 213
PAW > <u>set htyp -3</u>	use hatch for plot after cut
PAW > <u>nt/pl 30.y 1 -213 ! ! S</u>	plot y AFTER cut on same plot

Graphical cuts

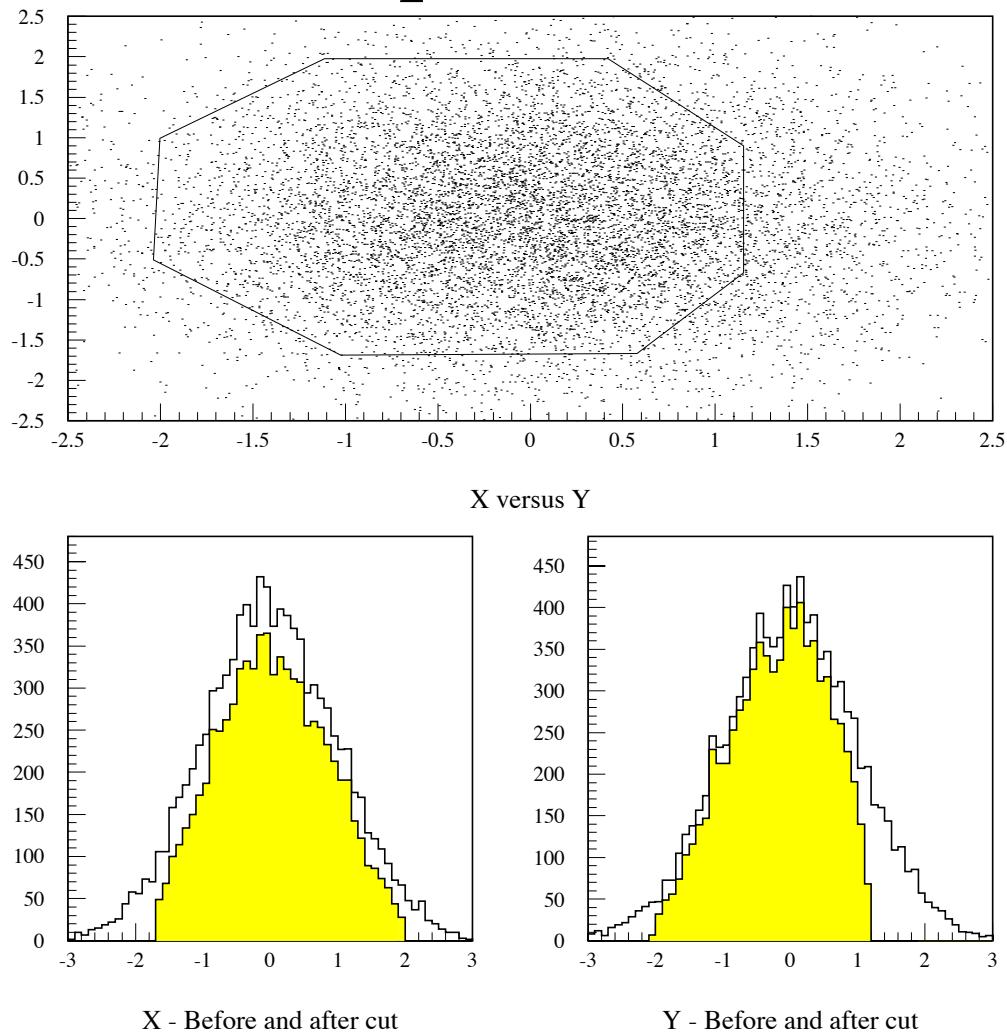


Figure 7.9: Graphical definition of cuts

COMIS selection function

In the definition of a selection criterion an external function (in the sense that it has not been compiled and linked together with PAW) can be used. This function is interpreted by the COMIS [1] package. The functions which are callable from within such a function are given below.

Type of function	List of callable routines
FORTRAN library	SQRT LOG LOG10 EXP SIN COS TAN ASIN ACOS ATAN2 ABS MOD MIN MAX INT REAL DBLE LEN INDEX
HBOOK package	HBOOK1 HBOOK2 HBOOKN HFILL HF1 HPRINT HDELET HRESET HFITGA HFITPO HFITEX HPROJ1 HPROJ2 HFN HGNPAR HROPEN PAOPEN PACLOS PAREAD PAWRIT HPAK HPAKE HUNPAK HGIVE HGN HGNF HF2 HFF1 HFF2 HBFUN1 HBFUN2 HRIN HROUT HI HIE HIX HIJ HIDALL HNOENT HX HXY HCOPY HSTATI HBPROF HOPERA HIDOPT HDERIV HRNDM1 HRNDM2 HBARX HBARY
ZEBRA package	FZIN FZOUT FZENDI FZENDO FZFILE RZCDIR RZLDIR RZFILE RZEND RZIN RZOUT RZVIN RZVOUT
HPLOT package	HPLOT HPLSYM HPLERR HPLEGO HPLNT HPLSUR HPLSOF HPLSET HPLGIV HPLLOC HPLSET HPLGIV HPLLOC
KUIP package	KUGETV KUDPAR KUVECT KILEXP KUTIME KUEXEL
HIGZ package	IPL IPM IFA IGTEXT IGBOX IGAXIS IGPIE IGRAPH IGHIST IGARC IGLBL IGRNG IGMETA IGSA IGSET IRQLC IRQST ISELNT ISFAIS ISFASI ISLN ISMK ISVP ISWN ITX ICLRWK ISCR
KERNLIB library	JBIT JBYT LENNOC RANNOR RNDM SBITO SBIT1 SBYT UCOPY UCTOH UHTOC VZERO
COMMON blocks	/PAWC/, /QUEST/, /KCWORK/, /PAWPAR/, /PAWIDN/

Table 7.3: Function callable and common blocks which can be referenced from an external function with PAW.

The command NTUPLE/UWFUNC allows a selection function for a Ntuple to be prepared more easily. It generates a function with a name specified by the user and with code making available the variables corresponding to the given Ntuple identifier via a COMMON block. As an example consider the Ntuple number 30 used previously.

Specifying a user selection function

```

PAW > NTUPLE/UWFUNC 30 SELECT.FOR PT | Generate SELECT.FOR
PAW > EDIT SELECT.FOR           | Look at file SELECT.FOR
      REAL FUNCTION SELECT(XDUMMY)
      REAL X , , Z
      COMMON/PAWIDN/IDNEVT,VIDN1,VIDN2,VIDN3,
      + X , , Z
      DIMENSION XDUMMY( 3)
      CHARACTER*8 CHTAGS( 3)
      DATA CHTAGS/' X , , Y , , Z /'
*
      SELECT=1.
      PRINT 1000, IDNEVT
      DO 10 I=1, 3
          PRINT 2000, I, CHTAGS(I), XDUMMY(I)
10    CONTINUE
*
1000 FORMAT(8H IDNEVT=,I5)
2000 FORMAT(5X,I3,5X,A,1H=,G14.7)
END

```

The user can add further FORTRAN code with the command EDIT. Remember that the value of the function can be used for weighting each event.

7.5.5 Examples

To put into practice the syntax explained above let us consider figure 7.10. We first plot variable Z with the binning automatically calculated by HBOOK. Then we define a histogram with identifier 300 into which we want HBOOK to plot the squared sums of the elements X and Y. This corresponds to the definition of the Z variable as can be seen in the FORTRAN listing in figure 7.5. As the MEAN and RMS are only calculated on the events within the histogram boundaries, they differ slightly between the top and bottom plot in figure 7.10.

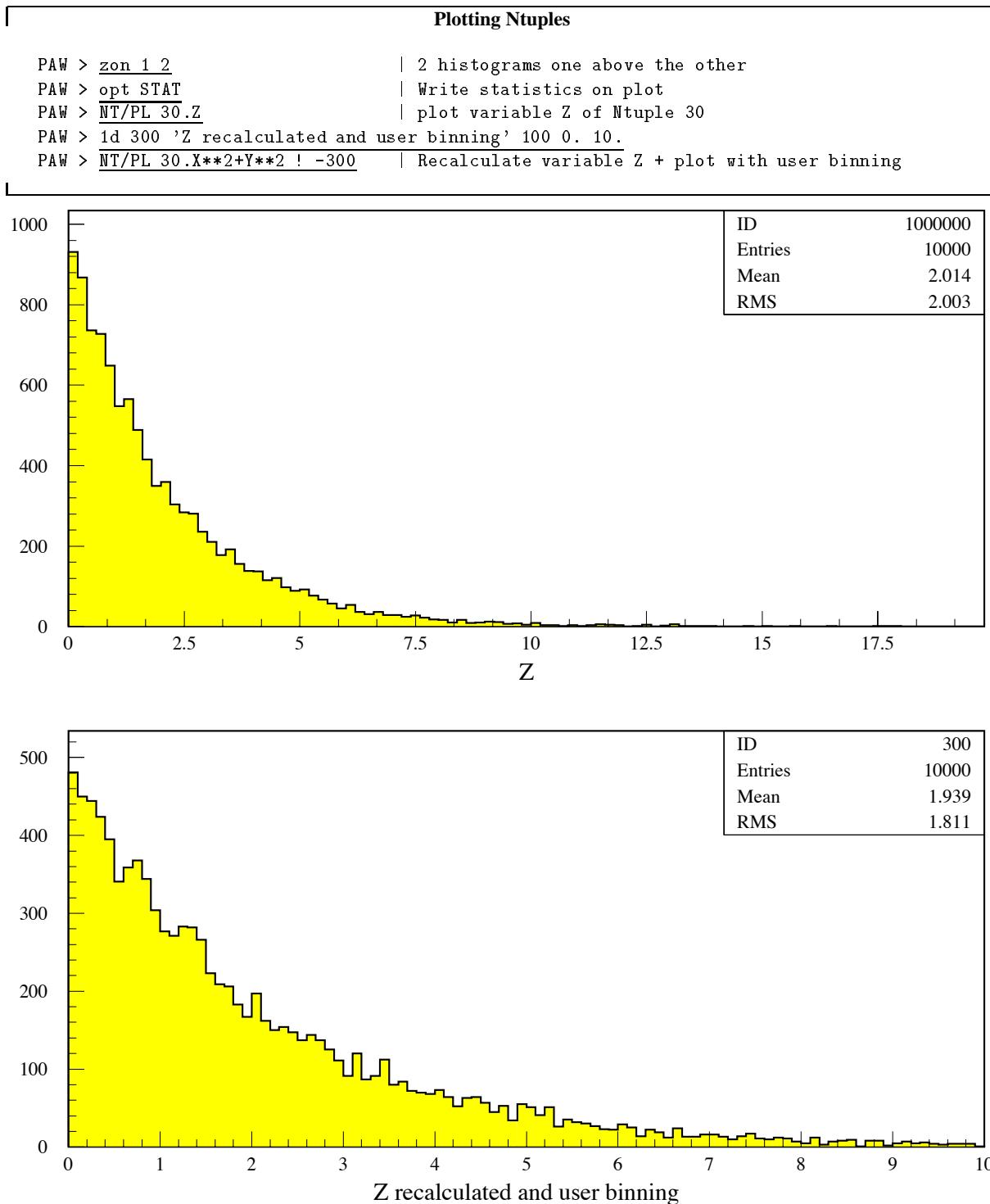


Figure 7.10: Read and plot Ntuple elements

More complex Ntuple presentations

PAW > <u>zon 2 2</u>	Divide plot in 4 zones
PAW > <u>opt STAT</u>	Select option to write statistics on plot
PAW > <u>set HTYP -3</u>	Define histogram hatch type
PAW > <u>1d 401 'NT/PL - X' 100. -2.5 2.5</u>	Book 1 dim histogram
PAW > <u>nt/pl 30.1 ! -401</u>	Plot variable 1 (x) using histogram 401
PAW > <u>1d 402 'NT/PL E option - Y' 100. -2.5 2.5</u>	1 dim histogram (different title)
PAW > <u>igset mtyp 21</u>	Select market type for points on plot
PAW > <u>nt/pl 30.y ! -402 !! E</u>	Plot y variable with Error bar option
PAW > <u>1d 403 'NT/PL B option - X' 40. -2.5 2.5</u>	1 dim histogram (different title + binning)
PAW > <u>set barw 0.4</u>	Define bar width for bar chart
PAW > <u>set baro 0.3</u>	Define bar origin for bar chart
PAW > <u>csel NB 0.33</u>	Print selection criterion on plot
PAW > <u>set hcol 1001</u>	Histogram colour black
PAW > <u>nt/pl 30.x y>0 -403 !! b</u>	Plot x variable as bar chart
PAW > <u>1d 404 'NT/PL PL option - Y' 100. -2.5 2.5</u>	1 dim histogram (different title)
PAW > <u>max 404 160</u>	Fix maximum for plotting hist 404
PAW > <u>nt/pl 30.y sqrt(z)>1 -404 !! pl</u>	Plot y variable with PL option

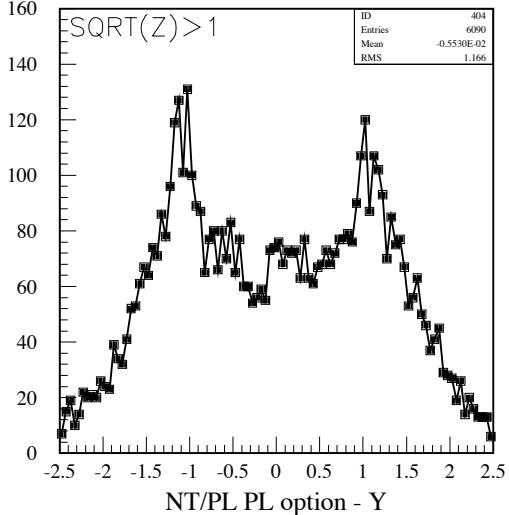
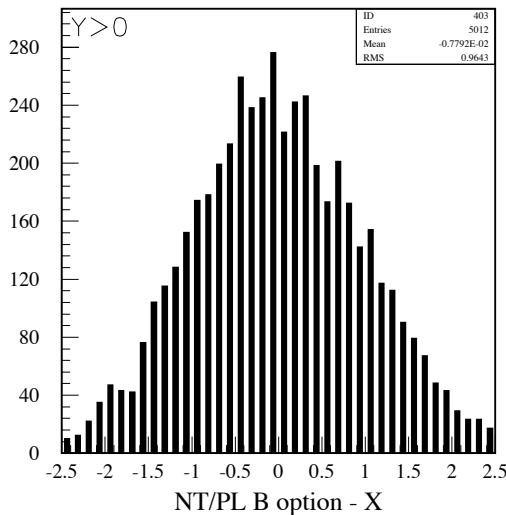
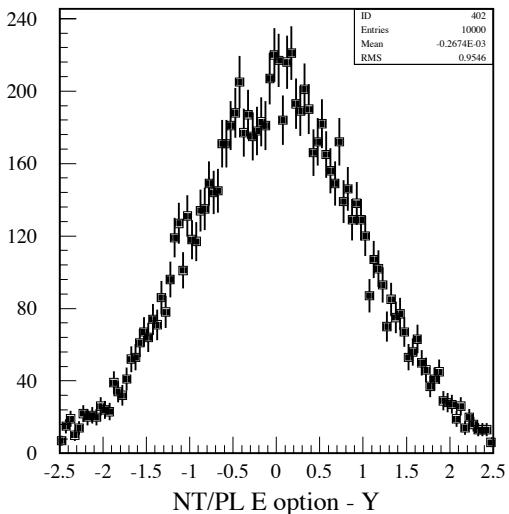
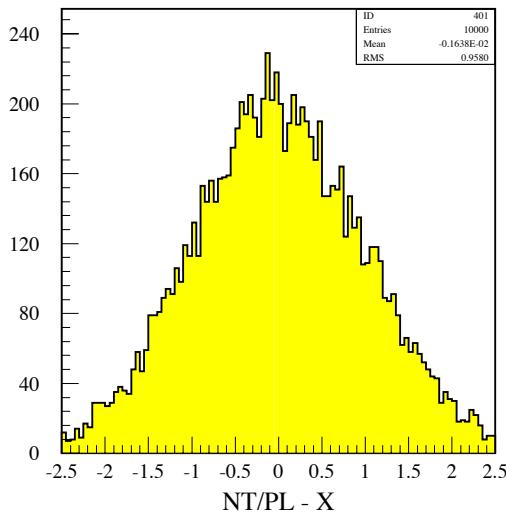


Figure 7.11: Selection functions and different data presentations

7.6 Fitting with PAW/HBOOK/MINUIT

Minuit[6]³ is conceived as a tool to find the minimum value of a multi-parameter function and analyze the shape of the function around the minimum. The principal application is foreseen for statistical analysis, working on chisquare or log-likelihood functions, to compute the best-fit parameter values and uncertainties, including correlations between the parameters. It is especially suited to handle difficult problems, including those which may require guidance in order to find the correct solution.

7.6.1 Basic concepts of MINUIT.

The MINUIT package acts on a multiparameter FORTRAN function to which one must give the generic name FCN. In the PAW/HBOOK implementation, the function FCN is called `HFCNH` when the command `Histo/Fit` (PAW) or the routine `HFITH` are invoked. It is called `HFCNV` when the command `Vector/Fit` or the routine `HFITV` are invoked. The value of FCN will in general depend on one or more variable parameters.

To take a simple example, suppose the problem is to fit a polynomial through a set of data points with the command Vector/Fit. Routine `HFCNV` called by `HFITV` calculates the chisquare between a polynomial and the data; the variable parameters of `HFCNV` would be the coefficients of the polynomials. Routine `HFITV` will request MINUIT to minimize `HFCNV` with respect to the parameters, that is, find those values of the coefficients which give the lowest value of chisquare.

7.6.2 Basic concepts - The transformation for parameters with limits.

For variable parameters with limits, MINUIT uses the following transformation:

$$P_{\text{int}} = \arcsin \left(2 \frac{P_{\text{ext}} - a}{b - a} - 1 \right) \quad P_{\text{ext}} = a + \frac{b - a}{2} (\sin P_{\text{int}} + 1)$$

so that the internal value P_{int} can take on any value, while the external value P_{ext} can take on values only between the lower limit a and the upper limit b . Since the transformation is necessarily non-linear, it would transform a nice linear problem into a nasty non-linear one, which is the reason why limits should be avoided if not necessary. In addition, the transformation does require some computer time, so it slows down the computation a little bit, and more importantly, it introduces additional numerical inaccuracy into the problem in addition to what is introduced in the numerical calculation of the FCN value. The effects of non-linearity and numerical roundoff both become more important as the external value gets closer to one of the limits (expressed as the distance to nearest limit divided by distance between limits). The user must therefore be aware of the fact that, for example, if he puts limits of $(0, 10^{10})$ on a parameter, then the values 0.0 and 1.0 will be indistinguishable to the accuracy of most machines.

The transformation also affects the parameter error matrix, of course, so MINUIT does a transformation of the error matrix (and the “parabolic” parameter errors) when there are parameter limits. Users should however realize that the transformation is only a linear approximation, and that it cannot give a meaningful result if one or more parameters is very close to a limit, where $\partial P_{\text{ext}} / \partial P_{\text{int}} \approx 0$. Therefore, it is recommended that:

- Limits on variable parameters should be used only when needed in order to prevent the parameter from taking on unphysical values.

³The following information about Minuit has been extracted from the Minuit documentation.

- When a satisfactory minimum has been found using limits, the limits should then be removed if possible, in order to perform or re-perform the error analysis without limits.

7.6.3 How to get the right answer from MINUIT.

MINUIT offers the user a choice of several minimization algorithms. The MIGRAD (Other algorithms are available with Interactive MINUIT, as described on Page 151) algorithm is in general the best minimizer for nearly all functions. It is a variable-metric method with inexact line search, a stable metric updating scheme, and checks for positive-definiteness. Its main weakness is that it depends heavily on knowledge of the first derivatives, and fails miserably if they are very inaccurate. If first derivatives are a problem, they can be calculated analytically inside the user function and communicated to PAW via the routine HDERIV.

If parameter limits are needed, in spite of the side effects, then the user should be aware of the following techniques to alleviate problems caused by limits:

Getting the right minimum with limits.

If MIGRAD converges normally to a point where no parameter is near one of its limits, then the existence of limits has probably not prevented MINUIT from finding the right minimum. On the other hand, if one or more parameters is near its limit at the minimum, this may be because the true minimum is indeed at a limit, or it may be because the minimizer has become “blocked” at a limit. This may normally happen only if the parameter is so close to a limit (internal value at an odd multiple of $\pm \frac{\pi}{2}$) that MINUIT prints a warning to this effect when it prints the parameter values.

The minimizer can become blocked at a limit, because at a limit the derivative seen by the minimizer $\partial F / \partial P_{\text{int}}$ is zero no matter what the real derivative $\partial F / \partial P_{\text{ext}}$ is.

$$\frac{\partial F}{\partial P_{\text{int}}} = \frac{\partial F}{\partial P_{\text{ext}}} \frac{\partial P_{\text{ext}}}{\partial P_{\text{int}}} = \frac{\partial F}{\partial P_{\text{ext}}} = 0$$

Getting the right parameter errors with limits.

In the best case, where the minimum is far from any limits, MINUIT will correctly transform the error matrix, and the parameter errors it reports should be accurate and very close to those you would have got without limits. In other cases (which should be more common, since otherwise you wouldn’t need limits), the very meaning of parameter errors becomes problematic. Mathematically, since the limit is an absolute constraint on the parameter, a parameter at its limit has no error, at least in one direction. The error matrix, which can assign only symmetric errors, then becomes essentially meaningless.

7.6.4 Interpretation of Parameter Errors:

There are two kinds of problems that can arise: the **reliability** of MINUIT’s error estimates, and their **statistical interpretation**, assuming they are accurate.

Statistical interpretation:

For discussion of basic concepts, such as the meaning of the elements of the error matrix, or setting of exact confidence levels, see [13, 14, 15].

Reliability of MINUIT error estimates.

MINUIT always carries around its own current estimates of the parameter errors, which it will print out on request, no matter how accurate they are at any given point in the execution. For example, at initialization, these estimates are just the starting step sizes as specified by the user. After a MIGRAD or HESSE step, the errors are usually quite accurate, unless there has been a problem. MINUIT, when it prints out error values, also gives some indication of how reliable it thinks they are. For example, those marked CURRENT GUESS ERROR are only working values not to be believed, and APPROXIMATE ERROR means that they have been calculated but there is reason to believe that they may not be accurate.

If no mitigating adjective is given, then at least MINUIT believes the errors are accurate, although there is always a small chance that MINUIT has been fooled. Some visible signs that MINUIT may have been fooled are:

- Warning messages produced during the minimization or error analysis.
- Failure to find new minimum.
- Value of EDM too big (estimated Distance to Minimum).
- Correlation coefficients exactly equal to zero, unless some parameters are known to be uncorrelated with the others.
- Correlation coefficients very close to one (greater than 0.99). This indicates both an exceptionally difficult problem, and one which has been badly parameterized so that individual errors are not very meaningful because they are so highly correlated.
- Parameter at limit. This condition, signalled by a MINUIT warning message, may make both the function minimum and parameter errors unreliable. See the discussion above “*Getting the right parameter errors with limits*”.

The best way to be absolutely sure of the errors, is to use “independent” calculations and compare them, or compare the calculated errors with a picture of the function. Theoretically, the covariance matrix for a “physical” function must be positive-definite at the minimum, although it may not be so for all points far away from the minimum, even for a well-determined physical problem. Therefore, if MIGRAD reports that it has found a non-positive-definite covariance matrix, this may be a sign of one or more of the following:

A non-physical region: On its way to the minimum, MIGRAD may have traversed a region which has unphysical behaviour, which is of course not a serious problem as long as it recovers and leaves such a region.

An underdetermined problem: If the matrix is not positive-definite even at the minimum, this may mean that the solution is not well-defined, for example that there are more unknowns than there are data points, or that the parameterization of the fit contains a linear dependence. If this is the case, then MINUIT (or any other program) cannot solve your problem uniquely, and the error matrix will necessarily be largely meaningless, so the user must remove the underdeterminedness by reformulating the parameterization. MINUIT cannot do this itself.

Numerical inaccuracies: It is possible that the apparent lack of positive-definiteness is in fact only due to excessive roundoff errors in numerical calculations in the user function or not enough precision. This is unlikely in general, but becomes more likely if the number of free parameters is very large, or if

the parameters are badly scaled (not all of the same order of magnitude), and correlations are also large. In any case, whether the non-positive-definiteness is real or only numerical is largely irrelevant, since in both cases the error matrix will be unreliable and the minimum suspicious.

An ill-posed problem: For questions of parameter dependence, see the discussion above on positive-definiteness.

Possible other mathematical problems are the following:

Excessive numerical roundoff: Be especially careful of exponential and factorial functions which get big very quickly and lose accuracy.

Starting too far from the solution: The function may have unphysical local minima, especially at infinity in some variables.

7.6.5 Fitting histograms

The general syntax of the command to fit histograms is:

```
HISTOGRAM id func [ chopt np par step pmin pmax errpar ]
```

Only the parameters, which are of more general use, are described in detail. The full description can be found in part 3 of this manual.

ID A histogram identifier (1-dim or 2-dim)
 A bin range may be specified, e.g. Histo/Fit 10(25:56) ...
 FUNC Name of a function to be fitted to the histogram.
 This function can be of various forms:

- 1 The name of a file which contains the user defined function to be minimized. Function name and file name must be the same. For example file FUNC.FOR is:

```
FUNCTION FUNC(X)    or FUNC(X,Y) for a 2-Dim histogram  

COMMON/PAWPAR/PAR(2)  

FUNC=PAR(1)*X +PAR(2)*EXP(-X)  

END
```

- 2 One of the keywords below (**1-dim histograms only**), which will use the parameterization described at the right for the fit.

```
G     Func=par(1)*exp(-0.5*((x-par(2))/par(3))**2)  

E     Func=exp(par(1)+par(2)*x)  

Pn    Func=par(1)+par(2)*x+par(3)*x**2...+par(n+1)*x**n, 0<n<20
```

- 3 A combination of the keywords above with the 2 operators + or *.

Note that in this case, the order of parameters in PAR must correspond to the order of the basic functions. Blanks are not allowed in the expression.

CHOPT All options of the HISTO/PLOT command plus the following additional ones:

- 0 Do not plot the result of the fit. By default the fitted function is drawn unless the option “N” below is specified.
- B Some or all parameters are bounded. In this case vectors STEP, PMIN, PMAX must be specified. Default is: All parameters vary freely.

- D The user is assumed to compute derivatives analytically using routine HDERIV. By default, derivatives are computed numerically.
 - L Use Log Likelihood method. Default is χ^2 method.
 - M Invokes interactive Minuit (See on Page 151)
 - N Do not store the result of the fit bin by bin with the histogram. By default the function is calculated at the centre of each bin and the fit results stored with the histogram data structure.
 - Q Quiet mode. No output printed about the fit.
 - V Verbose mode. Results are printed after each iteration. By default only final results are printed.
 - W Sets weights equal to 1.
- NP Number of parameters in fit ($0 \leq NP \leq 34$)
- PAR Vector containing the fit parameters.
Before the fit: Vector containing the initial values
After the fit: Vector containing the fitted values.
- STEP Vector with step size for fit parameters
- PMIN Vector with lower bounds for fit parameters
- PMAX Vector with upper bounds for fit parameters
- ERRPAR Vector with errors on the fitted parameters

When using predefined functions (case 2 for the FUNC parameter) initial values need not be specified when NP=0. In this case the parameter vector PAR, if specified, is only filled with the fitted parameters on **output**.

7.6.6 A simple fit with a gaussian

Example of simple fit with gaussian in PAW

```

PAW > opt stat      | Select option to show histogram statistics on plot
PAW > opt fit       | Select option to show fitted parameters on plot
PAW > hi/fit 10 G  | Fit histogram 10 with a single gaussian
*****
*                                         *
* Function minimization by SUBROUTINE HFITGA *
* Variable-metric method                   *
* ID =          10  CHOPT = T             *
*                                         *
*****
Convergence when estimated distance to minimum (EDM) .LT. 0.10E-03

FCN= 96.97320    FROM MIGRAD    STATUS=CONVERGED  CALLS= 549 EDM= 0.26E-03
                           STRATEGY= 1    ERROR DEF= 1.0000

INT EXT  PARAMETER                      STEP      FIRST
NO. NO.   NAME      VALUE     ERROR      SIZE      DERIVATIVE
 1 1 Constant      239.83    2.8178    0.00000   0.57627E-02
 2 2 Mean         -0.53038E-02 0.77729E-04  0.00000    22.025
 3 3 Sigma         0.98766   0.70224E-02  0.00000   -0.88534

CHISQUARE = 0.1021E+01  NPFIT = 98

```

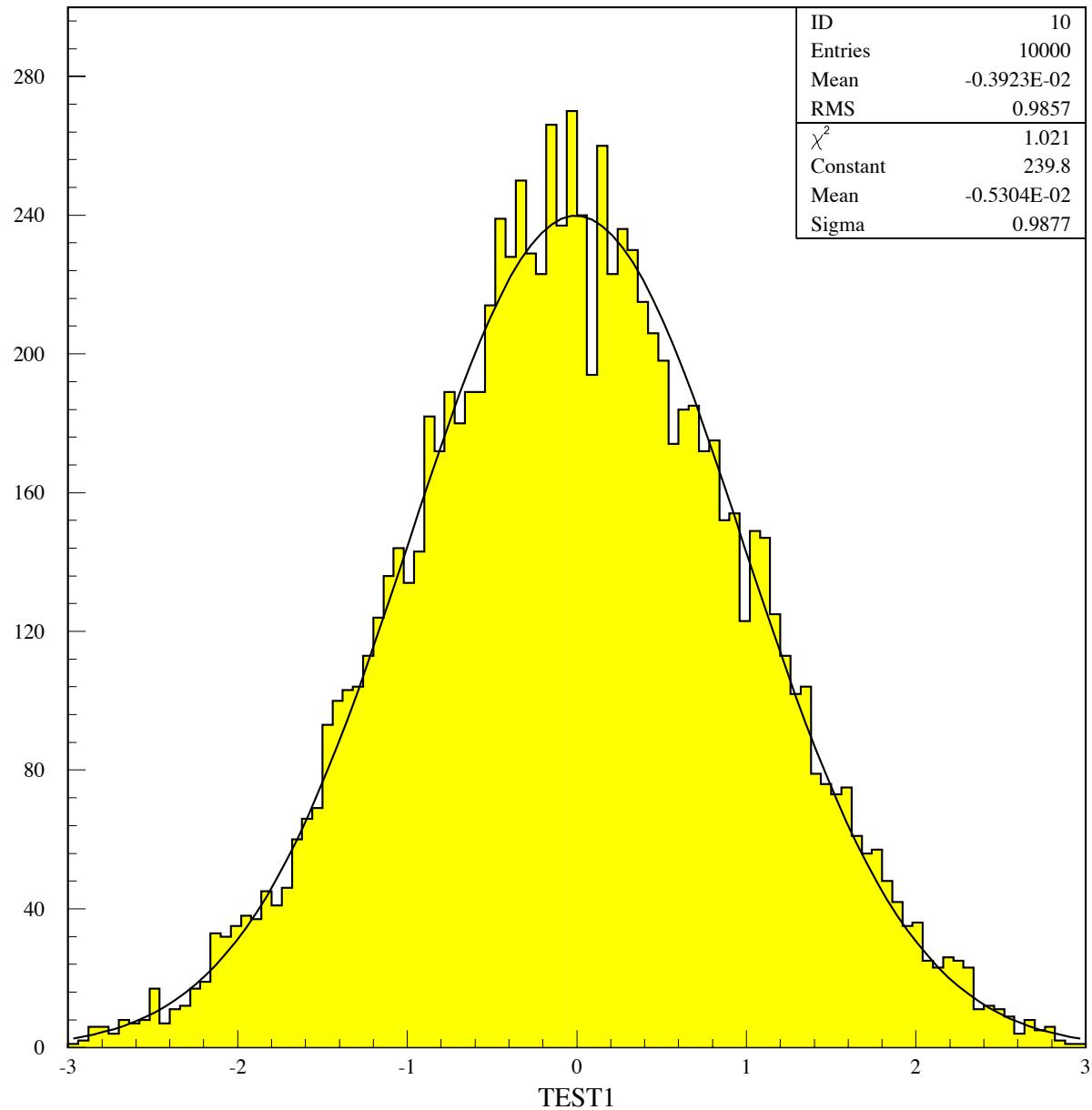


Figure 7.12: Example of a simple fit of a one-dimensional distribution

Fit parts of histogram separately					
PAW > <u>opt NSTA</u>	Turn off option showing statistics on plot				
PAW > <u>ve/cr par(6)</u>	Create a vector with 6 elements				
PAW > <u>set fit 111</u>	Show fitted parameters + errors on plot				
PAW > <u>hi/fit 110(1:50) G ! 0 par</u>	Fit first half with a gaussian and plot				
<hr/>					
***** * * Function minimization by SUBROUTINE HFITGA * * Variable-metric method * * ID = 110 CHOPT = TR * * * *****					
Convergence when estimated distance to minimum (EDM) .LT. 0.10E-03					
FCN= 90.66560 FROM MIGRAD STATUS=CONVERGED CALLS= 152 EDM= 0.68E-05 STRATEGY= 1 ERROR DEF= 1.0000					
INT EXT PARAMETER STEP FIRST NO. NO. NAME VALUE ERROR SIZE DERIVATIVE					
1 1 Constant 300.28 5.0681 0.13342 0.97075E-04					
2 2 Mean 0.30698 0.10511E-02 -0.13885E-04 -0.57797					
3 3 Sigma 0.73832E-01 0.67896E-03 -0.57602E-04 -4.6407					
CHISQUARE = 0.2159E+01 NPFIT = 45					
PAW > <u>hi/fit 110(50:99) G 0 0 par(4)</u> Fit second half with gaussian, do not plot					
<hr/>					
***** * * Function minimization by SUBROUTINE HFITGA * * Variable-metric method * * ID = 110 CHOPT = TR * * * *****					
Convergence when estimated distance to minimum (EDM) .LT. 0.10E-03					
FCN= 30.16534 FROM MIGRAD STATUS=CONVERGED CALLS= 221 EDM= 0.87E-04 STRATEGY= 1 ERROR DEF= 1.0000					
INT EXT PARAMETER STEP FIRST NO. NO. NAME VALUE ERROR SIZE DERIVATIVE					
1 1 Constant 153.27 3.0227 0.65005E-01 0.36877E-02					
2 2 Mean 0.70186 0.19599E-02 0.40388E-03 4.8103					
3 3 Sigma 0.11965 0.18242E-02 -0.25292E-03 6.9011					
CHISQUARE = 0.6418E+00 NPFIT = 50					
PAW > <u>hi/plot 110 SFUNC</u> Plot result of fit on Same plot					
PAW > <u>ve/pr par(1:6)</u> Print the fitted parameters in PAR					
PAR (1) = 300.2846					
PAR (2) = 0.3069752					
PAR (3) = 0.7383241E-01					
PAR (4) = 153.2716					
PAR (5) = 0.7018576					
PAR (6) = 0.1196475					

Parameter	Input value	Result of Figure 7.13	Result of Figure 7.14
<u>First Gaussian:</u>			
Height	1. (normalised)	$300. \pm 5.$	$308. \pm 5.$
Mean value	0.3	0.307 ± 0.001	0.303 ± 0.001
Width (sigma)	0.07	0.074 ± 0.001	0.070 ± 0.001
<u>Second Gaussian:</u>			
Height	0.5 (normalised)	$153. \pm 3.$	$154. \pm 4.$
Mean value	0.7	0.702 ± 0.002	0.703 ± 0.002
Width (sigma)	0.12	0.120 ± 0.002	0.119 ± 0.002

Table 7.4: Results for the fitted parameters of the gaussian distributions as compared to the initial values which the gaussian distributions were generated in the “batch” job in figure 7.3. The table also includes the result of the double gaussian fit in section 7.14

Example of a more complex fit

```

PAW > * Create vector of 6 elements and give initial values for combined fit of two gaussians
PAW > ve/cr par2(6) r 200 0.3 0.1 100 0.7 0.1 | initial values for the 6 fit parameters
PAW > set fit 111 | display fitted parameters plus errors
PAW > hi/fit 110(2:99) G+G ! 6 par2 | perform the fit (sum of 2 gaussians)

*****
*
* Function minimization by SUBROUTINE HFITH *
* Variable-metric method *
* ID = 110 CHOPT = R *
*
*****
Convergence when estimated distance to minimum (EDM) .LT. 0.10E-03

FCN= 57.41251 FROM MIGRAD STATUS=CONVERGED CALLS= 597 EDM= 0.10E-03
STRATEGY= 1 ERROR DEF= 1.0000

INT EXT PARAMETER STEP FIRST
NO. NO. NAME VALUE ERROR SIZE DERIVATIVE
1 1 P1 307.86 5.3896 1.3393 -0.51814E-03
2 2 P2 0.30265 0.10750E-02 0.18577E-03 3.5622
3 3 P3 0.70029E-01 0.86285E-03 0.19967E-03 11.689
4 4 P4 153.62 3.0170 0.73111 0.30406E-02
5 5 P5 0.70303 0.20652E-02 0.43051E-03 -1.2694
6 6 P6 0.11865 0.18645E-02 0.39360E-03 3.2237

CHISQUARE = 0.6524E+00 NPFIT = 94

```

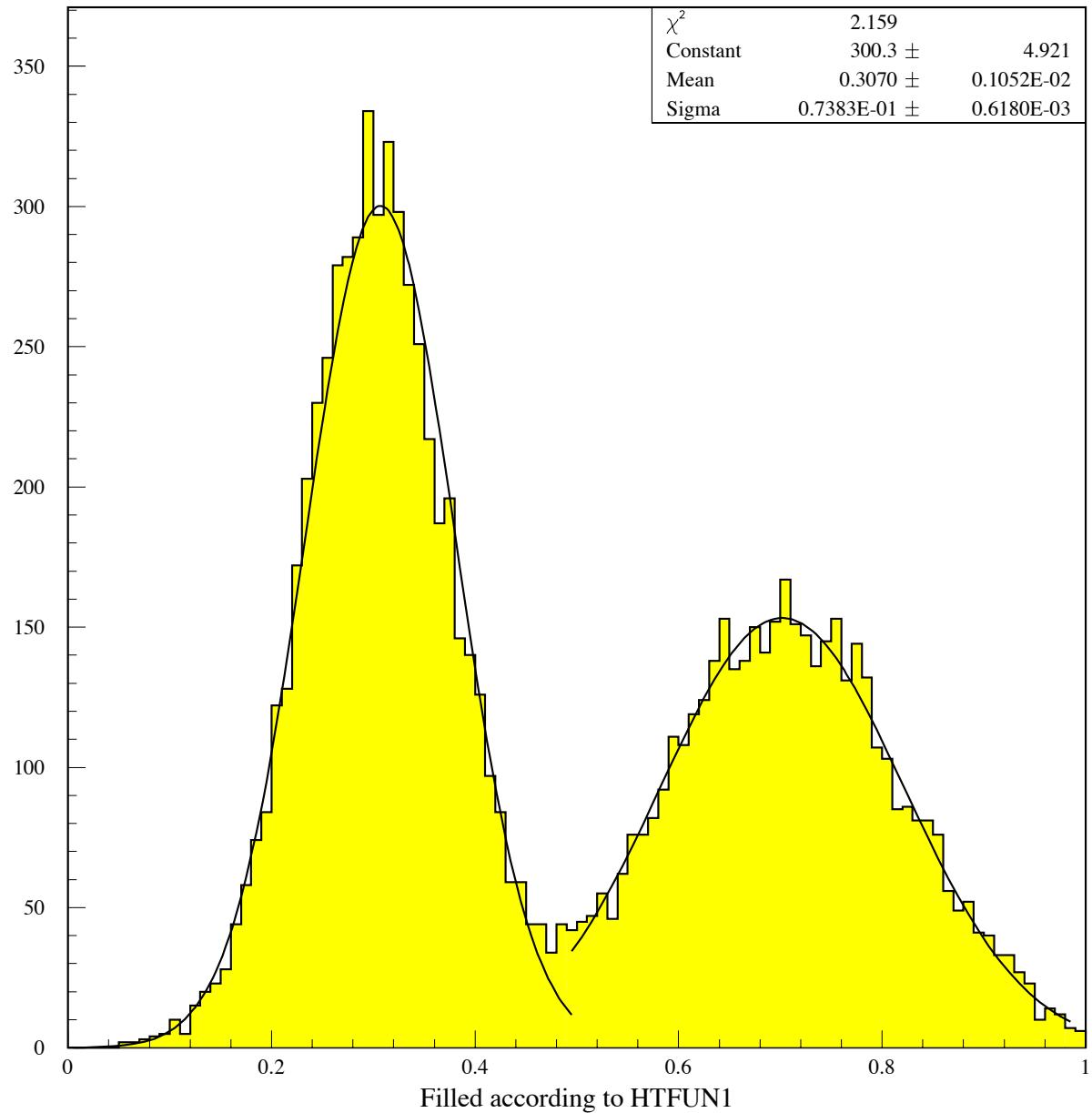


Figure 7.13: Example of a fit using sub-ranges bins

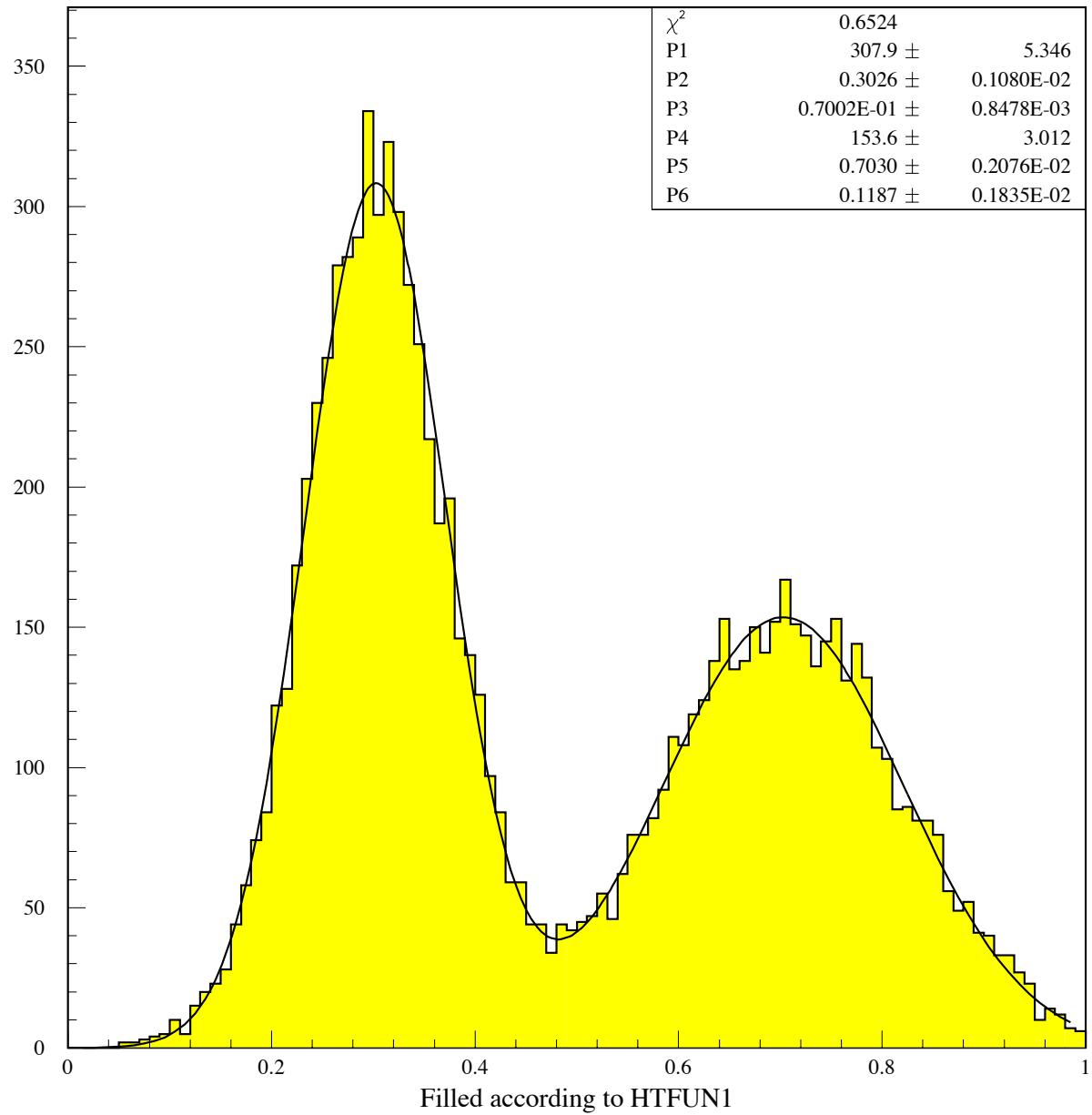


Figure 7.14: Example of a fit using a global double gaussian fit

7.7 Doing more with Minuit

When the HISTO/FIT or VECTOR/FIT command is invoked, PAW/HBOOK will set a default environment for Minuit. Control may be given to Minuit if the option “M” is specified in the command. In this case, the user may enter Minuit control statements.

Overview of available MINUIT commands

CLEAR

Resets all parameter names and values to undefined. Must normally be followed by a PARAMETER command or equivalent, in order to define parameter values.

CONtour par1 par2 [devs] [ngrid]

Instructs MINUIT to trace contour lines of the user function with respect to the two parameters whose external numbers are **par1** and **par2**. Other variable parameters of the function, if any, will have their values fixed at the current values during the contour tracing. The optional parameter [**devs**] (default value 2.) gives the number of standard deviations in each parameter which should lie entirely within the plotting area. Optional parameter [**ngrid**] (default value 25 unless page size is too small) determines the resolution of the plot, i.e. the number of rows and columns of the grid at which the function will be evaluated.

EXIT

End of Interactive MINUIT. Control is returned to PAW.

FIX parno

Causes parameter **parno** to be removed from the list of variable parameters, and its value will remain constant (at the current value) during subsequent minimizations, etc., until another command changes its value or its status.

HELP [SET] [SHOW]

Causes MINUIT to list the available commands. The list of SET and SHOW commands must be requested separately.

HESse [maxcalls]

Instructs MINUIT to calculate, by finite differences, the Hessian or error matrix. That is, it calculates the full matrix of second derivatives of the function with respect to the currently variable parameters, and inverts it, printing out the resulting error matrix. The optional argument [**maxcalls**] specifies the (approximate) maximum number of function calls after which the calculation will be stopped.

IMProve [maxcalls]

If a previous minimization has converged, and the current values of the parameters therefore correspond to a local minimum of the function, this command requests a search for additional distinct local minima. The optional argument [**maxcalls**] specifies the (approximate) maximum number of function calls after which the calculation will be stopped.

MIGrad [maxcalls] [tolerance]

Causes minimization of the function by the method of Migrad, the most efficient and complete single method, recommended for general functions (see also MINImize). The minimization produces as a by-product the error matrix of the parameters, which is usually reliable unless warning messages are produced. The optional argument [**maxcalls**] specifies the (approximate) maximum number of function calls after which the calculation will be stopped even if it has not yet converged. The optional argument [**tolerance**] specifies required tolerance on the function value at the minimum. The default tolerance is 0.1. Minimization will stop when the estimated vertical distance to the minimum (EDM) is less than $0.001 * [\text{tolerance}] * \text{UP}$ (see SET ERR).

MINImize [maxcalls] [tolerance]

Causes minimization of the function by the method of Migrad, as does the MIGrad command, but switches to the SIMplex method if Migrad fails to converge. Arguments are as for MIGrad.

MINOs [maxcalls] [parno] [parno] ...

Causes a Minos error analysis to be performed on the parameters whose numbers [**parno**] are specified. If none are specified, Minos errors are calculated for all variable parameters. Minos errors may be expensive to calculate, but are very reliable since they take account of non-linearities in the problem as well as parameter correlations, and are in general asymmetric. The optional argument [**maxcalls**] specifies the (approximate) maximum number of function calls *per parameter requested*, after which the calculation will be stopped for that parameter.

RELease parno

If **parno** is the number of a previously variable parameter which has been fixed by a command: **FIX parno**, then that parameter will return to variable status. Otherwise a warning message is printed and the command is ignored. Note that this command operates only on parameters which were at one time variable and have been FIXed. It cannot make constant parameters variable; that must be done by redefining the parameter with a PARAMETER command.

REStore [code]

If no [**code**] is specified, this command restores all previously FIXed parameters to variable status. If [**code**]=1, then only the last parameter FIXed is restored to variable status.

SCAn [parno] [numpts] [from] [to]

Scans the value of the user function by varying parameter number [**parno**], leaving all other parameters fixed at the current value. If [**parno**] is not specified, all variable parameters are scanned in sequence. The number of points [**numpts**] in the scan is 40 by default, and cannot exceed 100. The range of the scan is by default 2 standard deviations on each side of the current best value, but can be specified as from [**from**] to [**to**]. After each scan, if a new minimum is found, the best parameter values are retained as start values for future scans or minimizations. The curve resulting from each scan is plotted on the output unit in order to show the approximate behaviour of the function. This command is not intended for minimization, but is sometimes useful for debugging the user function or finding a reasonable starting point.

SEEk [maxcalls] [devs]

Causes a Monte Carlo minimization of the function, by choosing random values of the variable parameters, chosen uniformly over a hypercube centered at the current best value. The region size is by default 3 standard deviations on each side, but can be changed by specifying the value of [**devs**].

SET ERRordef up

Sets the value of **up** (default value= 1.), defining parameter errors. MINUIT defines parameter errors as the change in parameter value required to change the function value by **up**. Normally, for chisquared fits **up=1**, and for negative log likelihood, **up=0.5**.

SET LIMits [parno] [lolim] [uplim]

Allows the user to change the limits on one or all parameters. If no arguments are specified, all limits are removed from all parameters. If [**parno**] alone is specified, limits are removed from parameter [**parno**]. If all arguments are specified, then parameter [**parno**] will be bounded between [**lolim**] and [**uplim**]. Limits can be specified in either order, MINUIT will take the smaller as [**lolim**] and the larger as [**uplim**]. However, if [**lolim**] is equal to [**uplim**], an error condition results.

SET PARameter parno value

Sets the value of parameter **parno** to **value**. The parameter in question may be variable, fixed, or constant, but must be defined.

SET PRIntout level

Sets the print level, determining how much output MINUIT will produce. The allowed values and their meanings are displayed after a **SHOw PRInt** command. Possible values for **level** are:

- 1 No output except from SHOW commands
- 0 Minimum output (no starting values or intermediate results)
- 1 Default value, normal output
- 2 Additional output giving intermediate results.
- 3 Maximum output, showing progress of minimizations.

SET STRategy level

Sets the strategy to be used in calculating first and second derivatives and in certain minimization methods. In general, low values of **level** mean fewer function calls and high values mean more reliable minimization. Currently allowed values are 0, 1 (default), and 2.

SHOw XXXX

All **SET XXXX** commands have a corresponding **SHOw XXXX** command. In addition, the SHOw commands listed starting here have no corresponding SET command for obvious reasons. The full list of SHOw commands is printed in response to the command **HELP SHOw**.

SHOW CORrelations

Calculates and prints the parameter correlations from the error matrix.

SHOW COVariance

Prints the (external) covariance (error) matrix.

SIMplex [maxcalls] [tolerance]

Performs a function minimization using the simplex method of Nelder and Mead. Minimization terminates either when the function has been called (approximately) [**maxcalls**] times, or when the estimated vertical distance to minimum (EDM) is less than [**tolerance**]. The default value of [**tolerance**] is 0.1*UP (see SET ERR).

Chapter 8: Graphics (HIGZ and HPLOT)

8.1 HPLOT, HIGZ and local graphics package

Graphics input/output in PAW is handled by the two packages HPLOT (Histograms PLOTting) and HIGZ (High level Interface to Graphics and Zebra). HIGZ is the basic graphics system of PAW interfacing the local graphics package while HPLOT, sitting on top of HIGZ, is used for plotting HBOOK objects (Histograms, Ntuples, etc.). The figure below shows the hierarchy between HPLOT, HIGZ and the local basic graphics package (GKS, DI3000, X Windows, etc.).

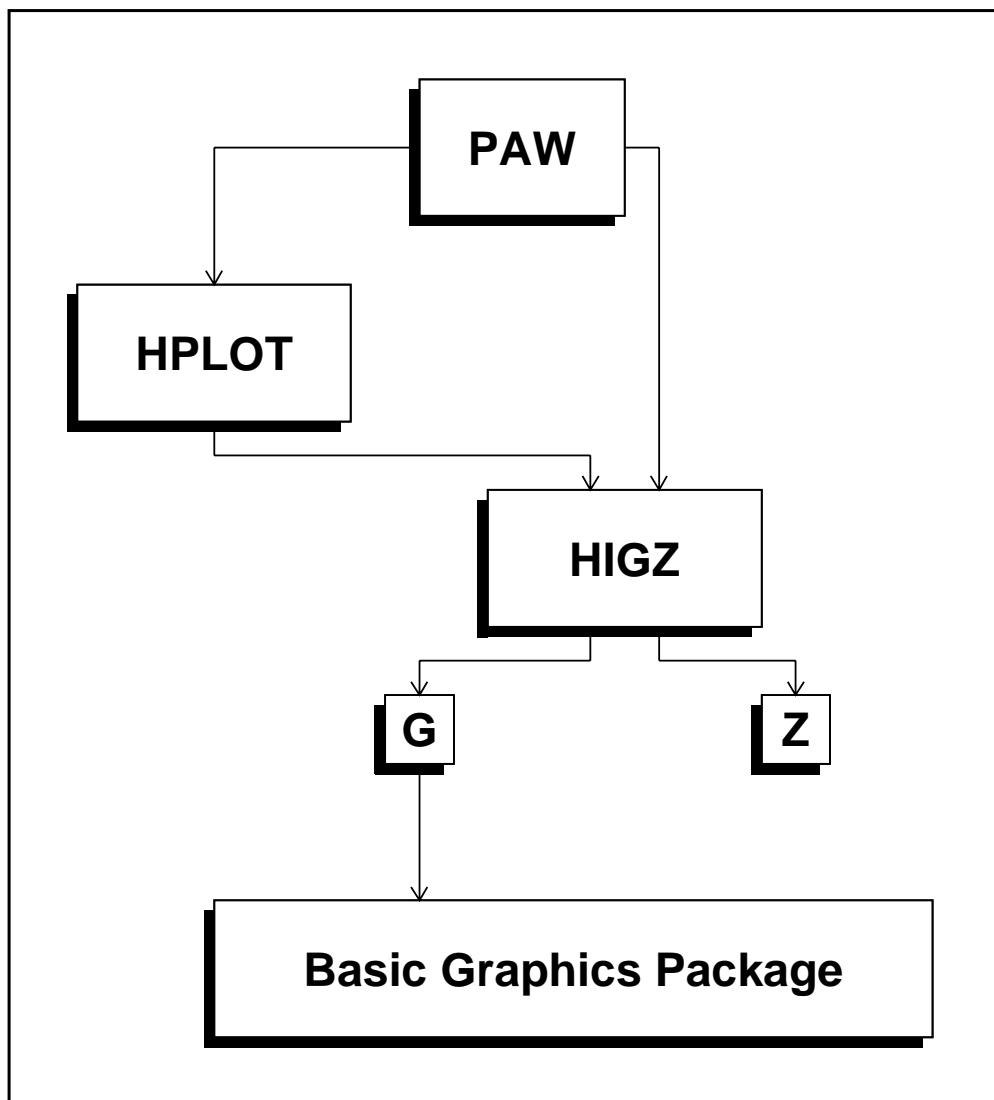


Figure 8.1: HPLOT and HIGZ in PAW

Graphics could be produced in PAW either directly by HIGZ commands or by HPLOT commands. In both cases, all the graphics is under the control of HIGZ. Two distinct modes are available in HIGZ: one is purely graphics (the G mode) interfacing the local graphics package, and the second (the Z mode) allows the management of the HIGZ structures (pictures). As an example, the simple PAW command HISTOGRAM/PL0T is handled at the different levels as follows:

PAW Level	HISTOGRAM/PLOT ID
HPLOT Level	Takes care of ZONE, SET, OPTION, etc.
HIGZ Level	Windows and Viewport, Axis, Boxes, Histogram, Text and Attributes
Basic graphics	Line, Text, Attributes, etc.

8.2 The metafiles

Metafiles are text files used as device independent sources of graphics output for printers of different type. PAW is able to produce two types of metafiles.

The first one is the local graphics package metafile (for example a GKS metafile). This file is produced by the local graphics package and it usually needs a special interpreter to be sent to the printers. For example, at CERN, the GKS metafile (workstation type 4) must be printed with GRPLOT

The second type of metafile is directly produced by HIGZ and is independent from the basic graphics package used. This type of metafile is a PostScript metafile (workstation type -111: portrait; -112: landscape) and could be sent directly to a PostScript printer (at CERN GKS metafiles type 12201 and 12202 could be also used to produce PostScript metafiles).

The command GRAPHICS/METAFILE LUN METAFL is designed to produce metafiles. LUN is the logical unit number of an open FORTRAN file and METAFL the metafile type. For example, the following four commands will produce a HIGZ/PostScript metafile with the name "PAW.PS" containing the graphics representation of histogram number 10:

```
PAW > FORTRAN/FILE 66 PAW.PS
PAW > GRAPHICS/META 66 -111
PAW > HISTO/PLOT 10
PAW > FORTRAN/CLOSE 66
```

8.3 The HIGZ pictures

The HIGZ pictures have three main goals:

- HIGZ graphics primitives and attributes can be stored in a ZEBRA structure in memory in order to display them later.
- They can be stored on direct access files (in a very compact way), in order to build a picture data base.
- They can be modified with the graphics editor.

8.3.1 Pictures in memory

The general command to manage pictures in memory is: PICTURE/IZPICT. This command has two parameters:

PNAME Picture name:

- CH Character string specifying picture name (must begin with a letter)
- N Picture number as displayed by PICT/LIST.
- *
- , All pictures in memory.
- , A blank indicates the current picture.

CHOPT Option value:

- AL Give a full listing of the pictures in memory.
- C Picture PNAME becomes the current picture.
- D Display the picture PNAME.
- F First picture in memory becomes the current picture.
- L List pictures in memory.
- M Make a new picture in memory with the name PNAME.
- N Next picture in memory becomes the current picture.
- P Print the contents of the picture PNAME.
- S Scratch picture PNAME from memory.

In addition, simpler and more mnemonic commands are available:

```
PAW > PICT/CREATE PNAME           | Create a picture in memory
PAW > PICT/LIST                  | List pictures in memory
1: PNAME <-- Current Picture
```

The last created picture in memory is called the **current** picture. All graphics primitives (line, text, histogram, etc.) produced by PAW commands will be stored in this picture if it is **active**, i.e. if mode Z is on.

```
PAW > SWITCH Z                  | Switch Z mode on
PAW > PICT/LIST
1: PNAME <-- Current Picture (Active)
```

Note that the command PICTURE/CREATE will switch automatically Z mode on.

```
PAW > PICT/PLOT PNAME
```

will display picture PNAME. If picture PNAME is not in memory and if the current working directory (as given by CDIR) is a picture file, PAW will try to take this picture from the file before displaying it.

HIGZ pictures can be created automatically by HPLOT via the command:

```
PAW > OPTION ZFL
```

If this command has been typed, each new plot produced by HPLOT will result in a HIGZ picture created in memory. The following example shows how for each HIST/PL0T ID command a new HIGZ picture is created with an automatic naming:

```
PAW > HIST/PL0T 10
PAW > HIST/PL0T 110
PAW > HIST/PL0T 20
PAW > PICT/LIST
  1: PICT1
  2: PICT2
  3: PICT3 <- Current Picture (Active)
```

A similar command is given by:

```
PAW > OPTION ZFL1
```

which works exactly like OPTION ZFL except that only the last created picture is kept in memory. For example, if we had typed OPTION ZFL1 instead of OPTION ZFL in the example above, the result would be:

```
PAW > PICT/LIST
  1: PICT3 <- Current Picture (Active)
```

The following example is a useful macro showing how to use the HIGZ pictures (via OPTION ZFL1) and the metafiles in order to produce a hard copy of the graphics screen:

Macro showing how to print current picture in PostScript

```
MACRO POST
FORTRAN/FILE 66 PAW.PS
META -66 -111
PICT/PL0T ''
CLOSE 66
SHELL PRINT PAW.PS
RETURN
```

Typing EXEC POST, the current HPLOT picture on the screen will be sent to the printer using the SHELL command which issues a system-dependent “print” command to the local operating system (e.g. lp or lpr on Unix).

Other available commands working on pictures in memory are:

```
PAW > PICT/RENAME PNAME PNAME2
PAW > PICT/COPY PNAME PNAME2
PAW > PICT/DELETE PNAME
```

- PNAME can be the complete name, the picture number in memory or ''.
- PNAME2 is the complete picture name.

8.3.2 Pictures on direct access files

HIGZ pictures are stored on direct-access files and hence access times to pictures are fast. Moreover, due to the fact that HIGZ uses high level primitives to describe the picture’s structural tree, a storage compaction factor as compared to the equivalent GKS metafiles of between 10 and 100 is routinely obtained.

As HIGZ is interfaced to various basic graphics packages, a picture file can be created on one system (e.g. DECGKS) and transported to another machine to be interpreted with a different graphics package (e.g GKSGRAL or DI3000).

All available commands to handle pictures with ZEBRA files are shown below. Note that in the example the picture names could be “*”, “ ” or a number.

Handling pictures with ZEBRA

```
PAW > * Open an existing picture file PICT.DAT on LUN 4 in Update mode
PAW > PICT/FILE 4 PICT.DAT ! U | Open the existing file PICT.DAT
PAW > LDIR | List the content of the file PICT.DAT
```

```
***** Directory ==> //LUN4 <==
```

```
Created 890512/1110 Modified 890622/1732
```

```
====> List of objects
```

PICTURE	NAME	CYCLE
UNIX		1
ZEBRA		1
CERN		1
MARKER		1

```
PAW > IZIN CERN | Put picture "CERN" in memory
```

```
PAW > PICT/LIST | List pictures in memory
1: CERN
```

```
PAW > IZOUT CERN | Store picture "CERN" in PICT.DAT
```

```
PAW > LDIR | List the content PICT.DAT
```

```
***** Directory ==> //LUN4 <==
```

```
Created 890512/1110 Modified 890622/1732
```

```
====> List of objects
```

PICTURE	NAME	CYCLE
UNIX		1
ZEBRA		1
CERN		1
		2
MARKER		1

```
PAW > PURGE | Purge the file PICTURES
```

```
PAW > SCRATCH ZEBRA | Delete the picture ZEBRA from PICT.DAT
PAW > LDIR | List the content of PICT.DAT
```

```
***** Directory ==> //LUN4 <==
```

```
Created 890512/1110 Modified 890622/1732
```

```
====> List of objects
```

PICTURE	NAME	CYCLE
UNIX		1
CERN		2
MARKER		1

8.4 HIGZ pictures generated in a HPLOT program

HIGZ pictures can be generated in a batch HPLOT program and later visualized in an interactive session with PAW. The HIGZ picture file, like any HBOOK file, can be exchanged between computers using the ZFTP facility, as described in on page 190. As the size of the picture data base (see page 156), and hence the associated disk storage requirements, is much smaller than the size of the metafile generated by the underlying graphics package, transfer times are drastically reduced. The example below show how to interactively visualize (with PAW) HIGZ pictures produced by HPLOT. In the same way we can visualize and edit pictures generated by any HIGZ based application (GEANT, event scanning programs, etc.)

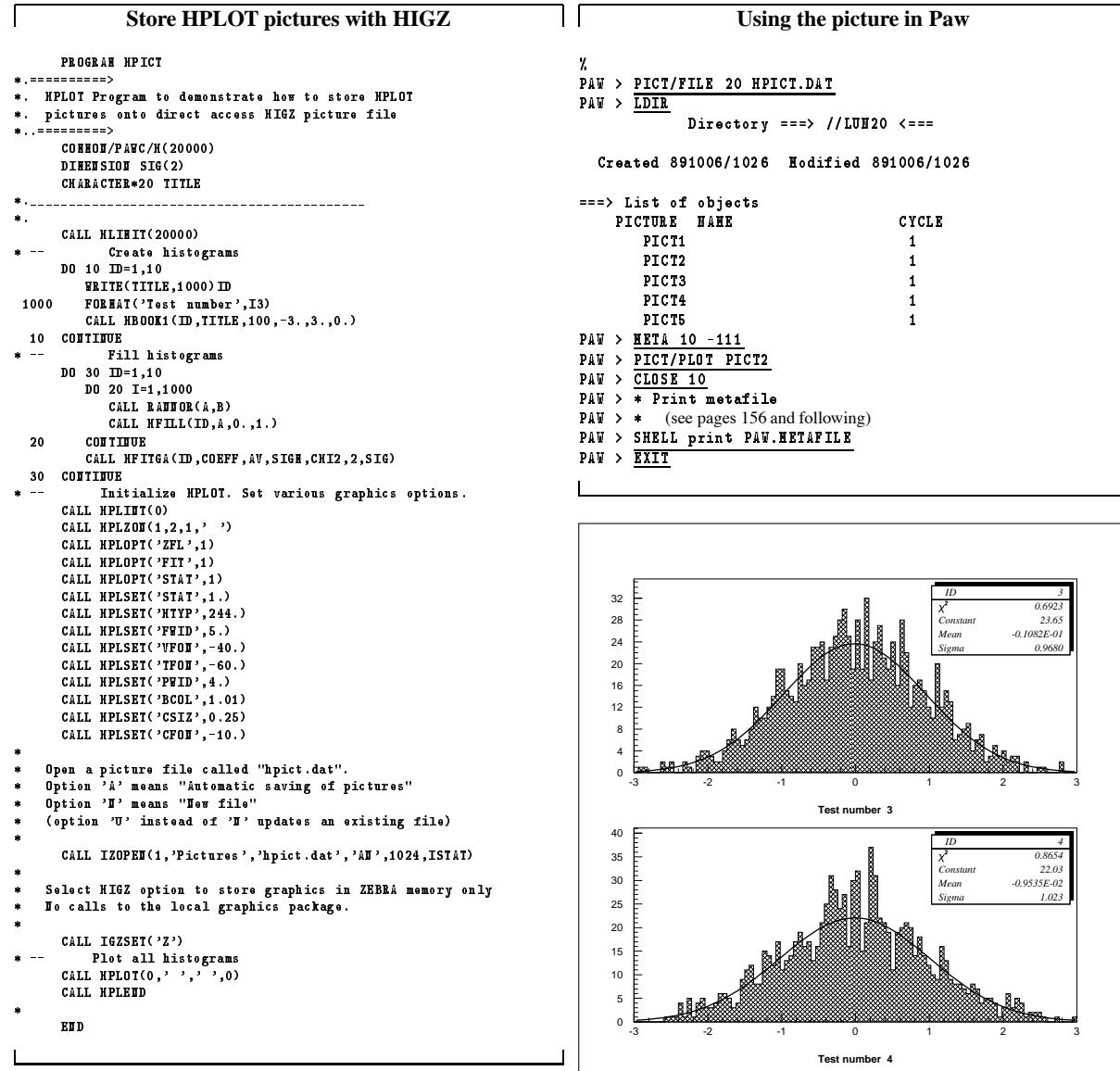


Figure 8.2: Visualising a HIGZ picture produced in a batch HPLOT program

8.5 Setting attributes

Attributes are parameters like: colour, character font, etc. which could be changed interactively in PAW via the commands PICTURE/IGSET, GRAPHICS/SET and GRAPHICS/OPTION. Each attribute is linked to one or more objects (lines, histogram, etc.). The aim of this section is to give a complete description of the attributes available in PAW and to clarify the differences between IGSET, which changes attributes at the HIGZ level, and SET and OPTION, which act at the HPLOT level.

IGSET (CHOPT, VAL)

Routine used to set the value of attributes related to primitives and/or macroprimitives. The first parameter is the mnemonic name of the parameter, the second is the value to be assigned. Note that all the basic primitives attributes can also be set with this routine.

CHOPT Character variable specifying the name of the parameter to be set (type CHARACTER*4). This is an **UPPERCASE** character string.

VAL **Floating point** value of the parameter (must be specified as a **REAL** number).

A value of 0.0 indicates that the parameter value must be reset to its default value.

Examples of IGSET commands

```
PAW > IGSET MTYP 20      | Change marker type to 20.  
                           | This new marker is used by all subsequent  
                           | commands using the current marker type.  
  
PAW > IGSET LWID        | Set the line width to its default value.  
  
PAW > IGSET             | Display actual and default values of all HIGZ attributes  
PAW > IGSET *          | Set ALL HIGZ attributes to their default values
```

OPTION (CHOPT)

The OPTION command has one optional parameter:

CHOPT Option name (four characters). Special values are:

- '*' Set all HPLOT options to their default values
- ' ' Display actual and default values of all HPLOT options

SET (CHOPT, VAL)

Sets an HPLOT parameter; see table 8.3 for details.

CHOPT Character variable of length 4 identifying the parameter to be redefined (must be given in uppercase). Special values are:

- '*' All parameters are set to their default values.
- 'SHOW' A list of all parameters and their values is printed.

VAR New value for the parameter specified. Special values are:

- 0. The corresponding parameters is set to its default value.

NAME	default	Explanation
'AURZ'	0.	If 1., the last current picture is automatically saved on disk when a new picture is created.
'AWLN'	0.0	Axis wire length. Default is length=0 (no grid)
'BARO'	0.25	Offset of the left edge of the bar with respect to the left margin of the bin for a bar chart (expressed as a fraction of the bin width).
'BARW'	0.50	Width of the bar in a bar chart (expressed as a fraction of the bin width).
'BASL'	0.01	Basic segment length in NDC space (0-1) by (0-1) for dashed lines
'BORD'	0.	Border flag. If = 1., a border is drawn in boxes, pie charts, . . .
'CHHE'	0.01	CHaracter HEight.
'CSHI'	0.02	Distance between each shifted drawing of a character (in percentage of character height) for characters drawn by TEXT
'FACI'	1.	Fill Area Colour Index.
'FAIS'	0.	Fill Area Interior Style (0.,1.,2.,3.).
'FASI'	1.	Fill Area Style Index.
'LAOF'	0.013	LAbels OFFset.
'LASI'	0.018	LAbels SIze (in World coordinates).
'LTYP'	1.	Line TYPe.
'LWID'	1.00	Line WIDth.
'MSCF'	1.00	Marker SCale Factor.
'MTYP'	1.	Marker TYPe.
'PASS'	1.	Text width (given by number of PASSes) of characters drawn by TEXT. The width is simulated by shifting the “pen” slightly at each pass.
'PICT'	1.	Starting number for automatic pictures naming.
'PLCI'	1.	PolyLine Colour Index.
'PMCI'	1.	PolyMarker Colour Index.
'TANG'	0.00	Text ANGLE (for calculating Character up vector).
'TMSI'	0.019	Tick Marks SIze (in world coordinates)
'TXAL'	0.	10*(horizontal alignment)+(vertical alignment).
'TXCI'	1.	TeXt Colour Index.
'TXFP'	10.	10*(TeXt Font) + (TeXt Precision). (0: hard, 1: string, 2: soft)
'*'		All attributes are set to their default values.
'SHOW'		The current and default values of the parameters controlled by IGSET are displayed.

Table 8.1: Parameters and default values for IGSET

CHOPT	alternative	Explanation
A4	A0/6	Page format for the plotter (A0,A1,A2,A3,A4,A5,A6)
BOX	NBOX	A box is (BOX) or is not (NBOX) drawn around picture
DVXR	DVXI	Integer (DVXI) or Real (DVXR) divisions for X axis
DVYR	DVYI	Integer (DVYI) or Real (DVYR) divisions for Y axis
HTIT	UTIT	HBOOK TITle (HTIT) or User TITle (UTIT) is printed
LINX	LOGX	LINEar or LOGarithmic scale in X
LINY	LOGY	LINEar or LOGarithmic scale in Y
LINZ	LOGZ	LINEar or LOGarithmic scale in Z
NAST	AST	Functions are drawn with (AST) or without (NAST) asterisks
NBAR	BAR	BAR charts for histogram
NCHA	CHA	Scatter plots are drawn with dots (NCHA) or one char/bin (CHA)
NDAT	DATE	DATE is printed (DATE) or not (NDAT) on each plot
NEAH	EAH	Error bars And Histogram are plotted (if both are present)
NFIL	FILE	FILE name is printed (FILE) or not (NFIL) on each plot
NFIT	NFIT	FIT parameters are printed (FIT) or not (NFIT) on each plot
NGRI	GRID	GRID or not grid (NGRI) on X and Y axis
NOPG	P	Page number is (P) or is not (NOPG) printed
NPTO	PTO	PTO (Please Turn Over)
NSQR	SQR	Size is set to the largest square (SQR)
NSTA	STA	STAtistics are printed (STA) or not (NSTA) on each plot
NTIC	TIC	Cross-wires are drawn (TIC) or not (NTIC) on each plot
NZFL	ZFL	Picture is (ZFL) or is not (NZFL) put in Z data base
SOFT	HARD	SOFTware or HARDware characters are used
TAB	NTAB	Table printed as TABles (TAB) or scatter plots (NTAB)
VERT	HORI	VERTical or HORIZONTAL orientation of paper

Table 8.2: Parameters and default values for OPTION

CHOPT	Default	Explanation (units are cm unless otherwise specified)
ASIZ	0.28	axis label size
BARO	0.25	bar offset for “bar charts”
BARW	0.5	bar width for “bar charts”
BCOL	1	zone fill area colour index
BTYP	0	zone fill area style index
BWID	1.	line width of the histogram rounding box
CSHI	0.03	character shift between two passes
CSIZ	0.28	comment size
DASH	0.15	length of basic dashed segment for dashed lines
DMOD	1	line style for histogram contour (see HPLOT)
GFON	2	global title font ($10*font+precision$)
GSIZ	0.28	global title size
HCOL	1	histogram fill area colour index ($10*font+precision$)
HMAX	0.90	histogram maximum for scale
HTYP	0	histogram fill area style index
KSIZ	0.28	Hershey character size
LFON	2	axis labels font ($10*font+precision$)
NDVX	510	number of divisions for X axis
NDVY	510	number of divisions for Y axis
PASS	1.	number of passes for software characters
PCOL	1	picture fill area colour index
PSIZ	0.28	page number size
PTYP	0	picture fill area style index
SSIZ	0.28	asterisk size (for functions)
TFON	2	general text (comments) font ($10*font+precision$)
TSIZ	0.00	histogram title size
VFON	2	axis values font ($10*font+precision$)
VSIZ	0.28	axis values size
XLAB	1.40	distance Y axis to labels
XMGL	2.00	X margin left
XMGR	2.00	X margin right
XSIZ	20.00	length of picture along X
XTIC	0.30	X axis tick mark length
XVAL	0.40	distance Y axis to axis values
XWIN	2.00	X space between zones
YGTI	1.50	Y position of global title
YHTI	1.20	Y position of histogram title
YLAB	0.80	distance X axis to labels
YMGL	2.00	Y margin low
YMGU	2.00	Y margin up
YNPG	0.60	Y position for number of page
YSIZ	20.00	length of picture along Y
YTIC	0.30	Y axis tick mark length
YVAL	0.20	distance X axis to axis values
YWIN	2.00	Y space between zones
2SIZ	0.28	scatter plot and table character. size

Table 8.3: Parameters and default values in SET

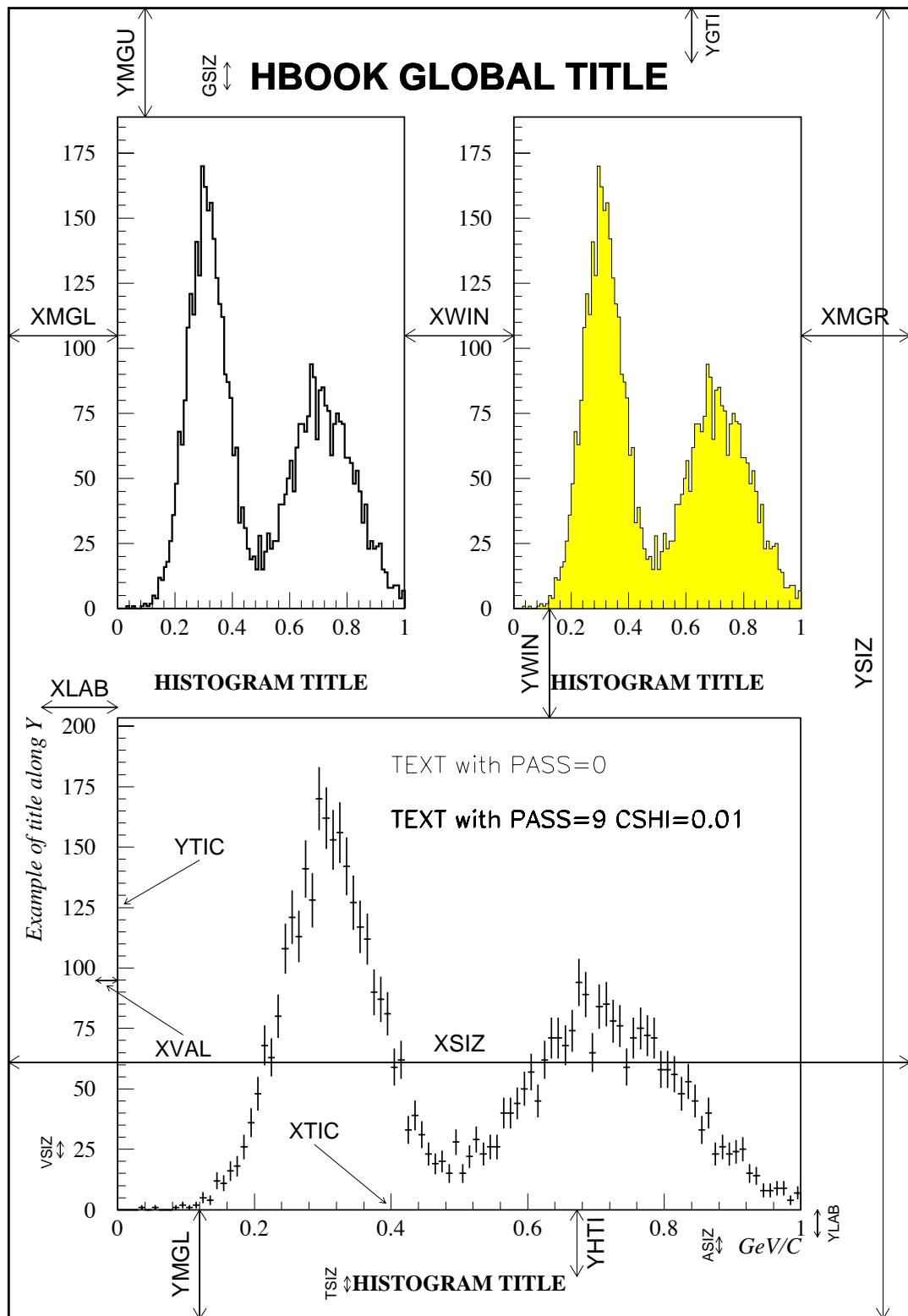


Figure 8.3: A graphical view of the SET parameters

8.6 More on labels

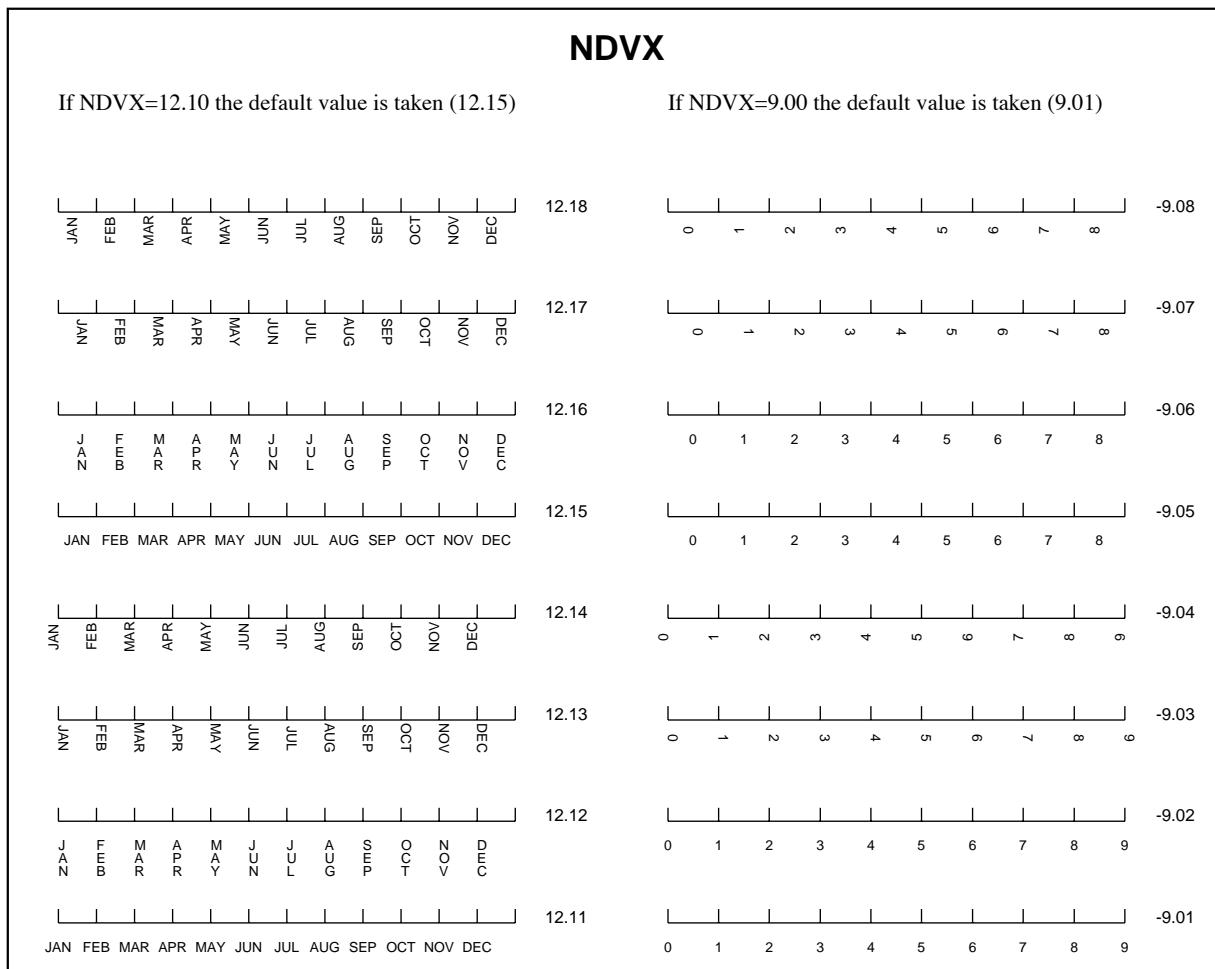


Figure 8.4: Example of labelling for horizontal axes

By default, labels used by **AXIS** and **PIE** are numeric labels. The command **GRAPHICS/PRIMITIVES/LABELS** (or **LABELS** for short), allows the user to define up to nine alphanumeric set of labels (numbered from 1 to 9). These labels can then be used in subsequent commands using **PIE** or **AXIS** primitives of HIGZ.

The **LABELS** command has three parameters:

LABNUM An integer between 1 and 9. It identifies the labels set.

NLABS The number of items to be placed on the labels (up to 50).

CHLABS NLABS character strings specifying the label items.

The label sets thus defined can be used for axes on all plots produced by PAW (HPLOT histograms, graphs, vectors drawing, etc.) via the SET NDVX (NDVY) command. These commands have the following structure:

Example of NXDV specification

```
SET NDVX i           e.g. SET NDVX 512
or
SET NDVX i.jk       e.g. SET NDVX 10.25
```

In the first case the number *i* contains 100 times the number of secondary divisions plus the number of primary divisions. (e.g. 512 means 12 primary and 5 secondary division. By adding 10000 times N3 to *i* a third level of divisions is available.

In the second case the number in front of the dot (*i*) indicates the total number of divisions, the first digit following the dot (*j*) the label identifier (LABNUM) (if this number is equal to 0 numeric labels are drawn). The second digit after the (*k*) dot indicates the position where the labels have to be drawn (i.e. the **text justification** parameter, in this case 5, indicating horizontally written text centered on the interval). Study figures 8.4 and 8.5 for details. These two figures show that the labels can be centered on the tick marks (1 to 4) or on the divisions (5 to 8). If the labels are centered on the tick marks, note that the number of items in the command LABELS must be equal to the number of tick marks (which is equal to the number of divisions **plus one**), otherwise the last alphanumeric label on the axis will be undefined.

By default, the number of primary divisions given by SET NDVX *n* or SET NDVY *n* is optimized to have a reasonable labelling. If the number of divisions has to be exactly equal to the number given by SET NDVX *n* or SET NDVY *n*, a negative value must be used i.e.:

Forcing an exact number of divisions

```
SET NDVX -i          e.g. SET NDVX -512
or
SET NDVX -i.jk      e.g. SET NDVX -10.25
```

For example to label each subsequent X-axis with the names of the months of the year centered in the middle of each bin one can use:

Example of alphanumeric label on an axis

```
PAW > LABEL 1 12 JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC
PAW > SET NDVX -12.15
```

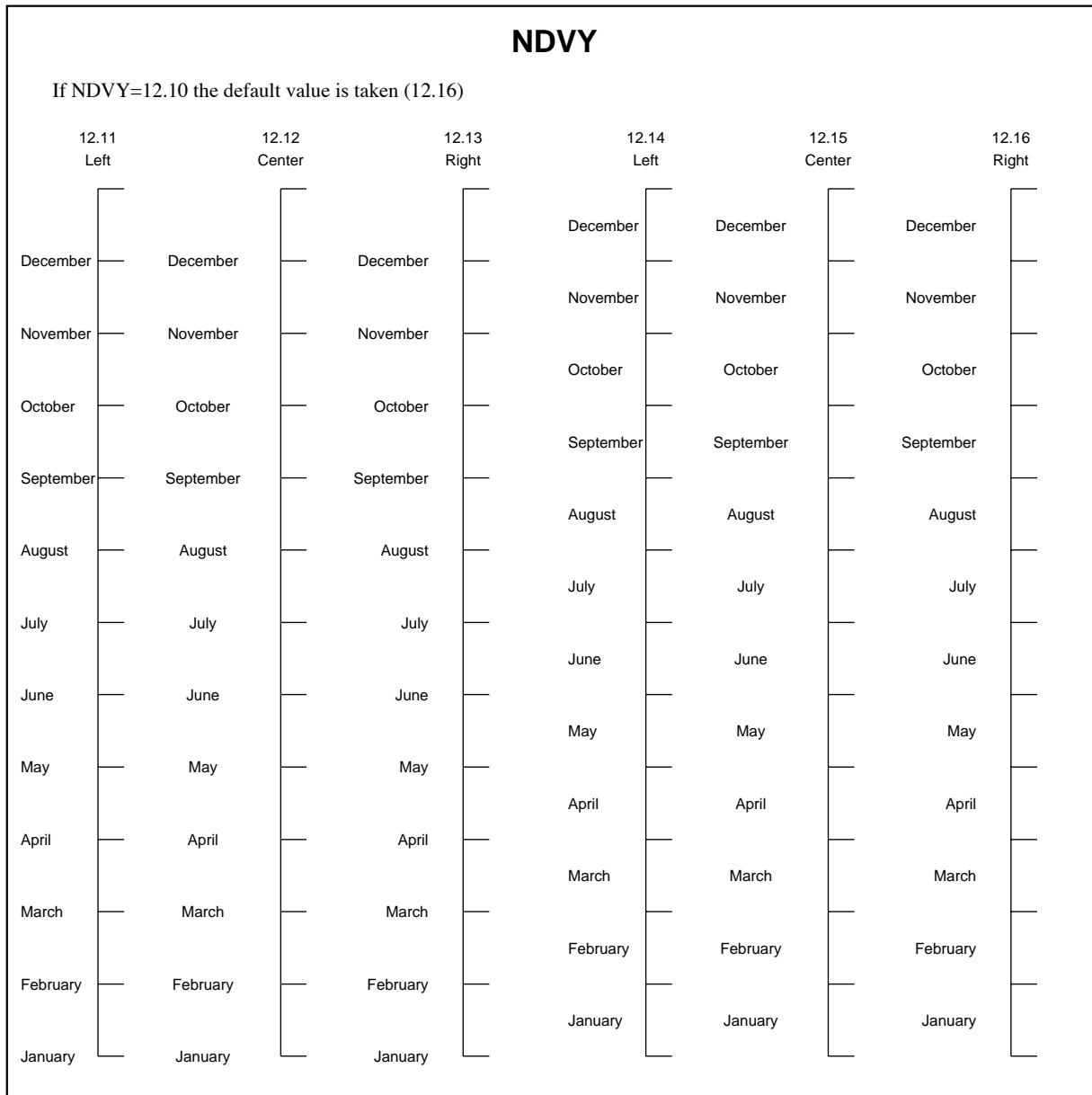


Figure 8.5: Example of labelling for vertical axes

8.7 Colour, line width, and fill area in HPLOT

The aspect of HPLOT pictures can be modified via the `xWID`, `xTYP` and `xCOL` attributes, where `x` can be `H`, `B`, `P`, or `F`, defined as follows:

- `B` zone Box
- `F` Function
- `H` Histogram
- `P` Page

The values given to the parameters `PTYP`, `BTYP`, `HTYP`, and `FTYP` are the HIGZ/GKS fill area interior styles. GKS styles are installation-dependent and even device-dependent. If the same result is desired on all devices, numbers greater than 100 (HIGZ styles: 8.7) should be used. Figure 8.6 shows how to use the `xTYP` parameter.

The parameters `PCOL`, `BCOL`, `HCOL` and `FCOL` are equivalent to `PTYP`, `BTYP`, `HTYP`, and `FTYP` respectively, but instead of changing the hatch style, they change the colour of the same areas.

If `PCOL`, `BCOL`, `HCOL` or `FCOL` are between 1 and 999, then only the contour of the corresponding area is changed. If they are between 1001 and 1999, then the surface is filled with the colour determined by the corresponding fill area colour index. and the corresponding value of the Fill Area Interior Style (for `HTYP`, `BTYP`, `PTYP` or `FTYP`) is automatically set to 1 (solid).

In addition, `BCOL` has two digits after the dot. The first one specifies the colour of the zone box shadowing and the second the colour of the statistic box shadowing.

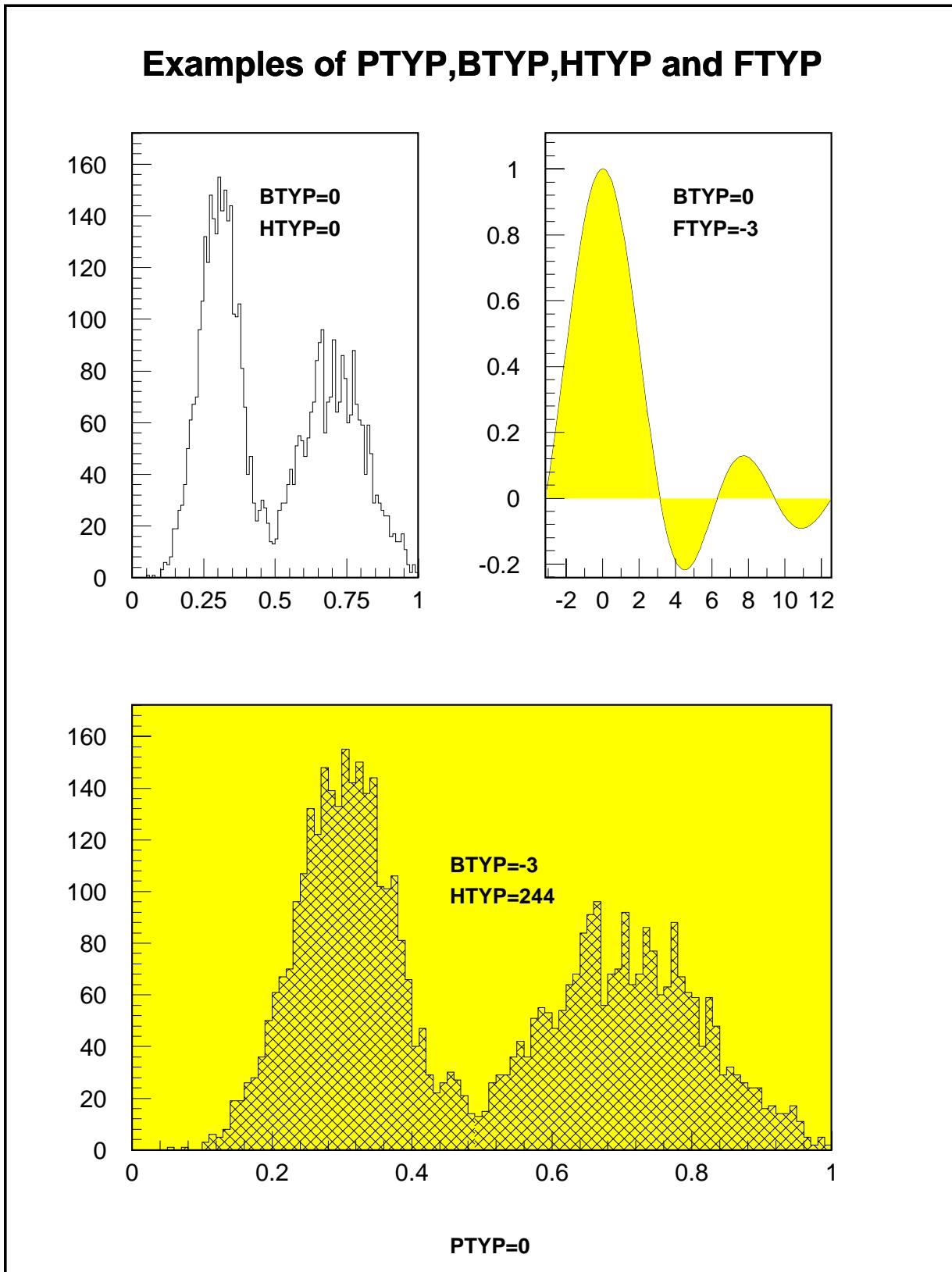


Figure 8.6: Example of fill area types in HPLOT

8.8 Information about histograms

Four options are available to plot additional informations on HPLOT pictures: DATE, FILE, STAT and FIT.

```
PAW > OPTION_DATE      | Plot date and hour on current HPLOT picture
PAW > OPTION_FILE       | Plot file name of current histogram
PAW > OPTION_STAT       | Plot statistics of current histogram
PAW > OPTION_FIT        | Plot Fit parameters of current histogram
```

For each of these OPTION commands a corresponding SET parameter is available:

```
PAW > SET_DATE i     | Default is 2
PAW > SET_FILE i      | Default is 1
```

where i defines the position of the date or file name:

- i = 1 : Top left corner of page/current histogram.
- i = 2 : Top right corner
- i = 3 : Bottom left corner
- i = 4 : Bottom right corner

For example the command:

```
PAW > SET_DATE 3
```

sets the position of the date to the bottom left corner of the HPLOT pictures.

```
PAW > SET_STAT i      | Default is 1111
```

where i corresponds to binary status bits OURMEIA as follows:

- O=1 Draw number of overflows
- U=1 Draw number of underflows
- R=1 Draw R.M.S.
- M=1 Draw mean value
- E=1 Draw number of entries
- I=1 Draw histogram identifier
- A=1 Draw the contents of all channels

For example the command:

```
PAW > SET_STAT 10
```

sets the statistics informations to be only the number of entries.

```
PAW > SET_FIT i        | Default is 101
```

where i corresponds to binary status bits CEP as follows:

- C=1 Draw χ^2
- E=1 Draw errors
- P=1 Draw fit parameters

For example to draw only the result of the χ^2 fit one would use:

```
PAW > SET_FIT 100
```

For all these OPTIONS, the **character size** is specified with the command SET CSIZ and the character font used with SET CFON.

8.9 Additional details on some IGSET commands

Store pictures in memory

If the AURZ mode is on, after typing the command:

```
PAW > IGSET AURZ 1
```

all subsequent created pictures are stored automatically in the last picture file opened via the command PICTURE/FILE.

Example of the use of pictures in memory

```
PAW > PICT/FILE 4 PICT.DAT ! N | Open a new picture file PICT.DAT
PAW > HIST/FILE 3 HEXAM.DAT | Open an existing histogram RZ file
PAW > LDIR | List the content of HEXAM.DAT
```

```
***** Directory ===> //LUN3 <==
```

```
Created 880104/1414 Modified 880104/1414
```

```
====> List of objects
```

HBOOK-ID	CYCLE	DATE/TIME	NDATA	OFFSET	REC1	REC2
10	1	880104/1414	75	725	32	
20	1	880104/1414	1815	800	32	33
30	1	880104/1414	1066	567	34	35

```
PAW > OPT ZFL | Each new plot will result in a HIGZ picture
PAW > IGSET AURZ 1 | Each new HIGZ picture is stored in PICT.DAT
PAW > HIST/PLOT 0 | All histograms in HEXAM.DAT are plotted
PAW > CDIR //LUN4 | Set the current working directory on PICT.DAT
PAW > LDIR | List the content of PICT.DAT
```

```
***** Directory ===> //LUN4 <==
```

```
Created 890928/1024 Modified 890928/1024
```

```
====> List of objects
```

PICTURE	NAME	CYCLE
PICT1		1
PICT2		1
PICT3		1

Text font and precision

Text font and precision attributes for use by later invocations of ITX are set with TXFP as follows:

```
PAW > IGSET TXFP (10*(Text font) + (text precision))
```

The same syntax is used in all xFON option of the HPLOT command SET. Text font selects the desired character font e.g. a roman font, a sans-serif font, etc. Text precision specifies how closely the graphics package implementation must follow the current size and orientation attributes. String (0) precision is most liberal (hardware), stroke (2) precision is most strict. Character precision is in the middle (1). The value of text font is dependent upon the basic graphics package used. However, font number 0, with precision 2 is always available, independently from the basic graphics package used (see figure 8.12). Hardware characters are available by specifying

```
PAW > IGSET TXFP 10 | Choose hardware characters, i.e. font 1, precision 0
```

Text alignment

The text alignment attributes for use by future invocations of ITX are set using the TXAL parameter as follows:

```
PAW > IGSET TXAL (10*(horizontal alignment) + (vertical alignment))
```

Text alignment controls the placement of the character string with respect to the specified position in the call to ITX. The horizontal alignment parameter ITXALH must be in the range 0-3 while the vertical alignment parameter ITXALV must be in the range 0-5. The following parameter definitions are standardized by the graphics package for text alignment: the horizontal alignment specifies which end of the string (or its geometric center) is aligned with the specified point in ITX. For a given horizontal alignment, the vertical alignment controls whether the top of tall characters (or the bottom of capital letters) line up with the specified point (see table 8.4).

Alignment parameter	Description
ITXALH=0	normal (usually same as 1)
ITXALH=1	left end of string at specified point
ITXALH=2	center of string at specified point
ITXALH=3	right end of string at specified point
ITXALV=0	normal
ITXALV=1	top of tallest chars plus any built in spacing
ITXALV=2	top of tallest chars
ITXALV=3	halfway between 2 and 4
ITXALV=4	bottoms of capital letters and most other chars
ITXALV=5	bottoms of descenders (e.g. "g", "y") plus any built in spacing.

Table 8.4: Text alignment parameters

Fill area style, marker and line type

The Fill Area Interior Style, The Fill Area Style Index, the Marker TYPe and the Line TYPe are set respectively using the IGSET parameters FAIS, FASI, MTYP and LTYPE.

Example

```
PAW > IGSET FAIS 3      | Fill area are hatched
PAW > IGSET FASI -101   | with the GKSGRAL style index -101
PAW > IGSET MTYP 25     | Marker type is an empty square
PAW > IGSET LTYP 15     | Line type is dotted
```

Figure 8.8 shows the available GKSGRAL fill area styles indeces.

In addition, HIGZ provides some portable fill area styles index coded using three digits $i j k$ as follows:

i : Distance between each hatch in mm

j : Angle between 90 and 180 degrees

k : Angle between 0 and 90 degrees

These numbers are coded according to table 8.5 and examples are shown in figure 8.7.

i	Distance	j	Angle	k	Angle
		0	180°	0	0°
1	0.75mm	1	170°	1	10°
2	1.50mm	2	160°	2	20°
3	2.25mm	3	150°	3	30°
4	3.00mm	4	135°	4	45°
5	3.75mm	5	not drawn	5	not drawn
6	4.50mm	6	120°	6	60°
7	5.25mm	7	110°	7	70°
8	6.00mm	8	100°	8	80°
9	6.75mm	9	90°	9	90°

Table 8.5: Codification for the HIGZ portable fill area interior styles

Example

```
PAW > IGSET FAIS 3      | Fill area interior style is hatched
PAW > IGSET FASI 190    | Hatch type is 190
```

These commands will yield hatching with two sets of lines at 90° and 0° spaced 1 mm apart.

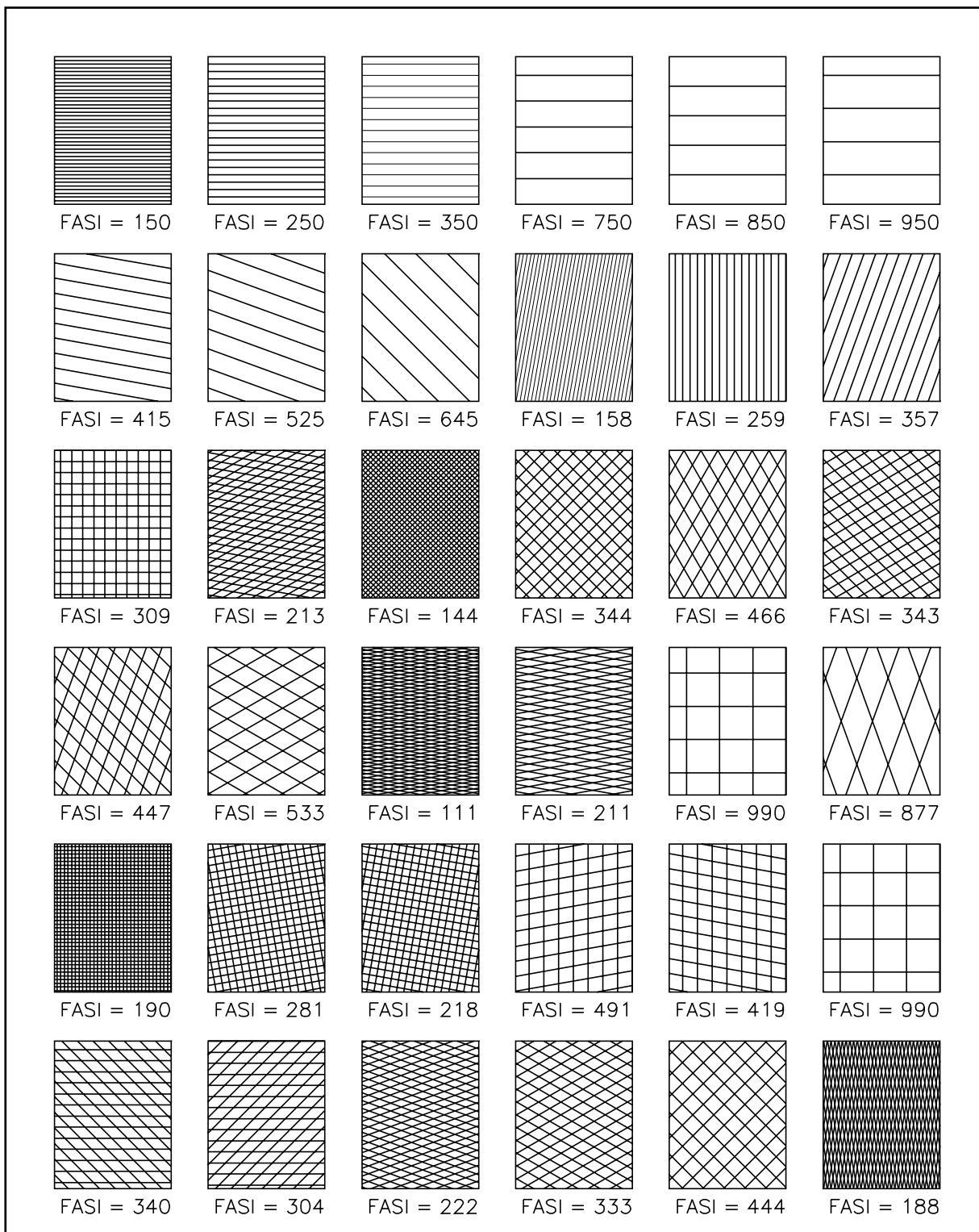


Figure 8.7: Examples of HIGZ portable hatch styles

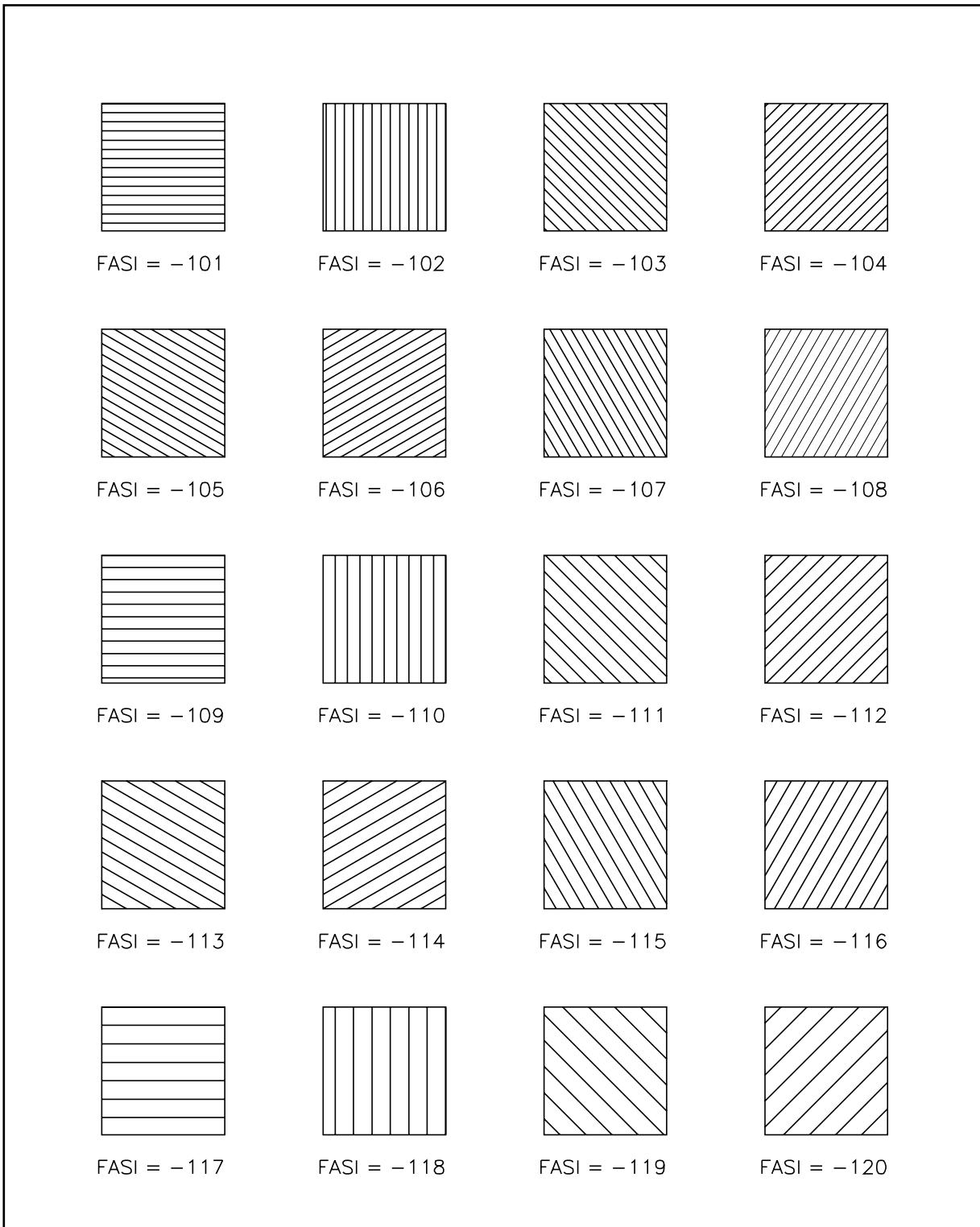


Figure 8.8: GKSGRAL Device independent hatch styles

Marker Index	Marker Type
31	*
30	☆
29	★
28	+
27	◊
26	△
25	□
24	○
23	▽
22	▲
21	
20	◎

Figure 8.9: HIGZ portable marker types

Line Index	Line Type
15
14
13	- - - - -
12	- - - - -

Figure 8.10: HIGZ portable line types

8.10 Text fonts

Text in PAW can be produced by two commands:

```
PAW > TEXT 10. 10. 'This is a text' 0.3
PAW > ITX 10. 10. 'This is a text'
```

The command TEXT draws software characters text, independently of the basic graphics package used by HIGZ. TEXT can produce over 200 different graphics signs, allowing to mix different types of characters (roman, greek, special, upper- and lowercase, sub- and superscript). The text to be printed is defined using a string of characters, separated by “escape” characters, which are defined in table 8.6. The repertoire of possible symbols is shown in figure 8.12.

<	go to lower case ¹	>	go to upper case (default)
[go to greek (roman = default)]	end of greek
"	go to special symbols	#	end of special symbols
^	go to superscript	?	go to subscript
!	go to normal level of script	&	backspace one character

Table 8.6: List of HPLOT escape sequences and their meaning

Boldface characters may be simulated by using the attributes PASS and CSHI. The meaning of these attributes is the following: every stroke used to display the character is repeated PASS times, at a separation CSHI*CHHE, where CHHE is the character height.

Software fonts are available using the TXFP attribute of the IGSETcommand as follows:

```
PAW > IGSET TXFP ffff
```

where ffff is the **font identifier** as defined in the figures on the following pages and p is the **software precision** parameter (i.e. 0 for hardware precision and 2 for software precision).

The command ITX draws a character string with the font given by IGSET TXFP. Via the ITX command the PostScript fonts described in Figures 8.13 and 8.14 using metafile types -111 and -112 are available. Figures 8.15, 8.16 and 8.17 display lists of PostScript characters accessible via an octal code preceded by a backslash, e.g. to print a “copyright” character © use:

Example on how to get special symbols with PostScript

```
PAW > IGSET TXFP -120 | Choose Symbol font
PAW > ITX 10. 10. '\323' | Copyright sign at position (10,10)
```

¹ Characters can also be entered directly in lower case.

Important Note

The characters '(', ')', and '\' are control characters for PostScript. To produce these characters in a PostScript metafile (-111 or -112) they must be escaped by a backslash.

Printing special PostScript characters

```
PAW > IGSET TXFP -20 | Choose font Times-Bold
PAW > ITX 10. 10. '\( | Open parenthesis at position (10,10)
```

French accented, German umlauted or other special **national characters** in any of the PostScript fonts are available by using their octal code as found in figure 8.15, e.g.:

Example of using national PostScript characters

```
PAW > IGSET TXFP -130 | Choose PostScript font Times-Roman
PAW > ITX 1. 4. 'K\355nstler in den gr\345\373ten St\311dten.'
PAW > ITX 1. 3. '\253\265 10'\372uvre on conna\333t 10'artisan\273'
PAW > ITX 1. 2. '\(proverbe fran\321ais\).' | quote escaped by @ character
PAW > ITX 1. 1. '\252\241Ma\337ana!\272, dit 10'\3231\325ve..'
```

The output would be:

Künstler in den größten Städten.
 «À l'œuvre on connaît l'artisan»
 (proverbe français).
 “¡Mañana!”, dit l'élève.

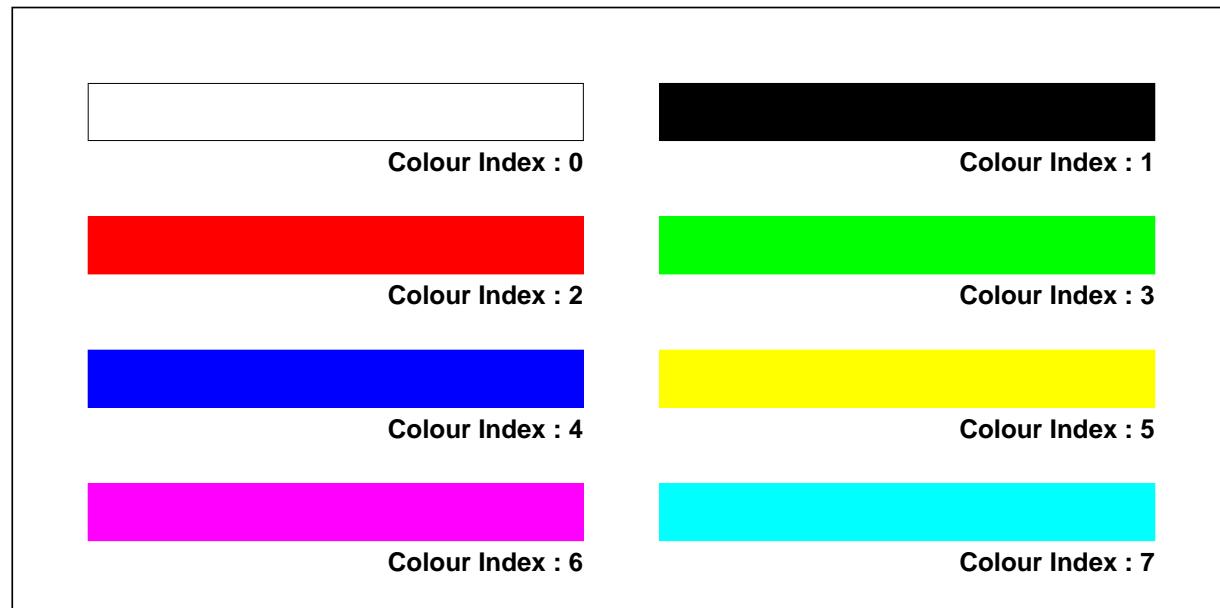


Figure 8.11: PostScript grey level simulation of the basic colours

Figure 8.12: HIGZ portable software characters (Font 0, Precision 2)

Figure 8.13: PostScript fonts

A	✡	a	✿	0	-pencil	[*
B	✚	b	◎	1	⌚]	*
C	❖	c	*	2	⌚	;	+
D	♣	d	✳	3	✓	.	-pencil
E	❖	e	✳	4	✓	/	-pencil
F	◆	f	✳	5	✗	\	*
G	❖	g	*	6	✗	{	‘
H	★	h	*	7	✗	}	“
I	☆	i	*	8	✗	:	+
J	●	j	*	9	✚	"	✂
K	☆	k	*	-	👉	<	✚
L	☆	l	●	=	✝	>	✝
M	☆	m	○	'	✿		,
N	☆	n	■	!	✂		
O	☆	o	□	@	✖		
P	☆	p	□	#	✂		
Q	*	q	□	\$	✂		
R	*	r	□	%	☎		
S	*	s	▲	^	✿		
T	*	t	▼	&	⌚		
U	✳	u	◆	*	👉		
V	*	v	❖	(✈		
W	*	w	▷)	✉		
X	*	x		_	✿		
Y	*	y	█	+	👉		
Z	*	z	█	~	‘		

Figure 8.14: Correspondence between ASCII and ZapfDingbats font (-14)

173	{	270	,	323	é	356	Ü
174		271	„	324	É	357	å
175	}	272	”	325	è	361	æ
176	~	273	»	326	È	362	Å
241	í	274	...	327	ê	363	ÿ
242	¢	275	%o	330	Ê	364	Ý
243	ƒ	276	â	331	ë	365	ı
244	/	277	ξ	332	Ë	366	á
245	¥	300	À	333	î	367	Á
246	f	301	`	334	Î	370	ł
247	§	302	'	335	ï	371	ø
250	¤	303	^	336	Ï	372	œ
251	'	304	~	337	ñ	373	ß
252	“	305	-	340	Ñ	374	ù
253	«	306	ˇ	341	Æ	375	Ù
254	<	307	·	342	ô	376	□
255	>	310	..	343	a		
256	fi	311	ä	344	Ô		
257	fl	312	°	345	ö		
260	à	313	¸	346	Ö		
261	—	314	Ã	347	û		
262	†	315	”	350	Ł		
263	‡	316	‘	351	Ø		
264	.	317	ˇ	352	Œ		
265	À	320	—	353	º		
266	¶	321	ç	354	Û		
267	•	322	Ç	355	ü		Font/Prec -13/0

Figure 8.15: Octal codes for PostScript characters in Times font

173	‘	270	❸	323	❽	356	⟲
174	’	271	❹	324	→	357	⟳
175	“	272	❺	325	→	361	⟳
176	”	273	❻	326	↔	362	⟳
241	⌚	274	❷	327	↑↓	363	➡➡
242	⌚	275	❸	330	↖	364	↖
243	⌚	276	❹	331	→	365	⇒
244	⌚	277	❽	332	↗	366	↗
245	♠	300	❶	333	→	367	↙
246	🂡	301	❷	334	→	370	↗
247	🂢	302	❸	335	→	371	↗
250	♣	303	❷	336	→	372	→
251	♦	304	❹	337	→→	373	↔
252	♥	305	❻	340	→→	374	→
253	♠	306	❷	341	→	375	→
254	❶	307	❸	342	➤	376	⇒⇒
255	❷	310	❹	343	➤		
256	❸	311	❽	344	➤		
257	❷	312	❶	345	↔		
260	❹	313	❷	346	↔		
261	❻	314	❶	347	↔		
262	❷	315	❷	350	↔		
263	❸	316	❹	351	↔		
264	❹	317	❻	352	↔		
265	❽	320	❷	353	↔		
266	❶	321	❸	354	↔		
267	❷	322	❹	355	↔		

Figure 8.16: Octal codes for PostScript characters in ZapfDingbats font (-14)

173	{	270	÷	323	©	356	}	
174		271	✗	324	™	357		
175		272	≡	325	∏	361		
176	~	273	≈	326	√	362		
241	Y'	274	· ·	327	•	363		
242	,	275	—	330	¬	364		
243	W	276	—	331	Λ	365		
244	/	277	↙↗↖↘↖	332	∨	366		
245	8	300	↔↔↔↔↔↔↔↔	333	↔↔↔↔↔↔↔↔	367		
246	f	301	↔↔↔↔↔↔↔↔	334	↔↔↔↔↔↔↔↔	370		
247	♣	302	↔↔↔↔↔↔↔↔	335	↔↔↔↔↔↔↔↔	371		
250	♦	303	↔↔↔↔↔↔↔↔	336	↔↔↔↔↔↔↔↔	372		
251	♥	304	↔↔↔↔↔↔↔↔	337	↔↔↔↔↔↔↔↔	373		
252	♠	305	↔↔↔↔↔↔↔↔	340	↔↔↔↔↔↔↔↔	374		
253	↔↔	306	↔↔↔↔↔↔↔↔	341	↔↔↔↔↔↔↔↔	375		
254	←	307	↔↔↔↔↔↔↔↔	342	↔↔↔↔↔↔↔↔	376		
255	↑	310	↔↔↔↔↔↔↔↔	343	©	}		
256	→	311	↔↔↔↔↔↔↔↔	344	™			
257	↓	312	↔↔↔↔↔↔↔↔	345	Σ			
260	○	313	↔↔↔↔↔↔↔↔	346				
261	±	314	↔↔↔↔↔↔↔↔	347				
262	"	315	↔↔↔↔↔↔↔↔	350				
263	W	316	↔↔↔↔↔↔↔↔	351				
264	×	317	↔↔↔↔↔↔↔↔	352				
265	∞	320	↔↔↔↔↔↔↔↔	353				
266	∂	321	↔↔↔↔↔↔↔↔	354				
267	•	322	↔↔↔↔↔↔↔↔	355				

Figure 8.17: Octal codes for PostScript characters in Symbol font

8.11 The HIGZ graphics editor

The HIGZ pictures in memory can be modified interactively with the HIGZ graphics editor. The command PICT/MODIFY invokes the HIGZ editor (see figure 8.18 for more details):

PAW > PICT/MODIFY PNAME

PNAME can be the complete name, the picture number in memory or ' '.

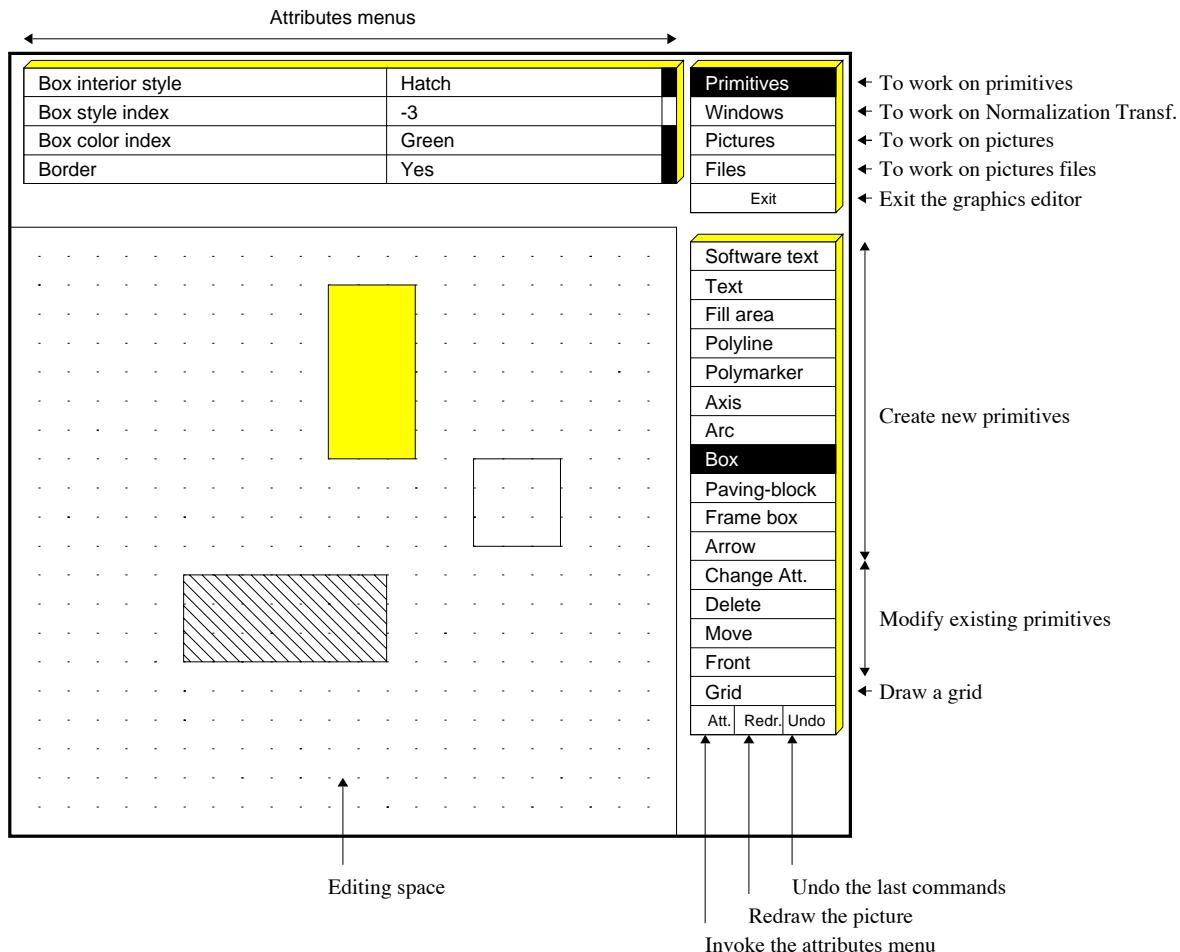


Figure 8.18: The HIGZ graphics editor

Chapter 9: Distributed PAW

With the increasing number of workstations, it happens more and more frequently that a user wants to run PAW on a mainframe or on a workstation. Several tools described in this chapter have been developed in order to use in the most convenient way all the resources available in an heterogeneous environment of workstations, superminis, data acquisition systems and mainframes.

- TELNETG:** A powerful terminal emulator. An alphanumeric window (line mode) is created on the local workstation (e.g. Apollo) to create a session (like with TELNET) on a remote computer (e.g. VAX). On the remote computer, a graphics program is run and a window is automatically created on the local workstation to receive the graphics output.
- 3270G** Same as the TELNETG emulator for the case of a connection with an IBM machine in full screen mode under VM/CMS.
- ZFTP** The ZEBRA file transfer program optimized to transport ZEBRA RZ or FZ files between machines with different data representations.

There exists also the possibility to access files on a **remote computer** from a PAW session on a workstation. PAW can be used in a **real time** environment. Access to HBOOK histograms being filled by a different process on the same machine (Global sections on VAX) or a computer on the network (e.g. OS9 modules). Both ZFTP and real time access to histograms on a remote computer require the implementation of a **PAW server** on this computer. The PAW server is automatically started from a PAW session, if PAW has been implemented with the relevant options (PATCHY [16] flag CZ). PAW and the PAW server must be linked with two special modules called **CZ** and **TCPAW** [17, 18].

CZ is a small FORTRAN package (about 300 lines). It provides an interface between the ZEBRA Input/Output routines and the high level transport routines of the TCPAW package.

TCPAW[17] is a networking package, written in C by Ben Segal (about 1500 lines). It provides a very simple FORTRAN-callable interface to TCP/IP services. It supports client and server modules running on UNIX, Apollo, VMS, VM/CMS and OS9 environments. Small parts of TCPAW are CERN specific but it would be perfectly possible to transport it elsewhere with minor modifications. The package currently requires the Wollongong (TWG) TCP/IP software to be present on VMS connected systems, the IBM FAL 1.2 Product on VM/CMS, and Microware TCP/IP on OS9. The UNIX systems Ultrix, CRAY Unicos, SUN OS, IBM AIX, Apollo/Aegis, Apple A/UX and HP-UX are supported as delivered.

9.1 TELNETG and 3270G

Figure 9.1 describes the functionality of these two programs. They allow to run a graphics application based on HIGZ (e.g. PAW, GEANT, etc.) on a host machine and to receive the graphics output on the local machine. TELNETG is designed to work with operating systems supporting a command line interface and 3270G for a full screen interface.

TELNETG and 3270G supports both graphics Input and Output. The graphics locator (commands LOCATE, VLOCATE, etc.) as well as the various KUIP graphics menu styles (G and GP) may be used. Both programs exploit the fact that the HIGZ macro primitives are very compact, therefore reducing the amount of information to be sent through the network. Compared to more conventional emulators (4014, 4207, etc.) gains in speed are typically a factor of 10 when drawing one-dimensional histograms and may reach a factor 100 for two-dimensional plots (lego, surface, scatterplot).

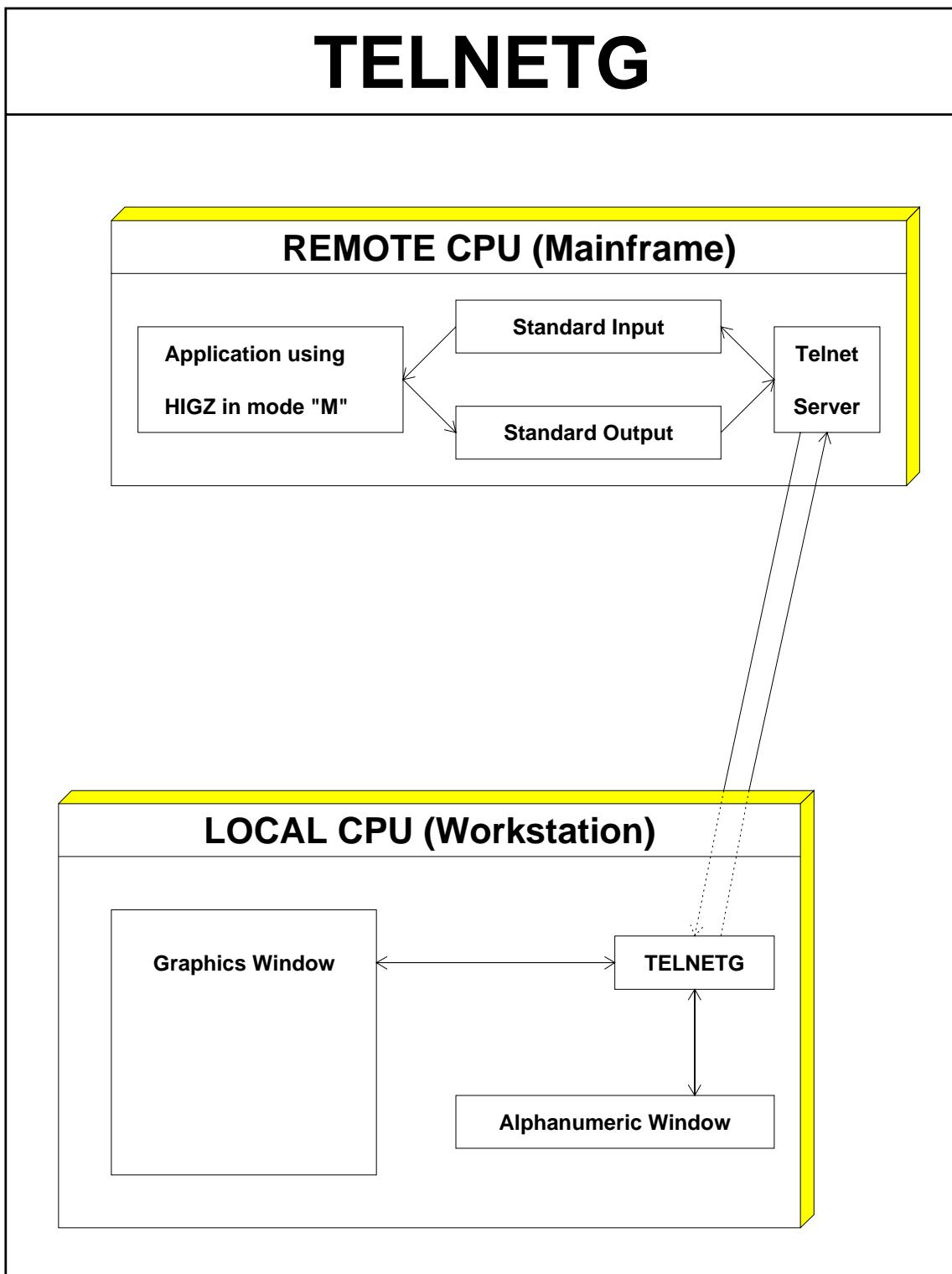


Figure 9.1: The TELNETG program

TELNETG combines a slightly modified version of the standard TELNET program written in the C language and an interface to the HIGZ system written in FORTRAN.

The following example shows how to use TELNETG from an Apollo to a VAX. The integer identifier of the workstation type must be preceded by a **minus sign** (e.g. for an Apollo DN3000):

Example of a TELNETG session

```
$ TELNETG vxcrna
Trying...
Open
      This is the CERN Central VAXcluster running VMS V5.1

Username: USERNAME
Password: PASSWORD(not echoed)
      Welcome to VAX/VMS version V5.1 on node VXCRNA
TERMINAL TYPE <? for HELP; No default>:D1
VxCrnA$ PAW
*****
*          *          *
*      W E L C O M E      to      P A W      *
*          *          *
*          Version 1.11/02  29 March 1991      *
*          *          *
*****  
Workstation type (?=HELP) <CR>=7878 : -10002
VERSION 7.4/2.6 OF GKSGRAL STARTED
PAW > hi/plot 10          | The graphics is sent to the Apollo
PAW > locate              | Graphics input using the Apollo mouse
```

9.2 ZFTP

The ZFTP program (ZEBRA File Transfer Program) provides the same functionality as the FTP program which is available like TELNET on all workstations and mainframes supporting TCP/IP. In addition ZFTP has been optimized to allow the transfer of ZEBRA binary files both sequential and direct access.

The direct access ZEBRA/RZ files (used for HBOOK histograms and HIGZ pictures) contain data in the local data representation. Because ZEBRA is an object oriented language supporting machine independent Input/Output, ZFTP is able to translate in flight all the ZEBRA data structures in a transparent way in the network buffers. ZFTP copies the RZ files on the local machine with the same parameters (RECL, quota, etc.) than on the remote machine. The original date and time of the objects is also preserved.

In addition to binary file transfer, ZFTP can also transfer alphanumeric text files (up to 80 characters/line). On IBM/VM-CMS, these files must be of type `RECFM=F,LRECL=80`.

The ZFTP user interface is based on KUIP and is the same on all systems. If several files have to be transferred (maybe on a regular basis), KUIP macros may be used. The following commands are available:

OPEN	To start a communication with a remote machine.
CLOSE	Close the current communication.
GETA	Transfer an Alphanumeric text file from the remote machine.
PUTA	Transfer an Alphanumeric text file to a remote machine.
GETRZ	Transfer a RZ file from a remote machine.
PUTRZ	Transfer a RZ file to a remote machine.
GETFZ	Transfer a FZ file from a remote machine.
PUTFZ	Transfer a FZ file to a remote machine.
RSHELL	Send a command to a remote machine.

Example of a ZFTP session

```
# Start execution of the program from inside the PAW directory
$ ZFTP
ZFTP > open CERNVM                                | Starts communication with CERNVM
| (prompt for username/password)
ZFTP > getrz RZFILE.DAT.D local.dat             | Transfer IBM file "RZFILE.DAT"
| to local file "local.dat"
ZFTP > puta local.car                            | Transfer local alphanumeric file
| "local.car" to IBM
| IBM file name will be "LOCAL CAR A"
ZFTP > quit
```

9.3 Access to remote files from a PAW session

When running PAW, it is often necessary to access files (e.g. HBOOK files) which reside on a different computer. The ZFTP program described above can be used if a very frequent access to the file is required. A more convenient mechanism is the possibility to access the files directly. On many systems, one may now use NFS [19] for this purpose. Under some circumstances, for example if the HBOOK file is not in exchange mode and it is to be accessed from a computer running a different operating system, an alternate approach is required. To fill this gap the PAW server is provided. This works using a conventional Client/Server model. The client (PAW) typically runs on a workstation. When the PAW command RLOGIN is invoked, a PAW server is automatically started on the remote machine, normally a mainframe or data server.

Once the RLOGIN REMOTE command has been executed, the PAW Current Directory is set to //REMOTE. The PAW client can now instruct the PAW server to attach a file using the RSHELL command (e.g. rshell file pawtest.dat). If an histogram with HBOOK ID=10 is on the remote file, than the PAW command Histo/Plot 10 will plot this histogram on the local workstation. The histogram resides on //PAWC like other histograms coming from local files.

The RSHELL command may be used to communicate with the PAW server. The expression typed following RSHELL is passed to the server. The current implementation of the PAW server recognizes the commands:

rshell file filename	Server connects filename
rshell cdir //lun11	Server changes current directory
rshell ld	Server lists current directory
rshell ld //	Server lists all connected files
rshell message	Server pass message to operating system

Access to remote files from a workstation

```
PAW > rlogin CERNVM
PAW > rshell file HRZTEST.DAT
PAW > histo/plot 10
PAW > histo/fit 20 G
PAW > rlogin VXCRNA
PAW > rshell file DISK$DL:[PAW]HEXAM.DAT;3
PAW > histo/plot 110
PAW > rshell file HRZTEST.DAT
PAW > histo/plot 110 s

PAW > rshell ld //
PAW > cdir //CERNVM
PAW > histo/plot 110
PAW > histo/plot //VXCRNA/110
PAW > cdir //PAWC
PAW > histo/list
PAW > Histo/delete 0
PAW > hrin //VXCRNA/0

PAW > cdir //CERNVM
PAW > rshell file NEW.DAT.D 1024 N
PAW > hrouut 0

| connect to CERNVM
| PAW server connects HRZTEST DAT A to //LUN11
| plot histogram 10 from CERNVM
| fit histo 20 with a gaussian and plot it
| connect to VXCRNA
| PAW server on VXCRNA connects file to //LUN11
| plot histogram 110 from VXCRNA
| PAW server on VXCRNA connects file to //LUN12
| plot histogram 110 from HRZTEST.DAT
| on VXCRNA on the existing picture
| list all files connected on VXCRNA
| Change current PAW directory to CERNVM
| plot histogram 110 from CERNVM
| plot histogram 110 from VXCRNA
| current directory to local memory
| list all histograms in //PAWC
| delete all histograms in memory
| read all histograms from VXCRNA
| file HRZTEST.DAT to //PAWC
| change directory to CERNVM
| creates a new file on the D disk
| write all histograms from //PAWC
| to CERNVM file NEW DAT D
```

9.4 Using PAW as a presenter on VMS systems (global section)

```

PROGRAM PRODUCE
PARAMETER MAXPAGES=100
COMMON/PAWC/IPAWC(128*MAXPAGES)
CHARACTER*8 GNAME
INTEGER*4 HCREATEG
*
      GNAME='GTEST'
      WAIT_TIME=1.
      NUMEVT=1000
*..... Create Global section
      NPAGES=HCREATEG(GNAME,IPAWC,128*MAXPAGES)
      IF(NPAGES.GT.0) THEN
         PRINT 1000,GNAME
1000   FORMAT(' Global Section: ',A,' created')
      ELSE
         IERROR=-NPAGES
         PRINT 2000,IERROR
2000   FORMAT(' Global Section Error', I6)
         GO TO 99
      ENDIF
      CALL HLIMIT(128*NPAGES)
*..... Book histos.
      CALL HBOOK1(10,'Test1$',50,-4.,4.,0.)
      CALL HBOOK1(20,'Test2$',50,-4.,4.,0.)
*..... Fill histos.
      DO 20 I=1,NUMEVT
         DO 10 J=1,100
            CALL RANNOR(A,B)
            CALL HFILL(10,A,0.,1.)
            CALL HFILL(20,B,0.,1.)
10      CONTINUE
         CALL LIB$WAIT(WAIT_TIME)
20      CONTINUE
*
99      STOP
      END

$ fort produce
$ link produce,SYS$INPUT/OPTIONS,-
cern$library:packlib/lib,kernlib/lib
PSECT=PAWC,PAGE

```

```

PAW > edit produce
macro produce ntimes=100
nt=[ntimes]
zone 1 2
histo/plot 10 K
histo/plot 20 K
loop:
histo/plot 10 U
histo/plot 20 U
wait ' ' 1
nt=[nt]-1
if nt>0 goto loop
return
PAW > global GTEST
PAW > exec produce ntimes=20

```

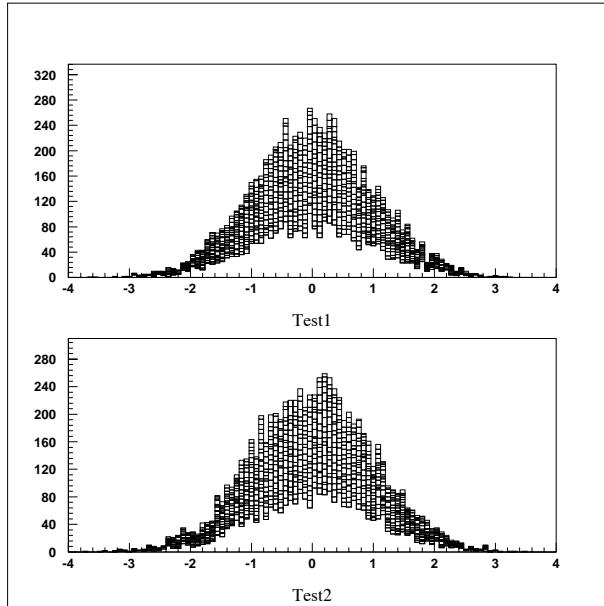


Figure 9.2: Visualise histograms in global section

In addition to the facilities described in the previous section, the standard version of PAW may be used as an online presenter on VMS systems using the mechanism of global sections. It is possible for two processes to reference the same histograms using **global sections**. For example, the first process may be a **histogram producer** (e.g. a monitoring task) and the second process **PAW**. As the histograms are being gradually filled by the first task, PAW can view them, and even reset them. To use the global sections, it is also necessary to "page align" the common which is in the global section. This is achieved in the "link step" when making the process (see example). The relevant statements are **SYS\$INPUT/OPTIONS** to tell the linker that some options follow the link statement, and **PSECT=PAWC,PAGE** which is the option to page align the **/PAWC/** common.

9.5 Using PAW as a presenter on OS9 systems

The technique described in previous sections may also be used to access HBOOK histograms being filled by a monitoring task on OS9 systems from a standard PAW session running on a machine with the TCP/IP software.

```

INDIRECT PAWC
PROGRAM PRODUCE
*
*      Monitoring task MT1 in processor OP2.
*
PARAMETER NWPAW=10000
COMMON/PAWC/IPAWC(NWPAW)
*
CALL HLIMIT(NWPAW)
*
*      Book histos.
*
CALL HBOOK1(10,'TEST1$',50,-3.,3.,0.)
CALL HBOOK1(20,'TEST2$',50,-3.,3.,0.)
*
*      Fill histos.
*
NUMEVT=10000
DO 20 I=1,NUMEVT
    DO 10 J=1,100
        CALL RANNOR(A,B)
        CALL HFILL(10,A,0.,1.)
        CALL HFILL(20,B,0.,1.)
10    CONTINUE
20    CONTINUE
*
99    STOP
END

```

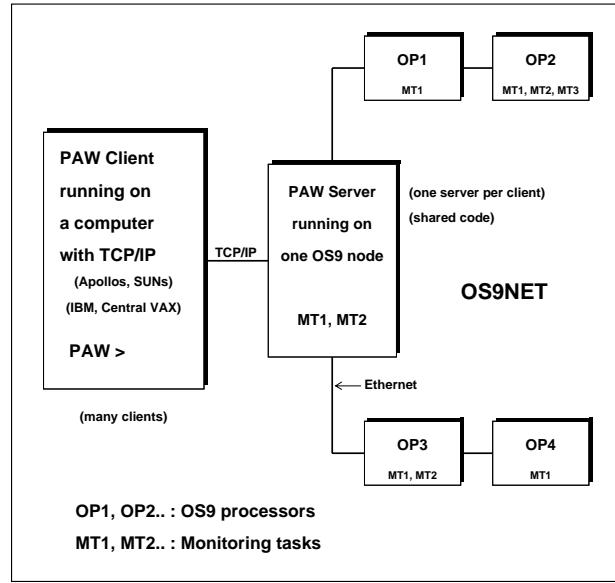


Figure 9.3: Visualising histograms on OS9 modules from PAW

Example of how to access OS9 modules from PAW

PAW > <u>rlogin 0-OPAL01</u>	connect to an OS9 machine
PAW > <u>rshell module OP2/MT1</u>	PAW server connects to OP2/MT1
PAW > <u>histo/plot 10</u>	(Processor OP2, Monitoring Task MT1)
PAW > <u>hrin 0</u>	plot histogram 10
PAW > <u>Histo/File 1 local.dat 1024 N</u>	read all histograms into //PAWC
PAW > <u>hrout 0</u>	create a new file local.dat
PAW > <u>rshell module OP3/MT2</u>	on the client machine
PAW > <u>Output 56 os9.listing</u>	save all histograms from //PAWC
PAW > <u>rshell ldir</u>	to the local file
PAW > <u>Output -56</u>	PAW server connects to another
	OS9 monitoring task
	Change output file on client
	list all histograms in MT2
	on file os9.listing
	Change output file to default (unit 6)
	file os9.listing is closed

Part III

PAW - Reference section

Notation used in the reference section

Optional parameters are enclosed in square brackets, e.g. [optpar]

The **type** of a parameter is indicated following its name as follows:

- C Character data
- I Integer data
- R Real (floating point) data

Supplementary information is given at the end of the line describing the parameter:

- D= Default value
e.g. D='S' for Character data or D=40 for Integer data
- R= Range of possible values
e.g. R=0:1 means that the variable's value lies between 0 and 1.
R=' ,L,P,*,+ ' enumerates the possible values for the given Character variable.

Chapter 10: KUIP

Command Processor commands.

HELP [item]

ITEM Command name *C D=’ ’*

Give the help of a command. If a command is supplied as parameter ITEM, its full explanation is given: syntax (as given by the command USAGE), explanation of the functionality, list of parameters with their attributes (prompt, type, default, range, etc.). 'HELP /' gives the help for all commands.

If HELP is entered without parameters, the dialogue style is switched to 'AN', guiding the user in traversing the tree command structure.

'HELP -EDIT' (or just 'HELP -E') switch the help in the edit mode. This mode is currently implemented on IBM/VM-CMS and Apollo: instead of writing the help text on the terminal output, it is written into a temporary file and the editor is invoked, XEDIT on IBM/VM-CMS or a new pad window on the Apollo (see also command SET_SHOW/HOST_EDITOR). 'HELP -NOEDIT' (or just 'HELP -N' or 'HELP -') switch the help in the standard mode.

USAGE [cmnd]

CMND Command name *C D=’ ’*

Give the syntax of a command. If CMND='/' the syntax of all commands is given.

MANUAL [item outfil docsys]

ITEM Command name or help (key)word *C D=’ ’*

OUTFIL Output file name *C D=’ ’*

DOCSYS Text formatting system *C D=’ ’ R=’ ,SGML,TEX,LATEX’*

Write on a file the text formatted help of a command. If OUTFIL=' ' the text is written to the terminal. It works like the command HELP ITEM, plus a text formatting option: if DOCSYS is not a blank the output is formatted according to the selected documentation and text formatting system (ex. SGML, TEX, LATEX, etc.). Example:

MANUAL / FILE.DOC SGML

will produce a file named FILE.DOC containing the SGML input to be processed by SGML in order to have the documentation of all the available commands.

LISTHELP [lstnam]

LSTNAM List name *C D=’ ’*

Help for lists. If a list is specified, display its attributes (D='dynamic', S='static', T='temporary', P='permanent') and a description of the list and its use. Otherwise display the names of all of the known lists. 'LISTHELP /' gives help for all lists.

EDIT fname

FNAME File name C

Invoke the default host editor. The file FNAME.KUMAC (the extension KUMAC is taken automatically unless FNAME contains already a dot) is given to the editor and, if it does not exist, a new file is created with this name. Use the command SET_SHOW/HOST_EDITOR to change the default editor.

LAST [n fname]

N N last commands to be saved I D=-1

FNAME File name C D=’ ’

Perform various operations with the history file.

If FNAME is not specified, the current history file is assumed by default (the startup history file name is LAST.KUMAC). To change the history file the command LAST 0 NEWFNAME must be entered.

If N=-1 (default case) the default host editor is called to edit the current history file, containing all the commands of the session.

If N=0 the history file FNAME is rewound and set as the current one (the command LAST 0 FNAME itself is not recorded).

If N>0 the last N commands of the session are saved in the current history file.

See also the command SET_SHOW/RECORDING.

MESSAGE [string]

STRING Message string C D=’ ’

Write a message string on the terminal. A useful command inside a macro. Several message strings can be given in the same command line, each of them separated by one or more spaces (the usual parameter separator); therefore multiple blanks will be dropped and only one will be kept. If multiple blanks have not to be dropped, the string must be surrounded by single quotes.

SHELL [cmd]

CMD Shell command C D=’ ’

Execute a command of the host operating system. To execute just one command enter SHELL COMMAND; otherwise enter just SHELL and wait for the system prompt. After the last command enter RETURN (the full word, not just <CR>) to go back to the application.

WAIT [string sec]

STRING Message string C D=’ ’

SEC Number of seconds R D=0

Make a pause (e.g. inside a macro). Wait a given number of seconds (if SEC.GT.0) or just until <CR> is entered (if SEC.LE.0). A message string is also written on the terminal before waiting.

UNITS

List all Input/Output logical units currently open. The files attached to them are also shown.

EXIT

End of the interactive session.

QUIT

End of the interactive session.

FUNCTIONS

KUIP System Functions. The function name (and arguments) is literally replaced, at run-time, by its current value. At present, the following functions are available:

\$DATE	Current date in format DD/MM/YY
\$TIME	Current time in format HH.MM.SS
\$CPTIME	CP time elapsed since last call (in sec)
\$RTIME	Real time elapsed since last call (in sec)
\$VDIM(VNAME, IDIM)	Physical length of vector VNAME on dimension IDIM (1..3)
\$VLEN(VNAME, IDIM)	As above, but for the logical length (i.e. stripping trailing zeroes)
\$NUMVEC	Current number of vectors
\$VEXIST(VNAME)	Index of vector VNAME (1..\$NUMVEC or 0 if VNAME does not exist)
\$SUBSTRING(STRING, IX, NCH) ...	STRING(IX:IX+NCH-1)
\$UPPER(STRING)	STRING changed to upper case
\$LOWER(STRING)	STRING changed to lower case
\$LEN(STRING)	Length of STRING, stripping leading/trailing blanks and single quotes
\$SIGMA(SIGMA_Expression)	Result of SIGMA_Expression, computed by SIGMA
\$ARGS	Command line at program invocation
\$KEYNUM	Address of latest clicked key in style GP
\$KEYVAL	Value of latest clicked key in style GP
\$LAST	Latest command line executed
\$ANUM	Number of aliases
\$ANAM(I)	Name of I-th alias
\$AVAL(I)	Value of I-th alias
\$STYLE	Current style as defined by SET/STYLE

10.1 ALIAS

Operations with aliases. Aliases are defined to provide shortcut abbreviations for the input line or some part of it. An alias name can be any string of characters (excepted the single quote and the blank) and whenever encountered in an input line it will be literally replaced by its value (another string of characters). Alias substitution does not apply in quoted strings. Aliases need separators to be recognized in the input line, namely:

blank / , = : . % ' ()

To juxtaposition aliases, a double slash can be used as concatenation sign. Be careful not defining aliases recursively.

CREATE *aname string [chopt]*

ANAME Alias name *C*
STRING Equivalent string *C*
CHOPT Option *C D=' ' R=' ,C'*

Create an alias named ANAME corresponding to STRING. Also switch ON the alias translation, i.e. ALIAS/TRANSLATION ON. If CHOPT='C' then the alias is a command alias, i.e. an alias that will only be translated when it is the first token on a command line. Example:

```
Alias/Create GG Graph/Struct/Scratch
Alias/Create FF File1/Name1/Name2
GG FF/ID
```

is equivalent to

```
Graph/Struct/Scratch File1/Name1/Name2/ID
```

```
Alias/Create LS DIR C
```

is equivalent to

```
DIR
```

only when LS is the first token on a command line. In the following case LS will not be translated

```
SHELL LS
```

LIST

List all aliases (name, equivalent string).

DELETE *alist*

ALIST Alias list *C*

Delete the definition of aliases in the list ALIST. The aliases are separated in the list by a comma and imbedded blanks are not allowed. If ALIST='*' then delete all aliases and the alias translation is switched OFF (i.e.: ALIAS/TRANSLATION OFF is executed).

TRANSLATION *[option]*

OPTION Option *C D='ON' R='ON,OFF,?'*

Switch ON/OFF the alias translation. If OFF, alias definitions are not used in parsing the command lines. It is automatically switched ON when an alias is created. If OPTION='?' the current value is shown. The startup value is OFF.

10.2 SET_SHOW

Set or show various KUIP parameters and options.

STYLE [option sgylen sgsize sgyspa sgbord wktype]

OPTION	Option	<i>C</i>	D='?' R='?',C,AN,AL,G,GW,GS,GP'
SGYLEN	max Y LENgth of each menu item box	<i>R</i>	D=0.025 R=0.005:0.25
SGSIZE	space available for the application	<i>R</i>	D=0.8 R=0:0.9
SGYSPA	max Y length of space between menus	<i>R</i>	D=0.02 R=-0.5:0.5
SGBORD	X or Y border for menus	<i>R</i>	D=0.015 R=0:0.25
WKTYPE	Graphics workstation type	<i>I</i>	D=0

Set the user dialogue style (or working mode). If OPTION='?' the current style is shown. The startup value is C (command mode). Currently available options are:

- C* for Command
- AN* for general Alpha menu (with Numbers)
- AL* for general Alpha menu (with Letters)
- G* for Graphics menu (with hardware character fonts)
- GW* for Graphics menu (with shadowed Width effect)
- GS* for Graphics menu (with Software character fonts)
- GP* for Graphics menu (with Panel keys only, i.e. no command tree menu),

When using OPTION='G' (or 'GW', 'GS', 'GP') the 4 parameters following the style can be defined to control the geometrical layout of the menus on the screen, and the the fifth one to set the graphics workstation type (without being prompted afterwards, if case of HIGZ was not initialized).

PANEL line [gkey]

LINE	Line number	<i>R</i>	D=0
GKEY	Graphics key value(s)	<i>C</i>	D=' '

Set up the panel of graphics keys (used by STYLE GP).

Examples:

PANEL 0	reset the panel
PANEL 2 A/L QUIT V/L	initialize line 2 with 3 graphics keys, respectively A/L, QUIT, V/L
PANEL 2 A/L ' ' V/L ' ' ' ' initialize line 2 with 5 graphics keys, and fill 1st and 3rd keys	
PANEL 2.04 MESSAGE	initialize 4th key of 2nd line to MESSAGE
PANEL 2.04	clear 4th key of 2nd line
PANEL -2.08	initialize line 2 with 8 graphics keys
PANEL -6.16	initialize line 6 with 16 graphics keys

Note that the key number on the right of the decimal point must always be defined with two digits.

Keys ending with a minus sign make an additional request of keyboard input; the complete command line will be the key text, with a blank at the place of the minus, concatenated with the additional keyboard input. Example:

PANEL 1.03 'VEC/PRI-'	entering VAB will execute VEC/PRI VAB.
-----------------------	--

Keys ending with a double minus sign behave as above but no blank is put at the place of the double minus. Example:

```
PANEL 1.03 'VEC/PRI V--'      | entering AB will execute VEC/PRI VAB
```

The dollar sign inside a key is replaced by additional keyboard input. Example:

```
PANEL 1.03 'VEC/PRI V($)'     | entering 11:20 will execute VEC/PRI V(11:20)
```

Maximum values for the key layout are: 1 panel, 10 lines/panel, 30 keys/line, 32 characters/key

COMMAND [chpath]

CHPATH Path name for command line *C D=’ ’*

Set a filter for the parsing of command lines. If it has been called, it means that whenever a command line is entered, if and only if it is not an existing command (not just ambiguous), it is inserted into the CPATH string, with \$n (n=1..9) being replaced by the n-th token of the command (tokens are separated by spaces), or \$* being replaced by the whole command line. Examples:

```
COMMAND 'V/CR $*(10),
AA                  =>  V/CR AA(10)
BB                  =>  V/CR BB(10)
V/LIST              =>  V/LIST

COMMAND 'VECTOR/PLOT $1 555 $2'
AA E                =>  VECTOR/PLOT AA 555 E
BB                 =>  VECTOR/PLOT BB 555

COMMAND             =>  shows its current value
COMMAND *            =>  reset (equivalent to COMMAND $*)
```

Note that COMMAND and subsequent command lines can be used inside macros, excepted when producing macro statements (like EXEC, IF, GOTO, etc.). For example, the above examples would work also inside macros, while COMMAND 'EXEC \$*' or COMMAND 'GOTO \$1' will not.

The same remark applies also to the command DEFAULT -Auto, which is in fact equivalent to COMMAND 'EXEC \$*'

The COMMAND logic is superseded by the DEFAULT -A (or -AR) logic.

APPLICATION [path cmdex]

PATH Application name *C D=’ ’*

CMDEX Exit command *C D=’EXIT’*

Set the application name. This means that all input lines will be concatenated to the string PATH (until the command specified by the parameter CMDEX is executed, which resets the application to the null string). The value of CMDEX may be specified if the default value EXIT has to be changed (i.e. because already used by the application). APPLICATION can also be inserted in a macro: in this case at least 4 characters must be specified (i.e. APPL).

ROOT [path]

PATH Root directory *C D='/'*

Set the root for searching commands. If PATH=? the current root is shown. This allows to access commands regardless of possible ambiguities with different menus. Commands are first searched starting from the current root: if a command is found it is executed. Only if a command is not found a second pass of search is done, starting now from the top root of the command tree (i.e. '/').

TIMING [option]

OPTION Option *C D='ON' R='ON,OFF,ALL'*

Set ON/OFF/ALL the timing of commands. If ON, the real time and the CPU time for the latest executed command (or macro) are presented. If ALL, the time is shown for each command being executed within a macro. The startup value is OFF.

PROMPT [option]

OPTION Prompt string *C D=' '*

Set the prompt string for the command mode dialogue. If OPTION is blank the current prompt is left unchanged.

BREAK [option]

OPTION Option *C D='ON' R='ON,OFF,TB,?'*

Set ON/OFF the break handling. If OPTION=? the current value is shown. The startup value is ON.

Hitting the keyboard interrupt (CTRL/C on VAX or CTRL/Q on the Apollo) under break ON condition, the current command or macro execution will be interrupted and the user will get again the application prompt.

BREAK TB switch ON the traceback of the routines called, with their line numbers, when an error occurs. This allows the detection of the routines which provoked the error.

COLUMNS [ncol]

NCOL Number of columns for terminal output *I D=80 R=0:132*

Set the maximum number of columns for terminal output. If NCOL=0 the current number of columns is shown. The startup value is 80.

RECORDING [nrec]

NREC Rate for recording on history file *I D=25 R=0:25*

Set the recording rate for the history file. Every NREC commands of the session the current history file is updated. If NREC=0 the history is not kept at all (i.e. the file is not written). See also the command LAST.

HOST_EDITOR [option top left width height dxpad dypad npads]

OPTION	Name of default host editor	<i>C</i>	D='?' R='?',EDT,TPU,DM,VI,WINDOW,PAD'
TOP	Top position of the edit window	<i>I</i>	D=20
LEFT	Left position of the edit window	<i>I</i>	D=20
WIDTH	Width of the edit window	<i>I</i>	D=0
HEIGHT	Height of the edit window	<i>I</i>	D=0
DXPAD	X offset for help PAD windows	<i>I</i>	D=30 R=0:
DYPAD	Y offset for help PAD windows	<i>I</i>	D=20 R=0:
NPADS	Maximum number of shifted pads	<i>I</i>	D=4 R=1:

Set the default host editor (only for VAX/VMS, Apollo and Unix). If OPTION='?' the current host editor is shown. The EDIT command will invoke this editor. The startup value is EDIT/EDT (VAX/VMS), dm (Apollo) or the value assigned to the EDITOR environment variable (Unix).

On the Apollo, if OPTION='WINDOW' or 'PAD' and some parameters are following, they will be used as edit window positions (in pixel units). If no parameters are following (i.e. typing just HOST WINDOW or HOST PAD) they will be asked graphically on a dummy window. OPTION='WINDOW' is used to specify window pad parameters used by commands like EDIT, LAST, etc., while OPTION='PAD' is used to specify help pad parameters used by the command HELP in EDIT mode.

This command is not meaningful on IBM/VM-CMS where it is always XEDIT.

HOST_SHELL [option]

OPTION	Name of default host shell	<i>C</i>	D='?' R='sh,csh,ksh,/com/sh'
--------	----------------------------	----------	------------------------------

Set the default host shell (only for Apollo and Unix machines). If OPTION='?' the current host shell is shown. The SHELL command will invoke this shell. The startup value is the value assigned to the SHELL environment variable.

VISIBILITY cmd [chopt1 chopt2]

CMD	Command name	<i>C</i>	D=''
CHOPT1	?, OFF, ON	<i>C</i>	D='?' R='?',OFF,ON'
CHOPT2	?, CLEAR, KEEP	<i>C</i>	D='?' R='?',CLEAR,KEEP'

Set or show the visibility attributes of a command.

If CHOPT1='OFF':

- the command is not executable anymore
- STYLE G draws a shadowed box on the command
- HELP may be still requested on the command

The startup value is ON.

CHOPT2 allows the user to customize the 'Style Motif':

- if CHOPT2='KEEP' the parameters window is not cleared upon execution

The startup value is CLEAR.

MODE mode

MODE KUIP Mode C D='M' R='M,A'

Set or Show KUIP mode for Command Area.

Chapter 11: MACRO

Macro Processor commands.

EXEC [mname]

MNAME Macro name C

Execute the command lines contained in the macro MNAME. As a file can contain several macros, the character '#' is used to select a particular macro inside a file as explained below.

If MNAME does not contain the character '#', the file MNAME.KUMAC is searched and the first macro is executed (it may be an unnamed macro if a MACRO statement is not found as first command line in the file).

If MNAME is of the form FILE#MACRO, the file named FILE.KUMAC is searched and the macro named MACRO is executed.

Examples:

```
EXEC ABC      to exec first (or unnamed) macro of file ABC.KUMAC
```

```
EXEC ABC#M    to exec macro M of file ABC.KUMAC
```

LIST [mname]

MNAME Macro name restrictions C D=' '

List all macros on disk. Macros are files with the extension KUMAC. MNAME may be specified to restrict the list to the macros containing such a string in the first part of their name, for example MACRO/LIST ABC will list all macros starting with ABC.

TRACE [option level prompt]

OPTION Option C D='ON' R='ON,OFF'

LEVEL Level C D=' ' R=' ,TEST,WAIT,FULL'

PROMPT Prompt string C D=' '

Set ON/OFF the trace of commands during macro execution. If TRACE='ON' the command being executed is written on the terminal, after the prompt defined in the parameter PROMPT. If LEVEL='TEST' the command is only echoed but not executed. If LEVEL='WAIT' the command WAIT is automatically inserted after the execution of each command. If LEVEL='FULL' all the names of macros and labels are printed at the end of macro interpretation. If PROMPT=' ' (default) the prompt written is replaced by a number of '>' equal to the current prompt length. At the end of the macro execution the prompt is switched back to the original one. The startup values are OPTION='OFF' and LEVEL=' '.

DEFAULTS [chpath]

CHPATH Path name for macro files C D=' '

Set or show various MACRO attributes.

Entered without parameters, it show the current "extra path" to the 'default working directory', as well as all macro parameters (name, default value) relative to the latest EXEC command entered.

If CHPATH is supplied, set its value representing the "extra path". The "extra path" is a string (e.g. the name of a disk directory) which is added in front of the macro file names, whenever they are used in the commands EDIT, EXEC and MACRO/LIST. The extra path is ignored if a dash sign '-' is put in front of the file name. If CHPATH='*', the extra path is reset and the host computer's current working directory is used. Example:

```

DEFAULT 'DISK$DL:[PAW]',
MACRO/LIST           | list macros in DISK$DL:[PAW]
EXEC FUNCTION        | executes DISK$DL:[PAW]FUNCTION.KUMAC
EXEC -MYMACRO        | executes MYMACRO.KUMAC
EXEC PALETTE         | executes DISK$DL:[PAW]PALETTE.KUMAC
DEFAULT *
EXEC PALETTE         | executes PALETTE.KUMAC

```

In addition to what described above, the command DEFAULTS may be used to control whether commands and/or macros are searched (and in which order):

```

DEFAULT -Command
CMD                  | CMD is executed (error if not found)
DEFAULT -Auto
CMD                  | try CMD first; if not found, try EXEC CMD
DEFAULT -AutoReverse
CMD                  | try EXEC CMD first; if not found, try CMD

```

The lower case letters following the minus sign are optional. The startup value (also re-set by DEFAULT *) is DEFAULT -C

The DEFAULT -A (or -AR) logic supersedes the COMMAND logic.

Important note:

Inside macros the DEFAULT -A (or -AR) logic is not active: DEFAULT -C is always assumed.

RECURSION [option]

OPTION Option C D='ON' R='ON,OFF'

Set ON/OFF the option to execute macros recursively. The startup value is OFF.

Chapter 12: VECTOR

Vector Processor commands. Vectors are equivalent to FORTRAN 77 arrays and they use the same notation except when omitting indexes (see last line below). Up to 3 dimensions are supported. Examples:

```
Vec(20) (mono-dimensional with 20 elements)
```

may be addressed by:

Vec	for all elements
Vec(13)	for element 13-th
Vec(12:)	for elements 12-th to last
Vec(:10)	for elements first to 10-th
Vec(5:8)	for elements 5-th to 8-th

```
Vec(3,100) (bi-dimensional with 3 columns by 100 rows):
```

may be addressed by:

Vec(2,5:8)	for elements 5-th to 8-th in 2-nd column
Vec(2:3,5:8)	for elements 5-th to 8-th in 2-nd to 3-rd columns
Vec(2,5)	for element 5-th in 2-nd column
Vec(:,3)	for all elements in 3-rd row
Vec(2)	for all elements in 2-nd column (SPECIAL CASE)

The latest line shows the special (and non-standard with FORTRAN 77) notation such that missing indexes are substituted to the right.

An 'invisible' vector called '?', mono-dimensional and of length 100, is always present. It is used for communicating between user arrays and KUIP vectors, being equivalenced with the real array VECTOR(100) in the labelled common block /KCWORK/.

```
CREATE vname [ type ]
```

VNAME	Vector name(length)	C
TYPE	Vector type	C D='R' R='R,I'

Create a vector named VNAME (elements are set to zero). The dimensions are taken from the name, for example VEC(20), VEC(3,100), VEC(2,2,10). Up to 3 dimensions are supported. Dimensions which are not specified are taken to 1, for example VEC(10) → VEC(10,1,1) and VEC → VEC(1,1,1). The vector may be of type Real or Integer. A vector is filled at the same time if parameters are given after the TYPE:

```
VEC/CREATE V(10) R 1 2 3 4 5 66 77 88 99 111  
VEC/CREATE W(20) R 1 2 3
```

In the last example only the first three elements are filled. Vector elements may be changed later with the command VECTOR/INPUT.

If many equal values have to be entered consecutively, one can specify just one value and precede it by a repetition factor and an asterisk. Example:

```
VEC/CREATE Z(20) R 5*1 2 4*3    --->    VEC/CREATE Z(20) R 1 1 1 1 1 2 3 3 3 3
```

Enter HELP VECTOR for more information on vector addressing.

LIST

List all vectors (name, dimensions, type).

DELETE vlist

VLIST Vector list *C D=*' '

Delete from memory all vectors in the list VLIST. The vectors are separated in the list by a comma and imbedded blanks are not allowed. An asterisk at the end of VLIST acts as string placeholder:

VEC/DEL AB*	---> deletes all vectors starting by AB
VEC/DEL *	---> deletes all vectors

COPY vnam1 vnam2

VNAM1 Source vector name *C*

VNAM2 Destination vector name *C*

Copy a vector into another one. Mixed vector type copy is supported (e.g. Integer → Real and viceversa). If VNAM2 does not exist it is created with the required dimensions, not necessarily the same as the source vector if a sub-range was specified. For example, if A is a 3 x 100 vector and B does not exist, COPY A(2,11:60) B will create B as a 50 elements mono-dimensional vector; a special (and non-standard with FORTRAN 77) notation is used such that, still using the above vectors, COPY A(2,1:100) B and COPY A(2) B have the same effect.

Note that VECTOR/COPY does not allow a range for the destination vector not specifying consecutive elements (i.e. along the first dimension):

VEC/COPY V(5)	W(3,4)	O.K.
VEC/COPY V1(2:3,5)	V2(4:5,9)	O.K.
VEC/COPY V1(5,2:3)	V2(4:5,9)	O.K.
VEC/COPY V1(3,3:4)	V2(4,4:5)	NOT allowed
VEC/COPY V1(2:3,5)	V2(2,4:5)	NOT allowed

Enter HELP VECTOR for more information on vector addressing.

INPUT vname

VNAME Vector name *C*

Enter values into a vector from the terminal. Example:

VEC/INPUT V(6:10) 1.1 2.22 3.333 4.4444 5.55555

If many equal values have to be entered consecutively, one can specify just one value and precede it by a repetition factor and an asterisk. Example:

VEC/INPUT V 5*1 2 4*3 ---> VEC/INPUT V 1 1 1 1 1 2 3 3 3 3

Enter HELP VECTOR for more information on vector addressing.

PRINT vname

VNAME Vector name C

Write to the terminal the content of a vector. Enter HELP VECTOR for more information on vector addressing.

READ vlist fname [format opt match]

VLIST	Vector list	C
FNAME	File name	C D=' '
FORMAT	Format	C D=' '
OPT	Options	C D='0C' R='0C,0, ,C'
MATCH	Matching pattern	C D=' '

Enter values into vector(s) from a file. A format can be specified, e.g. FORMAT='F10.5,2X,F10.5', or the free format is used if FORMAT is not supplied.

If vector(s) are not existing they will be created of the size as read from the file.

Vectors in the list VLIST are separated by a comma and imbedded blanks are not allowed. If subscripts are present in vector names, the smallest one is taken.

OPT is used to select between the following options:

- '0C' file is Opened, read and then Closed (default case)
- '0' file is Opened and then read (left open for further reading)
- ' ' file is read (already open, left so for further reading)
- 'C' file is read and then Closed (already open)

If the character 'Z' is present in OPT, the vector elements equal to zero after reading are set to the latest non-zero element value (for example reading 1 2 3 0 0 4 0 5 will give 1 2 3 3 3 4 4 5).

MATCH is used to specify a pattern string, restricting the vector filling only to the records in the file which verify the pattern. Example of patterns:

- /string/ match a string (starting in column 1)
- /string/ do not match a string (starting in column 1)
- /string/(n) match a string, starting in column n
- /string/(*) match a string, starting at any column

Enter HELP VECTOR for more information on vector addressing.

WRITE vlist [fname format chopt]

VLIST	Vector list	C
FNAME	File name	C D=' '
FORMAT	Format	C D='5(1X,G13.7)'
CHOPT	Options	C D='0C' R='0C,0, ,C'

Write to a file the content of vector(s). If FNAME= '' the content is written to the terminal. A format can be specified, e.g. FORMAT='F10.5,2X,F10.5', or the default one is used if FORMAT is not supplied.

Vectors in the list VLIST are separated by a comma and imbedded blanks are not allowed. If subscripts are present in vector names, the smallest one is taken.

CHOPT is used to select between the following options:

```
'OC'   file is Opened, written and then Closed (default case)
'O'    file is Opened and then written (left open for further writing)
' '   file is written (already open, left so for further writing)
'C'    file is written and then Closed (already open)
```

Enter HELP VECTOR for more information on vector addressing.

DRAW vname [id chopt]

```
VNAME  Vector name      C
ID     Histogram Identifier  C  D='12345'
CHOPT Options          C  D=' ' R=' ',C,S,+,B,L,P,*
Draw vector VNAME interpreting it as a histogram. Optionally save the contents in histogram ID.
CHOPT may be a combination of the following characters:
```

- 'C' Draw a smooth curve.
- 'S' Superimpose plot on top of existing picture.
- '+' Add contents of ID to last plotted histogram.
- 'B' Select Bar chart format.
- 'L' Connect channels contents by a line.
- 'P' Draw the current polymarker at each channel.
- '*' Draw a * at each channel.

HFILL vname id

```
VNAME  Vector name      C
ID     Histogram Identifier  C
Fill the existing histogram ID with vector VNAME. Note that the command VECTOR/PLOT can automatically book, fill and plot the contents of a vector.
```

PLOT vname [id chopt]

```
VNAME  Vector name      C
ID     Histogram Identifier  C  D='12345'
CHOPT Options          C  D=' ' R=' ',C,S,+,B,L,P,*
Each element of VNAME is used to fill an histogram which is automatically booked with 100 channels and then plotted. If VNAME has the form VNAME1%VNAME2 then a scatter-plot of vector VNAME1 versus VNAME2 is plotted. If ID is given different of 12345, then a 2-Dim histogram is created with 40 bins by 40 bins and filled. One can use the command VECTOR/HFILL to fill an already existing histogram. CHOPT may be a combination of the following characters:
```

- 'C' Draw a smooth curve.
- 'S' Superimpose plot on top of existing picture.
- '+' Add contents of ID to last plotted histogram.
- 'B' Select Bar chart format.
- 'L' Connect channels contents by a line.
- 'P' Draw the current polymarker at each channel.
- '*' Draw a * at each channel.

```
FIT x y ey func [ chopt np par step pmin pmax errpar ]
```

X	Vector of X coordinates	C
Y	Vector of Y coordinates	C
EY	Vector of errors on Y	C D='?'
FUNC	Function name	C
CHOPT	Character options	C D=' ' R=' ',O,N,Q,V,B,D,W,M'
NP	Number of parameters	I D=0 R=0:20
PAR	Vector of parameters	C
STEP	Vector of steps size	C
PMIN	Vector of lower bounds	C
PMAX	Vector of upper bounds	C
ERRPAR	Vector of errors on parameters	C

Fit a user defined function to the points defined by the two vectors X and Y and the vector of associated errors EY. See command Histo/Fit for explanation of parameters. Note that if option 'W' is specified or EY='?' (default), the array EY is ignored. Option 'L' is not available.

12.1 OPERATIONS

Simple arithmetic operations between vectors. In all the operations only the minimum vector length is considered, i.e. an operation between a vector A of dimension 10 and a vector B of dimension 5 will involve the first 5 elements in both vectors. If the destination vector does not exist, it is created with the same length as the source vector.

```
VBIAS vnam1 bias vnam2
```

VNAM1	Source vector name	C
BIAS	Bias value	R
VNAM2	Destination vector name	C
VNAM2(I) = BIAS + VNAM1(I)		

```
VSCALE vnam1 scale vnam2
```

VNAM1	Source vector name	C
SCALE	Scale factor	R
VNAM2	Destination vector name	C
VNAM2(I) = SCALE * VNAM1(I)		

```
VADD vnam1 vnam2 vnam3
```

VNAM1	First source vector name	C
VNAM2	Second source vector name	C
VNAM3	Destination vector name	C
VNAM3(I) = VNAM1(I) + VNAM2(I)		

VMULTIPLY vnam1 vnam2 vnam3VNAM1 First source vector name C VNAM2 Second source vector name C VNAM3 Destination vector name C

VNAM3(I) = VNAM1(I) * VNAM2(I)

VSUBTRACT vnam1 vnam2 vnam3VNAM1 First source vector name C VNAM2 Second source vector name C VNAM3 Destination vector name C

VNAM3(I) = VNAM1(I) - VNAM2(I)

VDIVIDE vnam1 vnam2 vnam3VNAM1 First source vector name C VNAM2 Second source vector name C VNAM3 Destination vector name C

VNAM3(I) = VNAM1(I) / VNAM2(I) (or 0 if VNAM2(I)=0)

Chapter 13: HISTOGRAM

Manipulation of histograms, Ntuples. Interface to the HBOOK package.

FILE lun fname [lrecl chopt]

LUN Logical unit number *I* R=1:128

FNAME File name *C*

LRECL Record length in words *I* D=1024

CHOPT Options *C* D=' ' R=' ,N,U'

Open an HBOOK direct access file.

For CHOPT=' ', existing file is opened (read mode only).

For CHOPT='N', a new file is opened.

For CHOPT='U', existing file is opened to be modified.

LIST [chopt]

CHOPT Options *C* D=' ' R=' ,I'

List histograms and Ntuples in the current directory. If CHOPT='I' a verbose format is used (HINDEX).

DELETE id

ID Histogram Identifier *C*

Delete histogram/Ntuple ID in Current Directory (memory). If ID=0 delete all histograms and Ntuples.

To delete histograms in disk files use command HIO/HSCRATCH.

PLOT [id chopt]

ID Histogram Identifier *C*

CHOPT Options *C* D=' ' R=' ,C,S,+,-,B,L,P,*,K,U,E,A'

Plot a single histogram or a 2-Dim projection. Each plotted histogram will start either a new picture or a new zone in the current picture. A channel range may be specified for 1-Dim and 2-Dim histograms.

Ex: Histo/plot 10(25:64) or Histo/plot 20(4:18,5:10). CHOPT may be a combination of the following characters:

- 'C' Draw a smooth curve.
- 'S' Superimpose plot on top of existing picture.
- '+' Add contents of ID to last plotted histogram.
- '-' Subtract contents of ID to last plotted histogram.
- '+-' Draw the delta with the last plotted histogram.
- 'B' Select Bar chart format.
- 'L' Connect channels contents by a line.
- 'P' Draw the current polymarker at each channel or cell.
- '*' Draw a * at each channel.
- 'K' Must be given if option 'U' is given later.
- 'U' Update channels modified since last call.
- 'E' Draw error bars and current marker.

```
'A'      Axis labels and tick marks are not drawn.
'BOX'    Draw 2-Dim with proportional boxes.
'COL'    Draw 2-Dim with a color table.
'Z'      Used with COL or SURF, it draws the color map.
'SURF'   Draw as a surface plot (angles are set via the command angle).
'SURF1'  Draw as a surface with color levels
'SURF2'  Same as SURF1 but without cell lines.
'SURF3'  Same as SURF but with the contour plot (in color) on top.
'SURF4'  Draw as a surface with Gouraud shading.
'LEGO'   Draw as a lego plot (angles are set via the command angle).
'LEGO1'  Draw lego plot with light simulation.
'LEGO2'  Draw lego plot with color levels.
'CONT'   Draw 2-Dim as a contour plot (15 levels).
'TEXT'   Draw 2-Dim as a table.
'CHAR'   Draw 2-Dim with characters (a la HBOOK).
'HIST'   Draw only histogram (no errors or associated function).
'FUNC'   Draw only the associated function (not the histogram).
```

1 Dim histograms could be plotted with option LEGO or SURF. In this case the angles are THETA=1 and PHI=-1. When option 'E' is used, the marker type can be changed with SMK, the marker size with SET KSIZ, the marker color with SPMCI. To plot projection X of ID type

HI/PLOT ID.PROX

To plot band 1 in Y of ID type

HI/PLOT ID.BANY.1

To plot slice 3 in Y of ID type

HI/PLOT ID.SLIY.3

ZOOM [id chopt icmin icmax]

ID	Histogram Identifier	<i>C</i>
CHOPT	Options	<i>C D=' ' R=' ,C,S,+ ,B,L,P,*'</i>
ICMIN	First channel	<i>I D=1</i>
ICMAX	Last channel	<i>I D=9999</i>

Plot a single histogram between channels ICMIN and ICMAX. Each plotted histogram will start either a new picture or a new zone in the current picture. If no parameters are given to the command, then the system waits for two points using the graphics cursor. To quit ZOOM, click the right button of the mouse or CRTL/E. CHOPT may be a combination of the following characters:

- 'C' Draw a smooth curve.
- 'S' Superimpose plot on top of existing picture.
- '+' Add contents of ID to last plotted histogram.
- 'B' Select Bar chart format.
- 'L' Connect channels contents by a line.
- 'P' Draw the current polymarker at each channel.
- '*' Draw a * at each channel.

MANY_PLOTS idlist

IDLIST List of histogram Identifiers *C*

Plot one or several histograms into the same plot. Plotted histograms are superimposed on the same zone of the picture.

PROJECT id

ID Histogram Identifier *C*

Fill all booked projections of a 2-Dim histogram. Filling is done using the 2-D contents of ID.

COPY id1 id2 [title]

ID1 First histogram Identifier *C*

ID2 Second histogram Identifier *C*

TITLE New title *C D=' '*

Copy a histogram (not Ntuple) onto another one. Bin definition, contents, errors, etc. are preserved. If TITLE is not given, ID2 has the same title as ID1.

FIT id func [chopt np par step pmin pmax errpar]

ID Histogram Identifier *C*

FUNC Function name *C D=' '*

CHOPT Options *C D=' ' R=' ,0,N,Q,V,B,L,D,W,M,E'*

NP Number of parameters *I D=0 R=0:34*

PAR Vector of parameters *C*

STEP Vector of steps size *C*

PMIN Vector of lower bounds *C*

PMAX Vector of upper bounds *C*

ERRPAR Vector of errors on parameters *C*

Fit a user defined (and parameter dependent) function to a histogram ID (1-Dim or 2-Dim) in the specified range. FUNC may be:

A- The name of a file which contains the user defined function to be minimized. Function name and file name must be the same. For example file FUNC.FOR is:

```
FUNCTION FUNC(X)  or FUNC(X,Y) for a 2-Dim histogram
COMMON/PAWPAR/PAR(2)
FUNC=PAR(1)*X +PAR(2)*EXP(-X)
END
```

Ex: His/fit 10 func.for ! 5 par

B- One of the following keywords (1-Dim only):

G : to fit Func=par(1)*exp(-0.5*((x-par(2))/par(3))**2)

E : to fit Func=exp(par(1)+par(2)*x)

Pn: to fit Func=par(1)+par(2)*x+par(3)*x**2.....+par(n+1)*x**n

Ex: His/fit 10 g

C- A combination of the keywords in B with the 2 operators + or *.

Ex: His/Fit 10 p4+g ! 8 par

His/Fit 10 p2*g+g ! 9 par

Note that in this case, the order of parameters in PAR must correspond to the order of the basic functions.

For example, in the first case above, par(1:5) apply to the polynomial of degree 4 and par(6:8) to the gaussian while in the second case par(1:3) apply to the polynomial of degree 2, par(4:6) to the first gaussian and par(7:9) to the second gaussian..

Blanks are not allowed in the expression.

For cases A and C, before the execution of this command, the vector PAR must be filled (via Vector/Input) with the initial values. For case B, if NP is set to 0, then the initial values of PAR will be calculated automatically. After the fit, the vector PAR contains the new values of parameters. If the vector ERRPAR is given, it will contain the errors on the fitted parameters. A bin range may be specified with ID.

Ex. Histo/Fit 10(25:56).

CHOPT : Possible options '0NQVBLDW' + HPLOT options

- '0' Do not plot the result of the fit. By default the fitted function is drawn unless the option 'N' below is specified.
- 'N' Do not store the result of the fit bin by bin with the histogram. By default the function is calculated at the middle of each bin and the fit results stored with the histogram data structure.
- 'Q' Quiet mode. No print
- 'V' Verbose mode. Results after each iteration are printed
By default only final results are printed.
- 'B' Some or all parameters are bounded. The vectors STEP,PMIN,PMAX must be specified.
Default is: All parameters vary freely.
- 'L' Use Log Likelihood.
Default is chisquare method.
- 'D' The user is assumed to compute derivatives analytically using the routine HDERIV.
By default, derivatives are computed numerically.
- 'W' Sets weights equal to 1. Default weights taken from the square root of the contents or from HPAKE/HBARX (PUT/ERRORS).
- 'M' The interactive Minuit is invoked.
- 'E' Performs a better Error evaluation (MIGRAD + HESSE + MINOS).

13.1 2D_PLOT

Plotting of 2-Dim histograms in various formats.

LEGO [id theta phi chopt]

ID	Histogram Identifier	C
THETA	Angle THETA in degrees	R D=30.
PHI	Angle PHI in degrees	R D=30.
CHOPT	Options	C D=' ' R=' ,1,2'

Draw a lego plot from 2-Dim or 1-Dim histograms. By default (CHOPT=' ') a hidden line algorithm is used. CHOPT='1' A hidden surface algorithm is used. The colour of the lego

is given by SET HCOL CI where CI is a colour index. For the top and the sides of the lego the same hue is used but with a different light.

CHOPT='2' A hidden surface algorithm is used. The colour of each bar

changes according to the value of Z. It is possible to change the set of colours used with SET HCOL c.L where L define a palette of colours given by the command ATT/PALETTE.

It is also possible to produce stacked lego plots. A stacked lego plot consists of a superimposition of several histograms, whose identifiers are given in the command LEGO separated by the character "+".

PAW > LEGO ID1+ID2+ID3 | Maximum number of ID's is 10. The colours of each IDn is given by the command ATT/PALETTE

Examples:

```
PAW > SET HCOL 2           | The colour the histogram is 2 (red)
PAW > LEGO 20               | Display a lego with lines
PAW > LEGO 20 !! 1          | Display a lego with different lights
PAW > LEGO 20 !! 2          | Display a lego with colours

PAW > PALETTE 1 3 2 3 4   | Create the palette number 1 with 3 elements: 2,3
PAW > SET HCOL 0.1          | The subsequent stack lego plots will use list 1
PAW > LEGO 10+20+30        | Plot a stack of lego plots with lines
PAW > LEGO 10+20+30 !! 1    | Plot a stack of lego plots with light
```

Notes: - The commands OPTION BAR, SET BARW and SET BARO act on lego plots

- The options 1 and 2 must be used only on selective erase devices.

SURFACE [id theta phi chopt]

ID	Histogram Identifier	C
THETA	Angle THETA in degrees	R D=30.
PHI	Angle PHI in degrees	R D=30.
CHOPT	Options	C D=' ' R=' ,1,2,3,4'

Draw a surface plot from 2-Dim or 1-Dim histograms. By default (CHOPT=' ') a hidden line algorithm is used. CHOPT='1' A hidden surface algorithm is used and each cell is filled

with a colour corresponding to the Z value (or grey scale with PostScript). It is possible to change the set of colours used with SET HCOL ic.L where L define a palette of colours given by the command ATT/PALETTE.

CHOPT='2' Is similar to option '1' except that the cell lines

are not drawn. This is very useful to draw contour plots with colours if THETA=90 and PHI=0.

CHOPT='3' A surface is drawn with a contour plot in color on top. The

contour plot is drawn with the colors defined with the command PALETTE.

CHOPT='4' A surface is drawn with Gouraud shading. With this command it is possible to draw color contour plots:

```
PAW > ATT/PAL 1 3 2 3 4    | Define the palette 1 with 3 elements
PAW > SET HCOL 0.1          | Set the list 1 as colours for histograms
PAW > SET NDVZ 4            | Set the number of Z divisions to 4
PAW > SURF id 90 0 2       | Draw the contour
```

Note: - The options 1 to 4 must be used only on selective erase devices.

CONTOUR [id nlevel chopt param]

ID	Histogram Identifier	C
NLEVEL	Number of contour lines	I D=10
CHOPT	Options	C D=' ' R=' ,0,1,2,S'
PARAM	Vector of contour levels	C

Draw a contour plot from a 2-Dim histogram. CHOPT may be a combination of the following characters:

- '0' use colour to distinguish contours.
- '1' use line style to distinguish contours.
- '2' line style and colour are the same for all contours.
- '3' The contour is drawn with filled colour levels. The levels are equidistant. The color indices are taken in the current palette (defined with the command PALETTE). If the number of levels (NLEVEL) is greater than the number of entries in the current palette, the palette is explored again from the beginning in order to reach NLEVEL.
- 'S' Superimpose plot on top of existing picture.

If PARAM is not given, contour levels are equidistant. If given, the vector PARAM may contain up to 50 values.

13.2 CREATE

Creation ("booking") of HBOOK objects in memory.

1DHISTO id title ncx xmin xmax [valmax]

ID Histogram Identifier *C*
 TITLE Histogram title *C* D=’ ’
 NCX Number of channels *I* D=100
 XMIN Low edge *R* D=0.
 XMAX Upper edge *R* D=100.
 VALMAX Maximum bin content *R* D=0.

Create a one dimensional histogram. The contents are set to zero. If VALMAX=0, then a full word is allocated per channel, else VALMAX is used as the maximum bin content allowing several channels to be stored into the same machine word.

PROFILE id title ncx xmin xmax ymin ymax [chopt]

ID Histogram Identifier *C*
 TITLE Histogram title *C* D=’ ’
 NCX Number of channels *I* D=100
 XMIN Low edge in X *R* D=0.
 XMAX Upper edge in X *R* D=100.
 YMIN Low edge in Y *R* D=-1.E30
 YMAX Upper edge in Y *R* D=1.E30
 CHOPT Options *C* D=’ ’ R=’ ,S’

Create a profile histogram. Profile histograms accumulate statistical quantities of a variable y in bins of a variable x. The contents are set to zero.

BINS id title ncx xbins [valmax]

ID Histogram Identifier *C*
 TITLE Histogram title *C* D=’ ’
 NCX Number of channels *I* D=100
 XBINS Vector of NCX+1 low-edges *C*
 VALMAX Maximum bin content *R* D=0.

Create a histogram with variable size bins. The low-edge of each bin is given in vector XBINS (NCX+1) values. The contents are set to zero. See 1DHISTO for VALMAX.

2DHISTO id title ncx xmin xmax ncy ymin ymax [valmax]

ID Histogram Identifier *C*
 TITLE Histogram title *C* D=’ ’
 NCX Number of channels in X *I* D=40
 XMIN Low edge in X *R* D=0.
 XMAX Upper edge in X *R* D=40.
 NCY Number of channels in Y *I* D=40
 YMIN Low edge in Y *R* D=0.
 YMAX Upper edge in Y *R* D=40.
 VALMAX Maximum bin content *R* D=0.

Create a two dimensional histogram. The contents are set to zero. See 1DHISTO for VALMAX.

PROX id

ID Histogram (2-Dim) Identifier *C*

Create the projection onto the x axis. The projection is not filled until the Histo/Project command is executed.

PROY id

ID Histogram (2-Dim) Identifier *C*

Create the projection onto the y axis. The projection may be filled with Histo/Project.

SLIX id nslices

ID Histogram (2-Dim) Identifier *C*

NSLICES Number of slices *I*

Create projections onto the x axis, in y-slices. The projection may be filled with Histo/Project.

SLIY id nslices

ID Histogram (2-Dim) Identifier *C*

NSLICES Number of slices *I*

Create projections onto the y axis, in x-slices. The projection may be filled with Histo/Project.

BANX id ymin ymax

ID Histogram (2-Dim) Identifier *C*

YMIN Low edge in Y *R*

YMAX Upper edge in Y *R*

Create a projection onto the x axis, in a band of y. The projection may be filled with Histo/Project.

BANY id xmin xmax

ID Histogram (2-Dim) Identifier *C*

XMIN Low edge in X *R*

XMAX Upper edge in X *R*

Create a projection onto the y axis, in a band of x. The projection may be filled with Histo/Project.

TITLE_GLOBAL [chtitl chopt]

CHTITL Global title *C D=' '*

CHOPT Options *C D=' ' R=' ,U'*

Set the global title. The global title is plotted at the top of each picture if the CHOPT=''. If CHOPT='U' and if the option 'UTIT' is on, a user title is plotted at the bottom of each histogram. The size and the Y position of the global title may be changed by the commands SET GSIZ and SET YGTI respectively. The size and the Y position of the user title may be changed by the commands SET TSIZ and SET YHTI respectively.

13.3 HIO

Input/Output operations of histograms.

HRIN id [icycle iofset]

ID Histogram Identifier *C*
 ICYCLE Cycle number *I* D=999
 IOFSET Offset *I* D=0

Read histogram/Ntuple ID from the current directory on direct access file to memory. An identical histogram is created but with an ID equal to that of the original histogram plus the offset IOFSET. Identifier may be '0' or '*' (for all histograms). If ICYCLE > 1000 and ID=0 read all histograms in all subdirectories as well. If IOFSET = 99999 then the contents of histogram ID on the disk file are added to the current histogram in memory if it exists. For example to add all histograms from FILE1 and FILE2 in memory, the sequence of commands can be:

```
PAW > Histo/File 1 FILE1
PAW > Hrin 0
PAW > Histo/File 2 FILE2
PAW > Hrin 0 ! 99999
```

HROUT id [chopt]

ID Histogram Identifier *C*
 CHOPT Options *C* D=' ' , R=' ', T'

Write histo/Ntuple ID from memory to current directory. Identifier may be '0' or '*' (for all histograms). If CHOPT='T' writes all histograms in subdirectories as well.

HSCRATCH id

ID Histogram Identifier *C*

Delete histogram ID in Current Directory on disk. If ID='0' or '*' delete all histograms. To delete histograms in memory use command HISTO/DELETE.

HFETCH id fname

ID Histogram Identifier *C*
 FNAME File name *C*

Fetch histogram ID from file FNAME. FNAME has been created by the old version of HBOOK3 (Unformatted).

HREAD id fname

ID Histogram Identifier *C*
 FNAME File name *C*

Read histogram ID from file FNAME. FNAME has been created by the old version of HBOOK3 (Formatted).

PRINT id [chopt]

ID Histogram Identifier *C*
 CHOPT Options *C D=' ' R=' ,S'*

Print histograms (line-printer format) on screen. The command OUTPUT_LP may be used to change the output file. If CHOPT='S', then only statistics (Number of entries, mean, RMS, underflow, overflow) are printed.

DUMP id

ID Histogram Identifier *C*
 Dump the histogram ZEBRA data structure on the terminal.

OUTPUT_LP [lun fname]

LUN Logical unit number *I D=6*
 FNAME File name *C D=' '*

Change the HBOOK "line printer" file name. If FNAME=' ' then OUTPUT is appended to an already opened file on unit LUN. If LUN is negative, the file is closed and subsequent output is directed to unit 6.

GLOBAL_SECT gname

GNAME Global section name *C D=' '*

Map the global section GNAME (VAX only). The current directory is changed to //GNAME.

GRESET id

ID Histogram Identifier *C*
 Reset histogram ID in the global section.

13.4 OPERATIONS

Histogram operations and comparisons.

ADD id1 id2 id3 [c1 c2]

ID1 First histogram Identifier *C*
 ID2 Second histogram Identifier *C*
 ID3 Result histogram Identifier *C*
 C1 Scale factor for ID1 *R D=1.*
 C2 Scale factor for ID2 *R D=1.*

Add histograms: ID3 = C1*ID1 + C2*ID2. Applicable to 1-Dim and 2-Dim histograms. See command HRIN to add histograms with same IDS from different files.

SUBTRACT id1 id2 id3 [c1 c2]

ID1 First histogram Identifier *C*
 ID2 Second histogram Identifier *C*
 ID3 Result histogram Identifier *C*
 C1 Scale factor for ID1 *R D=1.*
 C2 Scale factor for ID2 *R D=1.*

Subtract histograms: $ID3 = C1*ID1 - C2*ID2$. Applicable to 1-Dim and 2-Dim histograms.

MULTIPLY id1 id2 id3 [c1 c2]

ID1 First histogram Identifier *C*
 ID2 Second histogram Identifier *C*
 ID3 Result histogram Identifier *C*
 C1 Scale factor for ID1 *R D=1.*
 C2 Scale factor for ID2 *R D=1.*

Multiply histogram contents: ID3 = C1*ID1 * C2*ID2. Applicable to 1-Dim and 2-Dim histograms.

DIVIDE id1 id2 id3 [c1 c2]

ID1 First histogram Identifier *C*
 ID2 Second histogram Identifier *C*
 ID3 Result histogram Identifier *C*
 C1 Scale factor for ID1 *R D=1.*
 C2 Scale factor for ID2 *R D=1.*

Divide histograms: ID3 = C1*ID1 / C2*ID2. Applicable to 1-Dim and 2-Dim histograms.

RESET id [title]

ID Histogram Identifier *C*
 TITLE New title *C D=' '*
 Reset contents and errors of an histogram. Bin definition is not modified.

DIFF id1 id2 [chopt]

ID1 First Histogram Identifier *C*
 ID2 Second Histogram Identifier *C*
 CHOPT Options *C D='D' R='D,N,,0,U,L,R,B,T'*

Test of compatibility for two 1-Dim histograms ID1 and ID2. A probability PROB is calculated as a number between zero and one, where PROB near one indicates very similar histograms, and PROB near zero means that it is very unlikely that the two arose from the same parent distribution. For two histograms sampled randomly from the same distribution, PROB will be (approximately) uniformly distributed between 0 and 1. See discussion in HBOOK manual under "HDIFF- Statistical Considerations". By default (if no options are selected with CHOPT) the comparison is done only on the shape of the two histograms, without consideration of the difference in numbers of events, and ignoring all underflow and overflow bins. The string CHOPT allows specification of the following options:

N Include also comparison of the relative normalization of the two histograms, in addition to comparing the shapes.
 PROB is then a combined confidence level taking account of absolute contents.
 D Debug printout, produces a blank line and two lines of information at each call, including the ID numbers, the number of events in each histogram, the PROB value, and the maximum Kolmogorov distance between the two histograms.
 For 2-Dim histograms, there are two Kolmogorov distances

(see below). If 'N' is specified, there is a third line of output giving the PROB for shape alone, and for normalization.
 0 Overflow, requests that overflow bins be taken into account.
 U Underflow, requests that underflow bins be taken into account.

SMOOTH id [isel]

ID Histogram Identifier C
 ISEL Option flag I D=2
 Smooth histogram ID using the 353QH algorithm.

ISEL = 0,1 replace original histogram by smoothed.
 = 2 superimpose result of smoothing as a function when editing.

SPLINE id [isel knotx kx]

ID Histogram Identifier C
 ISEL Option flag I D=2
 KNOTX Number of knots I D=10
 KX Degree of the spline I D=3

Smooth 1-Dim or 2-Dim histogram ID using B-splines. If ID is a 1-Dim histogram then:

ISEL = 0,1 replace original histogram by smoothed.
 = 2 superimpose as a function when editing.

If ID is a 2-Dim histogram then original contents are replaced.

PARAM id [isel r2min maxpow]

ID Histogram Identifier C
 ISEL Control word I D=11
 R2MIN Min correlation coefficient R D=1.
 MAXPOW Max degree of polynomials I D=5 R=1:20

Perform a regression on contents of the 1-Dim histogram ID. Find the best parameterization in terms of elementary functions (regressors). See HBOOK guide HPARAM. Control word ISEL=1000*T+100*W+10*S+P

S = 1 resulting parametric fit superimposed on histogram
 0 no superposition
 P = 0 minimal output: the residual sum of squares is printed
 1 normal output: in addition, the problem characteristics and options are printed; also the standard deviations and confidence intervals of the coefficients.
 2 extensive output: the results of each iteration are printed with the normal output.
 W = 0 weights on histogram contents are already defined via HBARX or HPAKE. If not they are taken to be equal to the

```

    square-root of the contents.
1 weights are equal to 1.
T = 0 monomials will be selected as the elementary functions
1 Chebyshev polynomials with a definition region: [-1,1]
2 Legendre polynomials with a definition region: [-1,1]
3 shifted Chebyshev polynomials with a definition region: [0,1]
4 Laguerre polynomials with a definition region: [0,+infinite]
5 Hermite polynomials with a definition region: [-inf,+inf]
```

The FORTRAN code of the parameterization is written onto the file FPARAM.DAT.

HSETPR param value

PARAM Parameter name C D='FEPS'

VALUE Parameter value R D=0.001

Set various parameters for command PARAM.

13.5 GET_VECT

Fill a vector from values stored in HBOOK objects.

CONTENTS id vname

ID Histogram Identifier C

VNAME Vector name C

Get contents of histogram ID into vector VNAME.

ERRORS id vname

ID Histogram Identifier C

VNAME Vector name C

Get errors of histogram ID into vector VNAME.

FUNCTION id vname

ID Histogram Identifier C

VNAME Vector name C

Get function associated to histogram ID into vector VNAME.

ABSCISSA id vname

ID Histogram Identifier C

VNAME Vector name C

Get values of center of bins abscissa into vector VNAME.

REBIN id x y ex ey [n ifirst ilast]

ID	Histogram Identifier	<i>C</i>
X	Name of vector X	<i>C</i>
Y	Name of vector Y	<i>C</i>
EX	Name of vector EX	<i>C</i>
EY	Name of vector EY	<i>C</i>
N	Number of elements to fill	<i>I D=100</i>
IFIRST	First bin	<i>I D=1</i>
ILAST	Last bin	<i>I D=100</i>

Get contents and errors into vectors, grouping bins. Bin width and centers are also extracted. Allow to combine 2, 3 or more bins into one.

E.g.: REBIN 110 X Y EX EY 25 11 85
 will group by 3 channels 11 to 85 and return
 new abscissa, contents and errors.
 Errors in X are equal to 1.5*BINWIDTH.

N.B.:

REBIN ID X Y EX EY is a convenient way to return in
 one call abscissa, contents and errors for 1-Dim histogram.
 In this case the errors in X are equal to 0.5*BINWIDTH.

13.6 PUT_VECT

Replace histogram contents with values in a vector.

CONTENTS id vname

ID	Histogram Identifier	<i>C</i>
VNAME	Vector name	<i>C</i>

Replace contents of histogram with values of vector VNAME.

ERRORS id vname

ID	Histogram Identifier	<i>C</i>
VNAME	Vector name	<i>C</i>

Replace errors of histogram with values of vector VNAME.

13.7 SET

Set histogram attributes.

MAXIMUM id vmax

ID	Histogram Identifier	<i>C</i>
VMAX	Maximum value	<i>R</i>

Set the maximum value on the Y axis. To select again an automatic scale, just set VMAX less then the minimum.

MINIMUM id vminID Histogram Identifier *C*VMIN Minimum value *R*

Set the minimum value on the Y axis. To select again an automatic scale, just set VMIN greater than the maximum.

NORMALIZE_FACTOR id [xnorm]ID Histogram Identifier *C*XNORM Normalization factor *R D=1*

Set the contents/errors normalization factor. Only valid for histograms (1-Dim). (does not change contents, only presentation).

SCALE_FACTOR_2D id [xscale]ID Histogram Identifier *C*XSCALE Scale factor *R D=0*

Set the scale factor for histograms (2-Dim).

IDOPT id optionID Histogram Identifier *C*OPTION Options *C*

Set options for histogram ID. (* means default).

SETD* Set all options to the default values
 SHOW Print all the options currently set
 BLAC 1 Dim histogram printed with X characters
 CONT* 1 Dim histogram is printed with the contour option
 STAR 1 Dim histogram is printed with a * at the Y value
 SCAT* Print a 2 Dim histogram as a scatter-plot
 TABL Print a 2 Dim histogram as a table
 PROS* Plot errors as the Spread of each bin in Y for
 profile histograms
 PROE Plot errors as the mean of each bin in Y for
 profile histograms
 STAT Mean value and RMS computed at filling time
 NSTA* Mean value and RMS computed from bin contents only
 ERRO Errors bars printed as SQRT(contents)
 NERR* Do not print error bars
 INTE Print the values of integrated contents bin by bin
 NINT* Do not print integrated contents
 LOGY 1 Dim histogram is printed in Log scale in Y
 LINY* 1 Dim histogram is printed in linear scale in Y
 PCHA* Print channel numbers
 NPCH Do not print channel numbers
 PCON* Print bin contents

```
NPC0  Do not print bin contents
PLOW* Print values of low edge of the bins
NPL0  Do not print the low edge
PERR  Print the values of the errors for each bin
NPER* Do not print the values of the errors
PFUN  Print the values of the associated function bin by bin
NPFU* Do not print the values of the associated function
PHIS* Print the histogram profile
NPHI  Do not print the histogram profile
PSTA* Print the values of statistics (entries,mean,RMS,etc.)
NPST  Do not print values of statistics
ROTA  Print histogram rotated by 90 degrees
NROT* Print histogram vertically
1EVL  Force an integer value for the steps in the Y axis
AEVL* Steps for the Y axis are automatically computed
2PAG  Histogram is printed over two pages
1PAG* Histogram is printed in one single page
AUTO* Automatic scaling
```

Chapter 14: FUNCTION

Operations with Functions. Creation and plotting.

```
FUN1 id ufunc ncx xmin xmax [ chopt ]
```

ID	Histogram Identifier	C
UFUNC	Name of the function	C
NCX	Number of channels	I D=100 R=1:
XMIN	Low edge	R D=0.
XMAX	Upper edge	R D=100.
CHOPT	Options	C D='P'

Create a one dimensional histogram and fill the bins with the values of a (single-valued) function. The function UFUNC may be given in two ways:

-An expression of the variable x in case of a simple function.

Ex: FUN1 10 sin(x)/x 100 0 10

-UFUNC is the name of a COMIS function in a text file with the name UFUNC.FTN or UFUNC.FOR or UFUNC FORTRAN (Apollo, VAX, IBM).

If CHOPT='P' the function is drawn.

```
FUN2 id ufunc ncx xmin xmax ncy ymin ymax [ chopt ]
```

ID	Histogram (2-Dim) Identifier	C
UFUNC	Name of the function	C
NCX	Number of channels in X	I D=40 R=1:
XMIN	Low edge in X	R D=0.
XMAX	Upper edge in X	R D=40.
NCY	Number of channels in Y	I D=40 R=1:
YMIN	Low edge in Y	R D=0.
YMAX	Upper edge in Y	R D=40.
CHOPT	Options	C D='S' R='S, ,L,C'

Create a two dimensional histogram and fill the bins with the values of a (two-valued) function. The function UFUNC may be given in two ways:

-An expression of the variables x and y in case of a simple function.

Ex: FUN2 10 abs(sin(x**2+y**2)) 40 -2 2 40 -2 2 C

-UFUNC is the name of a COMIS function in a text file with the name UFUNC.FTN or UFUNC.FOR or UFUNC FORTRAN (Apollo, VAX, IBM).

If CHOPT='S' the function is drawn as a surface. If CHOPT='L' the function is drawn as a lego plot. If CHOPT='C' the function is drawn as a contour plot.

DRAW ufunc [chopt]UFUNC Name of function *C*CHOPT Options *C D=' ' R=' '*

Draw the function UFUNC in the current ranges specified by the command: RANGE XLOW XUP YLOW YUP ZLOW ZUP and with THETHA and PHI angles specified by the command ANGLE THETA PHI. The number of points to evaluate the function between XLOW, XUP YLOW, YUP, and ZLOW, ZUP can be changed by the command POINTS NPX NPY NPZ.

The function UFUNC may be given in two ways: - As an expression of the variables X, Y, Z in the case of a simple function.

Ex:

```
PAW > FUN/DRAW X*Y*Z           | equivalent to :
PAW > FUN/DRAW X*Y*Z=0
PAW > FUN/DRAW X**2+Y**2+Z**2=1
PAW > FUN/DRAW X**2+Y**2=1-Z**2
```

- As a COMIS function in a text file with the name UFUNC.FTN or UFUNC.FOR or UFUNC FORTRAN (Apollo, VAX, IBM).

Ex:

The file FTEST.FOR contains:

```
FUNCTION FTEST(X,Y,Z)
IF(X.LE.0..AND.Y.LE.0.)THEN
  FTEST=(X+0.5)**2+(Y+0.5)**2+(Z+0.5)**2-0.2
ELSE
  FTEST=(X-0.5)**2+(Y-0.5)**2+(Z-0.5)**2-0.1
ENDIF
END

PAW > RANGE -1 1 -1 1 -1 1 | Define the range as a cube between -1 1 in the 3
                               directions
PAW > POINTS 20 20 20      | FUN/DRAW will use 20 points in the 3 directions
PAW > FUN/DRAW FTEST.FOR   | Draw 2 spheres centered on (-0.5,-0.5,-0.5)
                               and (0.5,0.5,0.5) with the radius SQRT(0.2)
                               and SQRT(0.1)
```

PLOT ufunc xlow xup [chopt]UFUNC Name of function *C*XLOW Lower limit *R*XUP Upper limit *R*CHOPT Options *C D=' ' R=' ',C,S,+,L,P,*'*

Plot single-valued function UFUNC between XLOW and XUP. The function UFUNC may be given in two ways:

-An expression of the variable x in case of a simple function.

Ex: FUN/PLOT sin(x)/x 0 10

-UFUNC is the name of a COMIS function in a text file with the name UFUNC.FTN or UFUNC.FOR or UFUNC FORTRAN (Apollo, VAX, IBM). For example, if the file FTEST.FOR contains:

```
FUNCTION FTEST(X)
FTEST=SIN(X)*EXP(-0.1*X)
END
```

Then, FUN/PLOT FTEST.FOR 0 10, will interpret the Fortran code in the file FTEST.FOR and draw the function for x between 0 and 10.

The number of points to evaluate the function between XLOW and XUP can be changed by the command /FUN/POINTS. Only 1-Dim functions are supported. For 2-Dim use FUN2. CHOPT may be a combination of the following characters:

- 'C' Draw a smooth curve (default if CHOPT not specified)
- 'S' Superimpose plot on top of existing picture.
- '+' Add contents of ID to last plotted histogram.
- 'L' Connect channel contents by a line.
- 'P' Draw the current polymarker at each channel.
- '*' Draw a * at each channel.

POINTS [npx npy npz]

NPX Number of points on X axis *I D=20 R=2:1000*

NPY Number of points on Y axis *I D=20 R=2:1000*

NPZ Number of points on Z axis *I D=20 R=2:1000*

Change the number of points to be used by FUN/DRAW and FUN/PLOT. Note that the default for NPX is 20 for 3-Dim plots (FUN/DRAW) but it is 100 for 1-Dim plots (FUN/PLOT).

RANGE [xlow xup ylow yup zlow zup]

XLOW X Lower limit *R D=-1.*

XUP X Upper limit *R D=1.*

YLOW Y Lower limit *R D=-1.*

YUP Y Upper limit *R D=1*

ZLOW Z Lower limit *R D=-1.*

ZUP Z Upper limit *R D=1.*

Change the range used by FUN/DRAW.

ANGLE [theta phi]

THETA Angle THETA in degrees *R D=30.*

PHI Angle PHI in degrees *R D=30.*

Change the angle used by FUN/DRAW and HISTO/PLOT.

Chapter 15: NTUPLE

Ntuple creation and related operations.

CREATE idn title nvar chrzpa nprime varlist

IDN	Ntuple Identifier	C
TITLE	Ntuple title	C D=' '
NVAR	Number of variables	I D=1 R=1:512
CHRZPA	RZ path	C D=' '
NPRIME	Primary allocation	I D=1000
VARLIST	Names of the NVAR variables	C

Create an Ntuple. The Ntuple may be created either purely in memory or possibly using an automatic overflow to an RZ file. Memory allocation works in the following way. If CHRZPA = ' ', then a bank of NPRIME words is created. When the space in this bank is exhausted at filling time, a new linear structure of length NPRIME is created and this process will be repeated should the structure become exhausted. If CHRZPA contains the top directory name of an already existing RZ file (as declared with HISTO/FILE), then a bank of length NPRIME is also created, but at filling time, this bank is moved to the RZ file when full, and then it is overwritten by any new entries. The Ntuple can be filled by calling HFN from an interactively defined subroutine called by the command NTUPLE/LOOP or by NTUPLE/READ. The number of variables per data point is given in the parameter NVAR.

LIST

List all Ntuples in the Current Directory. Note that the command HISTO/LIST lists all histograms and Ntuples in the Current Directory.

PRINT idn

IDN Ntuple Identifier C

Print a summary about Ntuple IDN. Number of entries, variables names and limits are listed.

SCAN idn [chfunc nevent ifirst nvars varlis]

IDN	Ntuple Identifier	C
CHFUNC	User cut function	C D='0'
NEVENT	Number of events	I D=999999
IFIRST	First event	I D=1
NVARS	Number of variables to scan	I D=8 R=0:8
VARLIS	Names of the NVARS variables to scan	C

Scan the entries of an Ntuple subject to user cuts. Scan the variables for NEVENT events starting at IFIRST, requiring that the events satisfy cut CHFUNC. Up to 8 variables may be scanned, the default is to scan the first 8 variables. VARLIS may contain a list of the original variables or/and expressions of the original variables. For example, if IDN=30 has the 3 variables X,Y,Z, one can do:

```
PAW > scan 30
PAW > scan 30 z>10
PAW > scan 30 z>10 ! ! 5 z abs(x) y+z x func.for
```

where func.for is a COMIS function returning an expression of the original variables. This function func.for may be generated automatically by the PAW command:

PAW > uwfunc 30 func.for

LOOP idn uwfunc [nevent ifirst]

IDN	Identifier of Ntuple	<i>C</i>
UWFUNC	Selection function or cut number	<i>C D=' '</i>
NEVENT	Number of events	<i>I D=999999</i>
IFIRST	First event	<i>I D=1</i>

Invoke the selection function UWFUNC for each event starting at event IFIRST. In UWFUNC, the user can fill one or several histograms previously booked. The loop will be terminated if UWFUNC returns a negative value. For more information about UWFUNC, see command NTUPLE/PLOT.

MERGE idn1 idn2 [uwfunc nevent ifirst]

IDN1	Identifier of first Ntuple	<i>C</i>
IDN2	Identifier of second Ntuple	<i>C</i>
UWFUNC	Selection function or cut number	<i>C D=' '</i>
NEVENT	Number of events	<i>I D=999999</i>
IFIRST	First event	<i>I D=1</i>

Merge two Ntuples. Invoke the selection function UWFUNC for each of the NEVENT events starting at event IFIRST of Ntuple IDN1. Suppose you have 4 files containing Ntuple ID=10 and you want to merge the 4 files into the file 4, the sequence is:

```
PAW >Histo/file 1 file1
PAW >Histo/file 2 file2
PAW >Histo/file 3 file3
PAW >Histo/file 4 file4 1024 U
PAW >Ntuple/Merge //lun1/10 //lun4/10
PAW >Ntuple/Merge //lun2/10 //lun4/10
PAW >Ntuple/Merge //lun3/10 //lun4/10
PAW >Ntuple/plot 10.x .....
```

Only the events with UWFUNC>0 are appended to IDN2. IDN2 may be empty. Note that the Ntuple variables may be redefined inside UWFUNC. For more information about UWFUNC, see command NTUPLE/PLOT.

PROJECT idh idn [uwfunc nevent ifirst]

IDH	Identifier of histogram to fill	<i>C</i>
IDN	Identifier of Ntuple	<i>C</i>
UWFUNC	Selection function or cut number	<i>C D=' '</i>
NEVENT	Number of events	<i>I D=999999</i>
IFIRST	First event	<i>I D=1</i>

Project an Ntuple onto a 1-Dim or 2-Dim histogram, possibly using a selection function or predefined cuts. IDN may be given as IDN or IDN.X , IDN.Y%X , IDN.1, IDN.2%1. Y%X means variable Y of Ntuple IDN versus variable X. For more information about UWFUNC, see command NTUPLE/PLOT. The histogram IDH is not reset before filling. This allows several PROJECTs from different Ntuples.

```
READ idn fname [ format chopt nevent ]
```

IDN	Ntuple Identifier	C
FNAME	File name	C
FORMAT	Format	C D='*'
CHOPT	Options	C D=' '
NEVENT	Number of events	I D=1000000

Read Ntuple values from the alphanumeric file FNAME with the format specifications in FORMAT. Before executing this command, the Ntuple IDN must have been created with the command Ntuple/Create.

```
PLOT idn [ ufunc nevent ifirst nupd option ]
```

IDN	Ntuple Identifier	C
UWFUNC	Selection function	C D='0'
NEVENT	Number of events	I D=999999
IFIRST	First event	I D=1
NUPD	Frequency to update histogram	I D=1000000
OPTION	Options	C D=' ' R=' ,C,S,+,B,L,P,*,U,E,A'

Project and plot an Ntuple as a (1-Dim or 2-Dim) histogram with automatic binning (ID=1000000), possibly using a selection algorithm. See parameter CHOPT in command HISTO/PLOT for explanation of OPTION.

IDN may be given as IDN
 IDN.X
 IDN.Y%X
 IDN.1
 IDN.2%1
 IDN.expression1
 IDN.expression1%expression2

Y%X means a scatter-plot Y(I) versus X(I) where I is the event number. 2%1 means a scatter-plot variable 2 versus variable 1. In this example, X and Y are the names of the variables 1 and 2 respectively. Expression 1 is any numerical expression of the Ntuple variables. It may include a call to a COMIS function.

UWFUNC may have the following forms:

- 1- UWFUNC='0' or missing (only IDN given). No selection is applied.
- 2- UWFUNC is a CUT or combination of valid CUTS created by the command NTUPLE/CUTS. Ex:

UWFUNC=1	means use cut number 1
UWFUNC=1.AND.2	
UWFUNC=.NOT.(1.AND.2)	

```

UWFUNC=(1.OR.2).AND.3
3- UWFUNC is a FORTRAN expression
    Ex:    X>3.14.AND.(Y<Z+3.15)
4- UWFUNC is a variable name or an arithmetic expression
    Ex:    NT/PL0T 30.X Y  weight of each event is variable Y
           NT/PL0T 30.X X**2+Y**2
5- UWFUNC is the name of a selection function in a text file with
    the name UWFUNC.FTN, UWFUNC.FOR, UWFUNC FORTRAN (Apollo, VAX, IBM).

```

The command UWFUNC may be used to generate automatically this function. For example if IDN=30 is an Ntuple with 3 variables per event and 10000 events, then

```
NTUPLE/PL0T 30.X SELECT.FOR
```

will process the 10000 events of the Ntuple IDN=30. For each event, the function SELECT is called. It returns the weight of the event. Example:

```

FUNCTION SELECT(X)
DIMENSION X(3)
IF(X(1)**1+X(2)**2.LT.1.5)THEN
    SELECT=0.
ELSE
    SELECT=1.
ENDIF
END

```

The file SELECT.FOR (VAX), SELECT.FTN (Apollo) or SELECT FORTRAN (IBM) can be edited from PAW using the command EDIT. Note that if the suffix (.FTN, .FORTRAN or .FOR) is omitted, then COMIS will start from the precompiled version in memory and not from the file. Results of a selection can be saved in a MASK (See NTUPLE/MASK).

```

Ex: NT/PL0T 30.X Z<0.4>>MNAME(4)
means mark bit 4 in mask MNAME for all events satisfying
the condition Z<0.4

```

A MASK may also be given as input to a selection expression.

```

Ex: NT/PL0T 30.X MNAME(4).and.Z<0.4
means all events satisfying bit 4 of MNAME AND Z<0.4

```

It is possible to plot expressions of the original variables.

```

Ex 1: NT/PL0T 30.SIN(X)%SQRT(Y**2+Z**2) Z<0.4
plots a scatter-plot of variable U versus V for all events
satisfying the condition Z<0.4. U and V are defined as being
U=SIN(X) and V=SQRT(X**2+Y**2)

```

```

Ex 2: NT/PL0T 30.FUNC.FTN(X)%(SIN(Y)+3.) Z<0.2.and.TEST.FTN>6
plots a scatter-plot of variable U versus V for all events
satisfying the condition (Z<0.2 and the result of the COMIS
function TEST.FTN >6). U and V are defined as being
U=Result of the COMIS function FUNC.FTN, V=SIN(Y)+3.

```

The default identifier of the histogram being filled is IDF=1000000. At the next invocation of this command, it will be overwritten. If either NEVENT or IFIRST or NUPD are negative, then the identifier of the histogram being filled will be taken as IDF=-NEVENT or IDF=-IFIRST or IDF=-NUPD. IDF may have been created with H/CREATE. Before filling IDF, the contents of IDF are reset if IDF already exists. Use NTUPLE/PROJECT to cumulate several passes into IDF. Note that IDF not equal to 1000000 is a convenient way to force user binning. This option must be used when options '+', 'U', 'S' are specified in OPTION. Every NUPD events, the current status of the histogram is displayed.

```
CUTS icut [ option fname ]
```

ICUT	Cut number	<i>I</i>	R=0:100
OPTION	Options	<i>C</i>	D='P' R='P,G, ,S,-,R,W,D'
FNAME	File name	<i>C</i>	D=' '

Define cut number ICUT for an Ntuple. This cut can then be used in subsequent commands NTUPLE/PLOT, PROJECT.

```
OPTION='G' define a new cut ICUT using graphics input on the latest
      1-Dim or 2-Dim projection of the Ntuple.
      For a 1-Dim projection, give 2 points cutmin,cutmax.
      For a 2-Dim projection, give up to 20 points to delimit
      the selected area. The polygon will automatically
      be closed by PAW.

OPTION='P' Print definition of cut number ICUT.
      'S' same as P
      ''-'' Reset cut ICUT
      'R' read definition of cut ICUT from file FNAME.
      'W' write definition of cut ICUT on file FNAME (text file).
      'D' Draw cut contour.

OPTION='expression' Ex:    0.4<X<0.8.and.Y<SQRT(X)
```

Note that ICUT=0 means all cuts except for 'G' option. When option G is selected, graphical cuts are only operational for plots of the original Ntuple variables, not for expressions of these variables.

```
CSELECT [ chopt csize ]
```

CHOPT	Options	<i>C</i>	D='N' R='N, ,R,B,C'
CSIZE	Comment size	<i>R</i>	D=0.28

To write selection mechanism as a comment on the picture. If option N is given, then all subsequent NTUPLE/PLOT commands will print the selection mechanism with the options specified in CHOPT. By default, the comment is drawn left justified above the top zone line. The options are :

```
'R' comment is right adjusted to the current zone
'C' comment is centered to the current zone
'B' comment is drawn below the top zone line
```

Example:

```
CSEL          All coming NT/PLOT commands will draw a comment
```

```

        of size CSIZE=0.28cm Left justified.
CSEL NRB 0.4 All coming NT/PLOT commands will draw a comment
        of size 0.4 cm Right justified Below the top line.
CSEL CB      Draw previous selection mechanism Centered Below
        the top zone line.

```

MASK mname [chopt number]

```

MNAME   Mask name   C
CHOPT   Options     C  D=' ' R=' ',U,N,P,C,R'
NUMBER  Bit number  I  D=0

```

Perform Operations with masks. A mask is a direct-access file with the name MNAME.MASK. It must contain as many 32 bit words as there are events in the associated Ntuple. Masks are interesting when only a few events of a Ntuple are selected with a time consuming selection algorithm. For example if the command:

```
NT/PLOT 30.X Z<0.4.AND.SELECT.FTN>>MNAME(6)
```

then for all events in Ntuple 30 satisfying the condition above, the bit 6 in the corresponding mask words will be set. One can then use the mask as selection mechanism. Example:

```
NT/PLOT 30.X MNAME(6)
```

will produce the same results than the NT/PLOT command above, but will be much faster if only a small fraction of all the events is selected. MASKS are automatically saved across PAW sessions on files.

```

CHOPT=' ' Existing mask on file MNAME.MASK is attached for READ only.
CHOPT='U' Existing mask on file MNAME.MASK is attached for UPDATE.
CHOPT='N' A new mask on file MNAME.MASK is created for NUMBER events.
CHOPT='P' The comments for all active bits is printed.
CHOPT='C' Mask is closed.
CHOPT='R' Reset bit number NUMBER.If NUMBER=99, resets all bits.

```

Example:

```

MASK TEST N 10000
creates a new mask on file TEST.MASK with enough words to
process a Ntuple with 10000 events
MASK TEST UP
opens an existing mask for update and
prints the active selection bits with explanation

```

UWFUNC idn fname [chopt]

```

IDN    Ntuple Identifier  C
FNAME  File name         C
CHOPT  Options           C  D=' ' R=' ',E,P,T'

```

To generate the FORTRAN skeleton of a selection function. Example: If Ntuple ID=30 has variable names [X,Y,Z,ETOT,EMISS,etc] then:

NTUPLE/UWFUNC 30 SELECT.FOR will generate the file SELECT.FOR with:

```

FUNCTION SELECT(XDUMMY)
COMMON/PAWIDN/IDNEVT,VIDN1,VIDN2,VIDN3,X,Y,Z,ETOT,EMISS,etc
SELECT=1.
END

```

Then using the command EDIT one can modify this file which could then look something like (IDNEVT is the event number):

```

FUNCTION SELECT(XDUMMY)
COMMON/PAWIDN/IDNEVT,VIDN1,VIDN2,VIDN3,X,Y,Z,ETOT,EMISS,etc
IF(X**2+Y**2.GT.Z**2.OR.ETOT.GT.20.)THEN
    SELECT=1.
ELSE
    SELECT=0.
ENDIF
END

```

If in a subsequent command NTUPLE/PLOT, the selection function SELECT is used, then:

```

If NTUPLE/PLOT 30.ETOT SELECT.FOR
    VIDN1=ETOT
If NTUPLE/PLOT 30.SQRT(X**2+Y**2)%(ETOT-EMISS)
    VIDN1=ETOT-EMISS
    VIDN2=SQRT(X**2+Y**2)
If CHOPT='E' then the local editor is invoked on FNAME.
    ='P' code to print events is generated.
    ='T' Names of the Ntuple variables are generated in DATA.

```

LINTRA	idn [chopt nevent ifirst nvars varlis]
---------------	--

IDN	Ntuple Identifier	<i>C</i>
CHOPT	Options	<i>C D=' ' R=' ,N,P'</i>
NEVENT	Number of events	<i>I D=999999</i>
IFIRST	First event	<i>I D=1</i>
NVARS	Number of the most significant variables	<i>I D=20 R=0:20</i>
VARLIS	Names of the NVARS most significant variables	<i>C</i>

Data reduction on Ntuple. The method used is the PRINCIPAL COMPONENTS ANALYSIS. The Principal Components Analysis method consists in applying a linear transformation to the original variables of a ntuple. This transformation is described by an orthogonal matrix and is equivalent to a rotation of the original space to a new set of coordinates vectors, which hopefully provide easier identification and dimensionality reduction. This matrix is real positive definite and symmetric and has all its eigenvalues greater than zero. Among the family of all complete orthonormal bases, the basis formed by the eigenvectors of the covariance matrix and belonging to the largest eigenvalues corresponds to the most significant features for the description of the original ntuple. Reduction of the variables for NEVENT events starting at IFIRST The default is to take all the 20 first variables. CHOPT : Possible options 'NP'

```
'N' The variables are normalized
      This option is useful in the case the ranges of variables
      are very different
'P' Print more results about the analysis
```

This command creates a file : -> XTOXSI.FORTRAN or xtoksi.for,xtoksi.ftn. This file contains a Fortran function which computes the new variables. These new variables can be visualized in PAW with for example:

```
PAW > Ntuple/plot id.xtoksi.ftn(1)
PAW > Ntuple/plot id.xtoksi.ftn(1)%xtoksi.ftn(3)
```

Chapter 16: GRAPHICS

Interface to the graphics packages HPLOT and HIGZ.

SET [chatt value]

CHATT Attribute name *C* D='SHOW'
VALUE Attribute value *R* D=0

Set a specific HPLOT attribute. If CHATT='SHOW', print defaults and current values for all attributes. If CHATT='*', restore default values for all attributes. If VALUE=0, the attribute is set to its default value.

OPTION [choptn]

CHOPTN Option name *C* D='SHOW'

Set general plotting options for HPLOT. If CHOPTN='SHOW' print all current and default options. If CHOPTN='*', restore all default options.

METAFILE [lun metafl chmeta]

LUN Logical unit number *I* D=0
METAFL Metafile ID *I* D=0
CHMETA Metafile name *C* D=' '

Set the metafile logical unit and metafile type. This command controls the destination of the subsequent graphics output. Example:

```
LUN ==10 output only on metafile opened on unit 10;  
LUN = 0 output only on screen;  
LUN = 10 output on both screen and metafile opened on unit 10;
```

Use the command FORTRAN/FILE to open a new file, FORTRAN/CLOSE to close it. Note that PAW opens the file PAW.METAFILE on the unit 10 at initialization time.

```
METAFL= 4 Appendix E GKS.  
METAFL=-111 HIGZ/PostScript (Portrait).  
METAFL=-112 HIGZ/PostScript (Landscape).  
METAFL=-113 HIGZ/Encapsulated PostScript.  
METAFL=-114 HIGZ/PostScript Color (Portrait).  
METAFL=-115 HIGZ/PostScript Color (Landscape).  
METAFL=-777 HIGZ/LaTex Encapsulated.  
METAFL=-778 HIGZ/LaTex.
```

WORKSTATION iwkid [chopt iwtyp]

IWKID Workstation ID *I* D=1
CHOPT Options *C* D='0A'
IWTYP Workstation type *I* D=1

To create/delete workstations or change status.

```

CHOPT='O' Open a new workstation
CHOPT='C' Close a workstation
CHOPT='A' Activate a workstation
CHOPT='D' Deactivate a workstation
CHOPT='L' Give the list of open workstations
IWKID > 0 Do the action specified by CHOPT on the
workstation identified by IWKID.
IWKID=0 Do the action specified by CHOPT on all
workstations.
IWKID < 0 Do the action specified by CHOPT on the
workstation identified by -IWKID and the
complementary action on all the others.

```

SLIDE

Invoke the SLIDE package.

16.1 MISC

Miscellaneous HPLOT functions.

NEXT

Clear the screen. Initialize a new HIGZ picture if option ZFL or ZFL1 has been selected. Select the Normalization Transformation number 1 (cm).

CLR

Clear the screen.

LOCATE [ntpri chopt]

```

NTPRI Transformation with highest priority I D=-1
CHOPT Options C D='R' R='R,S,+'

```

Locate points on the screen using the graphics cursor and output coordinates on terminal. Control is returned when the BREAK (right) mouse button is clicked (or CRTL/E) or when 20 points are located. The optional parameter NTPRI may be specified to locate a point in the specific transformation number NTPRI. NTPRI=-1 (default) means that all the histogram transformation numbers (10, 20, etc.) have priority on transformation number 1.

```

CHOPT='R' Request mode is used to locate the points (default)
'S' Sample mode is used to locate the points
'I' Integrate an histogram between 2 bins
'+' use the tracking cross (default is cross-hair)

```

VLOCATE vecx vecy [chopt ntpri]

VECX	Vector for coordinates X	<i>C</i>
VECY	Vector for coordinates Y	<i>C</i>
CHOPT	Options	<i>C D=' ' R=' ,L,P,*,+,-,S'</i>
NTPRI	Transformation with highest priority	<i>I D=-1</i>

Locate a set of points using the graphics cursor. Return corresponding coordinates in vectors X and Y. If vectors X or Y do not exist, they are automatically created. Control is returned when the point is outside picture limits or when the BREAK (right) mouse button is clicked (or CRTL/E).

```
CHOPT=' ' use the cross-hair
'+' use the tracking cross
'-' use the rubber line
'L' connect points by a polyline
'P' draw the current polymarker at each point
'*' draw a * at each point
'S' sample mode is used. Allows to see the coordinates of
    point before clicking
```

The optional parameter NTPRI may be specified to locate a point in the specific transformation number NTPRI (see LOCATE).

HMOVE

Change the contents of a histogram channel using the cursor. Position the cursor to the channel to be changed, trigger graphics input, position the cursor to the new channel value (a rubber band box is used to visualize the change), trigger graphics input to fix the new value.

16.2 VIEWING

To define Normalization transformations. Either automatically (ZONE and SIZE) or 'by hand' (SVP, SWN and SELNT).

ZONE [nx ny ifirst chopt]

NX	Number of divisions along X	<i>I D=1</i>
NY	Number of divisions along Y	<i>I D=1</i>
IFIRST	First division number	<i>I D=1</i>
CHOPT	Option	<i>C D=' ' R=' ,S'</i>

Subdivide the picture into NX by NY zones, starting at zone IFIRST (count along X first). If CHOPT='S', redefine zones on current picture.

SIZE [xsize ysize]

XSIZE	Size along X	<i>R D=20.</i>
YSIZE	Size along Y	<i>R D=20.</i>

Set the size of the picture. On the terminal, the pictures will have the ratio YSIZE/XSIZE, and, if a metafile is produced, pictures will be YSIZE by XSIZEx cm. This command sets the parameters for the normalisation transformation number 1 to [0-XSIZE], [0-YSIZE].

SVP nt x1 x2 y1 y2

NT	Normalization transformation number	<i>I</i>
X1	Low X of viewport in NDC	<i>R D=0 R=0 : 1</i>
X2	High X of viewport in NDC	<i>R D=1 R=0 : 1</i>
Y1	Low Y of viewport in NDC	<i>R D=0 R=0 : 1</i>
Y2	High Y of viewport in NDC	<i>R D=1 R=0 : 1</i>

Set the viewport of the normalization transformation NT in the Normalized Device Coordinates (NDC).

SWN nt x1 x2 y1 y2

NT	Normalize transformation number	<i>I</i>
X1	Low X of window in WC	<i>R D=0</i>
X2	High X of window in WC	<i>R D=20</i>
Y1	Low Y of window in WC	<i>R D=0</i>
Y2	High Y of window in WC	<i>R D=20</i>

Set the window of the normalization transformation NT in World Coordinates (WC).

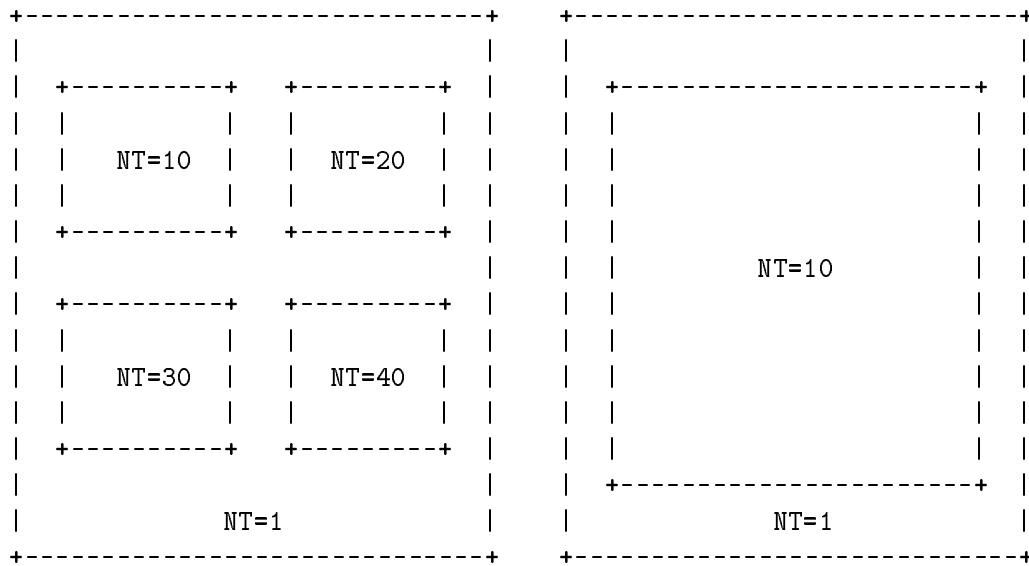
SELNT nt

NT Normalization transformation number I

Select a normalization transformation number.

If ZONE 2 2 is active , then:

If ZONE 1 1 is active, then:



16.3 PRIMITIVES

Call HIGZ drawing primitives

PLINE n x y

N Number of points *I*
 X Vector name for X coordinates *C*
 Y Vector name for Y coordinates *C*

Draw a polyline of N points X,Y in the current Normalization transformation. Use commands SLN, SLWSC and SPLCI (or IGSET) to change line attributes.

LINE x1 y1 x2 y2

X1 X first coordinate *R*
 Y1 Y first coordinate *R*
 X2 X second coordinate *R*
 Y2 Y second coordinate *R*

Draw a line connecting points (X1,Y1) and (X2,Y2) in the current Normalization transformation. Use commands SLN, SLWSC and SPLCI (or IGSET) to change line attributes.

FAREA n x y

N Number of points *I*
 X Vector name for X coordinates *C*
 Y Vector name for Y coordinates *C*

Fill the area defined by the N points X,Y in the current Normalization transformation. Use commands SFASI, SFAIS and SFACI (or IGSET) to change fill area attributes.

PMARKER n x y

N Number of points *I*
 X Vector name for X coordinates *C*
 Y Vector name for Y coordinates *C*

Draw polymarkers at the N points X,Y in the current Normalization transformation. Use commands SMK and SPMCI (or IGSET) to change polymarker attributes.

BOX x1 x2 y1 y2

X1 X coordinate of first corner *R*
 X2 X coordinate of second corner *R*
 Y1 Y coordinate of first corner *R*
 Y2 Y coordinate of second corner *R*

Draw and fill a box with the current fill area attributes. Use the current Normalization transformation.

FBOX x1 x2 y1 y2 x3 x4 y3 y4

X1 X coord of 1st corner of ext box *R*
 X2 X coord of 2nd corner of ext box *R*
 Y1 Y coord of 1st corner of ext box *R*
 Y2 Y coord of 2nd corner of ext box *R*
 X3 X coord of 1st corner of int box *R*
 X4 X coord of 2nd corner of int box *R*
 Y3 Y coord of 1st corner of int box *R*
 Y4 Y coord of 2nd corner of int box *R*

Draw and fill a frame (2 nested boxes) with the current fill area attributes. Use the current Normalization transformation.

ARROW x1 x2 y1 y2 [size]

X1 X coordinate of start point *R*
 X2 X coordinate of end point *R*
 Y1 Y coordinate of start point *R*
 Y2 Y coordinate of end point *R*
 SIZE Arrow size *R D=0.4*

Draw an arrow (X1,Y1) —> (X2,Y2) if SIZE>0. Draw an arrow (X1,Y1) <—> (X2,Y2) if SIZE<0. Use the current Normalization transformation.

AXIS x0 x1 y0 y1 wmin wmax ndiv [chopt]

X0 X axis origin in WC *R*
 X1 X end axis in WC *R*
 Y0 Y axis origin in WC *R*
 Y1 Y end axis in WC *R*
 WMIN Lowest value for labels *R*
 WMAX Highest value for labels *R*
 NDIV Number of divisions *I D=510*
 CHOPT Options *C D=' '*

Draw an axis in the current Normalization transformation.

```
NDIV=N1 + 100*N2 + 10000*N3
N1, N2, N3 = Number of 1st, 2nd, 3rd divisions respectively, eg:.
NDIV=0 --> no tick marks.
NDIV=2 --> 2 divisions, one tick mark in the middle
            of the axis.
CHOPT='G' : logarithmic scale, default is linear.
CHOPT='B' : Blank axis. Useful to superpose axis.
```

Orientation of tick marks on axis: Tick marks are normally drawn on the positive side of the axis. However, if X0=X1, then Negative .
 CHOPT='+' : tick marks are drawn on Positive side. (default)
 CHOPT='-' : tick marks are drawn on the negative side.
 i.e: '+-' --> tick marks are drawn on both sides of the axis.

CHOPT='U' : Unlabeled axis, default is labeled.
 Position of labels on axis. Labels are normally drawn on side opposite to tick marks. However:
 CHOPT= '=' on Equal side
 Orientation of labels on axis. Labels are normally drawn parallel to the axis. However if X0=X1, then Orthogonal if Y0=Y1, then Parallel
 CHOPT= 'P' : Parallel to the axis
 CHOPT= 'O' : Orthogonal to the axis (Top to Down).
 CHOPT= 'O' : Orthogonal to the axis (Down to Top).
 Position of labels on tick marks. Labels are centered on tick marks. However , if X0=X1, then they are right adjusted.
 CHOPT='R': labels are Right adjusted on tick mark.
 (default is centered)
 CHOPT='L': labels are Left adjusted on tick mark.
 CHOPT='C': labels are Centered on tick mark.
 CHOPT='M': In the Middle of the divisions.
 Direction of labels. Default is RIGHT
 CHOPT='Y': Down
 Format of labels. Blank characters are stripped, and then the label is correctly aligned. The dot,if last character of the string, is also stripped, unless
 CHOPT='.' Dot obligatory
 In the following, we have some parameters, like tick marks length and characters height (in percentage of the length of the axis).The default values are as follows:
 Primary tick marks: 3.0 %
 Secondary tick marks: 1.5 %
 Third order tick marks: .75 %
 Characters height for labels: 2%
 Characters spacing (related to height): 40%
 Labels offset: 4.0 %
 Type of labels. Labels are normally numeric . However, alphanumeric labels can be drawn (see command LABEL).
 CHOPT='T': Alphanumeric labels .
 Intrinsic parameters.
 CHOPT='S': Tick marks Size
 CHOPT='H': Labels Height
 CHOPT='D': Distance labels-axis
 Axis binning optimization. By default the axis binning is optimized .
 CHOPT='N': No binning optimization
 CHOPT='I': Integer labeling

ARC x1 y1 r1 [r2 phimin phimax]

X1 X coordinate of centre *R*
 Y1 Y coordinate of centre *R*
 R1 Inner radius *R*
 R2 Outer radius *R* D=-1.
 PHIMIN Minimum angle *R* D=0.
 PHIMAX Maximum angle *R* D=360.

Draw an arc of circle in the current Normalization transformation. If R1 is not equal to R2 the area between the two arcs of radius R1 and R2 is filled according to the current fill area attributes. The border is never drawn unless the interior style is hollow or the command IGSET BORD 1 has been called. If R1 is equal to R2 a polyline is drawn.

PIE x0 y0 radius n values [chopt iao ias iac]

X0 X coordinate of centre of the pie *R*
 Y0 Y coordinate of centre of the pie *R*
 RADIUS Radius of the pie chart *R*
 N Number of values *I*
 VALUES Vector name for N values *C*
 CHOPT Options *C* D=' ' R=' ',P,N,L'
 IAO Name of vector with offsets *C* D=' '
 IAS Name of vector with styles *C* D=' '
 IAC Name of vector with colors *C* D=' '

Draw a pie chart in the current Normalization transformation.

CHOPT Character variable specifying the option:
 'P' Labels of each slice will be in percentage.
 'N' Labels of each slice will be the numeric value in VALUES.
 'L' Labels of each slice will be the text given in command LABEL.

TEXT x y text size [angle chopt]

X X coordinate *R*
 Y Y coordinate *R*
 TEXT Text to be drawn *C*
 SIZE Text size *R* D=0.3
 ANGLE Comment angle *R* D=0
 CHOPT Justification option *C* D='L' R='L, ,C,R'

Draw text at position X,Y in the current normalization transformation using the software font IGTEXT. SIZE is always given in centimeters (as defined by the command SIZE). Boldface effects can be obtained using the parameters PASS and CSHI of the command SET.

CHOPT='L' Text is Left justified.
 CHOPT='C' Text is Centered.
 CHOPT='R' Text is Right justified.

The text color can be changed by IGSET TXCI.

ITX x y text

X	X coordinate	<i>R</i>
Y	Y coordinate	<i>R</i>
TEXT	Text to be drawn	<i>C</i>

Draw text at position X,Y in the current Normalization transformation, using the current font parameters. The font and the precision can be changed by IGSET TXFP. The character size can be changed by IGSET CHHE. The text color can be changed by IGSET TXCI. The text orientation can be changed with IGSET TXAL. The text angle can be changed by IGSET TANG.

LABELS labnum nlabs chlabs

LABNUM	Label identifier	<i>I</i>	D=1 R=1:9
NLABELS	Number of labels	<i>I</i>	D=0 R=0:50
CHLABELS	List of labels	<i>C</i>	D=' '

Define a list of labels to be used by subsequent commands such as PIE and AXIS. The position of the labels on the axis may be changed with SET NDVX (NDVY).

PAVE x1 x2 y1 y2 [dz isbox isfram chopt]

X1	X bottom left corner of box	<i>R</i>
X2	X top right corner of box	<i>R</i>
Y1	Y bottom left corner of box	<i>R</i>
Y2	Y top right corner of box	<i>R</i>
DZ	Box width	<i>R</i> D=0 .4
ISBOX	Box style	<i>I</i> D=0
ISFRAM	Frame style	<i>I</i> D=5
CHOPT	Option	<i>C</i> D=' TR'

Draw a paving-block (box with 3D effect). ISBOX (ISFRAM) may be 1000+ICOLOR where ICOLOR is the color index of the box (frame), otherwise the style index. If ISBOX (ISFRAM) =0, only the box contour is drawn with the current polyline attributes.

```

CHOPT='TR' (or just 'T') Top and Right frame are drawn (default)
CHOPT='TL' Top and Left frame
CHOPT='BR' (or just 'B') Bottom and Right frame
CHOPT='BL' Bottom and Left frame
CHOPT='L' Left frame only
CHOPT='R' Right frame only
CHOPT='T-' Top frame only pointing left
CHOPT='B-' Bottom frame only pointing left
CHOPT='S' Shadow mode
CHOPT='K' Key mode

```

HIST n x y [chopt]

N	Number of values	<i>I</i>
X	Vector name for X coordinates	<i>C</i>
Y	Vector name for Y coordinates	<i>C</i>
CHOPT	Options	<i>C</i> D='AHW' R='AHW,A,W,R,N,H,F,C,L,*,P,B, '

Draw an histogram defined by arrays X and Y. The number of components needed in vectors X and/or in Y may be dependent upon the value of CHOPT (see options 'R' and 'N').

CHOPT:

- 'A' X and Y axes are drawn (default).
- 'H' An histogram is drawn as a contour (default).
- 'W' The Window/Viewport parameters are automatically computed from the X and Y values (default).
- 'R' The histogram is Rotated, i.e. the values in X are used for the ordinate and the values in Y for the abscissa (default is the contrary).
If option R is selected (and option 'N' is not selected), the user must give:

2 values for Y (Y(1)=YMIN and Y(2)=YMAX)
N values for X, one for each bin.

Otherwise the user must give:

N values for Y, one for each bin.
2 values for X (X(1)=XMIN and X(2)=XMAX)

- If option 'N' is selected see below.
- 'N' Non equidistant bins (default is equidistant).
The arrays X and Y must be dimensioned as follows:
If option R is not selected (default) then give:
(N+1) values for X (limits of bins).

N values for Y, one for each bin.

Otherwise give:

(N+1) values for Y (limits of bins).
N values for X, one for each bin.

- 'F' The area delimited by the histogram is filled according to the fill area interior style and the fill area style index or colour index.

Contour is not drawn unless CHOPT='H' is also selected.

- 'C' A Smooth curve is drawn across points at the centre of each bin of the histogram.

- 'L' A straight Line is drawn across points at the centre of each bin of the histogram.

- '*' A star is plotted at the center of each bin of the histogram.

- 'P' Idem as '*' but with the current marker.

- 'B' A Bar chart with equidistant bins is drawn as fill areas.
(Contours are drawn). The bar origin and the bar

width can be controlled by the routine IGSET using the options BAR0 and BARW respectively.

To set Log scales in X and/or Y, use OPT LOGX/LOGY. Note that when an option is specified, it is also necessary to specify the options 'AW' or 'AHW' in order to start a new zone or/and draw the axes.

GRAPH n x y [chopt]

N	Number of values	<i>I</i>
X	Vector name for X coordinates	<i>C</i>
Y	Vector name for Y coordinates	<i>C</i>
CHOPT	Options	<i>C D='ALW' R='ALW,W,L,C,F, ,*,P,R,B'</i>

Draw a curve through a set of points.

- 'A' X and Y axes are drawn (default).
- 'L' Every point is connected with a straight line. (default)
- 'W' The Window/Viewport parameters are automatically computed from the X and Y values (default).
- 'C' The values in Y are plotted in the form of a smooth curve. A Spline approximation algorithm is used.
- 'F' A fill area is drawn. If the option 'CF' is used the contour of the fill area is smooth. The border of the fill area is drawn if the command IGSET BORD 1 has been typed. The fill area type may be changed via the IGSET parameters FASI and FASI
- 'R' The graph is Rotated, i.e. the values in X are used for the ordinate and the values in Y for the abscissa (default is the contrary).
- 'B' A Bar chart with equidistant bins is drawn as fill areas. (Contours are drawn). The bar origin and the bar width can be controlled by the routine IGSET using the options BAR0 and BARW respectively.
- '*' A star is plotted at every point.
- 'P' A marker is plotted at every point, according to current marker type and polymarker colour index.

To set Log scales in X and/or Y, use OPT LOGX/LOGY. Note that when an option is specified, it is also necessary to specify the options 'AW' or 'ALW' in order to start a new zone or/and draw the axes.

16.4 ATTRIBUTES

Change HIGZ/GKS attributes.

SLN [iln]

ILN Line style *I* D=1 R=1:

Set the line style.

SFAIS [ints]INTS Fill area interior style *I* D=0 R=0:3

Set the fill area interior style :

Hollow=0, Solid=1, Pattern=2, Hatch=3

SFASI [styli]STYLI Fill area style index *I* D=1

Set the fill area style index.

SFACI [ifaci]IFACI Fill area color index *I* D=1

Set the fill area color index.

SPLCI [iplci]IPLCI Polyline color index *I* D=1

Set the polyline color index.

SPMCI [ipmci]IPMCI Polymarker color index *I* D=1

Set the polymarker color index.

STXCI [itxci]ITXCI Text color index *I* D=1

Set the text color index.

STXFP [ifont iprec]IFONT Font number *I* D=0IPREC Font precision *I* D=2

Set text font and precision.

SCHH [chh]CHH Character height *R* D=0.28

Set the character height.

SLWSC [lw]LW Line width *I* D=1 R=1:

Set the line width.

SMK [mkt]

MKT Marker type *I* D=1

Set the marker type.

COLOR_TABLE icol [red green blue]

ICOL	Color Index	<i>I</i>	D=1
RED	Weight of red	<i>R</i>	D=0. R=0.:1.
GREEN	Weight of green	<i>R</i>	D=0. R=0.:1.
BLUE	Weight of blue	<i>R</i>	D=0. R=0.:1.

Define the color ICOL.

PALETTE palnb [nel list]

PALNB	Palette number	<i>I</i>	D=0 R=0:9
NEL	Number of elements in the palette	<i>I</i>	D=0 R=0:50
LIST	List of the palette elements	<i>I</i>	D=0

Define a palette of attributes. The palette number is used in the command SET. The command SET HCOL 0.1 defines the palette number 1 as colour indices used by the command LEGO in case of stacked lego plots and plotting of SURFACE with options 1 or 2, LEGO with option 2 and CONTOUR with option 3.

By default the palettes are initialized with 6 elements: 2,3,4,5,6,7.

If the number of elements (NEL) is equal to 0 (default), the palette is filled automatically according to the number of colours defined with the command IGSET NCOL. If NCOL is smaller than 8, the palette is filled with a subset of the 8 basic colours. If NCOL is greater than 8, the palette is filled with colours varying continuously from blue to red. This is called a "geographical" palette. Note that the command IGSET NCOL reset the colours 8 to NCOL with gray levels.

Examples:

```

PAW > IGSET NCOL 8      | Define the number of colours
PAW > PALETTE 1          | The palette 1 is filled with
                           | 8 elements: 0,5,7,3,6,2,4,1
PAW > IGSET NCOL 4      | Define the number of colours
PAW > PALETTE 1          | The palette 1 is filled with
                           | 4 elements: 0,5,7,3
PAW > IGSET NCOL 16      | Define the number of colours
PAW > PALETTE 1          | Fill palette 1 with 8 elements
                           | (8,9,10,11,12,13,14,15) varying
                           | continuously from blue to red

```

16.5 H PLOT

Draw various H PLOT objects (symbols, errors, key, etc.).

SYMBOLS x y n [isymb ssize]

X Vector of X coordinates *C*
Y Vector of Y coordinates *C*
N Number of points *I* D=1
ISYMB Symbol number *I* D=24
SSIZE Symbol size *R* D=0.28

Draw the same symbol at several points x,y in the current normalization transformation.

ERRORS x y ex ey n [isymb ssize]

X Vector of X coordinates *C*
Y Vector of Y coordinates *C*
EX Vector of X error bars *C*
EY Vector of Y error bars *C*
N Number of points *I* D=1
ISYMB Symbol number *I* D=24
SSIZE Symbol size *R* D=0.28

Draw a series of points using a symbol and error bars in horizontal and vertical direction in the current normalization transformation.

KEY x y [isymb text]

X X coordinate of comment *R*
Y Y coordinate of comment *R*
ISYMB Symbol number *I* D=24
TEXT Legend *C* D=' '

Draw one symbol and its explanation (legend) at a point x,y in the current normalization transformation.

TICKS [chopt xval yval]

CHOPT Options *C* D=' '
XVAL X position *R* D=1.E30
YVAL Y position *R* D=1.E30

Draw 'cross-wires' on a picture, optionally with tick marks and values. Cross-wires are lines perpendicular to the X and/or Y axis.

CHOPT is a string to denote which cross-wires to draw,
and where to draw the values:
' ' tick marks are drawn on the edges of the picture
'X' cross-wire drawn perpendicular to the X-axis
'Y' cross-wire drawn perpendicular to the Y-axis
'A' value drawn Above cross-wire
'B' value drawn Below cross-wire
'L' value drawn Left of cross-wire
'R' value drawn Right of cross-wire

```
XVAL intersection on the X-axis
YVAL intersection on the Y-axis
```

The values of XVAL are always histogram coordinates. The tick marks will be drawn on both side of the cross wire, unless the cross-wires are requested on the boundary of the box surrounding the histogram (i.e. at the extreme limits of the drawn histogram). In this case tick marks will only be drawn inside the box. The options 'A' and 'B' (for Above and Below) refer only to the cross-wire perpendicular to the Y axis. In each case only one cross-wire will be drawn. Similarly 'L' and 'R' (Left and Right) refer only to the cross-wires perpendicular to the X-axis. It is possible to redefine the length of tick marks on the X or Y axis with SET XTIC or SET YTIC. The position of the axis values may be changed with SET XVAL or SET YVAL.

ATITLE [xtit ytit]

```
XTIT X Axis title C D=' '
YTIT Y Axis title C D=' '
```

Draw axis titles on the axes of the present plot zone.

GRID

Draw a grid in cm.

NULL [xmin xmax ymin ymax chopt]

```
XMIN Low range in X R D=0.
XMAX High range in X R D=1.
YMIN Low range in Y R D=0.
YMAX High range in Y R D=1.
CHOPT Options C D=' ' R=' ',S,A,B'
```

Draw a frame box only. If XMIN, XMAX, etc. are given, draw a frame box with the window coordinates set to XMIN, XMAX, YMIN, YMAX. Axis labels and tick marks are drawn by default. If option 'S' is also specified, this command is a convenient way to redefine the scale for the current zone. If the option 'A' is given then axis labels and tick marks are not drawn. The box is not drawn if the option 'B' is given.

Chapter 17: PICTURE

Creation and manipulation of HIGZ pictures.

FILE lun fname [lrecl chopt]

LUN Logical unit number *I* R=1:128
FNAME File name *C*
LRECL Record length in words *I* D=1024
CHOPT Options *C* D=' ' R=' ',A,N,U,AN,AU'
Open a HIGZ direct access picture file.

For CHOPT=' ', existing file is opened.
For CHOPT='N', a new file is opened.
For CHOPT='U', existing file is modified.

If CHOPT='AU' or 'AN', pictures will be automatically saved on the direct access file. This automatic saving facility can be switched off using IGSET AURZ 0.

LIST

List all the HIGZ pictures currently stored in memory.

CREATE pname

PNAME Picture name *C*

Create a new picture, named PNAME, in memory. Note that all commands which start a new picture (clear workstation) automatically create pictures named PICT1, PICT2, etc. if the command OPTION ZFL or OPTION ZFL1 has been executed.

DELETE pname

PNAME Picture name *C* D=' '

Delete the picture PNAME from memory. PNAME='*' means all pictures.

SCRATCH pname [icycle]

PNAME Picture name *C* D=' '

ICYCLE Cycle number *I* D=9999

Delete the picture PNAME from current directory on disk.

PLOT [pname]

PNAME Picture name *C* D=' '

Plot the picture PNAME. PNAME=' ' means the current picture. PNAME='*' means all pictures.

MODIFY [pname chopt]

PNAME Picture name *C* D=' '

CHOPT Options *C* D=' ' R=' ',S,A'

Edit the picture PNAME. PNAME=' ' means the current picture. Various options can be selected with the graphics menu. This command is only available on workstations.

CHOPT='S' Software characters are used for the text in menus.

CHOPT='A' the option shAdow is used.

MERGE pname [x y scale chopt]

PNAME Picture name *C*
 X X coord(NDC) where to draw PNAME *R D=0*
 Y Y coord(NDC) where to draw PNAME *R D=0*
 SCALE Scale factor *R D=1.*
 CHOPT Options *C D=' ' R=' ',D'*

Add the picture PNAME to the current picture.

CHOPT='D' Picture PNAME is displayed during merging.

COPY pname1 pname2

PNAME1 Picture name *C*

PNAME2 New picture name *C*

Copy a picture.

RENAME pname1 pname2

PNAME1 Old picture name *C*

PNAME2 New picture name *C*

Rename a picture.

IZOUT [pname]

PNAME Picture name *C D=' '*

Write the picture PNAME to a direct access picture file (see command PICTURE/FILE). PNAME=' ' means the current picture. PNAME='*' means all pictures.

IZIN pname [icycle]

PNAME Picture name *C*

ICYCLE Cycle number *I D=9999*

Read picture into memory from a direct access picture file. (see command PICTURE/FILE). PNAME='*' means all pictures.

IZPICT pname [chopt]

PNAME Picture name *C*

CHOPT Options *C D='M' R='M, ,D,S,N,L,F,P,C'*

Perform various operations on a picture.

CHOPT:

'M' Make a new picture in memory with name PNAME.

An empty structure is created in memory and becomes the current picture. If PNAME = ' ', the picture is automatically named as PICTnnn, where the starting value of nnn is either 0 (default), or the value assigned by

IGSET to the parameter PICT.

- 'D' Display the picture PNAME in memory.
- 'S' Scratch the picture PNAME from memory. If PNAME = ' ' the current picture is scratched.
- 'N' The picture following the current picture in memory becomes the current picture. If the current picture is the last one in memory, the first picture in memory becomes the current picture.
- 'L' Give the list of the pictures in memory, following the sequence of their storage in memory.
- 'F' The First picture in memory becomes the current picture.
- 'P' Print the picture data structure. Useful to debug programs.
- 'C' Set Current picture. All calls to HIGZ graphic functions are stored in the current structure according to the option selected be IGZSET.

PNAME=' ' means the current picture. PNAME='*' means all pictures.

SWITCH [chopt]

CHOPT Options C D='G' R='G,Z,GZ'

Set the graphics switch to control plotting output to terminal (G) and/or picture in memory (Z).

If CHOPT='G' Graphics output only.
 If CHOPT='Z' graphics primitives stored in ZEBRA memory only.
 If CHOPT='GZ' both.

IGSET [chatt value]

CHATT Attribute name C D='SHOW'

VALUE Attribute value R D=0.

Set a HIGZ attribute. If CHATT='SHOW' print default and current values for all attributes. If CHATT='*' restore default values for all attributes. If VALUE=0, the attribute is set to its default value.

Chapter 18: ZEBRA

Interfaces to the ZEBRA RZ, FZ and DZ packages.

18.1 RZ

ZEBRA/RZ package: direct access Input/Output.

```
FILE lun fname [ lrecl chopt ]
```

LUN Logical unit number *I* R=1:128
FNAME File name *C*
LRECL Record length in WORDS *I* D=1024
CHOPT Options *C* D=' ' R=' ,U'

Open an existing direct access file.

CHOPT=' ' read only mode
CHOPT='U' update mode

```
MAKE lun fname [ lrecl nrec nwkey chform chtags ]
```

LUN Logical unit number *I* R=1:128
FNAME File name *C*
LRECL Record length in WORDS *I* D=1024
NREC Number of records *I* D=1000
NWKEY Number of words per Key *I* D=1
CHFORM Key format *C* D='I' R='I,B,A,H'
CHTAGS List of Tags *C* D='HBOOK-ID'

Open a new direct access file.

```
MDIR chdir [ nwkey chform chtags ]
```

CHDIR Directory name *C*
NWKEY Number of words per Key *I* D=1
CHFORM CHFORM *C* D='I'
CHTAGS List of Tags *C* D='HBOOK-ID'

Create a new RZ directory below the current directory.

```
DDIR chdir
```

CHDIR Directory name *C*

Delete the directory CHDIR from the current directory.

LDIR [chpath chopt]CHPATH Path name *C* *D*='CHOPT Options *C* *D*=' ',A,T'

List contents of a directory (memory or disk). To list all RZ files currently opened, type 'LD //'. Note that if the Current Directory is //PAWC, this command uses the same format as HISTO/LIST.

CHOPT='A' to list all the Ntuple extensions.

CHOPT='T' to list a directory Tree.

CDIR [chpath chopt]CHPATH Path name *C* *D*='CHOPT Options *C* *D*='

Change the current working directory (CWD). IF CHPATH is given make it the new CWD. Otherwise, print the pathname of the CWD.

Ex. CD dir1 ; make DIR1 the new CWD

CD //file1/dir2 ; make //FILE1/DIR2 the new CWD

CD ; print the name of the CWD

PURGE [keep]KEEP Number of cycles to be kept *I* *D*=1

Purge an RZ directory.

LOCK [chlock]CHLOCK Lock identifier *C* *D*='RZFILE'

Lock an RZ directory.

FREE [chlock]CHLOCK Lock identifier *C* *D*='RZFILE'

Free an RZ directory.

STAT chpathCHPATH Name of top directory *C*

Print space statistics for an RZ file.

18.2 FZ

ZEBRA/FZ package: sequential access Input/Output.

FILE lun fname [lrecl chopt]

LUN Logical unit number *I* R=1:128
 FNAME File name *C*
 LRECL Record length in words *I* D=900
 CHOPT Options *C* D='IX' R='IX,0,X,A,I,OX,IA,OA'
 Open an FZ sequential formatted or unformatted file.

CHOPT = 'I' Input file.
 CHOPT = 'O' Output file
 CHOPT = 'X' binary eXchange mode.
 CHOPT = 'A' Alphanumeric exchange mode.

TOFZ lun [chopt]

LUN Logical unit number of FZ file *I* R=1:128
 CHOPT Options *C* D=' '
 Copy the current directory tree onto an FZ file.

FRFZ lun [chopt]

LUN Logical unit number of FZ file *I* R=1:128
 CHOPT Options *C* D=' '
 Copy the FZ file into the current directory tree.

TOALPHA fname

FNAME Name of the FZ text file *C*
 Copy the current directory tree onto a FZ file. An alphanumeric format is used. The file FNAME can be exchanged between different machines.

FRALPHA fname

FNAME Name of the FZ text file *C*
 Copy the FZ alphanumeric file into the current directory.

18.3 DZ

ZEBRA/DZ package: debugging.

SHOW name [number chopt]

NAME Bank name *C*
 NUMBER Bank number *I* D=1
 CHOPT Options *C* D='BSV'

Display the contents of a bank or a data structure identified by its NAME and NUMBER. The output format of the data part is controlled by the internal or external I/O characteristic.

CHOPT='B' Print the bank.
 CHOPT='S' Print the bank contents from left to right Sideways
 with up to ten elements per line.
 CHOPT='V' Print the vertical (down) structure.
 CHOPT='D' Print the bank contents from top to bottom Downwards
 with five elements per line.
 CHOPT='L' Print the linear structure.
 CHOPT='Z' Print the data part of each bank in hexadecimal format

SURV [name [number]]

NAME Bank name *C*
 NUMBER Bank number *I* D=1

Print a survey of the structure identified by NAME, NUMBER.

SNAP [idiv chopt]

IDIV Division number *I* D=2 R=0:24
 CHOPT Options *C* D='M'

Snap of one or more divisions. Provides a snapshot of one or more divisions in a ZEBRA store. The kind of information provided is controlled by CHOPT.

CHOPT='M' Print Map entry for each bank
 CHOPT='E' Extend map entry to dump all links of each bank
 (otherwise only as many links as will fit on a line)
 CHOPT='F' Full. Dump all active banks, links and data
 CHOPT='K' Kill. Dropped banks to be treated as active
 (dropped banks are not normally dumped under D or F option)
 CHOPT='L' Dump all Link areas associated with the store
 CHOPT='W' Dump the Working space, links and data
 CHOPT='Z' Dump the information in hexadecimal.

VERIFY [idiv chopt]

IDIV Division number *I* D=0 R=0:24
 CHOPT Options *C* D='CLSU'

Check the structure of one or more ZEBRA divisions. The verification detail depends on the settings in CHOPT.

CHOPT='C' Check chaining of banks only
 CHOPT='L' Check validity of the structural links (implies 'C')
 CHOPT='S' Check the store parameters
 CHOPT='U' Check the validity of the up and origin (implies 'C')
 CHOPT='F' Errors are considered fatal and generate a call to ZFATAL

STORE [ixstor]

IXSTOR Store number *I* D=0 R=0:24

Display the structure of the ZEBRA store IXSTOR. Output the parameters characterizing the store, followed by a list of all divisions and all link areas associated with the store in question.

Chapter 19: FORTRAN

Interface to COMIS, SIGMA and FORTRAN Input/Output.

COMIS

Invoke the COMIS FORTRAN interpreter. COMIS allows to execute FORTRAN routines without recompiling and relinking. It communicates with PAW commands through vectors and functions. COMIS has its PAW-independent command structure. Example in command mode:

```
PAW > Comis
CS >   do 10 i=1,10
MND>       x=sqrt(i)*10.
MND>       print *,i,x
MND> 10 continue
MND>   END
CS > quit
PAW >
```

COMIS code may be inserted into a macro. Example:

```
Vector/Create Y(10) r 1 2 3 4 5 6 7 8 9 10
*
* In the following COMIS code, the statement "Vector Y" declares
* to COMIS an existing KUIP vector. KUIP dimension is assumed.
* The statement "Vector X(10)" creates a new KUIP vector.
* (Note that SUBROUTINEs must be declared before the MAIN program)
* (KUIP vectors cannot be created into the MAIN program)
*
APPLICATION COMIS QUIT
    SUBROUTINE DEMO
        Vector Y
        Vector X(10)
        do 10 i=1,10
            XX=i
            X(i)=Y(i)*sqrt(XX)*10.
    10 CONTINUE
    END
    CALL DEMO
    END
QUIT
Vector/print X      | Print KUIP vector created by COMIS
```

CALL urout

UROUT User routine C

Execute the routine UROUT. UROUT may be a routine compiled and linked with PAW. For example : CALL HPRINT(10). UROUT may also be the name of a file which can be edited interactively with the command EDIT. For example if file UROUT.FOR contains:

```
SUBROUTINE UROUT(N)
SUM=0.
DO 10 I=1,N
    SUM=SUM+I
10 CONTINUE
PRINT *,SUM
END
```

Then one can type CALL UROUT.FOR(10). The routine UROUT may also contains references to the library routines mentioned below. The following routines from the CERN Program Library can be called:

LOOP ntimes urout

NTIMES Number of calls *I* D=1
 UROUT User routine *C*

The routine UROUT is called NTIMES times. See command CALL for explanation of UROUT.

FILE lun fname

LUN Logical unit number *I* R=1:128
 FNAME File name *C*

Open a FORTRAN formatted text file.

CLOSE lun

LUN Logical unit number *I* R=1:128

Close the file on unit LUN. If the file has been opened with HISTO/FILE, PICTURE/FILE, etc, then before closing the unit, PAW will close correctly the file with CALL HREND or FZEND(O), ICLWK, etc.

REWIND lun

LUN Logical unit number *I* R=1:128

Rewind the file on unit LUN.

SIGMA

Invoke the SIGMA package. SIGMA is an array manipulation package using its own vector-oriented language, outside the PAW command conventions. SIGMA may be invoked in one of the three following ways:

- 1- Using the KUIP \$SIGMA function. Example:
 PAW > Vector/Create x(10) r 1 2 3 4 5 6 7 8 9 10
 PAW > Graph 10 x \$sigma(sqrt(x))
- 2- Using the SIGMA command. Example:
 PAW > sigma x=array(10,1#10)
 PAW > sigma y=sqrt(x)
 PAW > Graph 10 x y

3- Using the APPLication command. Example:

```
PAW > APPLication SIGMA
SIGMA > x=array(10,1#10)
SIGMA > y=sqrt(x)
SIGMA > exit
PAW > Graph 10 x y
```

Chapter 20: OBSOLETE

Obsolete commands.

20.1 HISTOGRAM

20.1.1 FIT

Fitting and smoothing (1-Dim or 2-Dim) histograms. Results are given as histogram-associated functions, and fit parameters printed on screen.

```
EXponential id [ isel iftlow iftup ]
```

ID histogram Identifier *C*
ISEL option flag *I* D=12
IFTLOW First channel *I* D=1
IFTUP Last channel *I* D=99999

Fit histogram ID with an exponential function between channels IFTLOW and IFTUP. Obsolete command. Use Command Hist/Fit instead. Control word ISEL = 100*W+10*P+S.

S=2 superimposes function to histogram
1 no superimposing
P=1 output from final iteration
>1 output at iterations from 0 to (P-1), N=0,1,2,..
0 no output
W=1 sets weights equal to 1
0 calculates statistical errors as
E=SQRT(CONTENTS) unless the 1-Dim histogram
ID is weighted with HBARX or HPAKE

```
GAUSS id [ isel iftlow iftup ]
```

ID histogram Identifier *C*
ISEL option flag *I* D=12
IFTLOW First channel *I* D=1
IFTUP Last channel *I* D=99999

Fit histogram ID with a Gaussian between channels IFTLOW and IFTUP. Obsolete command. Use Command Hist/Fit instead. Control word ISEL = 100*W+10*P+S.

S=2 superimposes function to histogram
1 no superimposing
P=1 output from final iteration
>1 output at iterations from 0 to (P-1), N=0,1,2,..
0 no output
W=1 sets weights equal to 1
0 calculates statistical errors as
E=SQRT(CONTENTS) unless the 1-Dim histogram
ID is weighted with HBARX or HPAKE

POLYNOMIAL id ncoeff [isel iftlow iftup]

ID	histogram Identifier	<i>C</i>
NCOEFF	Number of coefficients	<i>I</i> D=3
ISEL	option flag	<i>I</i> D=12
IFTLOW	First channel	<i>I</i> D=1
IFTUP	Last channel	<i>I</i> D=99999

Fit histogram ID with a polynomial between channels IFTLOW and IFTUP. NCOEFF is the degree of the polynomial plus one. Obsolete command. Use Command Hist/Fit instead. Control word ISEL = 100*W+10*P+S.

```
S=2 superimposes function to histogram
  1 no superimposing
P=1 output from final iteration
  >1 output at iterations from 0 to (P-1), N=0,1,2,..
  0 no output
W=1 sets weights equal to 1
  0 calculates statistical errors as
    E=SQRT(CONTENTS) unless the 1-Dim histogram
    ID is weighted with HBARX or HPAKE
```

FUNCTION id func np dpar [isel iftlow iftup step pmin pmax]

ID	Histogram Identifier	<i>C</i>
FUNC	Function name	<i>C</i>
NP	Number of parameters	<i>I</i>
DPAR	Vector of parameters	<i>C</i>
ISEL	option flag	<i>I</i> D=12
IFTLOW	First channel	<i>I</i> D=1
IFTUP	Last channel	<i>I</i> D=99999
STEP	Vector of steps size	<i>C</i>
PMIN	Vector of lower bounds	<i>C</i>
PMAX	Vector of upper bounds	<i>C</i>

Obsolete command. Use Command Hist/Fit instead. Fit a user defined (and parameter dependent) function to a histogram ID between channels IFTLOW and IFTUP. FUNC is the name of a file which contains the user defined function to be minimized. For example file FUNC.FOR is:

```
DOUBLE PRECISION FUNCTION FUNC(X)
DOUBLE PRECISION X,DPAR
COMMON/PAWPAR/DPAR(100)
FUNC=DPAR(1)*X +DPAR(2)*EXP(-X)
END
```

After the fit, the vector DPAR contains the new values of parameters. Control word ISEL = 10000*B+100*W+10*P+S.

```
S=2 superimposes function to histogram
  1 no superimposing
P=1 output from final iteration
  >1 output at iterations from 0 to (P-1), N=0,1,2,..
  0 no output
W=1 sets weights equal to 1
  0 calculates statistical errors as
    E=SQRT(CONTENTS) unless the 1-Dim histogram
    ID is weighted with HBARX or HPAKE
B=0 All parameters vary freely (vectors STEP,PMIN,PMAX not required)
  1 Some or all parameters are bounded
    STEP(I)=0 means parameter I is fixed to its initial value
```

Chapter 21: NETWORK

To access files on remote computers. To send messages to a remote process (ZEBRA server)

RLOGIN host

HOST Host name *C D= ' '*

Start a communication with a remote machine HOST. Current Directory will be changed to //HOST.

RSHELL message

MESSAGE Message to remote host *C D= ' '*

Send MESSAGE to current remote host. Note that the Current Directory must be //HOST (see RLOGIN). Some PAW commands (Histo/Plot, Histo/List) can communicate directly with HOST.

Appendix A: PAW tabular overview

Table A.1: Alphabetical list of PAW commands

Calling sequence	Page
1DHISTO (/HISTOGRAM/CREATE) id title ncx xmin xmax [valmax]	219
2DHISTO (/HISTOGRAM/CREATE) id title ncx xmin xmax ncy ymin ymax [valmax]	220
ABSCISSA (/HISTOGRAM/GET_VECT) id vname	227
ADD (/HISTOGRAM/OPERATIONS) id1 id2 id3 [c1 c2]	223
ANGLE (/FUNCTION) [theta phi]	233
APPLICATION (/KUIP/SET_SHOW) [path cmdex]	202
ARC (/GRAPHICS/PRIMITIVES) x1 y1 r1 [r2 phimin phimax]	248
ARROW (/GRAPHICS/PRIMITIVES) x1 x2 y1 y2 [size]	247
ATITLE (/GRAPHICS/HPLOT) [xtit ytit]	257
AXIS (/GRAPHICS/PRIMITIVES) x0 x1 y0 y1 wmin wmax ndiv [chopt]	247
BANX (/HISTOGRAM/CREATE) id ymin ymax	221
BANY (/HISTOGRAM/CREATE) id xmin xmax	221
BINS (/HISTOGRAM/CREATE) id title ncx xbins [valmax]	220
BOX (/GRAPHICS/PRIMITIVES) x1 x2 y1 y2	246
BREAK (/KUIP/SET_SHOW) [option]	203
CALL (/FORTRAN) urout	266
CDIR (/ZEBRA/RZ) [chpath chopt]	262
CLOSE (/FORTRAN) lun	267
CLR (/GRAPHICS/MISC)	243
COLOR_TABLE (/GRAPHICS/ATTRIBUTES) icol [red green blue]	255
COLUMNS (/KUIP/SET_SHOW) [ncol]	203
COMIS (/FORTRAN)	266
COMMAND (/KUIP/SET_SHOW) [chpath]	202
CONTENTS (/HISTOGRAM/GET_VECT) id vname	227
CONTENTS (/HISTOGRAM/PUT_VECT) id vname	228
CONTOUR (/HISTOGRAM/2D_PLOT) [id nlevel chopt param]	219
COPY (/HISTOGRAM) id1 id2 [title]	216
COPY (/PICTURE) pnam1 pnam2	259
COPY (/VECTOR) vnam1 vnam2	209
CREATE (/KUIP/ALIAS) aname string [chopt]	199
CREATE (/NTUPLE) idn title nvar chrzpa nprime varlist	234
CREATE (/PICTURE) pname	258
CREATE (/VECTOR) vname [type]	208
CSELECT (/NTUPLE) [chopt csiz]	238
CUTS (/NTUPLE) icut [option fname]	238
DDIR (/ZEBRA/RZ) chdir	261
DEFAULTS (/MACRO) [chpath]	206
DELETE (/HISTOGRAM) id	214
DELETE (/KUIP/ALIAS) alist	200
DELETE (/PICTURE) pname	258
DELETE (/VECTOR) vlist	209
DIFF (/HISTOGRAM/OPERATIONS) id1 id2 [chopt]	225
DIVIDE (/HISTOGRAM/OPERATIONS) id1 id2 id3 [c1 c2]	225
DRAW (/FUNCTION) ufunc [chopt]	231
DRAW (/VECTOR) vname [id chopt]	211
DUMP (/HISTOGRAM/HIO) id	223
EDIT (/KUIP) fname	197
ERRORS (/GRAPHICS/HPLOT) x y ex ey n [isymb ssize]	256
ERRORS (/HISTOGRAM/GET_VECT) id vname	227
ERRORS (/HISTOGRAM/PUT_VECT) id vname	228

Table A.1: Overview of PAW command sequences (continued)

Calling sequence	Page
EXEC (/MACRO) mname	206
EXIT (/KUIP)	198
EXPONENTIAL (/OBSOLETE/HISTOGRAM/FIT) id [isel iftlow iftup]	269
FAREA (/GRAPHICS/PRIMITIVES) n x y	246
FBOX (/GRAPHICS/PRIMITIVES) x1 x2 y1 y2 x3 x4 y3 y4	246
FILE (/FORTRAN) lun fname	267
FILE (/HISTOGRAM) lun fname [lrecl chopt]	214
FILE (/PICTURE) lun fname [lrecl chopt]	258
FILE (/ZEBRA/FZ) lun fname [lrecl chopt]	262
FILE (/ZEBRA/RZ) lun fname [lrecl chopt]	261
FIT (/HISTOGRAM) id func [chopt np par step pmin pmax errpar]	216
FIT (/VECTOR) x y ey func [chopt np par step pmin pmax errpar]	211
FRALPHA (/ZEBRA/FZ) fname	263
FREE (/ZEBRA/RZ) [chlock]	262
FRFZ (/ZEBRA/FZ) lun [chopt]	263
FUN1 (/FUNCTION) id ufunc ncx xmin xmax [chopt]	231
FUN2 (/FUNCTION) id ufunc ncx xmin xmax ncy ymin ymax [chopt]	231
FUNCTION (/HISTOGRAM/GET_VECT) id vname	227
FUNCTION (/OBSOLETE/HISTOGRAM/FIT) id func np dpar [isel iftlow iftup step pmin pmax]	270
FUNCTIONS (/KUIP)	199
GAUSS (/OBSOLETE/HISTOGRAM/FIT) id [isel iftlow iftup]	269
GLOBAL_SECT (/HISTOGRAM/HIO) gname	223
GRAPH (/GRAPHICS/PRIMITIVES) n x y [chopt]	253
GRESET (/HISTOGRAM/HIO) id	223
GRID (/GRAPHICS/HPLOT)	257
HELP (/KUIP) [item]	197
HFETCH (/HISTOGRAM/HIO) id fname	222
HFILL (/VECTOR) vname id	211
HIST (/GRAPHICS/PRIMITIVES) n x y [chopt]	251
HMOVE (/GRAPHICS/MISC)	244
HOST_EDITOR (/KUIP/SET_SHOW) [option top left width height dxpad dy pad npads]	203
HOST_SHELL (/KUIP/SET_SHOW) [option]	204
HREAD (/HISTOGRAM/HIO) id fname	222
HRIN (/HISTOGRAM/HIO) id [icycle iofset]	221
HROUT (/HISTOGRAM/HIO) id [chopt]	222
HSCRATCH (/HISTOGRAM/HIO) id	222
HSETPR (/HISTOGRAM/OPERATIONS) param value	227
IDOPT (/HISTOGRAM/SET) id option	229
IGSET (/PICTURE) [chatt value]	260
INPUT (/VECTOR) vname	209
ITX (/GRAPHICS/PRIMITIVES) x y text	250
IZIN (/PICTURE) pname [icycle]	259
IZOUT (/PICTURE) [pname]	259
IZPICT (/PICTURE) pname [chopt]	259
KEY (/GRAPHICS/HPLOT) x y [isymb text]	256
LABELS (/GRAPHICS/PRIMITIVES) labnum nlabs chlabs	251
LAST (/KUIP) [n fname]	198
LDIR (/ZEBRA/RZ) [chpath chopt]	261
LEGO (/HISTOGRAM/2D_PLOT) [id theta phi chopt]	217
LINE (/GRAPHICS/PRIMITIVES) x1 y1 x2 y2	246
LINTRA (/NTUPLE) idn [chopt nevent ifirst nvars varlis]	240
LIST (/HISTOGRAM) [chopt]	214
LIST (/KUIP/ALIAS)	200

Table A.1: Overview of PAW command sequences (continued)

Calling sequence	Page
LIST (/MACRO) [mname]	206
LIST (/NTUPLE)	234
LIST (/PICTURE)	258
LIST (/VECTOR)	208
LISTHELP (/KUIP) [1stnam]	197
LOCATE (/GRAPHICS/MISC) [ntpri chopt]	243
LOCK (/ZEBRA/RZ) [chlock]	262
LOOP (/FORTRAN) ntimes urout	267
LOOP (/NTUPLE) idn uwfunc [nevent ifirst]	235
MAKE (/ZEBRA/RZ) lun fname [lrecl nrec nwkey chform chtags]	261
MANUAL (/KUIP) [item outfil docsys]	197
MANY_PLOTS (/HISTOGRAM) idlist	215
MASK (/NTUPLE) mname [chopt number]	239
MAXIMUM (/HISTOGRAM/SET) id vmax	228
MDIR (/ZEBRA/RZ) chdir [nwkey chform chtags]	261
MERGE (/NTUPLE) idn1 idn2 [uwfunc nevent ifirst]	235
MERGE (/PICTURE) pname [x y scale chopt]	258
MESSAGE (/KUIP) [string]	198
METAFILE (/GRAPHICS) [lun metafl chmeta]	242
MINIMUM (/HISTOGRAM/SET) id vmin	228
MODE (/KUIP/SET_SHOW) mode	204
MODIFY (/PICTURE) [pname chopt]	258
MULTIPLY (/HISTOGRAM/OPERATIONS) id1 id2 id3 [c1 c2]	224
NEXT (/GRAPHICS/MISC)	243
NORMALIZE_FACTOR (/HISTOGRAM/SET) id [xnorm]	229
NULL (/GRAPHICS/HPLOT) [xmin xmax ymin ymax chopt]	257
OPTION (/GRAPHICS) [choptn]	242
OUTPUT_LP (/HISTOGRAM/HIO) [lun fname]	223
PALETTE (/GRAPHICS/ATTRIBUTES) palnb [nel list]	255
PANEL (/KUIP/SET_SHOW) line [gkey]	201
PARAM (/HISTOGRAM/OPERATIONS) id [isel r2min maxpow]	226
PAVE (/GRAPHICS/PRIMITIVES) x1 x2 y1 y2 [dz isbox isfram chopt]	251
PIE (/GRAPHICS/PRIMITIVES) x0 y0 radius n values [chopt iao ias iac]	249
PLINE (/GRAPHICS/PRIMITIVES) n x y	245
PLOT (/FUNCTION) ufunc xlow xup [chopt]	232
PLOT (/HISTOGRAM) [id chopt]	214
PLOT (/NTUPLE) idn [uwfunc nevent ifirst nupd option]	236
PLOT (/PICTURE) [pname]	258
PLOT (/VECTOR) vname [id chopt]	211
PMARKER (/GRAPHICS/PRIMITIVES) n x y	246
POINTS (/FUNCTION) [npx npy npz]	233
POLYNOMIAL (/OBSOLETE/HISTOGRAM/FIT) id ncoeff [isel iftlow iftup]	269
PRINT (/HISTOGRAM/HIO) id [chopt]	222
PRINT (/NTUPLE) idn	234
PRINT (/VECTOR) vname	209
PROFILE (/HISTOGRAM/CREATE) id title ncx xmin xmax ymin ymax [chopt]	220
PROJECT (/HISTOGRAM) id	216
PROJECT (/NTUPLE) idh idn [uwfunc nevent ifirst]	235
PROMPT (/KUIP/SET_SHOW) [option]	203
PROX (/HISTOGRAM/CREATE) id	221
PROY (/HISTOGRAM/CREATE) id	221
PURGE (/ZEBRA/RZ) [keep]	262
QUIT (/KUIP)	199

Table A.1: Overview of PAW command sequences (continued)

Calling sequence	Page
RANGE (/FUNCTION) [xlow xup ylow yup zlow zup]	233
READ (/NTUPLE) idn fname [format chopt nevent]	236
READ (/VECTOR) vlist fname [format opt match]	210
REBIN (/HISTOGRAM/GET_VECT) id x y ex ey [n ifirst ilast]	227
RECORDING (/KUIP/SET_SHOW) [nrec]	203
RECUSION (/MACRO) [option]	207
RENAME (/PICTURE) pname1 pname2	259
RESET (/HISTOGRAM/OPERATIONS) id [title]	225
REWIND (/FORTRAN) lun	267
RLOGIN (/NETWORK) host	272
ROOT (/KUIP/SET_SHOW) [path]	202
RSHELL (/NETWORK) message	272
SCALE_FACTOR_2D (/HISTOGRAM/SET) id [xscale]	229
SCAN (/NTUPLE) idn [chfunc nevent ifirst nvars varlis]	234
SCHH (/GRAPHICS/ATTRIBUTES) [chh]	254
SCRATCH (/PICTURE) pname [icycle]	258
SELNT (/GRAPHICS/VIEWING) nt	245
SET (/GRAPHICS) [chatt value]	242
SFACI (/GRAPHICS/ATTRIBUTES) [ifaci]	254
SFAIS (/GRAPHICS/ATTRIBUTES) [ints]	253
SFASI (/GRAPHICS/ATTRIBUTES) [styli]	254
SHELL (/KUIP) [cmd]	198
SHOW (/ZEBRA/DZ) name [number chopt]	263
SIGMA (/FORTRAN)	267
SIZE (/GRAPHICS/VIEWING) [xsize ysize]	244
SLIDE (/GRAPHICS)	243
SLIX (/HISTOGRAM/CREATE) id nslices	221
SLIY (/HISTOGRAM/CREATE) id nslices	221
SLN (/GRAPHICS/ATTRIBUTES) [iln]	253
SLWSC (/GRAPHICS/ATTRIBUTES) [lw]	254
SMK (/GRAPHICS/ATTRIBUTES) [mkt]	254
SMOOTH (/HISTOGRAM/OPERATIONS) id [isel]	226
SNAP (/ZEBRA/DZ) [idiv chopt]	264
SPLCI (/GRAPHICS/ATTRIBUTES) [iplci]	254
SPLINE (/HISTOGRAM/OPERATIONS) id [isel knotx kx]	226
SPMCI (/GRAPHICS/ATTRIBUTES) [ipmci]	254
STAT (/ZEBRA/RZ) clpath	262
STORE (/ZEBRA/DZ) [ixstor]	264
STXCI (/GRAPHICS/ATTRIBUTES) [itxci]	254
STXFP (/GRAPHICS/ATTRIBUTES) [ifont iprec]	254
STYLE (/KUIP/SET_SHOW) [option syleen sgysize sgyspa sgbord wktype]	200
SUBTRACT (/HISTOGRAM/OPERATIONS) id1 id2 id3 [c1 c2]	223
SURFACE (/HISTOGRAM/2D_PLOT) [id theta phi chopt]	218
SURV (/ZEBRA/DZ) name [number]	264
SVP (/GRAPHICS/VIEWING) nt x1 x2 y1 y2	244
SWITCH (/PICTURE) [chopt]	260
SWN (/GRAPHICS/VIEWING) nt x1 x2 y1 y2	245
SYMBOLS (/GRAPHICS/HPlot) x y n [isymb ssize]	255
TEXT (/GRAPHICS/PRIMITIVES) x y text size [angle chopt]	249
TICKS (/GRAPHICS/HPlot) [chopt xval yval]	256
TIMING (/KUIP/SET_SHOW) [option]	203
TITLE_GLOBAL (/HISTOGRAM/CREATE) [chtitl chopt]	221
TOALPHA (/ZEBRA/FZ) fname	263

Table A.1: Overview of PAW command sequences (continued)

Calling sequence	Page
TOFZ (/ZEBRA/FZ) lun [chopt]	263
TRACE (/MACRO) [option level prompt]	206
TRANSLATION (/KUIP/ALIAS) [option]	200
UNITS (/KUIP)	198
USAGE (/KUIP) [cmnd]	197
UWFUNC (/NTUPLE) idn fname [chopt]	239
VADD (/VECTOR/OPERATIONS) vnam1 vnam2 vnam3	212
VBIAS (/VECTOR/OPERATIONS) vnam1 bias vnam2	212
VDIVIDE (/VECTOR/OPERATIONS) vnam1 vnam2 vnam3	213
VERIFY (/ZEBRA/DZ) [idiv chopt]	264
VISIBILITY (/KUIP/SET_SHOW) cmd [chopt1 chopt2]	204
VLOCATE (/GRAPHICS/MISC) vecx vecy [chopt ntpri]	243
VMULTIPLY (/VECTOR/OPERATIONS) vnam1 vnam2 vnam3	212
VSCALE (/VECTOR/OPERATIONS) vnam1 scale vnam2	212
VSUBTRACT (/VECTOR/OPERATIONS) vnam1 vnam2 vnam3	213
WAIT (/KUIP) [string sec]	198
WORKSTATION (/GRAPHICS) iwkid [chopt iwtyp]	242
WRITE (/VECTOR) vlist [fname format chopt]	210
ZONE (/GRAPHICS/VIEWING) [nx ny ifirst chopt]	244
ZOOM (/HISTOGRAM) [id chopt icmin icmax]	215

Table A.2: Overview of PAW commands by function

Calling sequence	Page
FORTRAN	
CALL urout	266
CLOSE lun	267
COMIS	266
FILE lun fname	267
LOOP ntimes urout	267
REWIND lun	267
SIGMA	267
FUNCTION	
ANGLE [theta phi]	233
DRAW ufunc [chopt]	231
FUN1 id ufunc ncx xmin xmax [chopt]	231
FUN2 id ufunc ncx xmin xmax ncy ymin ymax [chopt]	231
PLOT ufunc xlow xup [chopt]	232
POINTS [npx npy npz]	233
RANGE [xlow xup ylow yup zlow zup]	233
GRAPHICS	
METAFILE [lun metafl chmeta]	242
OPTION [choptn]	242
SET [chatt value]	242
SLIDE	243
WORKSTATION iwkid [chopt iwtyp]	242
ATTRIBUTES	
COLOR_TABLE icol [red green blue]	255

Table A.2: Overview of PAW commands by function (continued)

Calling sequence	Page
PALETTE palnb [nel list]	255
SCHH [chh]	254
SFACI [ifaci]	254
SFAIS [ints]	253
SFASI [styli]	254
SLN [iln]	253
SLWSC [lw]	254
SMK [mkt]	254
SPLCI [iplci]	254
SPMCI [ipmci]	254
STXCI [itxci]	254
STXFP [ifont iprec]	254
HPLOT	
ATITLE [xtit ytit]	257
ERRORS x y ex ey n [isymb ssize]	256
GRID	257
KEY x y [isymb text]	256
NULL [xmin xmax ymin ymax chopt]	257
SYMBOLS x y n [isymb ssize]	255
TICKS [chopt xval yval]	256
MISC	
CLR	243
HMOVE	244
LOCATE [ntpri chopt]	243
NEXT	243
VLOCATE vecx vecy [chopt ntpri]	243
PRIMITIVES	
ARC x1 y1 r1 [r2 phimin phimax]	248
ARROW x1 x2 y1 y2 [size]	247
AXIS x0 x1 y0 y1 wmin wmax ndiv [chopt]	247
BOX x1 x2 y1 y2	246
FAREA n x y	246
FBOX x1 x2 y1 y2 x3 x4 y3 y4	246
GRAPH n x y [chopt]	253
HIST n x y [chopt]	251
ITX x y text	250
LABELS labnum nlabs chlabs	251
LINE x1 y1 x2 y2	246
PAVE x1 x2 y1 y2 [dz isbox isfram chopt]	251
PIE x0 y0 radius n values [chopt iao ias iac]	249
PLINE n x y	245
PMARKER n x y	246
TEXT x y text size [angle chopt]	249
VIEWING	
SELNT nt	245
SIZE [xsize ysize]	244
SVP nt x1 x2 y1 y2	244

Table A.2: Overview of PAW commands by function (continued)

Calling sequence	Page
SWN nt x1 x2 y1 y2	245
ZONE [nx ny ifirst chopt]	244
HISTOGRAM	
COPY id1 id2 [title]	216
DELETE id	214
FILE lun fname [lrecl chopt]	214
FIT id func [chopt np par step pmin pmax errpar]	216
LIST [chopt]	214
MANY_PLOTS idlist	215
PLOT [id chopt]	214
PROJECT id	216
ZOOM [id chopt icmin icmax]	215
2D_PLOT	
CONTOUR [id nlevel chopt param]	219
LEGO [id theta phi chopt]	217
SURFACE [id theta phi chopt]	218
CREATE	
1DHISTO id title ncx xmin xmax [valmax]	219
2DHISTO id title ncx xmin xmax ncy ymin ymax [valmax]	220
BANX id ymin ymax	221
BANY id xmin xmax	221
BINS id title ncx xbins [valmax]	220
PROFILE id title ncx xmin xmax ymin ymax [chopt]	220
PROX id	221
PROY id	221
SLIX id nslices	221
SLIY id nslices	221
TITLE_GLOBAL [chtitl chopt]	221
GET_VECT	
ABSCISSA id vname	227
CONTENTS id vname	227
ERRORS id vname	227
FUNCTION id vname	227
REBIN id x y ex ey [n ifirst ilast]	227
HIO	
DUMP id	223
GLOBAL_SECT gname	223
GRESET id	223
HFETCH id fname	222
HREAD id fname	222
HRIN id [icycle iofset]	221
HROUT id [chopt]	222
HSCRATCH id	222
OUTPUT_LP [lun fname]	223
PRINT id [chopt]	222
OPERATIONS	
ADD id1 id2 id3 [c1 c2]	223

Table A.2: Overview of PAW commands by function (continued)

Calling sequence	Page
DIFF id1 id2 [chopt]	225
DIVIDE id1 id2 id3 [c1 c2]	225
HSETPR param value	227
MULTIPLY id1 id2 id3 [c1 c2]	224
PARAM id [isel r2min maxpow]	226
RESET id [title]	225
SMOOTH id [isel]	226
SPLINE id [isel knotx kx]	226
SUBTRACT id1 id2 id3 [c1 c2]	223
PUT_VECT	
CONTENTS id vname	228
ERRORS id vname	228
SET	
IDOPT id option	229
MAXIMUM id vmax	228
MINIMUM id vmin	228
NORMALIZE_FACTOR id [xnorm]	229
SCALE_FACTOR_2D id [xscale]	229
KUIP	
EDIT fname	197
EXIT	198
FUNCTIONS	199
HELP [item]	197
LAST [n fname]	198
LISTHELP [lstnam]	197
MANUAL [item outfil docsys]	197
MESSAGE [string]	198
QUIT	199
SHELL [cmd]	198
UNITS	198
USAGE [cmnd]	197
WAIT [string sec]	198
ALIAS	
CREATE aname string [chopt]	199
DELETE alist	200
LIST	200
TRANSLATION [option]	200
SET_SHOW	
APPLICATION [path cmdex]	202
BREAK [option]	203
COLUMNS [ncol]	203
COMMAND [chpath]	202
HOST_EDITOR [option top left width height dxpad dypad npads]	203
HOST_SHELL [option]	204
MODE mode	204
PANEL line [gkey]	201
PROMPT [option]	203

Table A.2: Overview of PAW commands by function (continued)

Calling sequence	Page
RECORDING [nrec]	203
ROOT [path]	202
STYLE [option sgylen sgsize sgyspa sgbord wktype]	200
TIMING [option]	203
VISIBILITY cmd [chopt1 chopt2]	204
MACRO	
DEFAULTS [chpath]	206
EXEC mname	206
LIST [mname]	206
RECURSION [option]	207
TRACE [option level prompt]	206
NETWORK	
NTUPLE	
CREATE idn title nvar chrzpa nprime varlist	234
CSELECT [chopt csizes]	238
CUTS icut [option fname]	238
LINTR idn [chopt nevent ifirst nvars varlis]	240
LIST	234
LOOP idn uwfunc [nevent ifirst]	235
MASK mname [chopt number]	239
MERGE idn1 idn2 [uwfunc nevent ifirst]	235
PLOT idn [uwfunc nevent ifirst nupd option]	236
PRINT idn	234
PROJECT idh idn [uwfunc nevent ifirst]	235
READ idn fname [format chopt nevent]	236
RLOGIN host	272
RSHELL message	272
SCAN idn [chfunc nevent ifirst nvars varlis]	234
UWFUNC idn fname [chopt]	239
OBSOLETE	
HISTOGRAM	
EXPONENTIAL id [isel iftlow iftup]	269
FUNCTION id func np dpar [isel iftlow iftup step pmin pmax]	270
GAUSS id [isel iftlow iftup]	269
POLYNOMIAL id ncoeff [isel iftlow iftup]	269
PICTURE	
COPY pname1 pname2	259
CREATE pname	258
DELETE pname	258
FILE lun fname [lrec1 chopt]	258
IGSET [chatt value]	260
IZIN pname [icycle]	259
IZOUT [pname]	259
IZPICT pname [chopt]	259
LIST	258
MERGE pname [x y scale chopt]	258
MODIFY [pname chopt]	258

Table A.2: Overview of PAW commands by function (continued)

Calling sequence	Page
PLOT [pname]	258
RENAME pname1 pname2	259
SCRATCH pname [icycle]	258
SWITCH [chopt]	260
VECTOR	
COPY vnam1 vnam2	209
CREATE vname [type]	208
DELETE vlist	209
DRAW vname [id chopt]	211
FIT x y ey func [chopt np par step pmin pmax errpar]	211
HFILL vname id	211
INPUT vname	209
LIST	208
PLOT vname [id chopt]	211
PRINT vname	209
READ vlist fname [format opt match]	210
WRITE vlist [fname format chopt]	210
OPERATIONS	
VADD vnam1 vnam2 vnam3	212
VBIAS vnam1 bias vnam2	212
VDIVIDE vnam1 vnam2 vnam3	213
VMULTIPLY vnam1 vnam2 vnam3	212
VSCALE vnam1 scale vnam2	212
VSUBTRACT vnam1 vnam2 vnam3	213
ZEBRA	
DZ	
SHOW name [number chopt]	263
SNAP [idiv chopt]	264
STORE [ixstor]	264
SURV name [number]	264
VERIFY [idiv chopt]	264
FZ	
FILE lun fname [lrecl chopt]	262
FRALPHA fname	263
FRFZ lun [chopt]	263
TOALPHA fname	263
TOFZ lun [chopt]	263
RZ	
CDIR [chpath chopt]	262
DDIR chdir	261
FILE lun fname [lrecl chopt]	261
FREE [chlock]	262
LDIR [chpath chopt]	261
LOCK [chlock]	262
MAKE lun fname [lrecl nrec nwkey chform chtags]	261
MDIR chdir [nwkey chform chtags]	261
PURGE [keep]	262

Table A.2: Overview of PAW commands by function (continued)

Calling sequence	Page
STAT chpath	262

Bibliography

- [1] V. Berezhnoi (editor). *COMIS – Compilation and Interpretation System*, Program Library L210. CERN, 1988.
- [2] R.Brun. *HBOOK users guide (Version 4.15)*, Program Library Y250. CERN, 1992.
- [3] R.Bock et al. *HIGZ Users Guide*, Program Library Q120. CERN, 1991.
- [4] R.Brun and H.Renshall. *HPLOT users guide*, Program Library Y251. CERN, 1990.
- [5] R.Brun and P.Zanarini. *KUIP – Kit for a User Interface Package*, Program library I202. CERN, 1988.
- [6] F.James and M.Roos. *MINUIT – Users Guide*, Program Library D506. CERN, 1981.
- [7] R.Brun, M.Goossens, and J.Zoll. *ZEBRA Users Guide*, Program Library Q100. CERN, 1991.
- [8] L. Lamport. *L^AT_EX A Document Preparation System*. Addison-Wesley, 1986.
- [9] Adobe. *PostScript Language Manual (Second Edition)*. Addison Wesley, 1990.
- [10] Graphics section. *Guide to computer graphics at CERN*, DD/US/1987. CERN, 1990.
- [11] R.Brun, F.Bruyant, M.Maire, A.C.McPherson, and P.Zanarini. *GEANT3* (DD/EE/81-1). CERN, 1987.
- [12] M. Brun, R. Brun, and F. Rademakers. *CMZ - A Source Code Management System*. CodeME S.A.R.L., 1991.
- [13] F.James. *Interpretation of the errors on parameters as given by MINUIT*, Supplement to “CERN Program Library Long writeup D506”. CERN, 1978.
- [14] F.James. Determining the statistical Significance of experimental Results. Technical Report DD/81/02 and CERN Report 81–03, CERN, 1981.
- [15] W.T.Eadie, D.Drijard, F.James, M.Roos, and B.Sadoulet. *Statistical Methods in Experimental Physics*. North-Holland, 1971.
- [16] H.J. Klein and J. Zoll. *PATCHY Reference Manual*, Program Library L400. CERN, 1988.
- [17] B.Segal. *The TCPAW package*. CERN, 1989.
- [18] R.Brun and B.Segal. *A distributed Physics Analysis workbench*. CERN, 1989.
- [19] Sun Microsystems. *Network File System Version 2*. Sun Microsystems, 1987.

Index

- *
 - IGSET parameter, 162
- * comment character, 25
- : subarray separation character, 27
- NETWORK, 272
- \$SIGMA, 104, 110
 - _ continuation character, 27
 - | comment character, 25
- 1DHIST0(/HISTOGRAM/CREATE), **219**
- 2DHIST0(/HISTOGRAM/CREATE), **220**
- 2SIZ
 - SET parameter, 164
- 3270G, 187
- A4
 - HPLOPT option, 163
- [*], macro argument, 98
- [@], macro return code, 98
- [#], macro argument number, 98
- [%var], indexed positional parameters, 98
- abbreviation, 10, 18, 27, 92, 94, 101
 - command, 92
- ABSCISSA(/HISTOGRAM/GET_VECT), **227**
- accents
 - French, 179
- active picture, 157
- ADD(/HISTOGRAM/OPERATIONS), **223**
- addressing of vectors, 108
- ALIAS, 101, 102
- alias, 10, 65, 101, 105
- alignment
 - horizontal, 172
 - vertical, 172
- alphanumeric
 - labels, 69, 166
 - menu mode, 95
- alphanumeric mode, 95
- ANAM, 103
- ANGLE(/FUNCTION), **233**
- ANUM, 103
- ANY, 112
 - ANY (SIGMA), **113**
- Apollo, 15
- application SIGMA, 110
- APPLICATION(/KUIP/SET_SHOW), **202**
- arc
 - border, 162
- ARC(/GRAPHICS/PRIMITIVES), **248**
- ARGS, 103
- arithmetic
 - expression, 96
 - operator, 96
- ARRAY, 107
- array, 107
 - filling, 111
 - in SIGMA, 110
- ARRAY (SIGMA), **111**
- ARROW(/GRAPHICS/PRIMITIVES), **247**
- ASIZ
 - SET parameter, 164
- asterisk size (for functions), 164
- ATITLE(/GRAPHICS/HPLOT), **257**
- attribute, 161
- AURZ
 - IGSET parameter, 162
 - SET parameter, 172
- automatic
 - storage of pictures, 172
- automatic naming of pictures, 162
- AVAL, 103
- AWLN
 - IGSET parameter, 162
- AXIS, 166
- axis
 - divisions, 167
 - labels
 - font and precision, 164
 - size, 164
 - labels offset, 162
 - labels size, 162
 - tick marks size, 162
 - values
 - font and precision, 164
 - size, 164
- AXIS(/GRAPHICS/PRIMITIVES), **247**
- band, 13
- BANX(/HISTOGRAM/CREATE), **221**
- BANY(/HISTOGRAM/CREATE), **221**
- bar

chart, 69, 163
 histogram
 offset, 164
 width, 164
BAR0
 IGSET parameter, 162
 SET parameter, 164
BARW
 IGSET parameter, 162
 SET parameter, 164
bash shell, 5
basic operator in SIGMA, 111
BASL
 IGSET parameter, 162
batch, 3, 16, 58
BCOL
 SET parameter, 164, 169
binning for Ntuple plots, 63
BINS(/HISTOGRAM/CREATE), 220
block (lego) plot, 53
boldface, 178
book histogram, 13
boolean value in SIGMA, 111
BORD
 IGSET parameter, 162
BOX
 HPLOPT option, 163
box
 around picture, 163
 border, 162
 fill area
 colour, 164
BOX option, 45
BOX(/GRAPHICS/PRIMITIVES), 246
break, 106
BREAK(/KUIP/SET_SHOW), 203
BREAKL, 97
BTYP
 SET parameter, 164, 169
BWID
 SET parameter, 164
CALL(/FORTRAN), 266
CASE, 97
case sensitivity, 25
CDF Command Definition File, 10
CDIR, 157
CDIR, 124
CDIR(/ZEBRA/RZ), 262
CERN Program Library
 NEW, 15
 OLD, 15
 PRO, 15
change directory, 123
character
 height, 178
 key size, 164
 shift, 164
CHHE
 IGSET parameter, 162
chisquare, 11
client, 193
CLOSE(/FORTRAN), 267
CLR(/GRAPHICS/MISC), 243
CMS, 15
 colon as separation character in array, 27
COLOR_TABLE(/GRAPHICS/ATTRIBUTES), 255
colour, 161, 169
COLUMNS(/KUIP/SET_SHOW), 203
COMIS, 12, 39, 45, 107, 109, 137
comis, 47
COMIS(/FORTRAN), 266
command
 STYLE AL, 95
 STYLE AN, 95
 STYLE U, 95
 abbreviation, 10, 18, 27, 33, 92
 definition file (CDF), 10
 mode, 23, 92
 parameter, 93
 list, 92
 mandatory, 18
 optional, 18
 search path, 15
 structure, 17, 91
 STYLE G, 95
 STYLE GP, 95
 STYLE M, 95
COMMAND(/KUIP/SET_SHOW), 202
comment
 and statistic size, 164
 font and precision, 164

star, 25
vertical bar, 25
common /PAWC/, 123
components
 of PAW, 9
 of vector, 100
CONTENTS(/HISTOGRAM/GET_VECT), 227
CONTENTS(/HISTOGRAM/PUT_VECT), 228
continuation character, 25
contour plot, 43, 53, 55
CONTOUR(/HISTOGRAM/2D_PLOT), 219
control operator in SIGMA, 111
COPY(/HISTOGRAM), 216
COPY(/PICTURE), 259
COPY(/VECTOR), 209
correlation, 11
Courier font, 181
CPTIME, 103
create
 histogram, 47
 Ntuple, 58
 vector, 107
CREATE(/KUIP/ALIAS), 199
CREATE(/NTUPLE), 234
CREATE(/PICTURE), 258
CREATE(/VECTOR), 208
cross-wires, 163
CSELECT(/NTUPLE), 238
CSHI
 IGSET parameter, 162
 SET parameter, 164, 178
CSIZ
 SET parameter, 164
current
 directory, 123
 picture, 157
cut, 7, 12, 65, 69, 132, 134
 graphical, 135
CUTS(/NTUPLE), 238
CZ, 187

DASH
 SET parameter, 164
dash mode for lines, 164
data presentation
 one-dimensional, 51
 two-dimensional, 53, 71
data structure, 123
DATE, 103
DATE
 HPLOPT option, 171
 SET parameter, 171
date, 171
 and hour on pictures, 163, 171
 position, 164
DDIR(/ZEBRA/RZ), 261
DECNET, 15, 187
default setting, 10, 163
default value, 93
DEFAULTS, 101
DEFAULTS(/MACRO), 206
DEL, 112
DELETE(/HISTOGRAM), 214
DELETE(/KUIP/ALIAS), 200
DELETE(/PICTURE), 258
DELETE(/VECTOR), 209
DEL (SIGMA), 113
delta function, 113
DI3000, 11
dialogue style, 10, 92
DIFF, 114
DIFF, 112
DIFF(/HISTOGRAM/OPERATIONS), 225
DIFF (SIGMA), 114
diologue
 style, 10
directory, 8
 change, 123
 current, 123
display, 15
distance
 x axis
 to labels, 164
 to to axis values, 164
 y axis
 to labels, 164
 to to axis values, 164
DIVIDE(/HISTOGRAM/OPERATIONS), 225
divisions, 167
DMOD
 SET parameter, 164
DO, 97

Domain, 15
DRAW(/FUNCTION), 231
DRAW(/VECTOR), 211
 driver, 15
DST, 12, 122, 125
 Data Summary Tape, 12
DUMP(/HISTOGRAM/HIO), 223
DVXR
 HPLOPT option, 163
DVYR
 HPLOPT option, 163
EDIT, 138
EDIT(/KUIP), 197
 editor, 186
ELSE, 97
 emacs, 5
 error
 bars, 163
 errors on fitted parameters, 142
ERRORS(/GRAPHICS/HPLOT), 256
ERRORS(/HISTOGRAM/GET_VECT), 227
ERRORS(/HISTOGRAM/PUT_VECT), 228
 event, 12, 57
 exception
 condition, 106
 handling, 106
 exchange input/output, 11
 exclamation mark, 94
 exclamation mark character
 place-holder, 18, 27
EXEC, 96–98, 101, 105, 158
 exec, 96
EXEC(/MACRO), 206
EXIT(/KUIP), 198
EXITM, 97
EXITM, 98
EXPONENTIAL(/OBSOLETE/HISTOGRAM/FIT), 269
 expression in SIGMA, 104
FACI
 IGSET parameter, 162
FAIS
 IGSET parameter, 162
 SET parameter, 174
FAREA(/GRAPHICS/PRIMITIVES), 246
FASI
 IGSET parameter, 162
 SET parameter, 174
FILE
 HPLOPT option, 171
 SET parameter, 171
 file name
 on pictures, 163, 171
 position, 164
FILE(/FORTRAN), 267
FILE(/HISTOGRAM), 214
FILE(/PICTURE), 258
FILE(/ZEBRA/FZ), 262
FILE(/ZEBRA/RZ), 261
 fill
 area, 169
 interior style, 174
 style index, 174
 histogram, 13
 vector, 107
 fill area
 colour index, 162
 interior style, 162
 style index, 162
 first page number, 164
FIT
 HPLOPT option, 171
 SET parameter, 171
 fit, 11, 13, 35, 141
 parameters on pictures, 163, 171
 values to be plotted, 164
 vector, 109
FIT(/HISTOGRAM), 216
FIT(/VECTOR), 211
 font, 161
 identifier, 178
 PostScript, 178
 software precision, 178
 text, 178
 fonts, 179, 181–185
FOR, 97
FORTRAN, 266–269
 interpreter, 45
FRALPHA(/ZEBRA/FZ), 263

- FREE(/ZEBRA/RZ), **262**
French accent, 179
FRFZ(/ZEBRA/FZ), **263**
FTP, 190
FTYP
 SET parameter, 169
FUN1(/FUNCTION), **231**
FUN2, 43
FUN2(/FUNCTION), **231**
FUNCTION, 231–234
function, 13, 39, 47, 103, 105
 fill area
 colour, 164
 type, 164
 in SIGMA, 112
 line width, 164
 one-dimensional, 39
 two-dimensional, 39, 43
FUNCTION(/HISTOGRAM/GET_VECT), **227**
FUNCTION(/OBSOLETE/HISTOGRAM/FIT), **270**
FUNCTIONS(/KUIP), **199**
- GAUSS(/OBSOLETE/HISTOGRAM/FIT), **269**
GDDM, 15
GDDM (IBM), 11
German umlaut, 179
GFON
 SET parameter, 164
GKS, 11, 15, 21, 156
 Graphical Kernel System, 14
GL (Silicon Graphics), 11
global
 section, 124, 187, 192
 title
 font and precision, 164
 size, 164
GLOBAL_SECT(/HISTOGRAM/HIO), **223**
GMR3D (Apollo), 11
GOTO, 97, 105
GOTO macro
 control, 99
 statement, 96
GPR, 15
GPR (Apollo), 11
GRAPH, 109
GRAPH(/GRAPHICS/PRIMITIVES), **253**
- graphical
 cut, 135
 menu mode, 95
 output, 109
GRAPHICS, 242–258
graphics
 editor, 186
 terminal, 15
graphics mode, 95
Greek, 178
GRESET(/HISTOGRAM/HIO), **223**
grid, 163
 line type, 164
GRID(/GRAPHICS/HPLOT), **257**
GRPLOT, 156
GSIZ
 SET parameter, 164
- hardware characters, 163
hatch style, 24, 33, 51, 174–176
HBOOK, 10, 58, 122, 137
 histogram file, 59
 Title, 163
HBOOKN, 58
HCDIR, 123, 124
HCOL
 SET parameter, 164, 169
HDERIV, 142
HELP, 15, 18
HELP(/KUIP), **197**
Helvetica font, 181
HESSE, 143
HFCNH, 141
HFCNV, 141
HFETCH(/HISTOGRAM/HIO), **222**
HFILL(/VECTOR), **211**
HFITH, 141
HFITV, 141
HIFIT, 151
HIGZ, 10, 21, 123, 137, 155, 156, 160, 161
 G mode, 156
 graphics editor, 186
 software font, 181
 Z mode, 156, 157
HIST, 109
HIST(/GRAPHICS/PRIMITIVES), **251**

HIST/PLOT, 158
 HISTOFILE, 129
 HISTOGRAM, 214–231
 histogram, 7, 13, 47, 122

- bar option, 31
- booking, 13
- creation, 31
- file, 59
- fill area
 - colour, 164
 - type, 164
- filling, 13
- hatch style, 33
- line width, 164
- maximum for scale, 164
- operations, 51
- presentation, 169
- title size, 164

HISTOGRAM/PLOT, 156
 history file, 10, 106
 history mechanism, 106
 HLIMIT, 58, 123
 HMAX

- SET parameter, 164

HMOVE(/GRAPHICS/MISC), **244**
 host, 15
 HOST_EDITOR(/KUIP/SET_SHOW), **203**
 HOST_SHELL(/KUIP/SET_SHOW), **204**

HPLOPT

- A4, 163
- BOX, 163
- DATE, 171
- DVXR, 163
- DVYR, 163
- FILE, 171
- FIT, 171
- HTIT, 163
- LINX, 163
- LINY, 163
- LINZ, 163
- NAST, 163
- NBAR, 163
- NCHA, 163
- NDAT, 163
- NEAH, 163
- NFIL, 163
- NFIT, 163
- NGRI, 163
- NOPG, 163
- NPT0, 163
- NSQR, 163
- NSTA, 163
- NTIC, 163
- NZFL, 163
- SOFT, 163
- STAT, 171
- TAB, 163
- VERT, 163

HPOINT, 10, 122, 137, 155, 160, 161
 HREAD(/HISTOGRAM/HIO), **222**
 HRFILE, 123
 HRIN, 123
 HRIN(/HISTOGRAM/HIO), **221**
 HROPEN, 59
 HROUT, 58, 123
 HROUT(/HISTOGRAM/HIO), **222**
 HPUT, 58
 HSCRATCH(/HISTOGRAM/HIO), **222**
 HSETPR(/HISTOGRAM/OPERATIONS), **227**

HTIT

- HPOINT option, 163

HTYP

- SET parameter, 164, 169

IBM, 15
 IBM 3192G graphics terminal, 15
 IDOPT(/HISTOGRAM/SET), **229**
 IF, 97
 IF macro

- control, 99
- statement, 96

IGSET (), **161**
 IGSET, 161, 162, 174, 178
 IGSET

- *, 162
- AURZ, 162
- AWLN, 162
- BAR0, 162
- BARW, 162
- BASL, 162
- BORD, 162
- CHHE, 162

CSHI, 162
 FACI, 162
 FAIS, 162
 FASI, 162
 LAOF, 162
 LASI, 162
 LTYP, 162
 LWID, 162
 MSCF, 162
 MTYP, 162
 PASS, 162
 PICT, 162
 PLCI, 162
 PMCI, 162
 SHOW, 162
 TANG, 162
 TMSI, 162
 TXAL, 162
 TXCI, 162
 TXFP, 162
IGSET(PICTURE), 260
 indexed positional parameters, [%var], 98
 initialisation, 16
 input histograms, 49
 input/output, 11
INPUT(VECTOR), 209
 integer or real divisions on axis, 163
 interactive, 3
 interrupt, 106
IQUEST, 59
ITX, 172, 173, 178
ITX(GRAPHICS/PRIMITIVES), 250
IZIN(PICTURE), 259
IZOUT(PICTURE), 259
IZPICT, 157
IZPICT(PICTURE), 259
KERNLIB, 137
KEY(GRAPHICS/HPLOT), 256
 KEYNUM, 103
 KEYVAL, 103
 KSIZ
 SET parameter, 164
KUIP, 10, 91, 123, 137, 197–206
 STYLE, 95
 system function
 CPTIME, 103
 DATE, 103
 LEN, 103
 LOWER, 103
 NUMVEC, 103
 RTIME, 103
 SIGMA, 103
 SUBSTRING, 103
 TIME, 103
 UPPER, 103
 VDIM, 103
 VEXIST, 103
 VLEN, 103
 vector, 109
KUSER, 95
 label, 41, 166
 in KUIP macro, 96
 text justification, 167
label:, 97
LABELS, 166
LABELS(GRAPHICS/PRIMITIVES), 251
LAOF
 IGSET parameter, 162
LASI
 IGSET parameter, 162
LAST, 106
LAST, 103, 106
LAST.KUMACOLD, 106
LAST.KUMAC, 106
LAST(KUIP), 198
LDIR, 129
LDIR(ZEBRA/RZ), 261
LEGO
 plot, 71
LEGO(HISTOGRAM/2D_PLOT), 217
LEN, 103
 length of
 basic dashed segment, 164
 X axis, 164
 Y axis, 164
LFON
 SET parameter, 164
 library functions in SIGMA, 120
 limits on fitted parameters, 142
 line

type, 174, 177
 width, 169
LINE(/GRAPHICS/PRIMITIVES), 246
 linear scale, 163
 lines, 161
LINTRA(/NTUPLE), 240
LINX
 HPLOPT option, 163
LINY
 HPLOPT option, 163
LINZ
 HPLOPT option, 163
 list histograms, 49
LIST(/HISTOGRAM), 214
LIST(/KUIP/ALIAS), 200
LIST(/MACRO), 206
LIST(/NTUPLE), 234
LIST(/PICTURE), 258
LIST(/VECTOR), 208
LISTHELP(/KUIP), 197
LOCATE(/GRAPHICS/MISC), 243
LOCK(/ZEBRA/RZ), 262
 logarithmic scale, 163
 logical
 expression, 96
 operator, 96
 logical operator in SIGMA, 111
LOGON . KUMAC, 106
 loop, 41, 67
 goto, 35
 if, 35
 label, 35
LOOP(/FORTRAN), 267
LOOP(/NTUPLE), 235
LOWER, 103
LS, 114
LS, 112
LS (SIGMA), 114
LTYPE
 IGSET parameter, 162
LTYPE
 SET parameter, 174
//LUN1, 124
LVMAX, 112
LVMAX (SIGMA), 115
LVMIM, 112
LVMIN (SIGMA), 115
LWID
 IGSET parameter, 162
MACHINE, 103
MACRO, 206–208
MACRO, 97, 98
Macro
 argument number
 [#], 98
 arguments
 [*], 98
 return code
 [@], 98
 macro, 10, 13, 96, 97
 GOTO, 96
 IF, 96
 label, 96
 ON ERROR GOTO, 96
 parameter, 10, 41
 READ, 96
 RETURN, 96
 statements, 96, 97
 variable, 99
 variables, 105
Macro Flow Control, 99
MAKE(/ZEBRA/RZ), 261
 mandatory parameter, 18
MANUAL(/KUIP), 197
MANY_PLOTS(/HISTOGRAM), 215
 marker, 29
 type, 174, 177
MASK, 133
 mask, 7, 13, 57, 67, 132, 134
MASK(/NTUPLE), 239
MAX, 112
MAXIMUM(/HISTOGRAM/SET), 228
MAX (SIGMA), 115
MAXV, 112
MAXV (SIGMA), 116
MDIR(/ZEBRA/RZ), 261
 menu, 10, 17
 alphanumeric, 95
 graphical
 panel, 95
 pull-down, 95

mode, 95
merge pictures, 79
MERGE(/NTUPLE), 235
MERGE(/PICTURE), 258
MESSAGE(/KUIP), 198
METAFILE, 156
metafile, 7, 14, 21, 45, 156
METAFILE(/GRAPHICS), 242
MIGRAD, 142, 143
MIN, 112
minimisation, 11, 141
MINIMUM(/HISTOGRAM/SET), 228
MIN (SIGMA), 115
MINUIT, 11, 141
MINV, 112
MINV (SIGMA), 116
MIPS, 3
mode
 HIGZ
 G mode, 156
 Z mode, 156, 157
MODE(/KUIP/SET_SHOW), 204
MODIFY, 186
MODIFY(/PICTURE), 258
MOTIF, 95
MSCF
 IGSET parameter, 162
MTYP
 IGSET parameter, 162
 SET parameter, 174
multiple dialogue, 92
MULTIPLY(/HISTOGRAM/OPERATIONS), 224

NAST
 HPLOPT option, 163
national character, 179
native input/output, 11
NBAR
 HPLOPT option, 163
NCHA
 HPLOPT option, 163
NCO, 112
NCO (SIGMA), 116
NDAT
 HPLOPT option, 163
NDVX
 SET parameter, 164, 167
NDVY
 SET parameter, 164
NEAH
 HPLOPT option, 163
NETWORK, 272
NEXT(/GRAPHICS/MISC), 243
NFIL
 HPLOPT option, 163
NFIT
 HPLOPT option, 163
NGRI
 HPLOPT option, 163
NOPG
 HPLOPT option, 163
NORMALIZE_FACTOR(/HISTOGRAM/SET), 229
NPTO
 HPLOPT option, 163
NSQR
 HPLOPT option, 163
NSTA
 HPLOPT option, 163
NTCUT, 134, 135
NTCUTS, 133
NTIC
 HPLOPT option, 163
NTMASK, 134
NTPLOT, 134
NTUPLE, 234–242
Ntuple, 7, 12, 65, 122, 131
 create, 58, 61
 cut, 69, 132
 examples, 57
 loop, 67
 mask, 67, 132
 plot, 61
 print, 61
 read
 text file, 61
 weight, 132
 write Ntuple RZ file, 61
NTUPLEPLOT, 132
Ntuples
 read, 63
NULL(/GRAPHICS/HPLOT), 257
number of

divisions for
 X axis, 164
 Y axis, 164
 passes for software characters, 164
NUMVEC, 103
NZFL
 HPLOPT option, 163
OBSOLETE, 269–272
OF ERROR, 97
ON ERROR, 97
ON ERROR GOTO, 96
ON ERROR GOTO, 96, 97
ON ERROR GOTO macro statement, 96
 one-dimensional
 data presentation, 51
 function, 39
 online help, 10
 operand, 96
 operating system, 8
 operation on vectors, 25, 108
 operator in SIGMA, 111
OP (SIGMA), **113**
OPTION (), **161**
OPTION, 156, 158, 161, 171
OPTION(/GRAPHICS), **242**
 optional parameter, 18
ORDER, 112
 order, 94
ORDER (SIGMA), **117**
OS, 103
OS9, 193
 module, 124, 187
OSF, 95
OSI, 187
 output histogram, 47
OUTPUT_LP(/HISTOGRAM/HIO), **223**
 overflow on RZ file, 60

 page
 format, 163
 number, 163
 number size, 164
PALETTE(/GRAPHICS/ATTRIBUTES), **255**
PAWMAIN, 123
 panel, 95
 menu, 17

PANEL(/KUIP/SET_SHOW), **201**
 paper orientation, 163
PARAM(/HISTOGRAM/OPERATIONS), **226**
 parameter, 10, 93
 errors (fit), 142
 list, 92
 parameter names, 94
 parameters, 97
PASS
 IGSET parameter, 162
 SET parameter, 164, 178
 path, 15
PAVE(/GRAPHICS/PRIMITIVES), **251**
PAW, 141
 access, 15
 command structure, 91
 entities, 21
 examples, 23
 initialisation, 16
 object, 21
 server, 187, 193
 structure, 9
 /PAWC/ common, 123
 /PAWC/ common, 123, 124
 //PAWC directory, 124
PAWINT, 123
PAWLOGON, 15–17
PCOL
 SET parameter, 164, 169
PG terminal type, 15
PICT
 IGSET parameter, 162
PICT/LIST, 157
PICTURE, 258–261
 picture, 8, 14, 156, 163
 fill area
 colour, 164
 type, 164
 line width, 164
PICTURE/CREATE, 157
PICTURE/FILE, 172
PID, 103
PIE, 109
PIE, 166
 pie chart, 81
PIE(/GRAPHICS/PRIMITIVES), **249**

place-holder
 exclamation mark character, 18, 27

PLCI
 IGSET parameter, 162

PLINE(/GRAPHICS/PRIMITIVES), **245**

PLOT
 commands, 21

PL0T(/FUNCTION), **232**

PL0T(/HISTOGRAM), **214**

PL0T(/NTUPLE), **236**

PL0T(/PICTURE), **258**

PL0T(/VECTOR), **211**

PL0THIS, 124

plotting
 E option, 51
 P and L options, 51

PMARKER(/GRAPHICS/PRIMITIVES), **246**

PMCI
 IGSET parameter, 162

POINTS(/FUNCTION), **233**

polyline
 colour index, 162
 type, 162
 width, 162

polymarker, 51
 colour index, 162
 scale factor, 162
 type, 162

POLYNOMIAL(/OBSOLETE/HISTOGRAM/FIT), **269**

PostScript, 14, 21, 45, 179, 181–185
 fonts, 178

prefix SIGMA, 110

presenter, 192, 193

PRINT
 commands, 21

PRINT(/HISTOGRAM/HIO), **222**

PRINT(/NTUPLE), **234**

PRINT(/VECTOR), **209**

PROD, 112

PROFILE(/HISTOGRAM/CREATE), **220**

PROF (SIGMA), **117**

PROJECT(/HISTOGRAM), **216**

PROJECT(/NTUPLE), **235**

projection, 13

PROMPT(/KUIP/SET_SHOW), **203**

PROX(/HISTOGRAM/CREATE), **221**

PROY(/HISTOGRAM/CREATE), **221**

PSIZ
 SET parameter, 164

PTO (Please Turn Over), 163

PTYP
 SET parameter, 164, 169

pull-down menu, 17, 95

PURGE(/ZEBRA/RZ), **262**

QUAD, 112

QUAD (SIGMA), **118**

QUEST, 96

QUIT(/KUIP), **199**

RANGE(/FUNCTION), **233**

READ, 97, 98

READ macro statement, 96

READ(/NTUPLE), **236**

READ(/VECTOR), **210**

real time, 124

REBIN(/HISTOGRAM/GET_VECT), **227**

recalling previous commands, 106

RECORDING, 106

RECORDING(/KUIP/SET_SHOW), **203**

RECUSION(/MACRO), **207**

recursive, 101

remote
 access, 129, 187
 file, 191
 login, 191, 193
 shell, 191, 193

RENAME(/PICTURE), **259**

REPEAT, 97

replay, 11

RESET(/HISTOGRAM/OPERATIONS), **225**

RETURN, 97

RETURN macro statement, 96

REWIND(/FORTRAN), **267**

RLOGIN, 191, 193

RLOGIN(/NETWORK), **272**

ROOT(/KUIP/SET_SHOW), **202**

RSHELL, 191, 193

RSHELL(/NETWORK), **272**

RTIME, 103

RZ
 file, 59

RZ file, 11

SCALE_FACTOR_2D(/HISTOGRAM/SET), 229
SCAN, 131
 scan, 65
SCAN(/NTUPLE), 234
 scatter plot, 49
 (BOX option), 53, 55
 and table character size, 164
 table, 122
SCHH(/GRAPHICS/ATTRIBUTES), 254
SCRATCH(/PICTURE), 258
 selection, 65
 function, 132, 134, 137
SELNT(/GRAPHICS/VIEWING), 245
 separator, 101
 server, 193
SET
 NDVX, 71
 NDVY, 71
SET (), 161
SET, 156, 161, 167, 171, 172
IGSET
 2SIZ, 164
 ASIZ, 164
 AURZ, 172
 BAR0, 164
 BARW, 164
 BCOL, 164, 169
 BTYP, 164, 169
 BWID, 164
 CSHI, 164, 178
 CSIZ, 164
 DASH, 164
 DATE, 171
 DMOD, 164
 FAIS, 174
 FASI, 174
 FCOL, 169
 FILE, 171
 FIT, 171
 FTYP, 169
 GFON, 164
 GSIZ, 164
 HCOL, 164, 169
 HMAX, 164
 HTYP, 164, 169
 KSIZ, 164
 LFON, 164
 LTYPE, 174
 MTYP, 174
 NDVX, 164, 167
 NDVY, 164
 PASS, 164, 178
 PCOL, 164, 169
 PSIZ, 164
 PTYP, 164, 169
 SSIZ, 164
 STAT, 171
 TFON, 164
 TSIZ, 164
 TXAL, 173
 TXFP, 172, 178
 VFON, 164
 VSIZ, 164
 XLAB, 164
 XMGL, 164
 XMGR, 164
 XSIZ, 164
 XTIC, 164
 XVAL, 164
 XWIN, 164
 YGTI, 164
 YHTI, 164
 YLAB, 164
 YMGL, 164
 YMGU, 164
 YNPG, 164
 YSIZ, 164
 YTIC, 164
 YVAL, 164
 YWIN, 164
SET , 161
SET(/GRAPHICS), 242
SET/PROMPT, 106
SFACI(/GRAPHICS/ATTRIBUTES), 254
SFAIS(/GRAPHICS/ATTRIBUTES), 253
SFASI(/GRAPHICS/ATTRIBUTES), 254
SHELL, 158
 shell
 bash, 5
 tcsh, 5
SHELL(/KUIP), 198
SHIFT, 97

SHOW
 IGSET parameter, 162
SHOW(/ZEBRA/DZ), 263
SIGMA, 12, 73, 75, 107, 108, 110–121
 \$SIGMA, 110
 access, 110
 APPLication SIGMA, 110
 array, 110
 filling, 111
 structure, 111
 basic operator, 111
 boolean value, 111
 control operator, 111
 expression, 104
 function, 112
 library functions, 120
 logical operator, 111
 prefix SIGMA, 110
 vector, 110
SIGMA, 103
SIGMA(/FORTRAN), 267
SIZE(/GRAPHICS/VIEWING), 244
slice, 13
SLIDE(/GRAPHICS), 243
slides, 86
SLIX(/HISTOGRAM/CREATE), 221
SLIY(/HISTOGRAM/CREATE), 221
SLN(/GRAPHICS/ATTRIBUTES), 253
SLWSC(/GRAPHICS/ATTRIBUTES), 254
SMK(/GRAPHICS/ATTRIBUTES), 254
SM00TH(/HISTOGRAM/OPERATIONS), 226
SNAP(/ZEBRA/DZ), 264
SOFT
 HPLOPT option, 163
software
 characters, 163, 180
 fonts, 178
special parameters, 98
special symbols, 20, 178
SPLCI(/GRAPHICS/ATTRIBUTES), 254
SPLINE(/HISTOGRAM/OPERATIONS), 226
SPMCI(/GRAPHICS/ATTRIBUTES), 254
SSIZ
 SET parameter, 164
STAT
 HPLOPT option, 171
 SET parameter, 171
 statistic
 analysis, 11
 parameters on pictures, 163, 171
 values to be plotted, 164
STORE, 93
STORE(/ZEBRA/DZ), 264
structure of PAW, 9
STXCI(/GRAPHICS/ATTRIBUTES), 254
STXFP(/GRAPHICS/ATTRIBUTES), 254
STYLE
 AN, 95
 G, 95
 GP, 95
STYLE, 92, 103
style, 7
style of dialogue, 10
STYLE(/KUIP/SET_SHOW), 200
subarray uses colon as separation character, 27
subrange, 55
subscript, 178
SUBSTRING, 103
SUBTRACT(/HISTOGRAM/OPERATIONS), 223
SUMV, 112
SUMV (SIGMA), 119
superscript, 178
surface plot, 43, 53, 55
SURFACE(/HISTOGRAM/2D_PLOT), 218
SURV(/ZEBRA/DZ), 264
SVP(/GRAPHICS/VIEWING), 244
SWITCH
 Z, 157
SWITCH(/PICTURE), 260
SWN(/GRAPHICS/VIEWING), 245
Symbol font, 181, 185
symbols, 20
SYMBOLS(/GRAPHICS/HPLOT), 255
system function, 103
system functions, 105
TAB
 HPLOPT option, 163
TANG
 IGSET parameter, 162
TCP/IP, 15, 129, 187, 193

TCPAW, 187
 tcsh shell, 5
 Tektronix, 15
 TELNET, 187, 190
 TELNETG, 187
 TEXT, 162, 178
 text
 (and title) font and precision, 164
 alignment, 162
 angle, 162
 character height, 162
 colour index, 162
 data, 21
 font, 162, 172, 178
 precision, 162, 172
 width, 162
TEXT(/GRAPHICS/PRIMITIVES), 249
TFON
 SET parameter, 164
 tick marks, 41, 167
TICKS(/GRAPHICS/HPLOT), 256
TIME, 103
 Times font, 181, 183
TIMING(/KUIP/SET_SHOW), 203
 title font and precision, 164
TITLE_GLOBAL(/HISTOGRAM/CREATE), 221
TMSI
 IGSET parameter, 162
tn3270, 15
TOALPHA(/ZEBRA/FZ), 263
TOFZ(/ZEBRA/FZ), 263
TRACE(/MACRO), 206
TRANSLATION(/KUIP/ALIAS), 200
TSIZ
 SET parameter, 164
 two-dimensional
 data presentation, 53, 71
 function, 39
TXAL
 IGSET parameter, 162
 SET parameter, 173
TXCI
 IGSET parameter, 162
TXFP
 IGSET parameter, 162
 SET parameter, 172, 178
 umlaut
 German, 179
 underscore, 27
UNITS(/KUIP), 198
 Unix, 5
 unix, 15
UNTIL, 97
UPPER, 103
USAGE command, 20
USAGE(/KUIP), 197
user
 interface, 23
 routine, 47
 title, 163
user mode, 95
UWFUNC, 137
UWFUNC(/NTUPLE), 239
VADD(/VECTOR/OPERATIONS), 212
variable, 105
VAX, 15, 187
VAX/VMS, 192
Vaxstation, 15
VBIAS(/VECTOR/OPERATIONS), 212
VDIM, 103
VDIVIDE(/VECTOR/OPERATIONS), 213
VECTOR, 208–214
VECTOR, 107
vector, 13, 25, 107
 address, 108
 arithmetic, 108, 110
 component, 100
 create, 107
 draw, 31
 fill, 107
 in SIGMA, 110
 operations, 33, 110
 plot, 31
 print, 35
VEFIT, 151
VERIFY(/ZEBRA/DZ), 264
version, 15
VERT
 HPLOPT option, 163
VEXIST, 103
VFON

SET parameter, 164
VISIBILITY(/KUIP/SET_SHOW), 204
VLEN, 103
VLOCATE(/GRAPHICS/MISC), 243
VM-CMS, 15
VMAX, 112
VMAX (SIGMA), 119
VMIN, 112
VMIN (SIGMA), 119
VMS, 15, 192
VMULTIPLY(/VECTOR/OPERATIONS), 212
VSCALE(/VECTOR/OPERATIONS), 212
VSIZ
 SET parameter, 164
VSUBTRACT(/VECTOR/OPERATIONS), 213
VSUM, 112
VSUM (SIGMA), 120

WAIT(/KUIP), 198
weight, 132
weighting factor, 134
WHILE, 97
wildcard, 33
workstation, 3, 15
 type, 16
workstation type, 156
WORKSTATION(/GRAPHICS), 242
WRITE(/VECTOR), 210

X axis
 colour, 164
 tick marks length, 164
X margin
 left, 164
 right, 164
X space between windows, 164
X windows, 11, 15
X11, 15
XLAB
 SET parameter, 164
XMGL
 SET parameter, 164
XMGR
 SET parameter, 164
XSIZ
 SET parameter, 164
XTIC
 SET parameter, 164

SET parameter, 164
XVAL
 SET parameter, 164
XWIN
 SET parameter, 164

Y axis
 colour, 164
 tick marks length, 164
Y margin
 low, 164
 up, 164
Y position of
 global title, 164
 histogram title, 164
 page number, 164
Y space between windows, 164
YGTI
 SET parameter, 164
YHTI
 SET parameter, 164
YLAB
 SET parameter, 164
YMGL
 SET parameter, 164
YMGU
 SET parameter, 164
YNPG
 SET parameter, 164
YSIZ
 SET parameter, 164
YTIC
 SET parameter, 164
YVAL
 SET parameter, 164
YWIN
 SET parameter, 164

ZapfDingbats font, 181, 182, 184
ZEBRA, 8, 11, 59, 123, 137, 187, 261–266
 FRALFA, 21
 FZ file, 21
 RZ file, 21, 190
 TOALFA, 21
ZFL (option), 157
ZFL1 (option), 158
ZFTP, 190

zftp, 187
ZONE, 156
zone, 29
ZONE(/GRAPHICS/VIEWING), 244
ZOOM(/HISTOGRAM), 215