CERN Program Library Entry Q121

PAW - Physics Analysis Workstation

The Complete Reference

Version 1.07 (October 1989)

René Brun
Olivier Couet
Carlo Vandoni
Pietro Zanarini

Michel Goossens (Editor)

CERN Geneva, Switzerland

Copyright Notice

PAW - Physics Analysis Workstation

CERN Program Library entry **Q121**

Copyright CERN, Geneva 1989

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.

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.

Fifth corrected printing - June 1991

About this guide

Preliminary remarks

This **Complete Reference** of PAW (for **P**hysics **A**nalysis **W**orkstation), 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.

Notation

Throughout this manual, data or commands to be **entered** are underlined.

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 contibuted 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.

This document has been produced using a SGML [15] tagging scheme, the BOOKMASTER [16] macros and the IBM SCRIPT/DCF [13, 14] text-processing system on VM/CMS. All pictures shown are produced with PAW and are included in PostScript [17] format in the manual.

Related Manuals

This document can be complemented by the following manuals:

COMIS, Compilation and Interpretation System [1]
HBOOK User Guide — Version 4 [7]
HIGZ — High level Interface to Graphics and ZEBRA [2]
HPLOT User Guide — Version 5 [6]
KUIP — Kit for a User Interface Package [9]
MINUIT — Function Minimization and Error Analysis [20]
ZEBRA — Data Structure Management System [4]

Contents

PART 1. PAW - STEP BY STEP **1**

Chapter 1. A few words on PAW **3**
1.1 PAW terminology 3
1.2 The structure of the PAW system 4

Chapter 2. General principles **7**
2.1 Access to PAW 7
2.2 Initialising PAW 8
2.3 Command structure 9
2.4 Getting help 10
2.5 Special symbols for PAW 11
2.6 PAW entities and their related commands 12

Chapter 3. PAW by examples **13**
3.1 Vectors and elementary operations 14
3.2 One and two-dimensional functions 21
3.3 Using histograms 25
3.4 Examples with Ntuples 30
3.5 The SIGMA application and more complex examples 39

PART 2. PAW - COMMANDS AND CONCEPTS **49**

Chapter 4. User interface - KUIP **51**
4.1 The PAW command structure 51
4.2 Multiple dialogue styles 52
4.3 Macros 56
4.4 Aliases 60
4.5 System functions 62
4.6 On aliases, system functions and macro variables 63
4.7 Recalling previous commands 64
4.8 Exception condition handling 64

Chapter 5. Vectors **65**
5.1 Vector creation and filling 65
5.2 Vector addressing 66
5.3 Vector arithmetic operations 66
5.4 Vector arithmetic operations using SIGMA. 66
5.5 Using KUIP vectors in a COMIS routine 66
5.6 Usage of vectors with other PAW objects 67
5.7 Graphical output of vectors 67
5.8 Fitting the contents of a vector 67

Chapter 6. SIGMA **68**
6.1 Access to SIGMA 68
6.2 Vector arithmetic operations using SIGMA 69
6.3 SIGMA functions 70
6.4 Available library functions 77

Chapter 7. HBOOK **78**
7.1 Introduction 78
7.2 Basic ideas 79
7.3 HBOOK batch as the first step of the analysis 82
7.4 Using PAW to analyse data 85
7.5 Ntuples: A closer look 87

7.6 Fitting with PAW/HBOOK/MINUIT . . . 94
7.7 Doing more with MINUIT . . . 102

Chapter 8. Graphics (HIGZ and HPLOT) . . . 105
8.1 HPLOT, HIGZ and local graphics package . . . 105
8.2 The metafiles . . . 106
8.3 The HIGZ pictures . . . 106
8.4 HIGZ pictures generated in a HPLOT program . . . 109
8.5 Attributes setting . . . 110
8.6 More on labels . . . 114
8.7 Colour, line width, and fill area in HPLOT . . . 116
8.8 Information on histograms . . . 118
8.9 Additional details on some IGSET commands . . . 119
8.10 Text fonts . . . 124
8.11 The HIGZ graphics editor . . . 132

Chapter 9. Distributed PAW . . . 133
9.1 TELNETG and 3270G . . . 134
9.2 ZFTP . . . 136
9.3 Access to remote files from a PAW session . . . 137
9.4 Using PAW as a presenter on VMS systems . . . 138
9.5 Using PAW as a presenter on OS9 systems . . . 139

PART 3. PAW - REFERENCE SECTION . . . 141

Chapter 10. FORTRAN . . . 143

Chapter 11. FUNCTION . . . 146

Chapter 12. GRAPHICS . . . 148
12.1 MISC . . . 149
12.2 VIEWING . . . 150
12.3 PRIMITIVES . . . 151
12.4 ATTRIBUTES . . . 157
12.5 HPLOT . . . 158

Chapter 13. HISTOGRAM . . . 160
13.1 2D_PLOT . . . 164
13.2 CREATE . . . 164
13.3 HIO . . . 166
13.4 OPERATIONS . . . 167
13.5 GET_VECT . . . 170
13.6 PUT_VECT . . . 171
13.7 SET . . . 171

Chapter 14. KUIP . . . 173
14.1 ALIAS . . . 175
14.2 SET_SHOW . . . 176

Chapter 15. MACRO . . . 179

Chapter 16. NETWORK . . . 181

Chapter 17. NTUPLE . . . 182

Chapter 18. PICTURE . . . 188

Chapter 19. VECTOR . . . 191
19.1 OPERATIONS . . . 194

Chapter 20. ZEBRA . . . 196

20.1 RZ . . . 197
20.2 FZ . . . 198
20.3 DZ . . . 199

Appendix A. PAW commands and parameters . . . 200

Bibliography . . . 205

Index of entries . . . 206

Figures

1. PAW entities and their related commands . . . 12
2. Simple vector commands . . . 14
3. Further vector commands and writing vectors to disk . . . 15
4. Various data representations . . . 16
5. Difference between VECTOR/DRAW and VECTOR/PLOT . . . 17
6. Vector operations . . . 18
7. Simple macro with loop and vector fit . . . 19
8. Plotting one-dimensional functions . . . 21
9. One-dimensional functions and loops . . . 22
10. Plotting two-dimensional functions . . . 23
11. Plotting a two-dimensional function specified in a external file . . . 24
12. Creation of one- and two-dimensional histograms . . . 25
13. Reading histograms on an external file . . . 26
14. One-dimensional plotting and histogram operations . . . 27
15. Two-dimensional data representations . . . 28
16. The use of sub-ranges in histogram specifiers . . . 29
17. Ntuples - Creation and output to a file . . . 33
18. Ntuples - Automatic and user binning . . . 34
19. Ntuples - A first look at selection criteria . . . 35
20. Ntuples - Masks and loop . . . 36
21. Ntuples - Using cuts . . . 37
22. Ntuples - Two dimensional data representation . . . 38
23. Using the SIGMA processor - Trigonometric functions . . . 39
24. Using the SIGMA processor - More complex examples . . . 40
25. Histogram operations (Keep and Update) . . . 41
26. Merging several pictures into one plot . . . 42
27. Pie charts with hatch styles and PostScript colour simulation . . . 43
28. A complex graph with PAW . . . 44
29. Making slides with PAW using PostScript . . . 46
30. Example of the PAW command tree structure . . . 51
31. Example of KUIP command line mode . . . 52
32. Example of STYLE G . . . 54
33. Example of STYLE GP . . . 55
34. The layout of the /PAWC/ dynamic store . . . 79
35. The ZEBRA data structure used for two-dimensional histograms . . . 80
36. Schematic presentation of the various steps in the data analysis chain . . . 82
37. Writing data to HBOOK with the creation of a HBOOK RZ file . . . 83
38. Output generated by job HTEST . . . 83
39. Adding data to a HBOOK RZ file . . . 84
40. Reading a HBOOK direct access file . . . 85
41. Plot of one- and two-dimensional histograms . . . 86
42. Print and scan Ntuple elements . . . 87
43. Graphical definition of cuts . . . 90
44. Read and plot Ntuple elements . . . 92
45. Selection functions and different data presentations . . . 93
46. Example of a simple fit of a one-dimensional distribution . . . 98

47. Example of a fit using sub-ranges bins . . . 99
48. Example of a fit using a global double gaussian fit . . . 101
49. HPLOT and HIGZ in PAW . . . 105
50. Visualising a HIGZ picture produced in a batch HPLOT program . . . 109
51. A graphical view of the SET parameters . . . 113
52. Example of labelling for horizontal axes . . . 114
53. Example of labelling for vertical axes . . . 115
54. Example of fill area types in HPLOT . . . 117
55. Examples of HIGZ portable hatch styles . . . 121
56. GKSGRAL Device independent hatch styles . . . 122
57. HIGZ portable marker types . . . 123
58. HIGZ portable line types . . . 123
59. HIGZ portable software characters (Font 0, Precision 2) . . . 125
60. GKSGRAL software fonts . . . 126
61. PostScript grey level simulation of the basic colours . . . 126
62. PostScript fonts . . . 127
63. Correspondence between ASCII and ZapfDingbats font (-14) . . . 128
64. Octal PostScript characters in Times font . . . 129
65. Octal PostScript characters in ZapfDingbats font (-14) . . . 130
66. Octal PostScript characters in Symbol font . . . 131
67. The HIGZ graphics editor . . . 132
68. Example of a TELNETG session . . . 134
69. The TELNETG program . . . 135
70. Example of a ZFTP session . . . 136
71. Access to remote files from a workstation . . . 137
72. Visualising histograms in a global section . . . 138
73. Visualising histograms in an OS9 module . . . 139
74. Access to OS9 modules from PAW . . . 139
75. Description of the IDOPT parameters . . . 172

Tables

1. Special symbols . . . 11
2. Definition of the variables of the CERN staff Ntuple . . . 30
3. KUIP Macro Statements . . . 56
4. System Functions . . . 62
5. SIGMA functions . . . 70
6. Syntax for specifying Ntuple variables . . . 88
7. Syntax of a selection function used with a Ntuple . . . 88
8. Functions callable and common blocks which can be referenced from an external function with PAW . . . 91
9. Comparison of the results of the fit for the double gaussian distribution . . . 100
10. Parameters and default values in IGSET . . . 110
11. Parameters and default values in OPTION . . . 111
12. Parameters and default values in SET . . . 112
13. Text alignment parameters . . . 120
14. Codification for the HIGZ portable fill area interior styles . . . 120
15. List of HPLOT escape characters and their meaning . . . 124
16. PAW commands and parameters . . . 200

Part 1. PAW - Step by Step

Chapter 1. A few words on PAW **3**
1.1 PAW terminology 3
1.2 The structure of the PAW system 4

Chapter 2. General principles **7**
2.1 Access to PAW 7
2.2 Initialising PAW 8
2.3 Command structure 9
2.4 Getting help 10
2.5 Special symbols for PAW 11
2.6 PAW entities and their related commands 12

Chapter 3. PAW by examples **13**
3.1 Vectors and elementary operations 14
3.2 One and two-dimensional functions 21
3.3 Using histograms 25
3.4 Examples with Ntuples 30
3.5 The SIGMA application and more complex examples 39

Chapter 1. A few words on PAW

Personal workstations equipped with a 1 Mbit bitmap display, a speed of more than 4 MIPS, with at least 8 Mbytes of main memory and 150 Mbytes of local disk space (e.g. APOLLO, SUN, DEC and VAX workstations) are now becoming available at an affordable price for individual users. In order to exploit the full functionality of these workstations, at the beginning of 1986 the **P**hysics **A**nalysis **W**orkstation 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 better 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.

1.1 PAW terminology

PAW is a system for the interactive analysis of data, which integrates various data management and data presentation tools and packages, developed mainly for High Energy Physics applications over the last few years. Therefore, a number of terms used in this manual may sound unfamiliar to the non-HEP user, or they may have a meaning which is not rigorously the same as used in other fields. Several terms used in the next sections are defined below.

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), filled by explicit entry of data or can be derived from other histograms. The information stored about 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 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.

Ntuple Two-dimensional array, 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. Selection criteria can be applied to each entry of an element and a complete Ntuple can be analysed in a fast, efficient and interactive way.

DST Data Summary Tape, also, Data Summary File. A collection of experimental data, organized in a sequential way.

Event A subset of data in a DST, related to a particular physics event.

Cut A selection function used when analysing Ntuples.

Mask A file containing a bit pattern for each element of a Ntuple corresponding to the result of selection criteria applied to the Ntuple.

Macro A text file containing PAW commands and logical constructs to control the flow of execution. Parameters can be supplied to it when calling the 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.

GKS The **G**raphical **K**ernel **S**ystem 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 page description language in wide spread use on many laser printers, e.g. the popular Apple Laserwriter. The pictures and text of this manual have been produced in that language.

Function Sequence of one or more statements with a FORTRAN-like syntax entered on the command line or via an external file.

1.2 The structure of the PAW system

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). The PAW environment assists physicists in the analysis and presentation of their data. It provides interactive graphical presentation, statistical and mathematical tools. In principle PAW should be useful to any physicist with only a basic knowledge of computers. Indeed, to help the novice, extensive online guidance is built into the user interface and, if needed, a menu based dialogue can be requested. On the other hand, due to its extensibility with macros and user functions via the FORTRAN COMIS interface, PAW can become a powerful and efficient tool in the hands of a more experienced user.

On the inner front cover **the various components of PAW** are shown schematically. All these components are available in a library and can be called from any user application program either in batch or interactive mode. A short description of each of them follows.

KUIP - The user interface package

The purpose of KUIP (**K**it for a **U**ser **I**nterface **P**ackage) 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.

HBOOK and HPLOT - The histograming 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

HIGZ - The graphics interface package

A **H**igh level **I**nterface to **G**raphics and **Z**EBRA (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.

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.

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.

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.

SIGMA - The array manipulation language

A scientific computing programming language SIGMA (**S**ystem for **I**nteractive **G**raphical **M**athematical **A**pplications), 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.

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.

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.

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.

Apollo SR10

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.

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:    PAW ( X11 HOST yourhost
Vax/VMS:   PAW/X11/host=yourhost
Apollo:    paw -d X11 -h yourhost
```

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
```

Important modes to run PAW

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

```
On Apollo do:  PAW -b macroname
On VMS    do:  PAW/BATCH=macroname
On VM     do:  PAW (BATCH=macroname
```

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

```
On Apollo do:  PAW -n
On VMS    do:  PAW/NOLOG
On VM     do:  PAW (NOLOG
```

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 1.11/02  29 March 1991          *
 *                                                  *
 ****************************************************
 Workstation type (?=HELP) [CR]=10002 :
```

At CERN, with GKSGRAL, the following types may be used (for more details, see [10]):

```
      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)
  8601-6:  Vaxstation GPX
 10001-2:  Apollo DNXXXX monochrome (GPR)
 10003-4:  Apollo DNXXXX colour (GPR)
  9701-8:  Apollo DNXXXX (GSR)
```

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 the Apollo the file pawlogon.kumac, should be put into the directory user_data. 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[9] 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: * BANX             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 142. Moreover an item preceded by a **star** indicates a **terminal leaf** in the command tree, i.e. an **executable** command (see 4.1, “The PAW command structure” on page 51 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.
```

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 1 have a special connotation when working with PAW .

Table 1. Special symbols

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 the beginning of a command line: Unix C shell-like **history** list (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.

2.6 PAW entities and their related commands

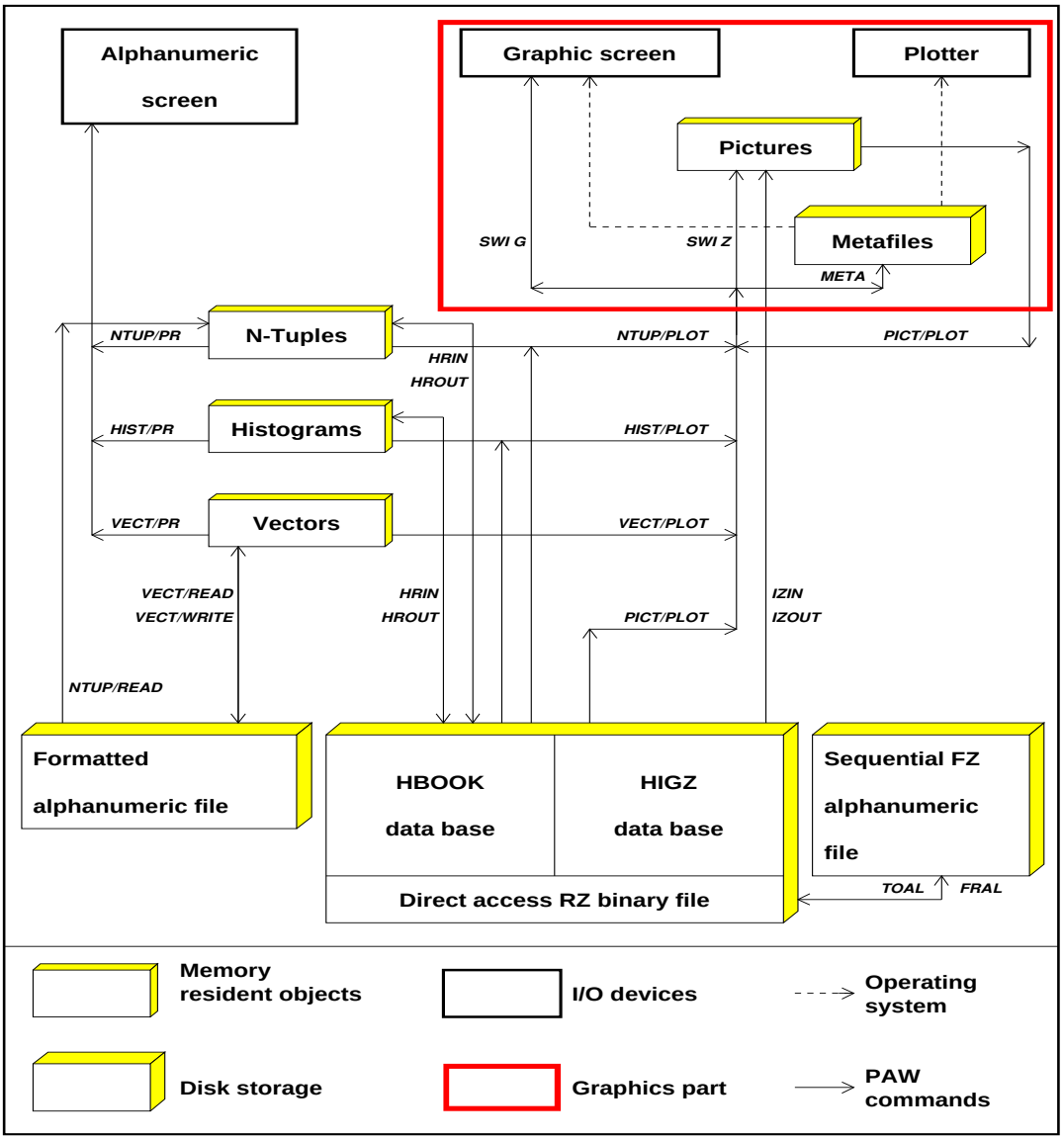


Figure 1. PAW entities and their related commands

Relations which exist between various PAW entities as described in section 1.1, “PAW terminology” and the operations which can be performed upon them have been schematically represented in Figure 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 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.

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 the 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 2.1, “Access to PAW” on page 7).

   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 `-111`. 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.

```
PAW > ************************************************************************
PAW > *                        TUTORIAL EXAMPLE PAWEX1                       *
PAW > * Some simple vector commands                                          *
PAW > * Lines starting with a * are comments and are ignored by PAW         *
PAW > ************************************************************************
PAW >                          | similarly everything following a VERTICAL BAR is ignored
PAW >                          | so this character can be used to provide INLINE COMMENTS
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 the values of the 10 elements
PAW >                          | ****  PAW commands are NOT CASE SENSITIVE
PAW > VECTOR/CREATE VX(20) R 1. 2. 3. 4. 5. 6. _             | Underscore _ is continuation character
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 the 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
```

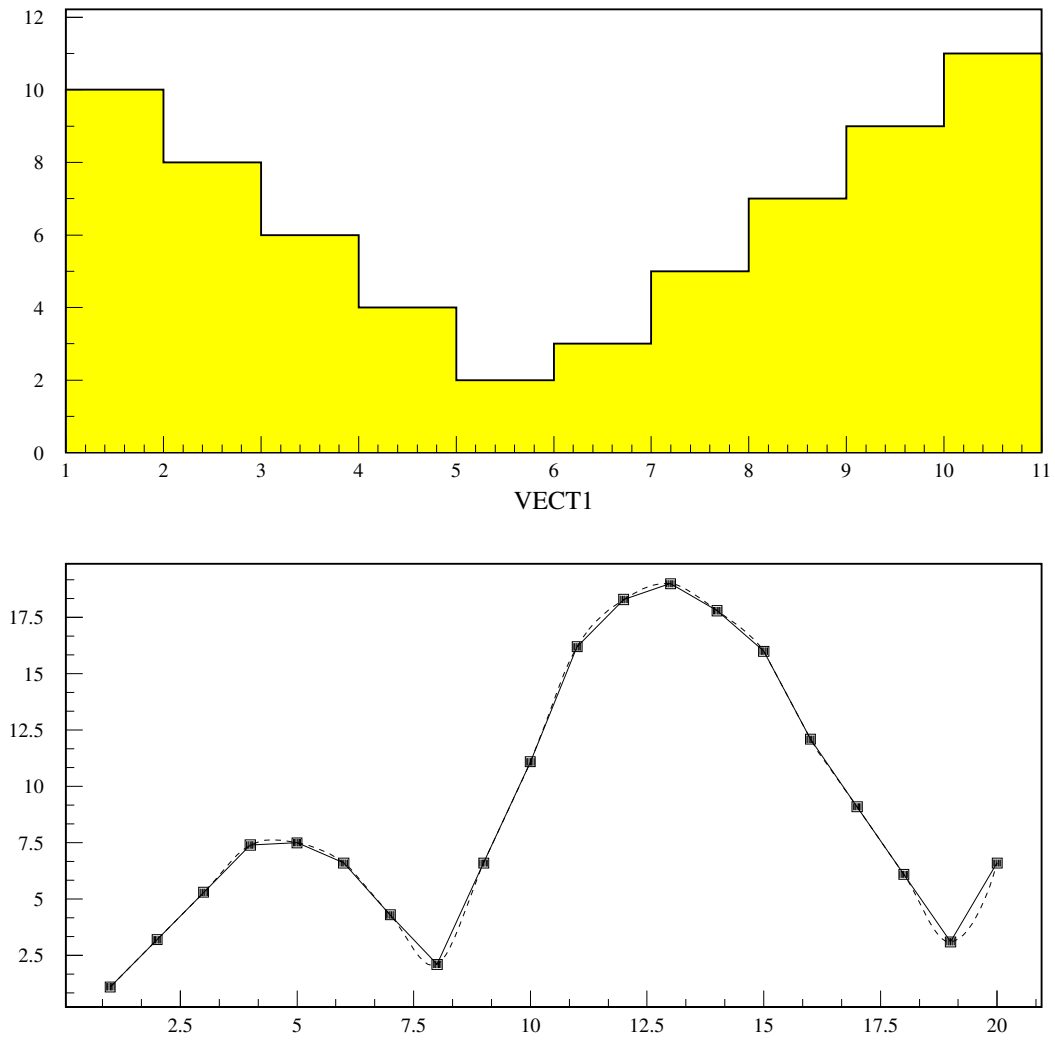


Figure 2. Simple vector commands

```
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 commands can be ABBREVIATED to their shortest non-ambiguous form
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 > ve/dr VECT(1:10,3) ! SC                     | Once more third row on VECT, now as continuous curve
PAW > set dmod 13                                 | Set line style
PAW > igset MTYP 29                               | Define marker type
PAW > igset MSCF 3.                               | Marker scale factor
PAW > set HWID 6                                  | Redefine histogram line width
PAW > vector/draw VECT1 ! LPS                     | Draw contents of VECT1 as line with marker
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 > 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
```

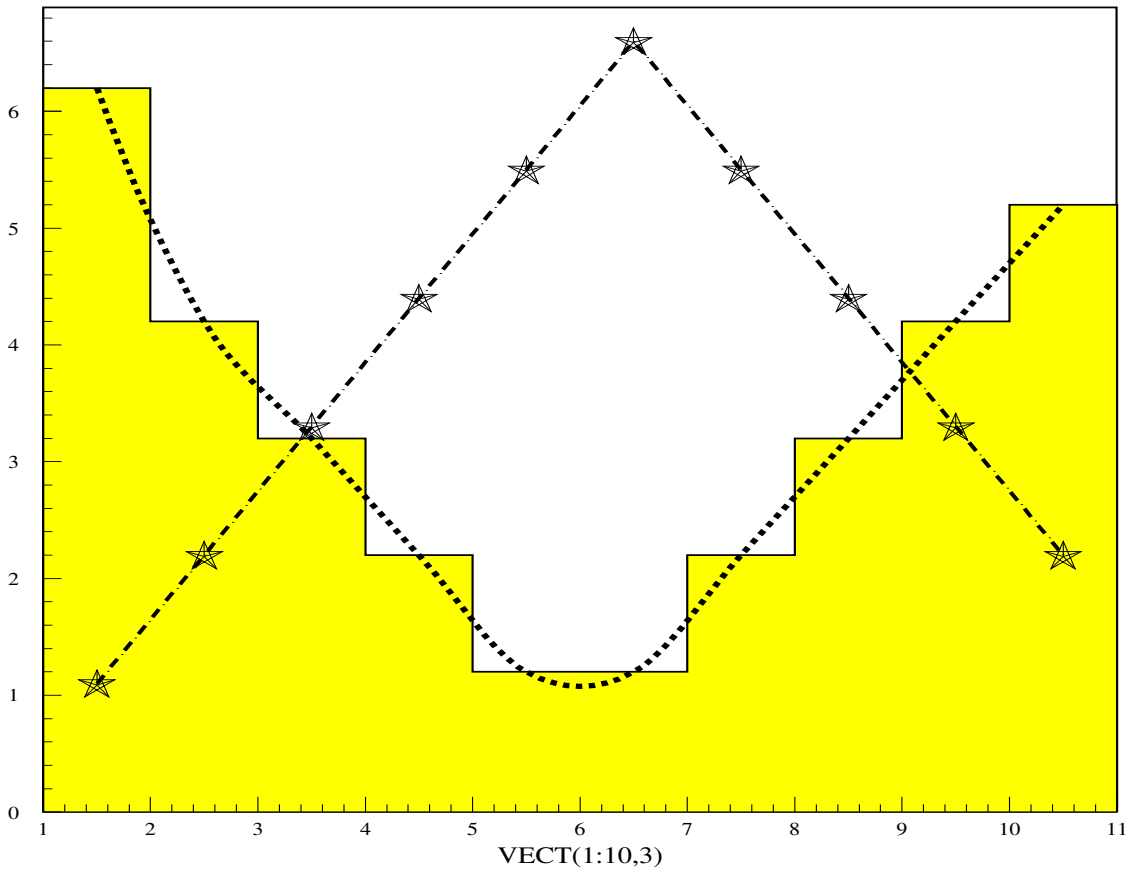


Figure 3. Further vector commands and writing vectors to disk

```
PAW > ********************************************************************************
PAW > *                        TUTORIAL EXAMPLE PAWEX3                               *
PAW > * Example showing various possible data representation with VECTOR/DRAW  *
PAW > ********************************************************************************
PAW > zone 2 3                                  | Divide picture page in 2 by 3 images
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
```

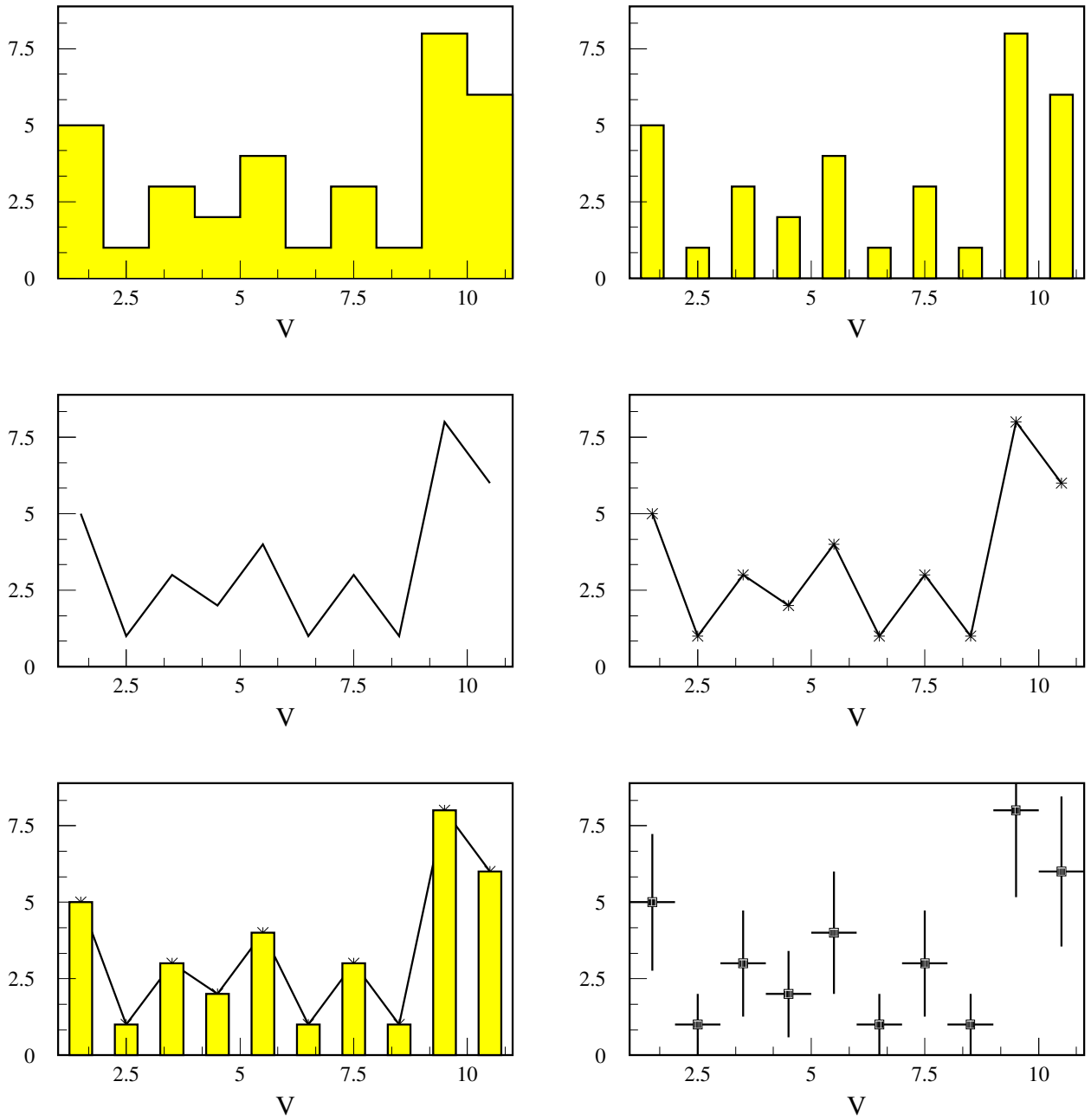


Figure 4. Various data representations

```
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 > * PLOT interprets contents of vector elements as values to be histogrammed
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
```

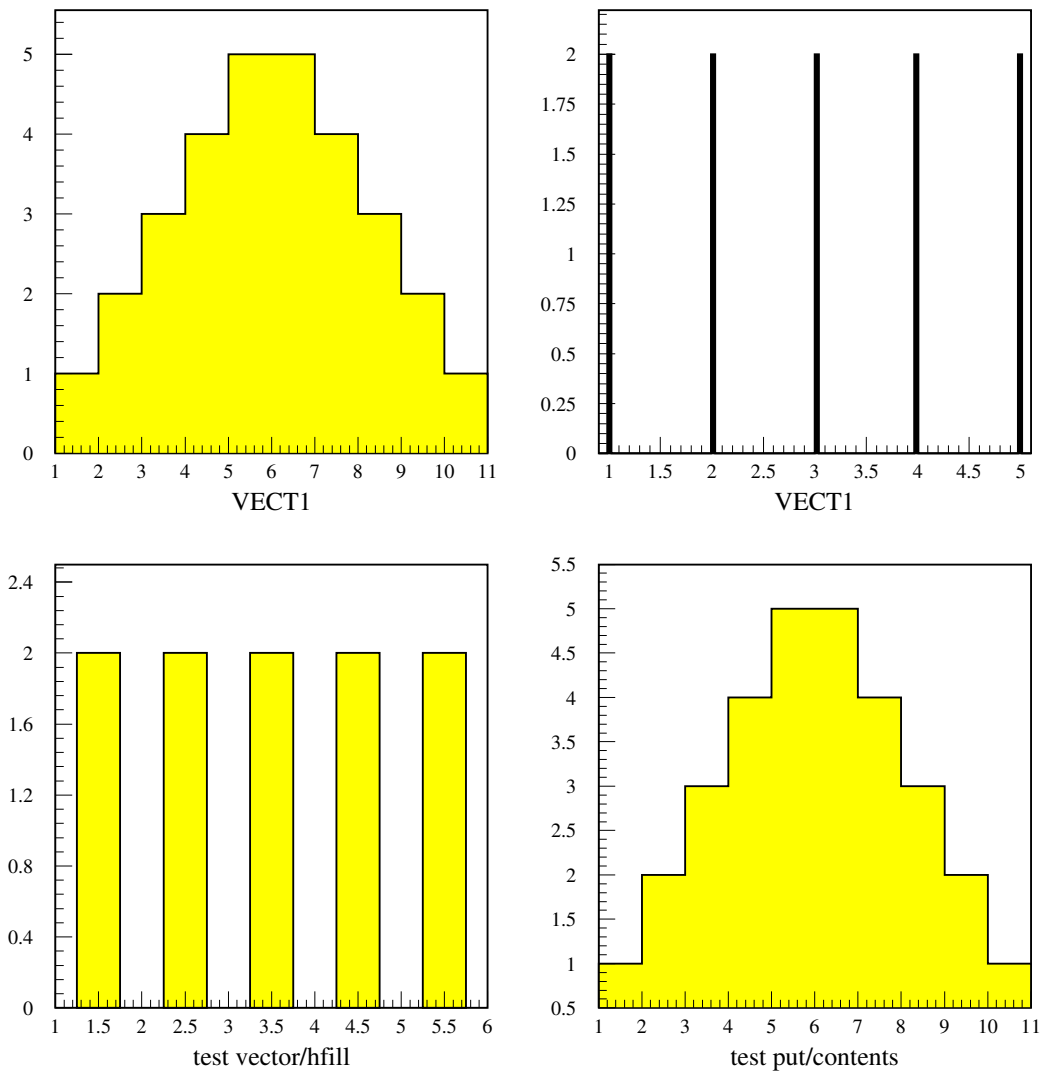
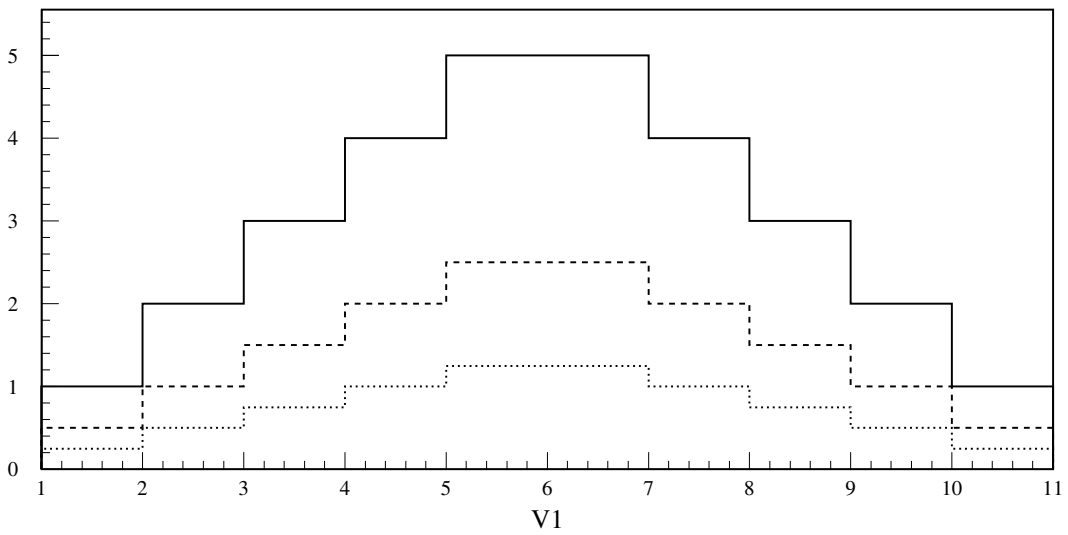


Figure 5. Difference between VECTOR/DRAW and VECTOR/PLOT

```
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 > *   Differences between various plots are shown using different line styles
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 > *   Differences between various plots are shown using different hatch styles
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
```



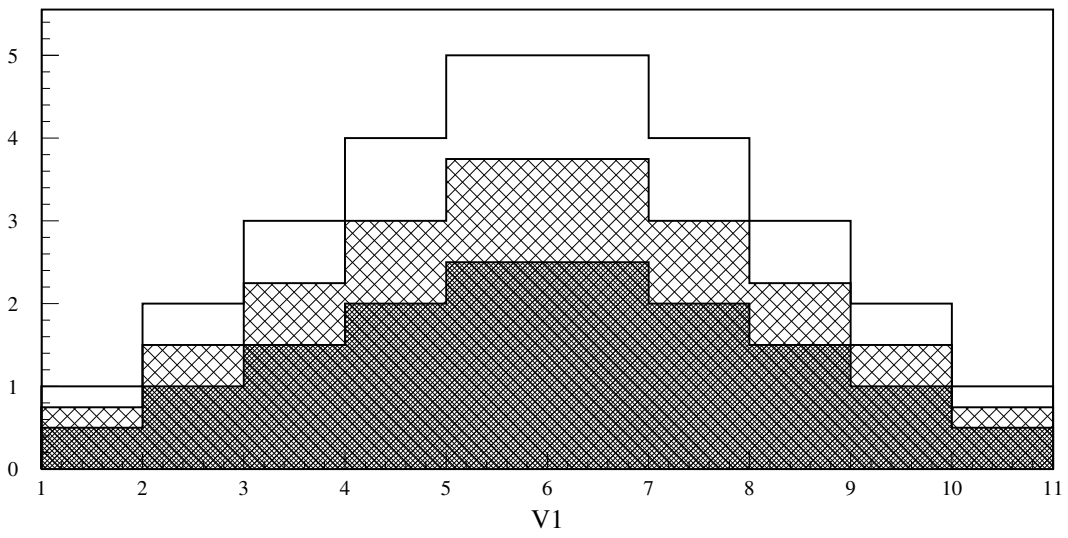
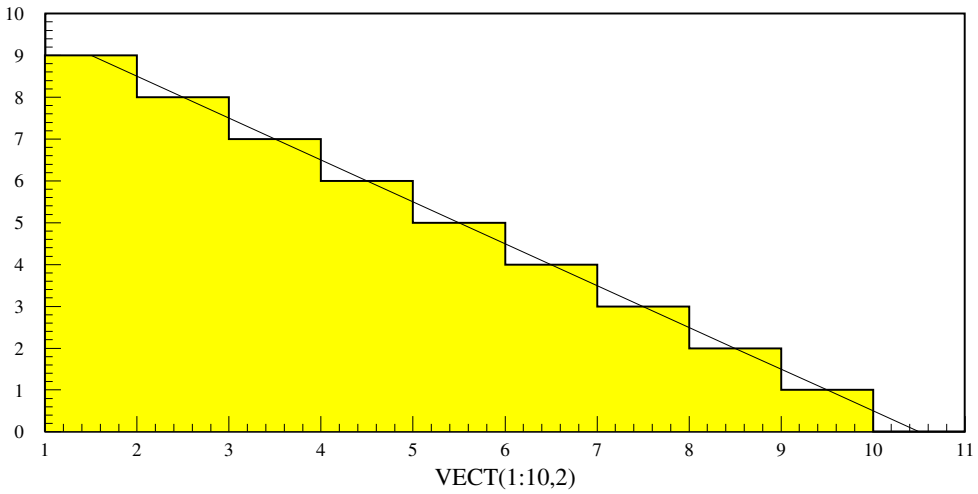


Figure 6. Vector operations

```
PAW > EDIT PAWEX6                                        | Edit MACRO PAWEX6.KUMAC
MACRO PAWEX6
*****************************************************************************
*                      TUTORIAL EXAMPLE PAWEX6                              *
* Example showing a simple macro, with a loop and a VECTOR fit.             *
*****************************************************************************
ve/create VECT(10,3)                                    | Create a 2 dimensional vector
ve/read VECT 'vector.data'                              | Read vector values from external file
                                                        |  created in example PAWEX2
ve/print VECT(1:10,3)                                   | Print last row of VECT (see Example 2)
vbias vect(1:10,1) 0.5 vect(1:10,1)
zone 1 2
ip = 2                                                  | Loop counter
set htyp -3
* Start of a loop over the three rows of vector VECT
LOOP:                                                   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  ve/draw VECT(1:10,[ip])                               | draw next row                       |
  order = ip - 1                                        | Define order for fit                |
  ve/fit VECT(1:10,1) VECT(1:10,[IP]) ! P[order] WS     | Fit polynomial to given row         |
  ip = ip + 1                                           | Increment counter                   |
  if ip < 4        GOTO LOOP                            | loop test >>>>>>>>>>>>>>>>>>>>>>>>>>>
ve/delete VECT                                          | Delete VECT
RETURN
```



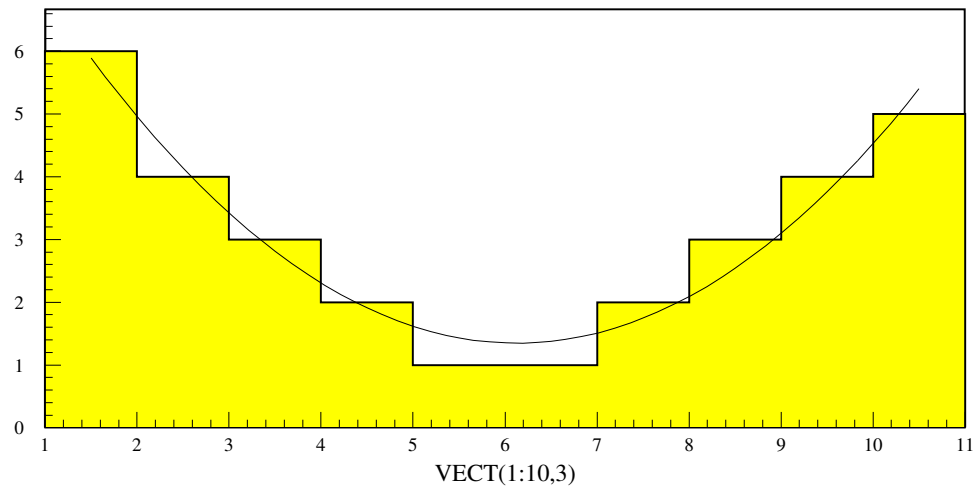


Figure 7 (Part 1 of 2). Simple macro with loop and vector fit

```
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
```

Figure 7 (Part 2 of 2). Simple macro with loop and vector fit

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**.

```
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
```

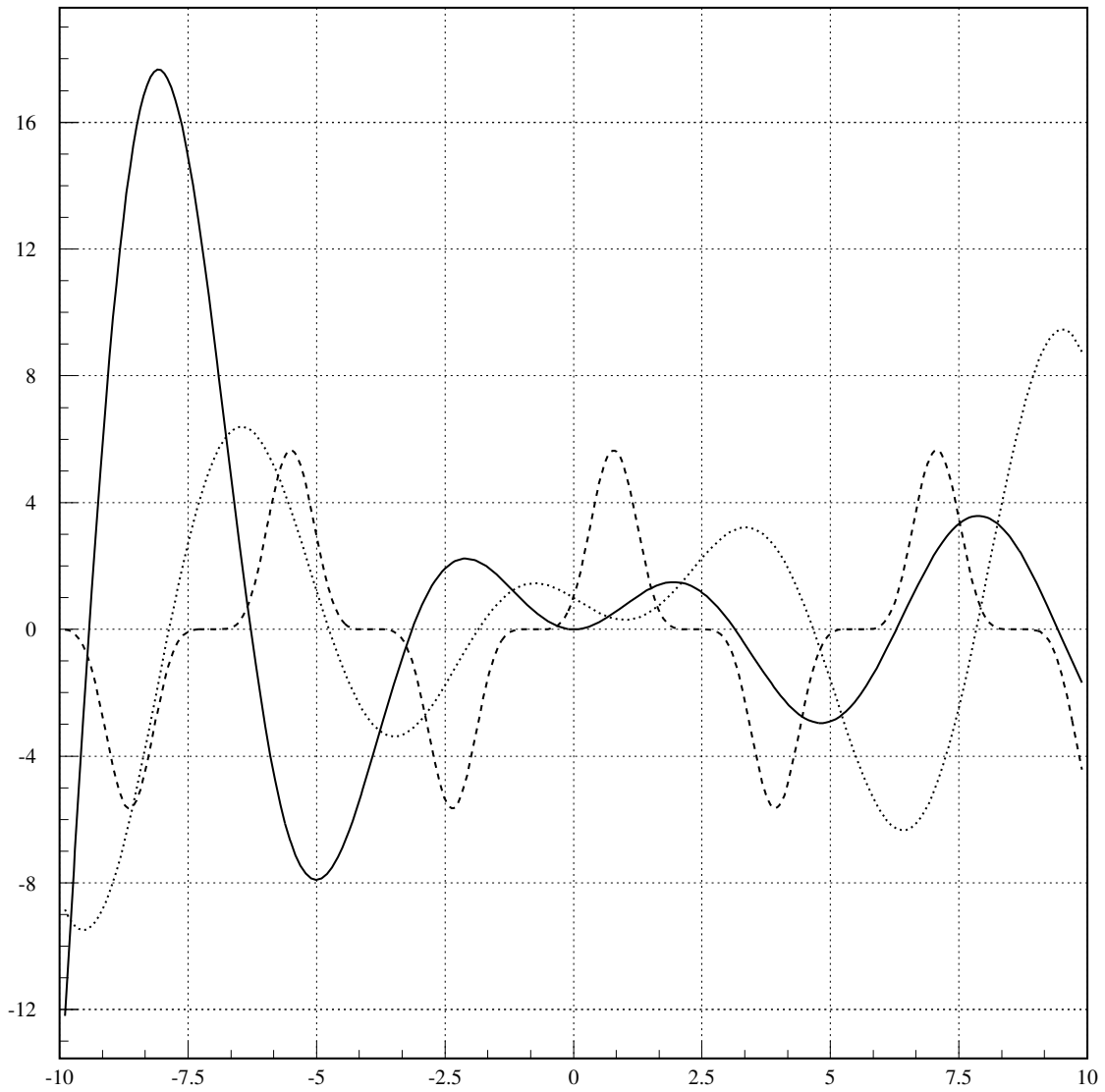


Figure 8. Plotting one-dimensional functions

```
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 > EXEC pawex8 12                       | Execute Macro file with first parameter = 12
PAW > opt htit                                          | Reset to histogram title
```

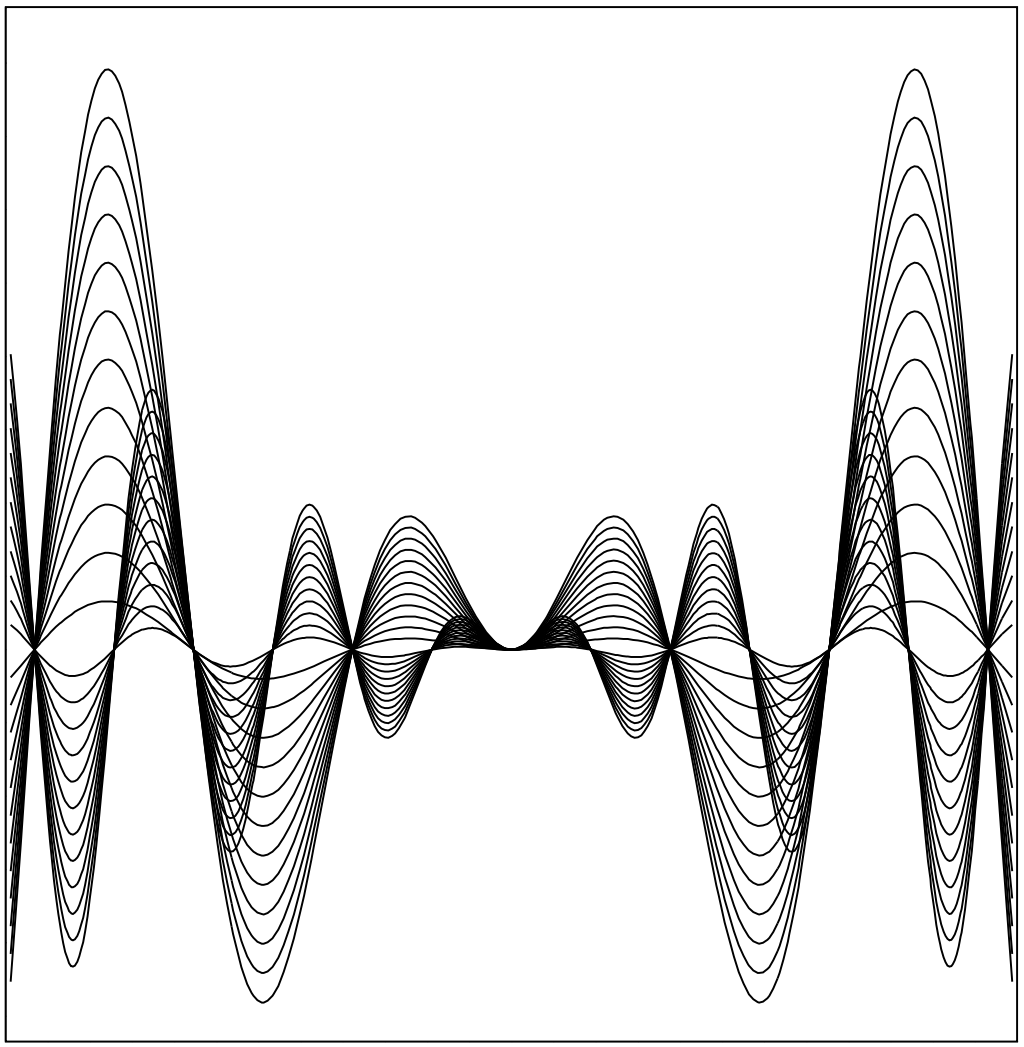
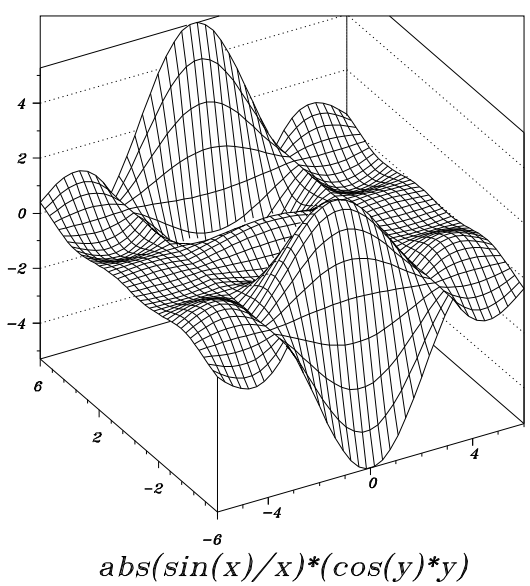
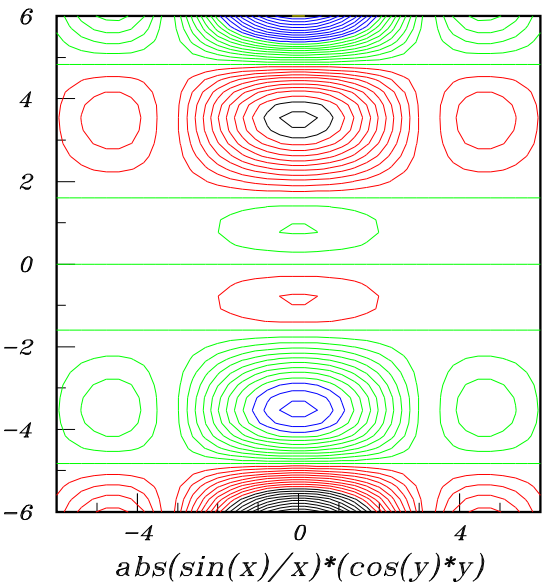
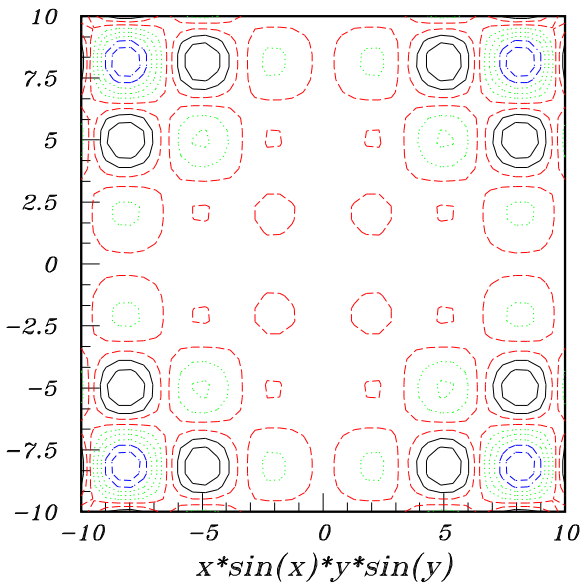


Figure 9. One-dimensional functions and loops

```
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
```







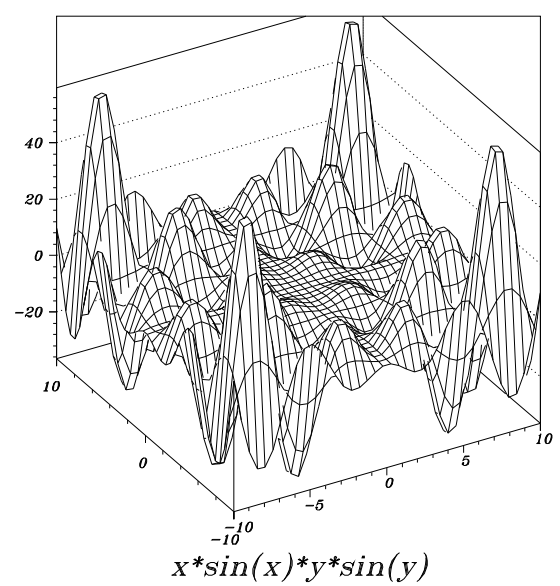


Figure 10. Plotting two-dimensional functions

```
PAW > ********************************************************************************
PAW > *                        TUTORIAL EXAMPLE PAWEX10                              *
PAW > * Calculate and plot (BOX option) Mandelbrot distribution                      *
PAW > ********************************************************************************
PAW > edit mandel.ftn                                      | 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 > fun2 10 mandel.ftn 100 -2.4 .8 100 -1.2 1.2 ' '   | Calculate function and plot
PAW > fortran/file 66 mandel.ps                          | Open "metafile" for PostScript image
PAW > metafile 66 -111                                   | HIGZ/PostScript portrait mode
PAW > hi/pl 10 BOX                                       | Plot histogram with BOX option
PAW > close 66                                           | Close metafile
```

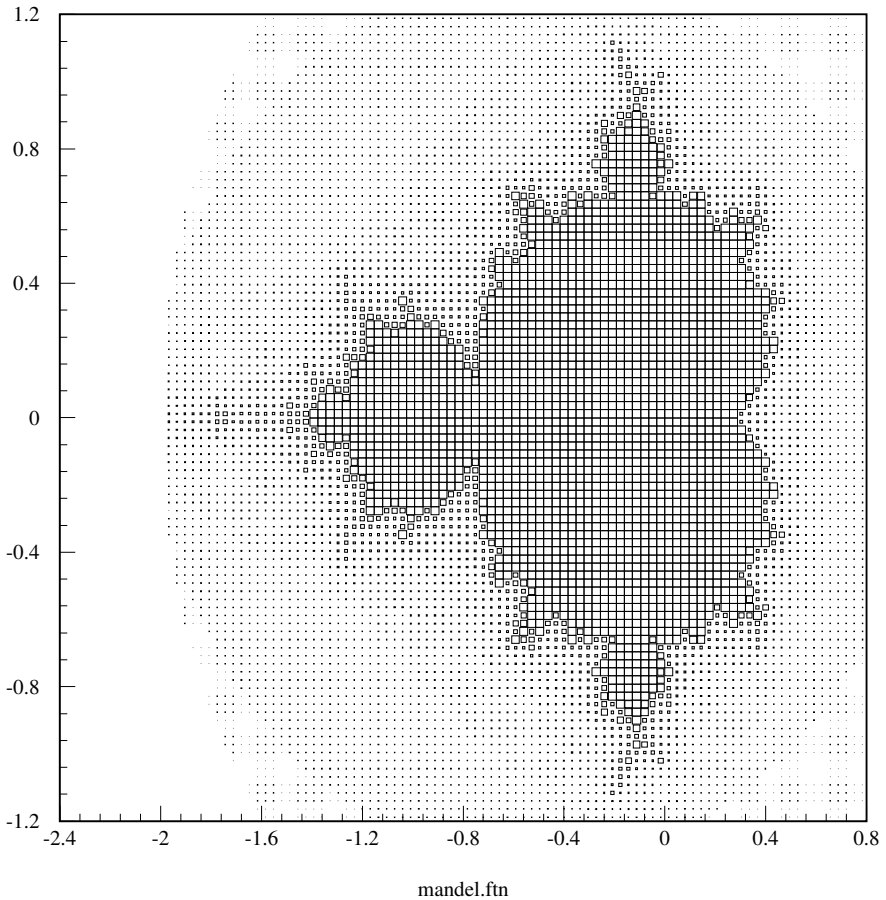
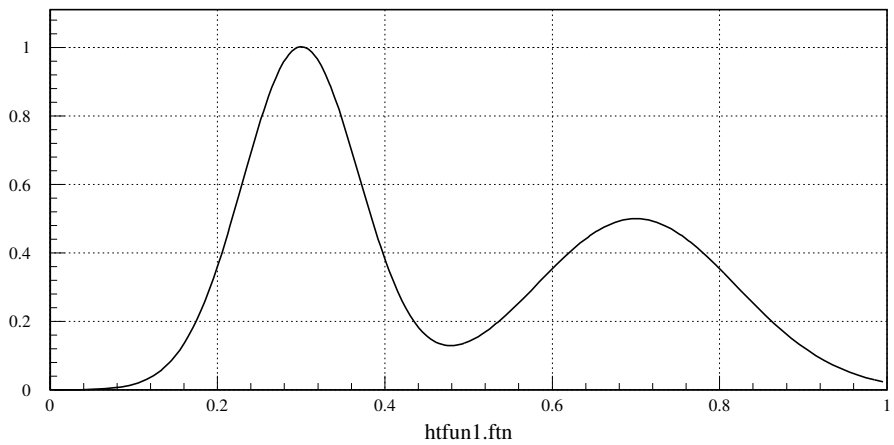


Figure 11. Plotting a two-dimensional function specified in a external file

3.3 Using histograms

```
 PAW > ************************************************************************
 PAW > *                      TUTORIAL EXAMPLE PAWEX11                        *
 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, fill 1-dimensional histogram 100 according to 1-dim function htfun1 and plot
 PAW > fun1 100 htfun1.ftn 100. 0. 1.
 PAW > ldhisto  110 'Test 1-dim Histo' 100 0. 1. 1000.    | Create a 1-dimensional histogram
 PAW > call urout.ftn(5000)                               | Fill histogram 110 according to function 100 (5000 events)
 PAW > * Fill and plot as a contour 2-dimensional histogram 200 according to the 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 > hrout 0                                            | Write all histograms
 PAW > close 1                                            | Close the output unit
 PAW > hi/de 0                                            | Delete all histograms in memory
```



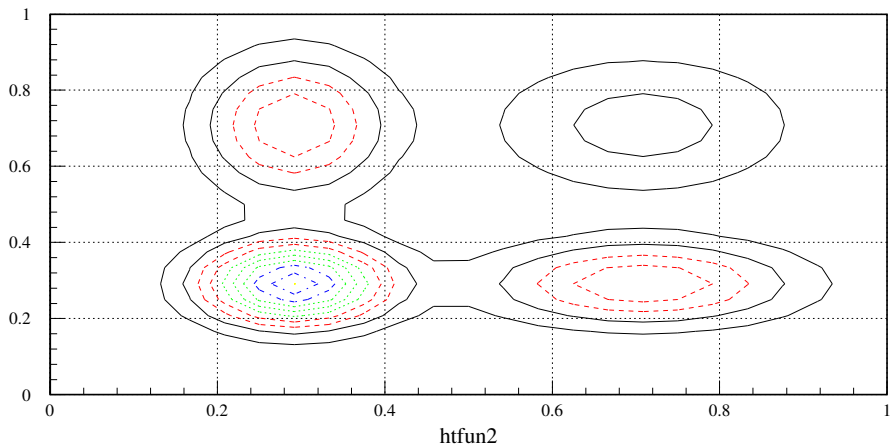


Figure 12. Creation of one- and two-dimensional histograms

```
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
```

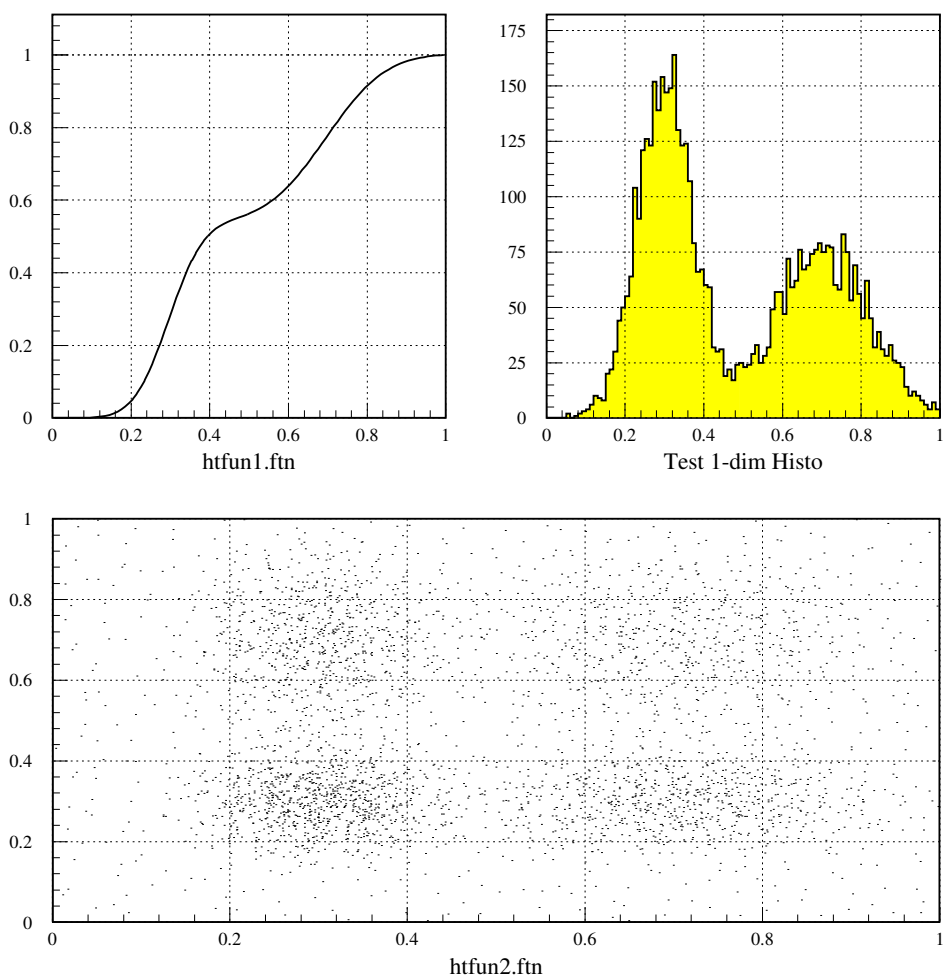


Figure 13. Reading histograms on an external file

```
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 120 on same plot
PAW > text 0.55 95. 'LEP Very Preliminary' 0.35 25.        | Add text to plot
PAW > hrout 0                                              | Write all histograms to the histogram file
PAW > close 1                                              | Close input/output unit
```

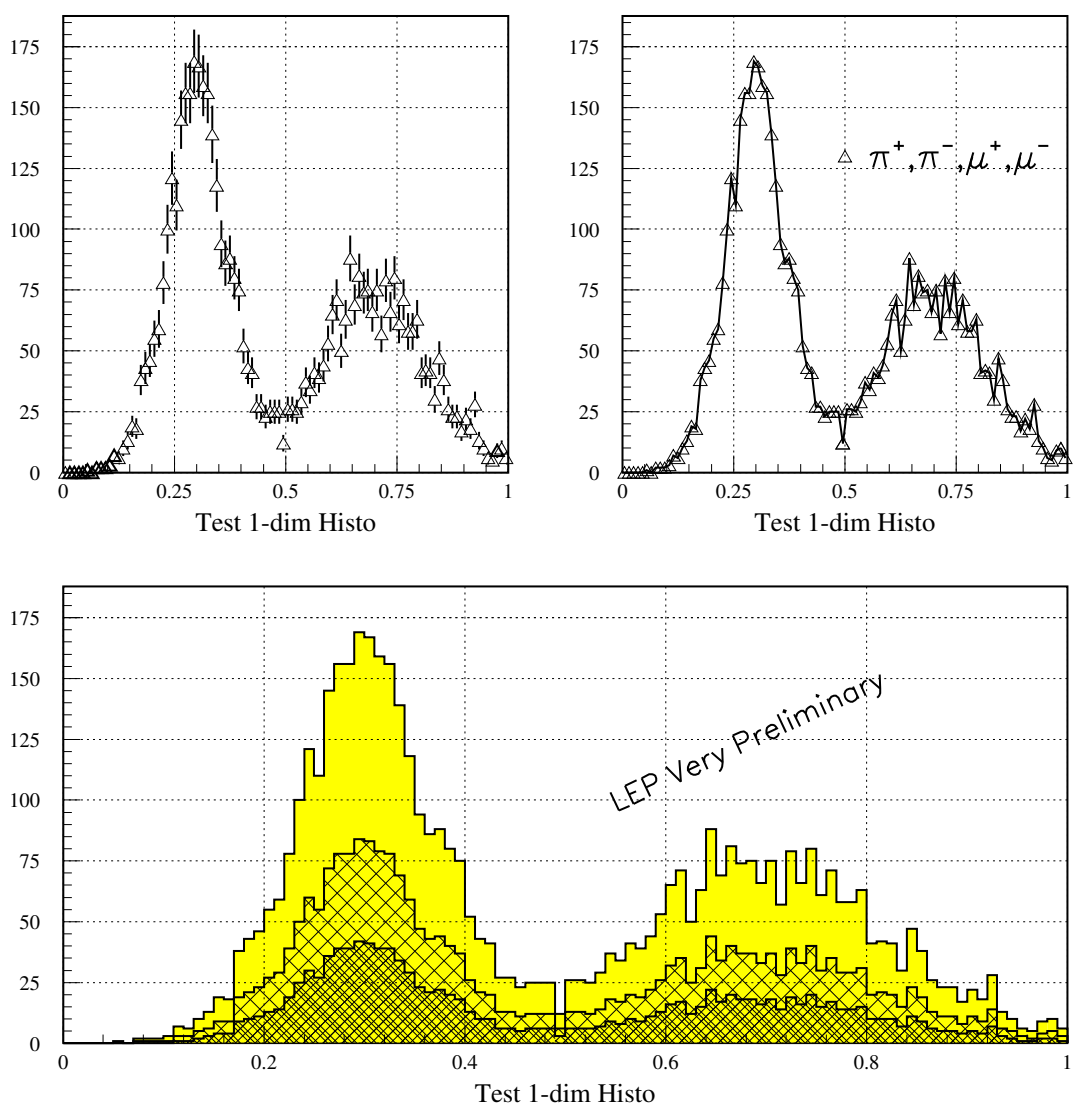


Figure 14. One-dimensional plotting and histogram operations

```
PAW > ********************************************************************************
PAW > *                      TUTORIAL EXAMPLE PAWEX14                                *
PAW > * Different representations of 2-dim histograms                                *
PAW > ********************************************************************************
PAW > histogram/file 1 pawhists.rzdat              | Open histogram file created in PAWEX11, modified in PAWEX13
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
```

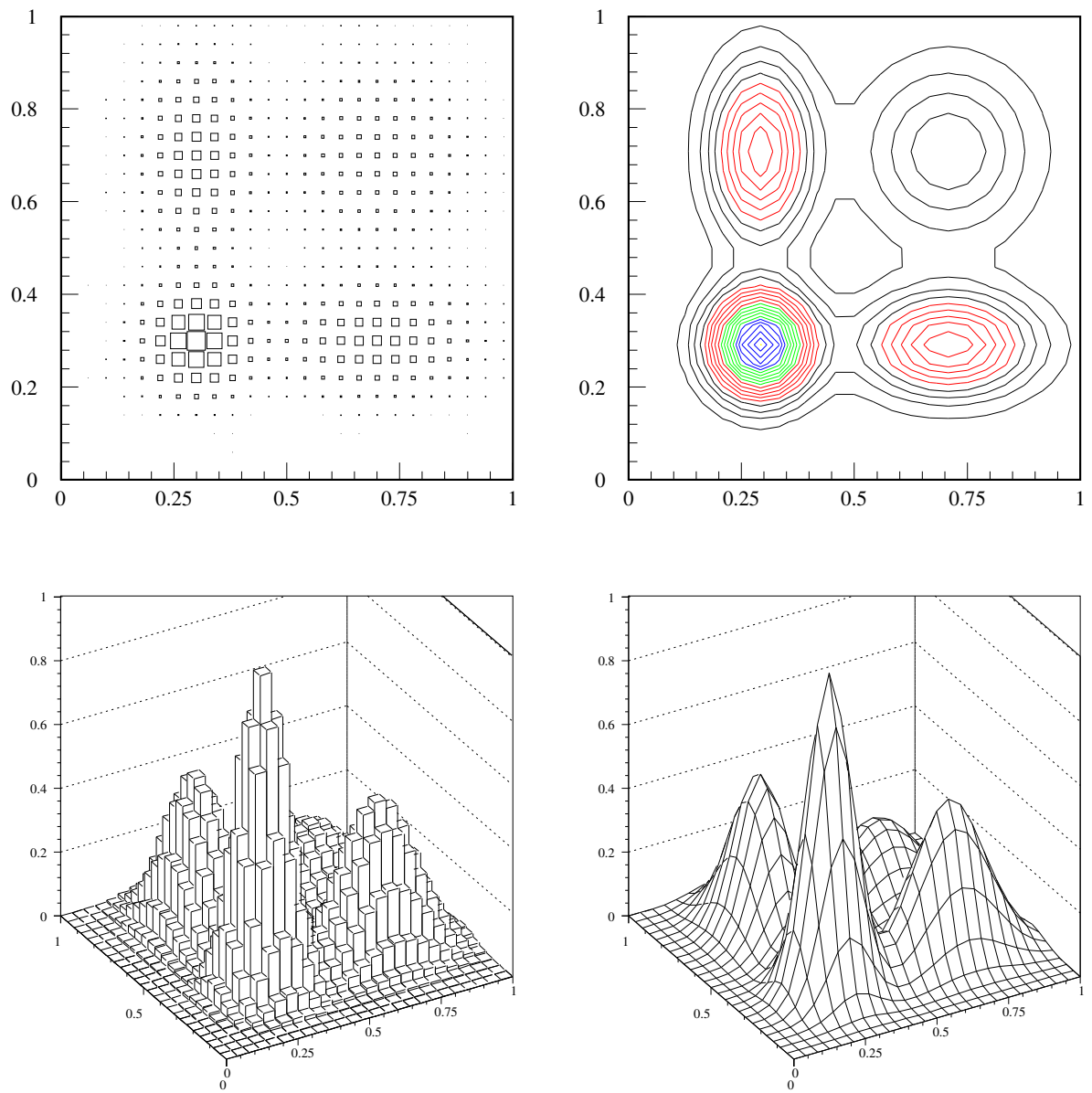


Figure 15. Two-dimensional data representations

```
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
```

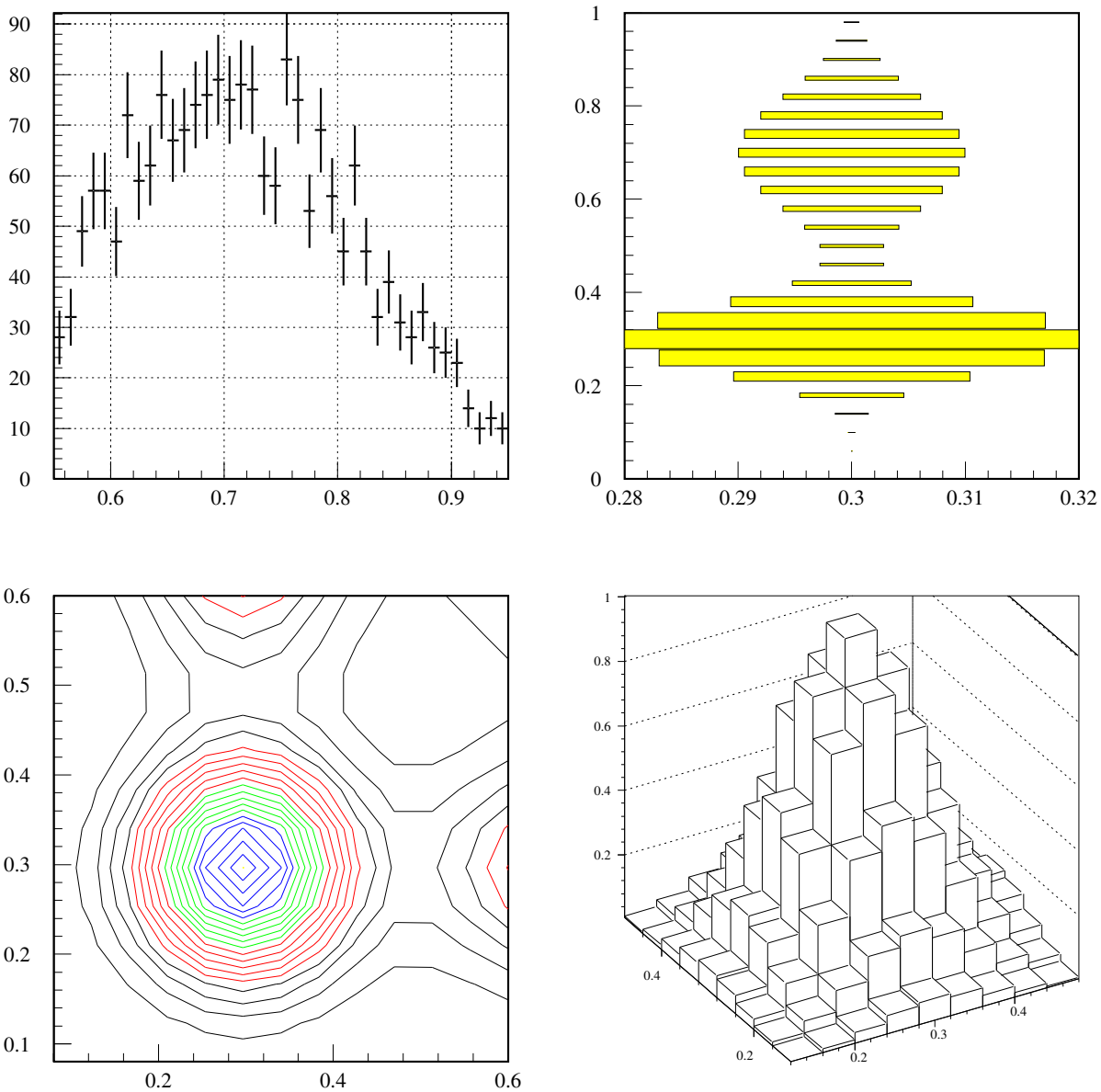


Figure 16. The use of sub-ranges in histogram specifiers

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, “Ntuples: A closer look” on page 87 for more details).

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 2.

Table 2. Definition of the variables of the CERN staff Ntuple

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)

Creating Ntuples

Ntuples can be created interactively, using PAW (see Figure 17 on page 33) or in an independent batch job, using HBOOK, using the following calling sequence [7]

```
CALL HBOOKN(IDN,TITLE,NCOLUM,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 17 on page 33 is as follows:

```
      PROGRAM APTUPLE
      PARAMETER (NCOLUM=11)
      CHARACTER*8 CHTAGS(NCOLUM)
      COMMON/PAWC/H(70000)
      REAL XTUPLE(NCOLUM)
      DATA NPRIME/1000/
*
      DATA CHTAGS/'category','division','flag','age','service',
     + 'children','grade','step','nation','hrweek','cost'/
*
      CALL HLIMIT(70000)
      OPEN(UNIT=1,FILE='APTUPLE.DAT',STATUS='OLD')
      CALL HBOOKN(10,'CERN Population',NCOLUM,' ',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 `NCOLUM` 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 the 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:

```
                     . . . . . .
      OPEN(UNIT=1,FILE='aptuple.dat',STATUS='OLD')
      LRECL=1024
      CALL HROPEN(2,'APTUPLE','aptuple.rzdat','N',LRECL,ISTAT)
      CALL HBOOKN(10,'CERN Population',NCOLUM,'APTUPLE',NPRIME,CHTAGS)
                     . . . . . .
 1000 CALL HROUT(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
```

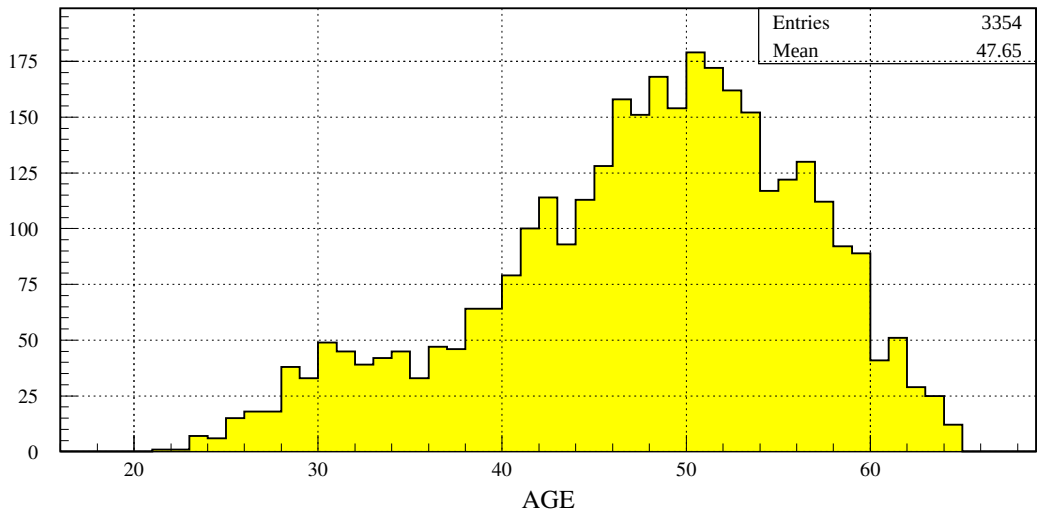
It is seen that 38107 words were needed to store all the data associated to the Ntuple. The file aptuple.rzdat created in batch by the program or the one created interactively in Figure 17 on page 33 are completely interchangeable. Generally speaking, a Ntuple (or any HBOOK histogram RZ file) can be created in batch on a mainframe and then transferred to a machine running PAW. See Chapter 9, “Distributed PAW” on page 133 for more details.

Listing the directory of the disk-resident Ntuple

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

```
 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
```

```
PAW > ************************************************************************
PAW > *                      TUTORIAL EXAMPLE PAWEX16                        *
PAW > * Creation of a Ntuple and first look at its contents                  *
PAW > ************************************************************************
PAW > Ntuple/create 10 'CERN Population' 11 ' ' 3500 _      | Create Ntuple 10 with 11 variables
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 > hrout 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
```



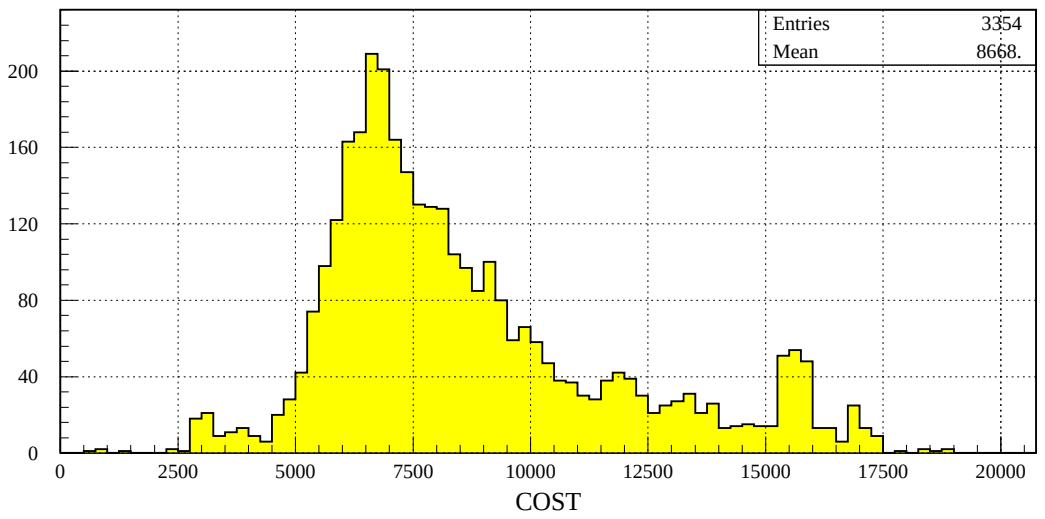


Figure 17. Ntuples - Creation and output to a file

```
PAW > ********************************************************************************
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 > zone 2 2                                                   | Divide plot into two by two squares
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 > Ntuple/pl 10.age                                           | Age distribution with automatic binning
PAW > hi/create/1dhisto 11 'Age - User binning' 45 20. 65.       | Book histogram 11
PAW > set ndvx -509                                              | Exactly 5 secondary and 9 primary divisions
PAW > Ntuple/project 11 10.Age                                   | Age distribution with user binning
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 > Ntuple/plot 10.cost                                        | Cost distribution with automatic binning
PAW > set ndvx -504                                              | Exactly 5 secondary and 4 primary divisions
PAW > Ntuple/pl 10.Cost ! -12                                    | Cost distribution with user binning
PAW > close 2                                                    | Close histogram file
PAW > zone                                                       | Reset zone
PAW > hi/de 0                                                    | Delete histograms from memory
```

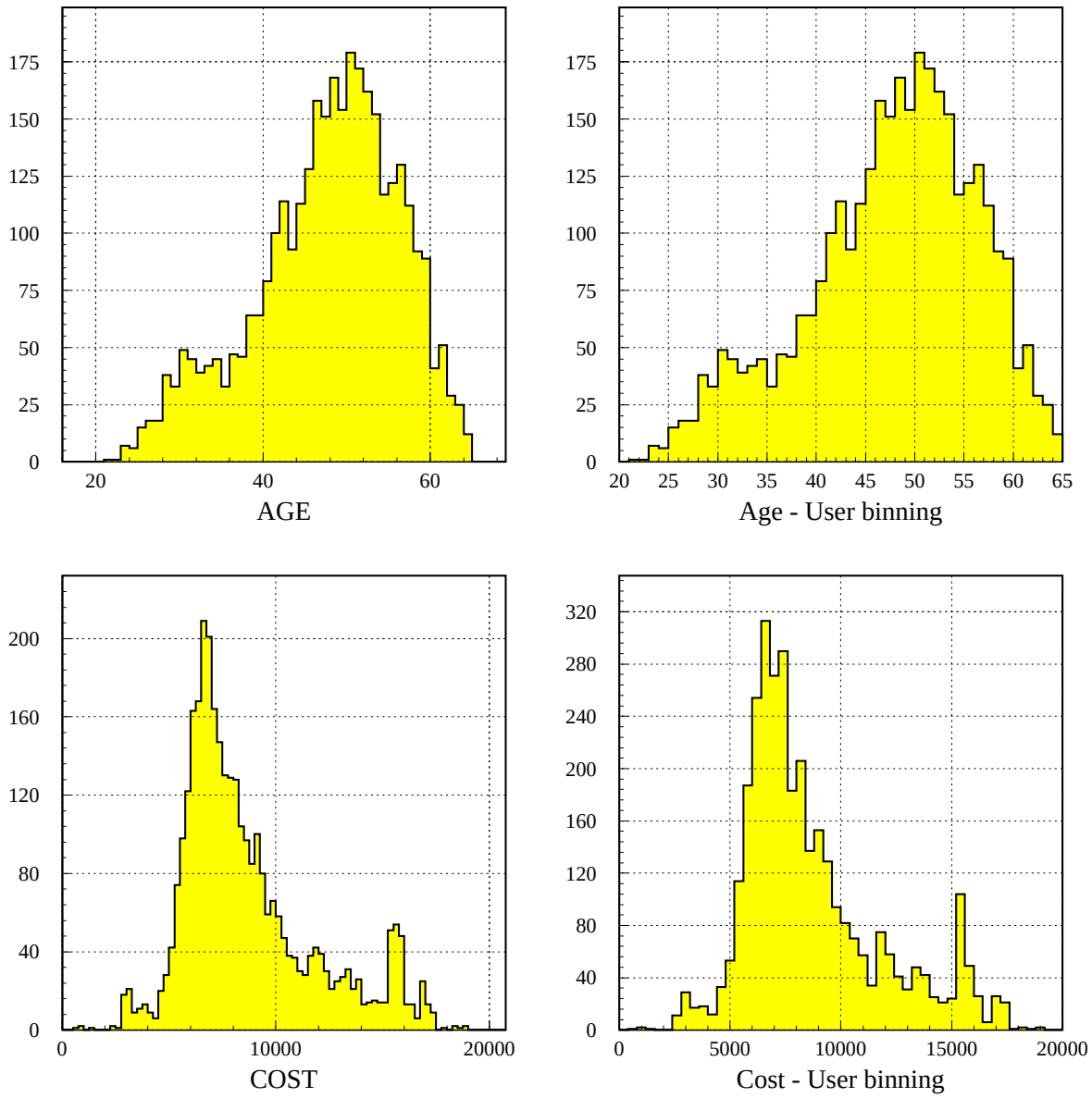


Figure 18. Ntuples - Automatic and user binning

```
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 > NT/SCAN 10 nation=NATFR.and.division=DIVEP _        | Scan and look for French CERN staff in EP division
PAW >_ ! ! 5 age service children grade step             |   printing the variables indicated
        ********************************************************************************
        * 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                                        | Specify maximum for histogram 200
PAW > set ndvx 507                                       | Maximum 5 secondary and 7 primary divisions
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 > Nt/pl 10.Service nation=NATFR -200 ! ! S           | Number of years at CERN for French staff members
PAW > set htyp 144                                       | Change hatch style
PAW > Nt/pl 10.Service _
PAW >_ division=DIVEP.and.nation=NATFR -200 ! ! S        | Number of years at CERN for French Staff in EP
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
```

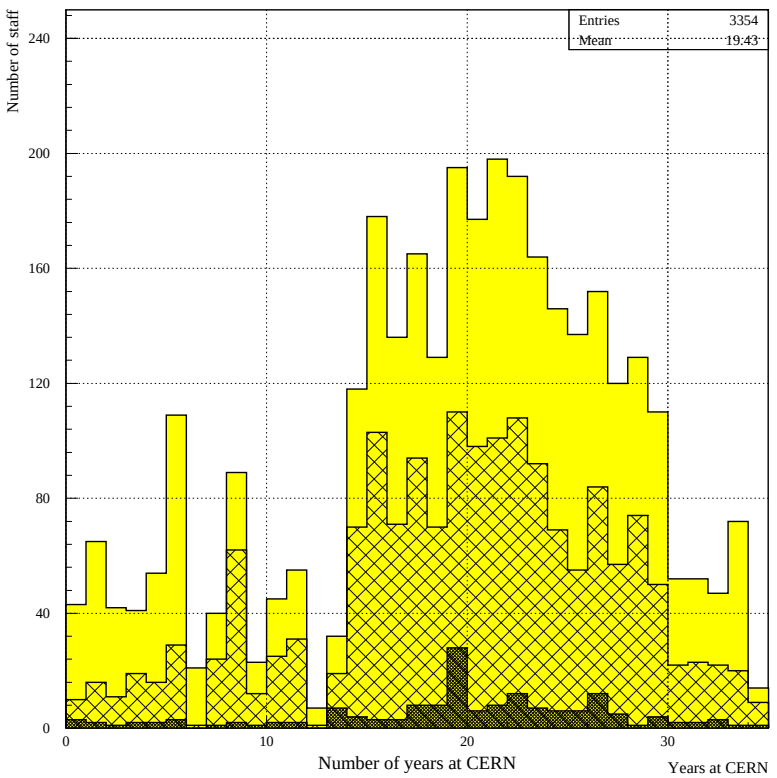


Figure 19. Ntuples - A first look at selection criteria

```
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 > set stat 110                                     | Indicate number of entries and average
PAW > opt grid                                         | Draw grid on picture
PAW > set grid 1003                                    | Only horizontal grid lines with line type 3
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 > set NDVX -12.05                                  | Exactly 12 divisions with text centered in X
PAW > set NDVY 507                                     | We want 5 secondary and 7 primary divisions in Y
PAW > set htyp -3                                      | Dotted hatch style (postscript)
PAW > Ntuple/plot 10.grade ! -20                       | Plot grade distribution into histogram 10
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 the histogram
PAW > itx 10.7 575 'Staff at end of grade'              | Add some more text to the histogram
PAW > ATITLE 'Grade' 'Number of Staff'                 | Axis title
PAW > set htyp 0                                       | Reset hatch style
PAW > hi/de 0                                          | Delete histograms from memory
```

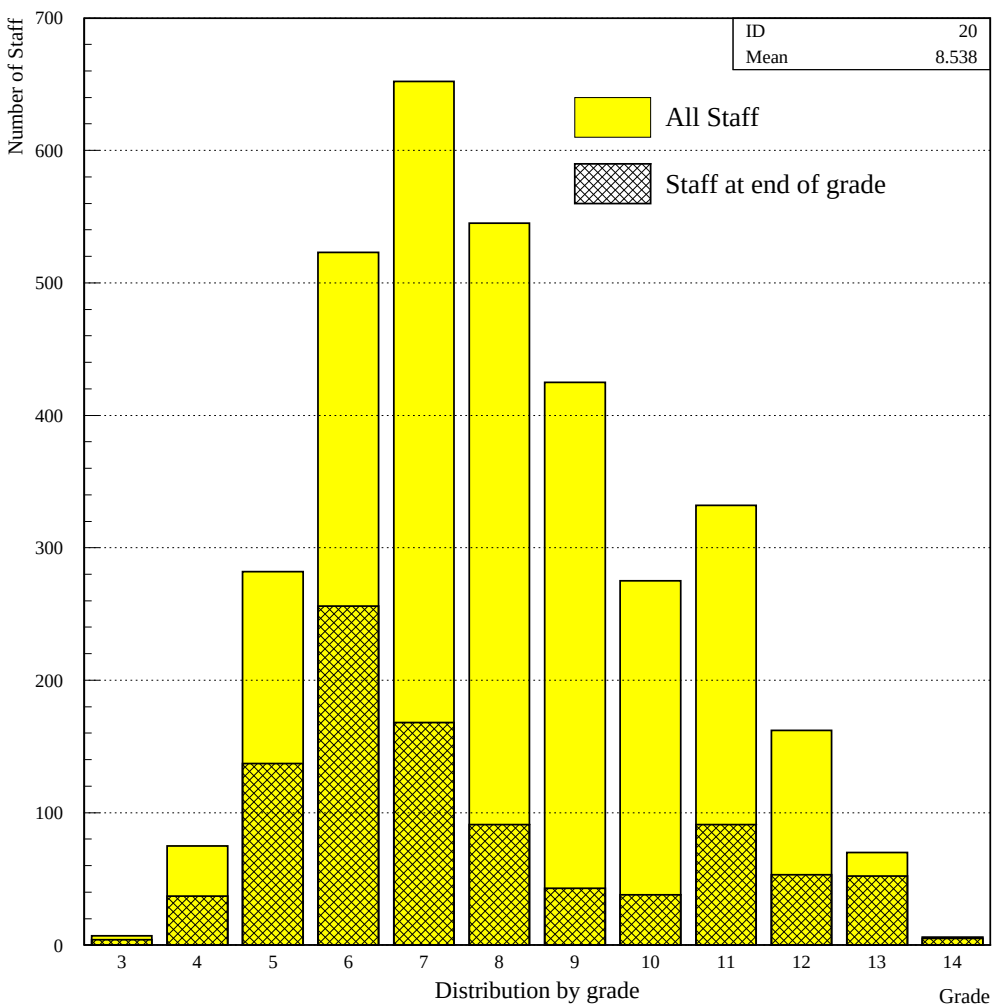


Figure 20. Ntuples - Masks and loop

```
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 > set grid 1003                                          | Only horizontal grid lines with line type 3
PAW > opt bar                                                | Activate the bar chart option
PAW > opt stat                                               | Print statistics on plot
PAW > set stat 10                                            | Show 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 > Ntuple/plot 10.division ! -20                          | Plot all staff per division into histogram 20
PAW > set htyp -3                                            | Choose hatch style
PAW > Ntuple/plot 10.division 2 -20 ! ! s                    | Plot staff per division (satisfying cut 2)
PAW > igset baro 0.5                                         | Reset bar origin
PAW > set htyp 145                                           | Change hatch style
PAW > Ntuple/plot 10.division 1 -20 ! ! s                    | Plot staff per division (satisfying cut 1)
PAW > set htyp 154                                           | Change hatch style
PAW > Ntuple/plot 10.division 1.and.2 -20 ! ! s              | Plot staff per division (satisfying cuts 1 and 2)
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
```

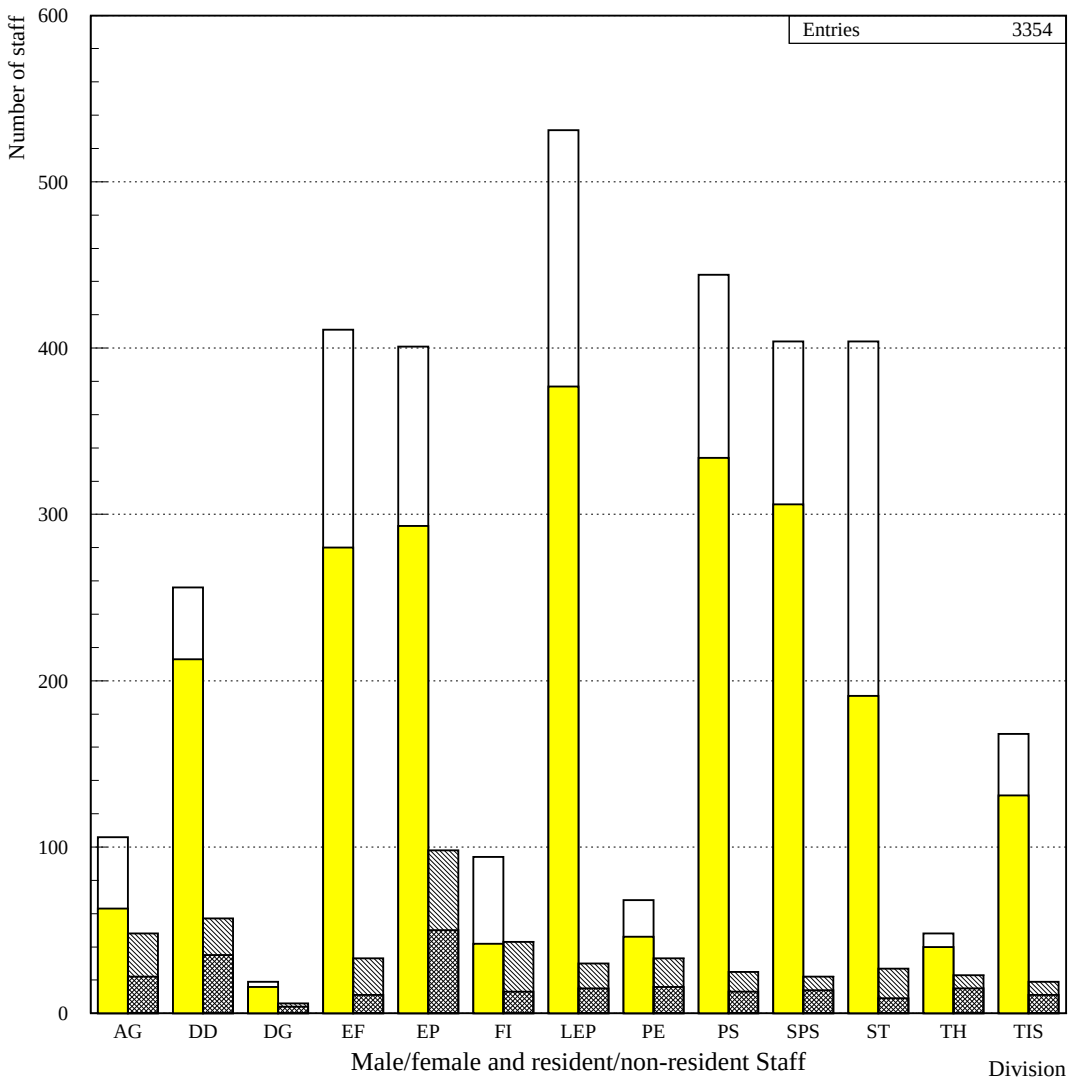


Figure 21. Ntuples - Using cuts

```
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 > * Lego plot of staff near end of grade (step ≥7) and choose viewing angles
PAW > * Note that the 2-dim Ntuple projection and histogram sub-range specifications are reversed
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
```

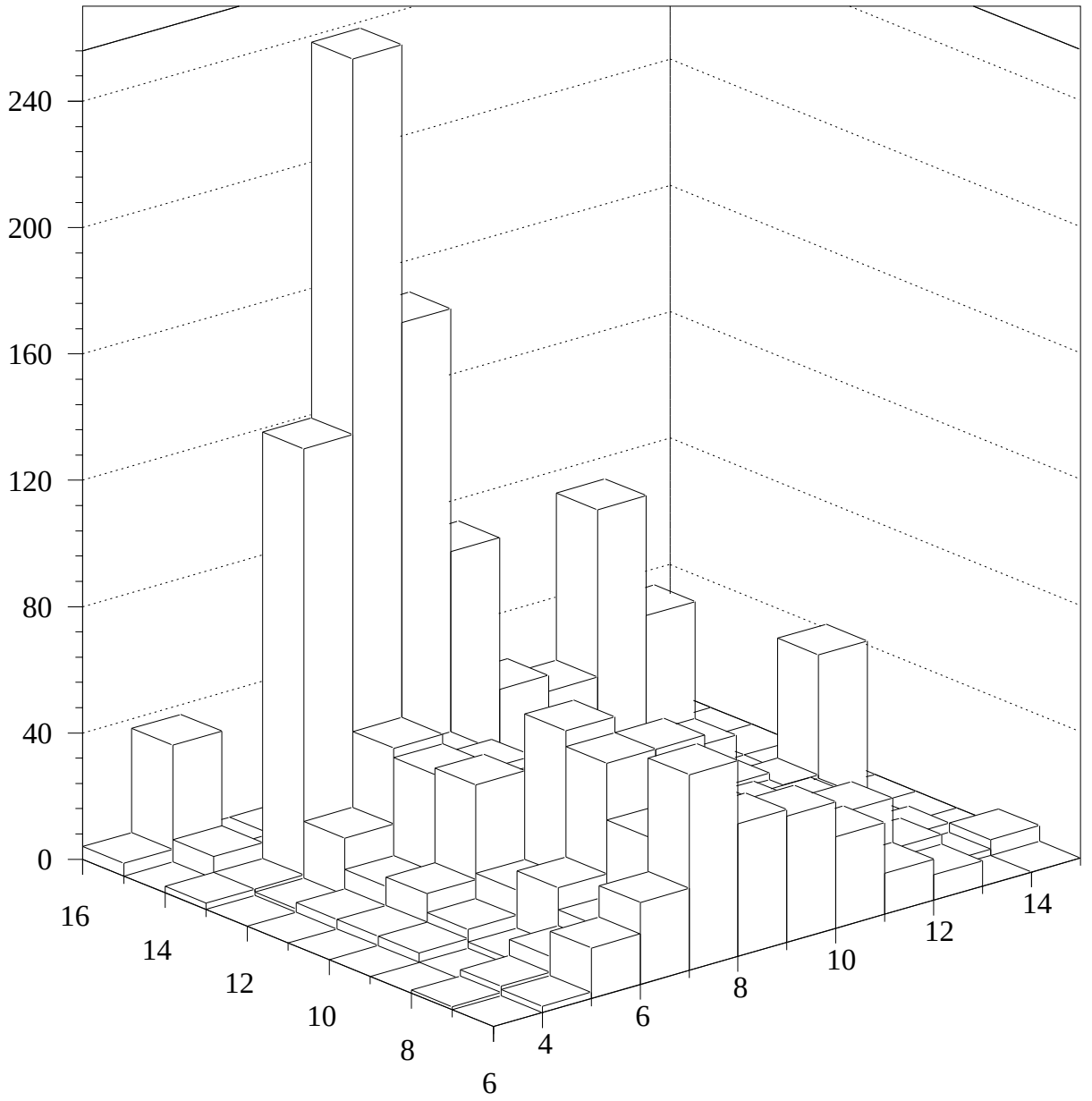


Figure 22. Ntuples - Two dimensional data representation

3.5 The SIGMA application and more complex examples

```
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 ;SET NDVX 520            | Draw grid on picture and ask for 20 primary divisions
PAW > sigma x=array(200,0#2*PI)         | Define array x with 200 elements values between 0 and 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 > SIGMA x=array(300,0#8)            | Define array x with 300 elements filled with values between 0 to 8
PAW > SIGMA g=cosh(x)+sin(1/(.1+X*X))   | Hyperbolic cosine oscillations are invisible
PAW > gra 300 X G                       | Plot the vectors x and g
PAW > SIGMA  x=array(300,0#3)           | Define array x with 300 elements filled with values between 0 to 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 > SIGMA  x=array(300,0#1)           | Define array x with 300 elements filled with values between 0 to 1
PAW > SIGMA g=cosh(x)+sin(1/(.1+X*X))   | Calculate the ordinate
PAW > gra 300 X G                       | Plot vectors x against g
```

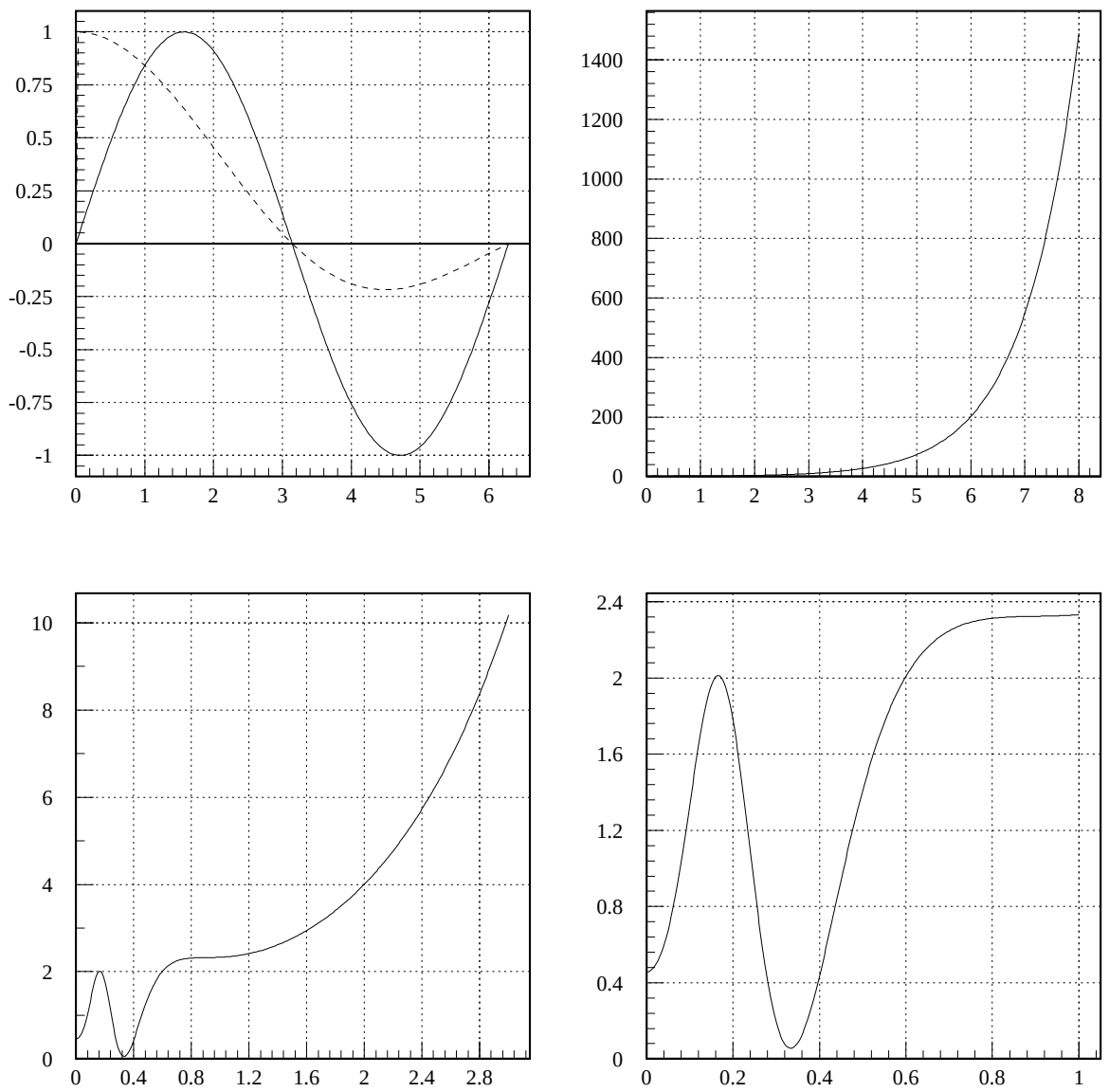


Figure 23. Using the SIGMA processor - Trigonometric functions

```
PAW > ************************************************************************
PAW > *                      TUTORIAL EXAMPLE PAWEX23                        *
PAW > * Examples of the SIGMA processor - 2                                  *
PAW > ************************************************************************
PAW > zone 2 2                          | Divide picture two by two
PAW > sigma x=array(200,0#5)            | Define array x with 200 elements filled with values between 0 to 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 ;SET NDVX 520            | Draw grid on picture and ask for 20 primary divisions
PAW > gra 200 x y                       | 2-dim representation y versus x
PAW > opt NGRI                          | No grid on picture
PAW > SIGMA x=array(200,0#2*pi)         | Array between 0 and 2 pi
PAW > SIGMA s=sin(x)                    | Define some more arrays (automatic creation of array s)
PAW > SIGMA S2=S/2                      |                         (automatic creation of array s2)
PAW > SIGMA c=cos(x)                    |                         (automatic creation of array c)
PAW > SIGMA c2=c/2                      |                         (automatic creation of array c2)
PAW > SIGMA s4=s/4                      |                         (automatic creation of array s4)
PAW > SIGMA c4=c/4                      |                         (automatic creation of array c4)
PAW > gra 200 s c                       | Then plot all arrays in same plot
PAW > gra 200 s2 c l                    |           ditto
PAW > gra 200 s4 c l                    |           ditto
PAW > gra 200 s c2 l                    |           ditto
PAW > gra 200 s2 c2 l                   |           ditto
PAW > gra 200 s4 c2 l                   |           ditto
PAW > gra 200 s c4 l                    |           ditto
PAW > gra 200 s2 c4 l                   |           ditto
PAW > gra 200 s4 c4 l                   |           ditto
PAW > sigma a=array(100,0#59.77)        | A array of 100 elements with values between 0 and 59.77 (19*pi)
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
```

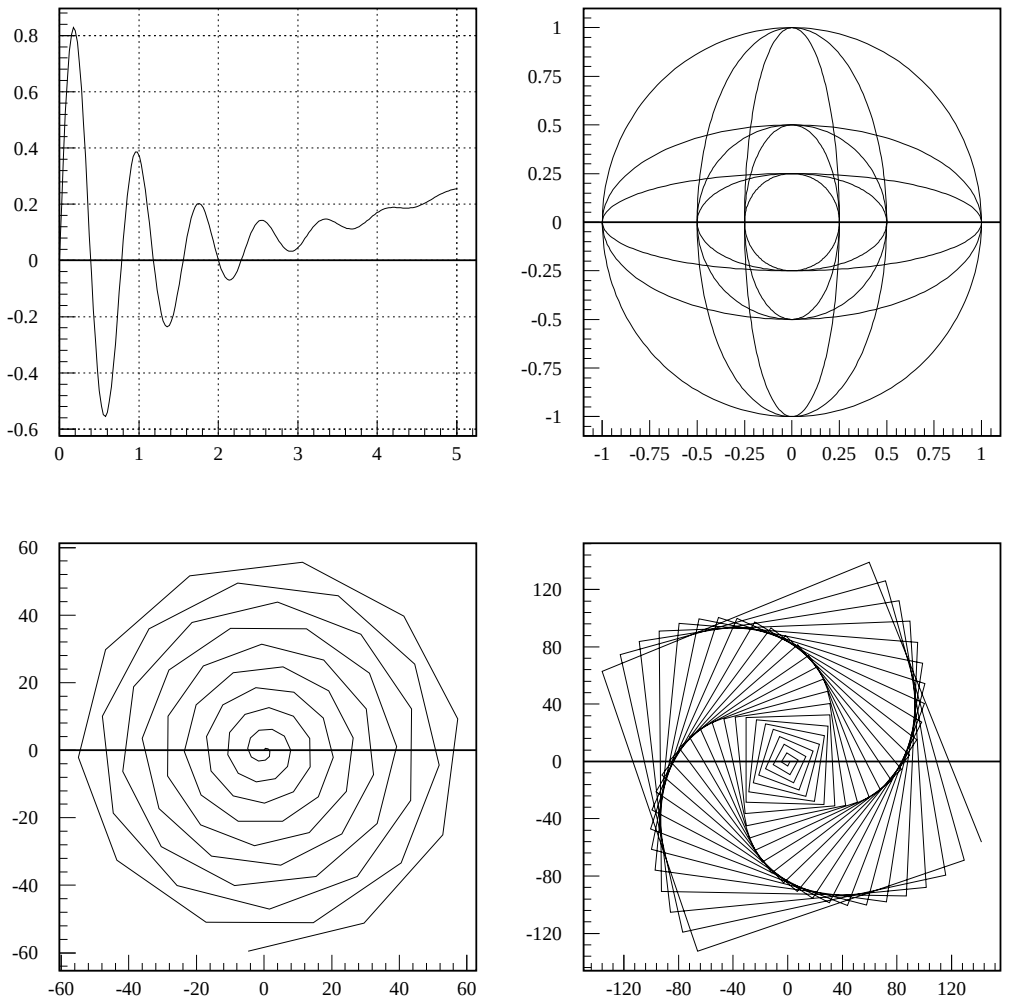


Figure 24. Using the SIGMA processor - More complex examples

```
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
```

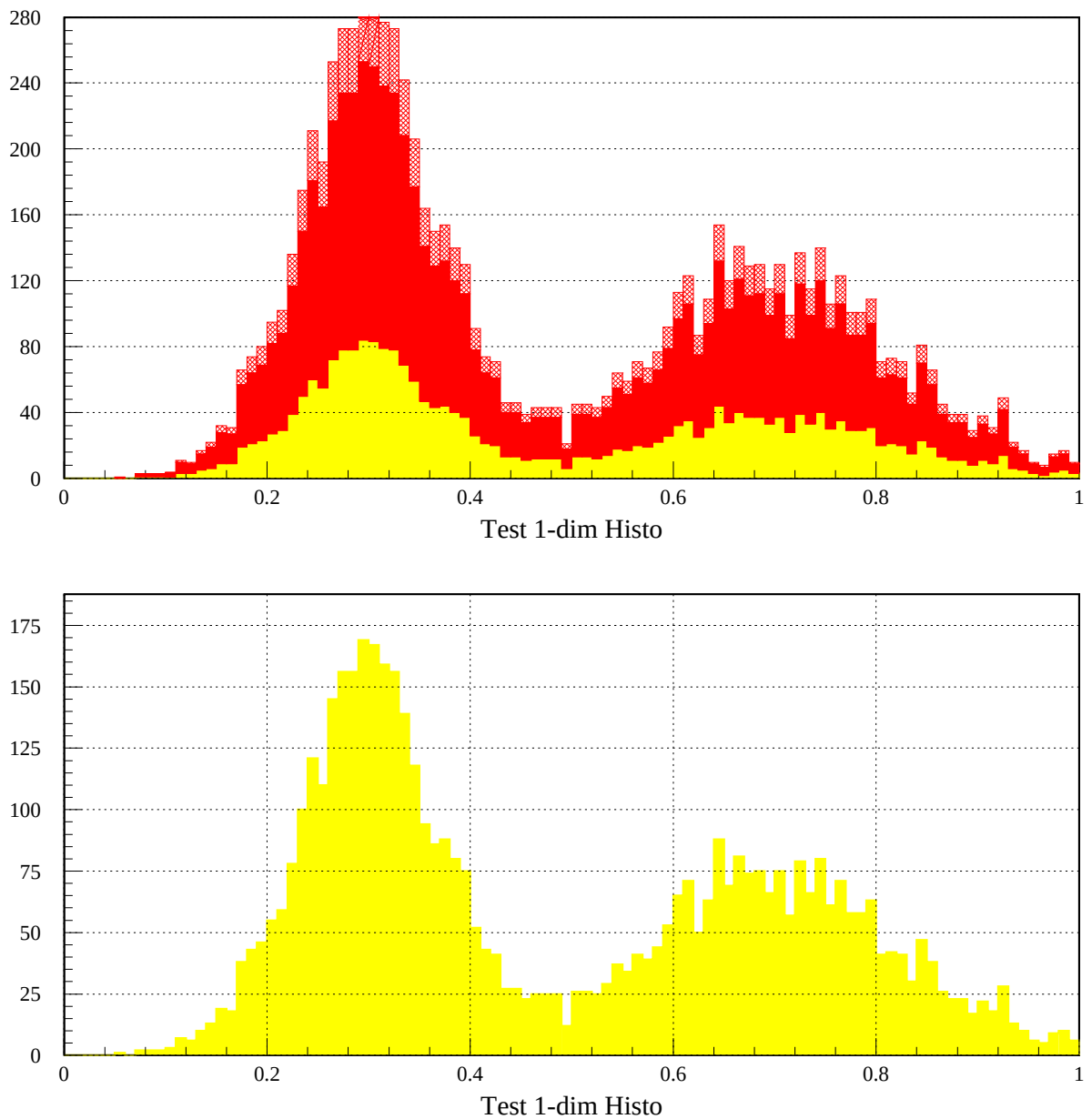


Figure 25. Histogram operations (Keep and Update)

```
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                                  | Font for axes (GKSGRAL font -104, precision 2)
      set lfon -1042                                  | Font for axis labels
      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
```

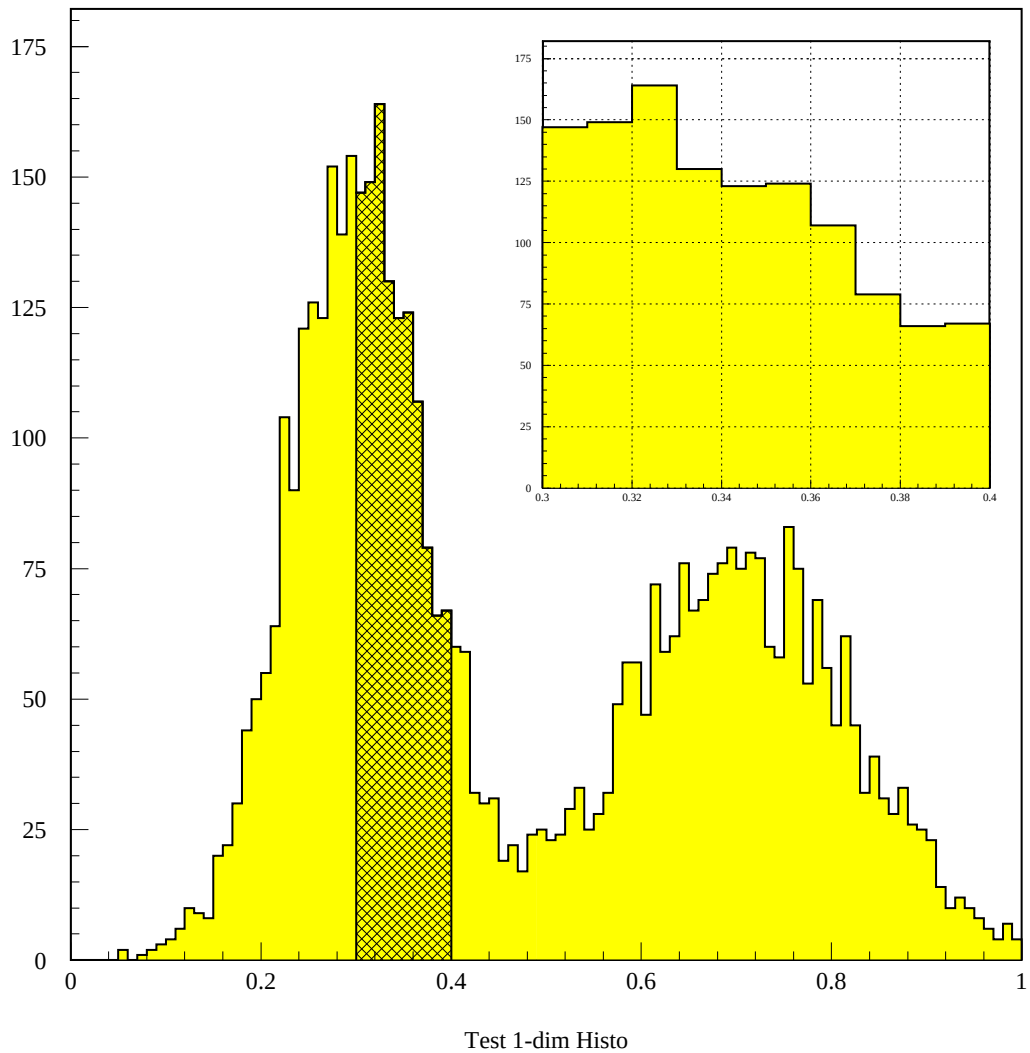


Figure 26. Merging several pictures into one plot

```
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
```

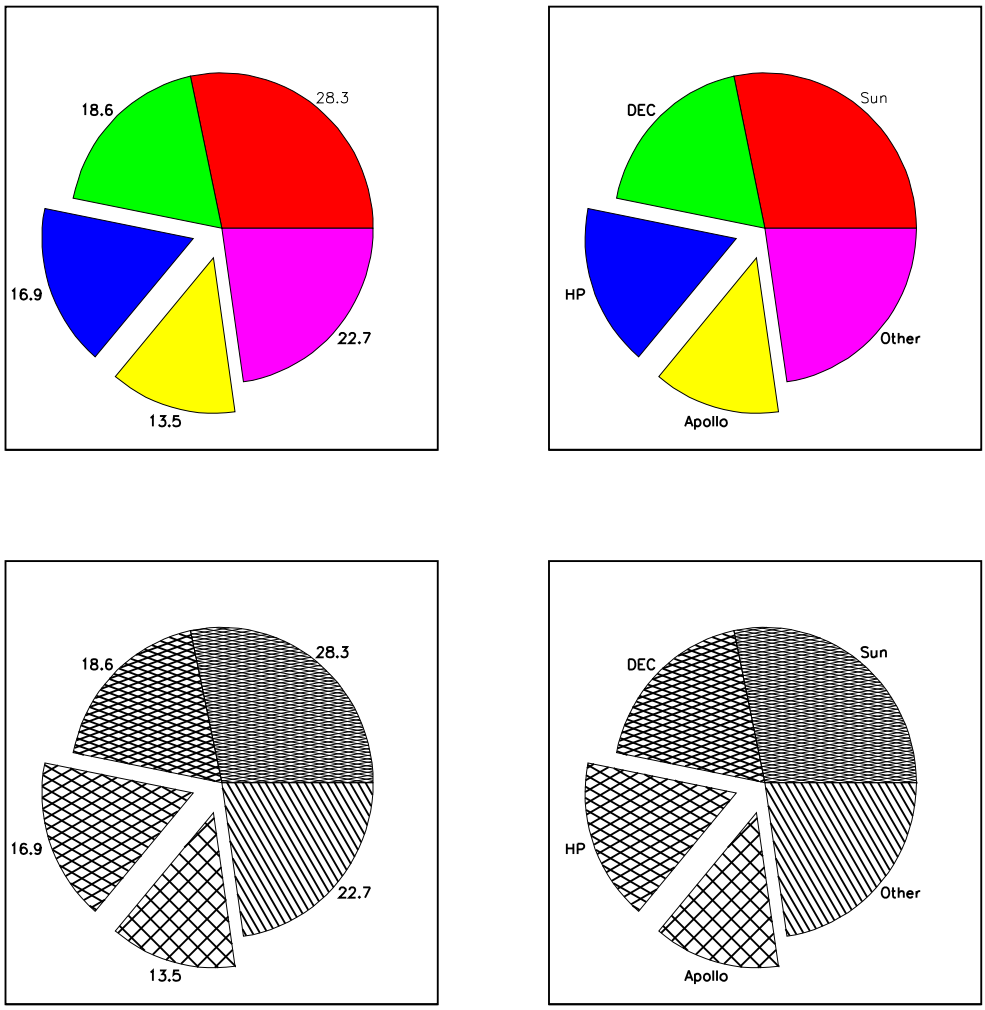


Figure 27. Pie charts with hatch styles and PostScript colour simulation

```
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                                             | Logaritmic 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                                         | Qaudruple line width
      title_gl 'CERN Central Computer Usage'               | Global title of picture
      vector/create vy(30) R 9.2 11.8 34.9 60.7 87.1 217.8 360 1250 2500 4006 _  | IBM168 hours used for each year since 1960
        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
      SET NDVX -30.05                                      | Exactly 30 divisions in X with labels centered on divisions
      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 '                           | Write text
      arrow 81. 79.  10000. 10000. -0.11                   | Draw arrow
      itx 80.  12000. 'IBM 3032'                           | Write text
      arrow 85. 81.  18000. 18000. -0.11                   | Draw arrow
      itx 83.  20000. 'IBM 3081'                           | Write text
      igset txal 10                                        | Left align text
      arrow 84. 82.  27000. 27000. -0.11                   | Draw arrow
      itx 82.  30000. 'SIEMENS 7880'                       | Write text
      igset txal 20                                        | Center text horizontally
      arrow 90. 84.  42000. 42000.  0.11                   | Draw arrow
      itx 87.  50000. 'SIEMENS 7890'                       | Write text
      arrow 90. 85.  68000. 68000.  0.11                   | Draw arrow
      itx 87.5 72000. 'IBM 3090'                           | Write text
      arrow 90. 88.  100000. 100000.  0.11                 | Draw arrow
      itx 89. 110000. 'CRAY'                               | Write text

      arise=$sigma(int((exp(f1(2))-1)*100+0.5))//'% per Annum rise'   | Get slope of annual increase from fit 1
      xhand=68.                                            | X coordinate for hand and text
      yhand=$sigma(exp(f1(1)+f1(2)*[xhand]))               | Calcualte y coordinate from fit 1
      EXEC DRAW X=[xhand] Y=[yhand] TEXT=[arise]           | Draw the hand and text
      arise=$sigma(int((exp(f2(2))-1)*100+0.5))//'% per Annum rise'   | Get slope of annual increase from fit 2
      xhand=84.                                            | X coordinate for hand and text
      yhand=$sigma(exp(f2(1)+f2(2)*[xhand]))               | Calcualte y coordinate from fit 2
      EXEC DRAW X=[xhand] Y=[yhand] TEXT=[arise]           | Draw the hand and text
      atitle 'Year ' 'IBM 168 Units used '                 | Add axis title to picture
      RETURN

      MACRO DRAW
      igset TXAL 30                                        | Right align
      igset TANG -35.                                      | Text angle
      igset TXFP -140                                      | Zapfdinbats font
      igset CHHE  0.50                                     | Character height
      itx $SIGMA([X]-0.9) [Y] +                            | A hand in the font ZapfDingbats
      igset TXAL 30                                        | right align text
      igset TANG  0.                                       | Text angle
      igset TXFP -30                                       | Font for accompanying text
      igset CHHE  0.22                                     | Character height
      y = y * 1.70                                         | Start text somewhat above hand
      itx [X] [Y] [TEXT]                                   | The text to be written
      RETURN
PAW > EXEC COMPTIME                                    | Execute macro for computer time
```

Figure 28 (Part 1 of 2). A complex graph with PAW

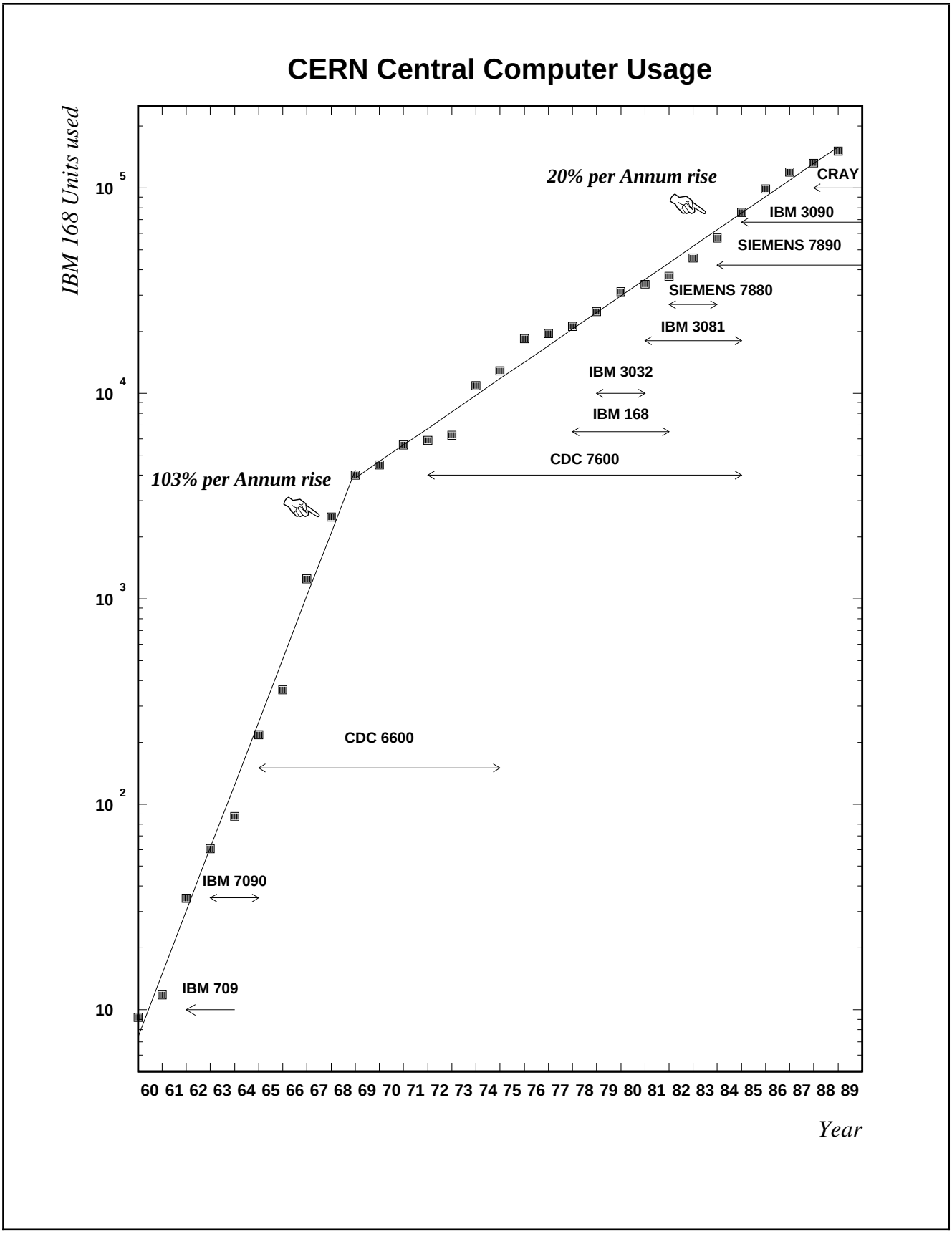


Figure 28 (Part 2 of 2). A complex graph with PAW

```
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
     igset txfp -70                                             | Defined new font (Helvetica-bold-oblique)
     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)
     igset txal 33                                              | Text alignment Hor-Right / Vert-Centered
     itx 6 7 'Efficiency ='                                     | Write text
     igset txal 20                                              | Text alignment Hor-Centered / Vert-Base
     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
     igset txfp -240                                            | Defined new font (ZapfDingbats-bold)
     igset chhe 0.6                                             | Set character height
     igset txal 30                                              | Text alignment Hor-Right / Vert-Base
     itx 1.5 17 P                                               | Write text
     itx 1.5 15 P                                               | Write text
     igset chhe 0.3                                             | Set character height
     igset txal 20                                              | Text alignment Hor-Centered / Vert-Base
     igset txfp -70                                             | Defined new font (Helvetica-bold-oblique)
     itx 9 4 'Many time consuming applications today have:'     | Write textOn apollo execute
     itx 9 3 'Efficiency > 0.9'                                 | Write text
     return
PAW > * Define alias defining main text fonr and precision (PostScript HELVETICA BOLD) for the slide text
PAW > alias/create mainfont -60                                 | Create alias
PAW > * Define alias to print PostScript metafile (See “Pictures in memory” on page 106 and following, where macro POST is described)
PAW > alias/create @ 'exec post'                                | Create alias
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 *                                            | Delete Alias definition
```

Figure 29 (Part 1 of 2). Making slides with PAW using PostScript

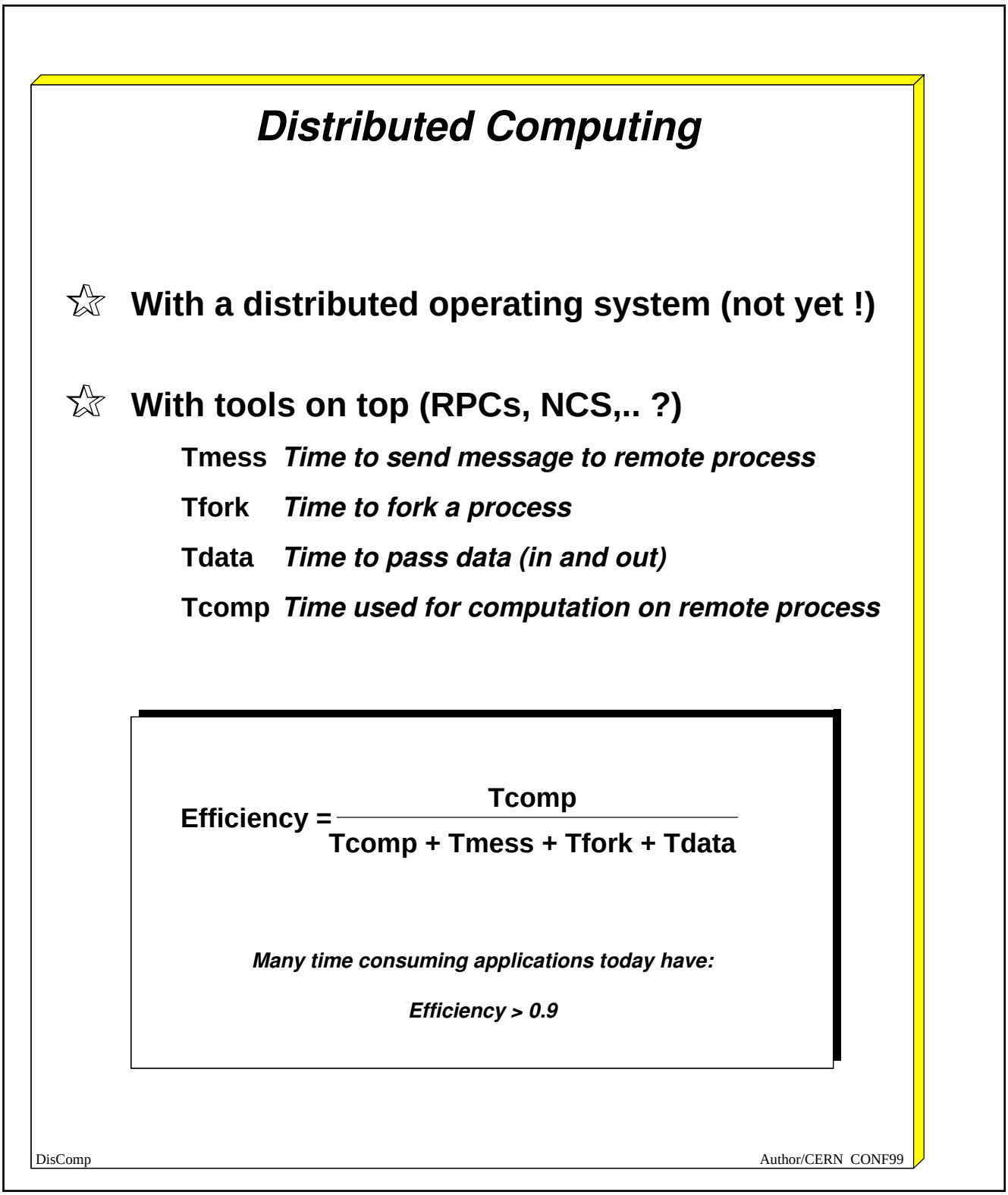


Figure 29 (Part 2 of 2). Making slides with PAW using PostScript

Part 2. PAW - Commands and Concepts

Chapter 4. User interface - KUIP **51**
4.1 The PAW command structure 51
4.2 Multiple dialogue styles 52
4.3 Macros 56
4.4 Aliases 60
4.5 System functions 62
4.6 On aliases, system functions and macro variables 63
4.7 Recalling previous commands 64
4.8 Exception condition handling 64

Chapter 5. Vectors **65**
5.1 Vector creation and filling 65
5.2 Vector addressing 66
5.3 Vector arithmetic operations 66
5.4 Vector arithmetic operations using SIGMA. 66
5.5 Using KUIP vectors in a COMIS routine 66
5.6 Usage of vectors with other PAW objects 67
5.7 Graphical output of vectors 67
5.8 Fitting the contents of a vector 67

Chapter 6. SIGMA **68**
6.1 Access to SIGMA 68
6.2 Vector arithmetic operations using SIGMA 69
6.3 SIGMA functions 70
6.4 Available library functions 77

Chapter 7. HBOOK **78**
7.1 Introduction 78
7.2 Basic ideas 79
7.3 HBOOK batch as the first step of the analysis 82
7.4 Using PAW to analyse data 85
7.5 Ntuples: A closer look 87
7.6 Fitting with PAW/HBOOK/MINUIT 94
7.7 Doing more with MINUIT 102

Chapter 8. Graphics (HIGZ and HPLOT) **105**
8.1 HPLOT, HIGZ and local graphics package 105
8.2 The metafiles 106
8.3 The HIGZ pictures 106
8.4 HIGZ pictures generated in a HPLOT program 109
8.5 Attributes setting 110
8.6 More on labels 114
8.7 Colour, line width, and fill area in HPLOT 116
8.8 Information on histograms 118
8.9 Additional details on some IGSET commands 119

8.10 Text fonts 124
8.11 The HIGZ graphics editor 132

Chapter 9. Distributed PAW 133
9.1 TELNETG and 3270G 134
9.2 ZFTP 136
9.3 Access to remote files from a PAW session 137
9.4 Using PAW as a presenter on VMS systems 138
9.5 Using PAW as a presenter on OS9 systems 139

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:

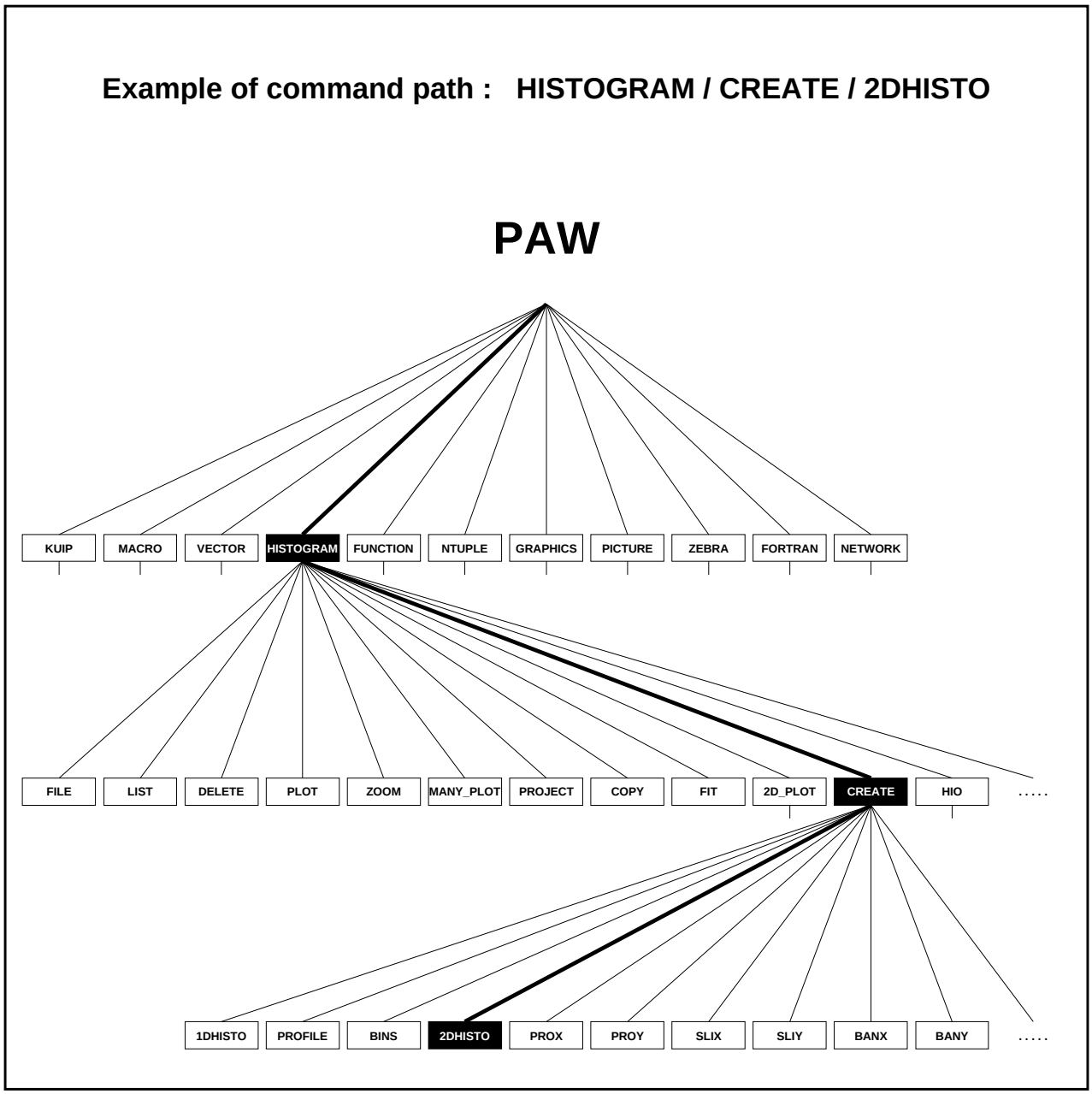


Figure 30. Example of the PAW command tree structure

4.2 Multiple dialogue styles

PAW is based on the KUIP[9] 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`.

Command line mode

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

The general syntax of an input command line is a command name optionally followed by a parameter list. The command name and the parameter list are separated by one or more blanks (therefore, no blank should appear within the command name). The parameters in the parameter list are again separated by one or more blanks; no blank should then appear within a parameter, unless the whole parameter is enclosed in single quotes, like for example `'This parameter has blanks'` or a blank parameter `' '`.

The command name is a structured name representing the path along the tree structure handled by KUIP. Each element of the path, called command element, is separated from the others by one slash, like for example in `/HISTOGRAM/CREATE/2DHISTO` (the first slash at the beginning of a command can always be omitted).

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 called menus.

Command abbreviations

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

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

```
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
```

Figure 31. Example of KUIP command line mode

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 command USAGE:

```
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 parameters always has a **default value**, which is used when the user does not specify the parameter. In the above example 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 escape 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 for instance 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
```

Menu modes

In addition to the command line mode, KUIP also allows various menu modes:

- **Alphanumeric**, entered by STYLE AN or STYLE AL.

  Selection is from a list by number or by letter. This mode still uses the terminal keyboard as input, but the user, instead of typing directly the full command, is guided into the tree of the command structure which is scanned through its branches until a leaf is found. The resulting path name corresponds then to the command which may be invoked.

- **Graphics**: This mode is particularly interesting for workstations. It should not be used with simple terminals.

  - Pull-down menus, entered by STYLE G; fixed layout, reflecting the command structure
  - Panels of function keys, entered by STYLE GP; interactive user definable multiple layouts.

Style G

Under STYLE G the user is still guided through the command tree as for the alphanumeric styles, but the mouse is used to navigate on pull down menus in the graphics window. In the figure below, the shadowed part is available for the graphics application.

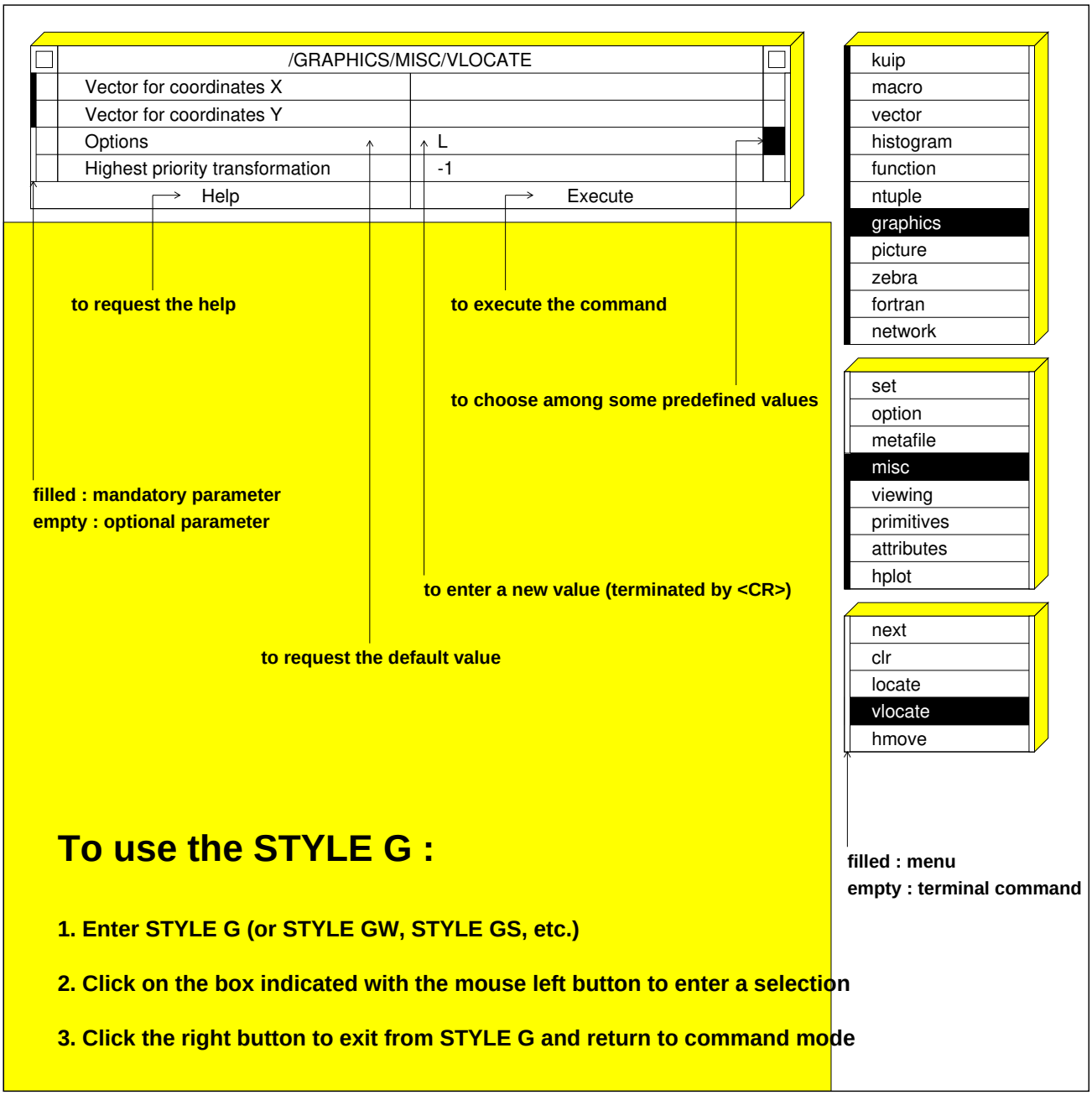


Figure 32. Example of STYLE G

Style GP

Panels of graphics function keys may be defined with the command PANEL:

```
panel 0
panel 1 Edit-     Exec- Shell-     Vector/List  Alias/List -     HELP-   EXIT
panel 2 'Ldir //' Cdir- Histo/List Histo/Plot-  Histo/Plot Lego  Surface Contour
panel 3 Set-      Opt-  Igset-     ' '          Locate     Zone- Next    Null
panel 4 'Histo/Plot $ E | Plot error bars'      'Exec paneldoc--'
```

Then enter STYLE GP, move the mouse to a graphics function key and click the left mouse button; the text written on the key will be sent for execution as if it were typed in via the keyboard. Parameters can also be added interactively by the user.

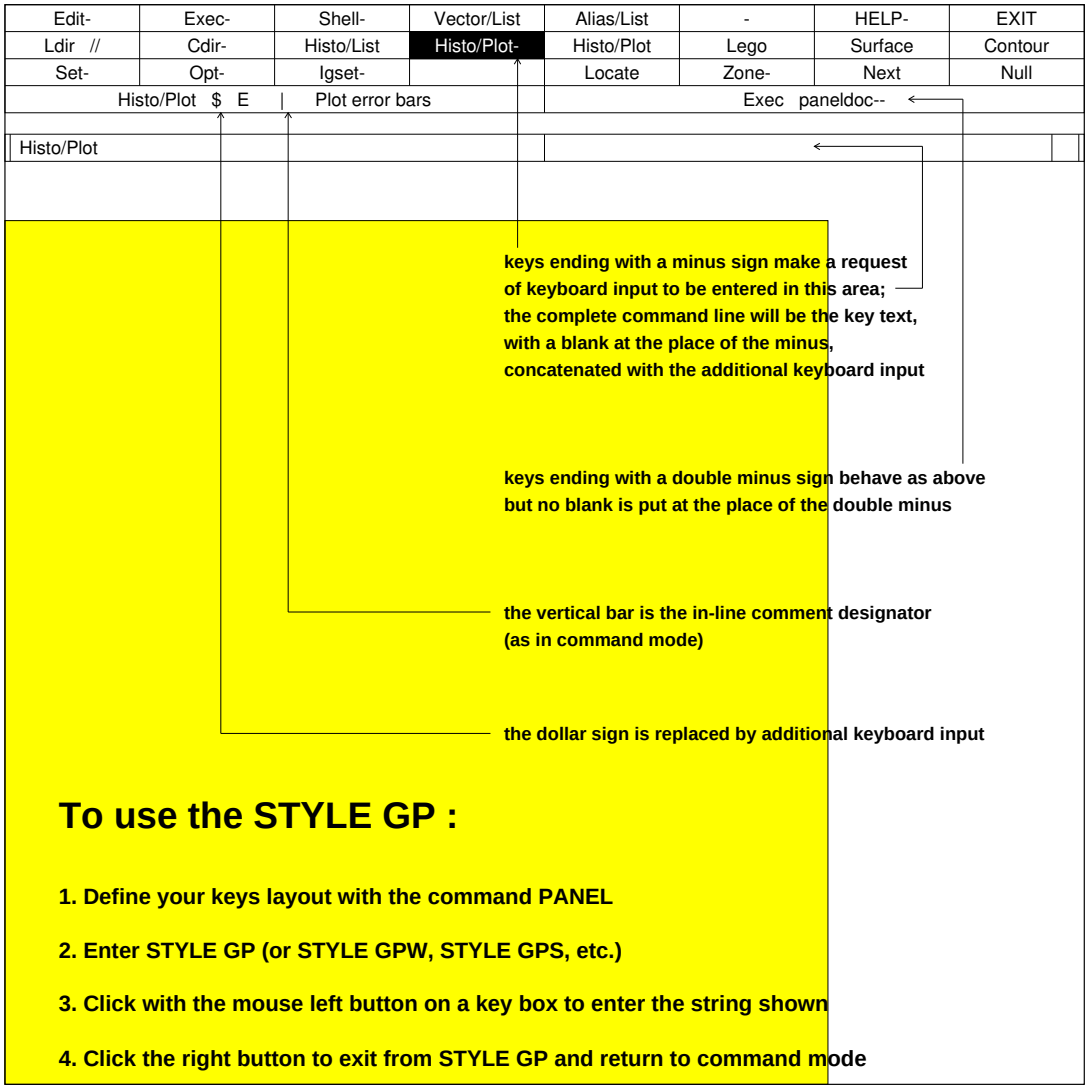


Figure 33. Example of STYLE GP

A panel can be redefined from within itself. For example by selecting the key Exec paneldoc-- and entering the number 1, the macro file paneldoc1.kumac will be used as input and executed. If it contains the following lines:

```
panel 0
panel 1 'Histo/Plot 1' 'Histo/Plot 2' 'Histo/Plot 3' 'Histo/Plot 4'
panel 2 'Histo/Plot 5' 'Histo/Plot 6' 'Histo/Plot 7' 'Histo/Plot 8'
panel 3 'Cdir //LUN1' 'Cdir //LUN2'
panel 4 'Exec paneldoc | to go back'
```

then the panel will be modified accordingly.

4.3 Macros

A macro is a set of command lines stored in a file, which can be created/edited with the command EDIT and executed with EXEC:

```
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

Examples:

```
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 commands the following special 'macro statements' are valid only inside macros (excepted EXEC, valid inside and outside):

Table 3. KUIP 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
label **:**	a label must terminate with a colon
GOTO label	jump to label
ON ERROR GOTO label	resume at label on error condition
IF logical_expression **GOTO** label	conditional statement
par **=** arithmetic_expression	assignment statement

In the last line of above table par is an argument passed with the command EXEC (either from the command mode or from another macro) or just a local variable of the macro.

arithmetic_expression and logical_expression are simple expressions defined as follows (with '|' standing for 'or' and juxtaposition standing for 'and'):

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

A label is any string in a line that is terminated by a colon (therefore labels must be alone and not intermixed with commands on the same line). Its definition is local to the macro in which the label is contained, therefore the same label can be used on different macros with no clash problem.

The ON ERROR GOTO statement is activated by error conditions of most PAW commands.* In executing a macro, the latest ON ERROR GOTO executed is the active one (i.e. the previous definition is superseded).

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

Positional parameters can be passed to a macro, separated by blanks, e.g.

```
PAW > EXEC ABC P1 123 'This is P3'
```

Inside a macro, positional parameters can be retrieved by including in brackets the number representing their order in the list. Using the above EXEC sequence for the following macro:

```
MACRO ABC
  MESSAGE [1] [3] [2]
RETURN
```

will print:

```
P1 This is P3 123
```

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 executing the command line. Example:

```
MACRO OPEN
  HISTO/FILE 20 DST[1].DAT
RETURN
```

called by:

```
PAW > EXEC OPEN 123TEST
```

will execute the command:

```
HISTO/FILE 20 DST123TEST.DAT
```

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

```
MACRO OPEN LUN=20 NAME=JUNK EXT=DAT LRECL=1024 CHOPT=' '
  HISTO/FILE [LUN] [NAME].[EXT] [LRECL] [CHOPT]
RETURN
```

Entering now:

```
PAW > EXEC OPEN EXT=TEMP LUN=10
```

will execute the command:

```
HISTO/FILE 10 JUNK.TEMP 1024
```

Parameters can also be read at macro run time. For example:

```
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
```

will ask the user to set parameters when READ is being executed. If just <CR> is entered, the values remain unchanged. The order of priority for macro parameters is such that the values given in the EXEC statement supersede those given in the MACRO statement.

Macro flow control

The flow of a macro can be controlled by using the IF and the GOTO macro statements. The parameters of a macro, described above, can be tested and their value modified. This possibility is valid in general for any macro **variable**, independently from the fact that the variable is a macro parameter passed to the macro or that the variable is just defined locally within the macro.

Inside a macro, variables can be set as in FORTRAN, but the distinction of their type is irrelevant: an automatic mechanism is used to distinguish between integer, real or character type variables. As an example consider the following macro:

```
   MACRO DOC1
      A=10
      NN=1.5
      TOT=A+NN
      IF TOT > 11 GOTO GREAT
      MESSAGE 'KUIP arithmetic is wrong'
      GO TO 10
GREAT:
      MESSAGE Sum of [A] and [NN] is [TOT]
10:
   RETURN
```

Its execution gives (hopefully!):

```
PAW > EXEC DOC1
Sum of 10 and 1.5 is 11.5
```

On the other hand, the macro:

```
   MACRO DOC2
      A=10
      NN=1.5
      TOT=A+XX
      MESSAGE Result of sum is [TOT]
   RETURN
```

produces the following result:

```
PAW > EXEC DOC2
Result of sum is A+XX
```

i.e. when numeric values cannot be used, the right part of an assignment statement is left unchanged as a character string. This because to the undefined variable XX is not assigned any numeric value (unlike FORTRAN).

An example of macro using arithmetic and logical operations is the following:

```
   MACRO PLOT 1=' ' 2=' '
      IF [1]<>' ' GOTO first_given
      MESSAGE 'Give starting ID'
      READ 1
first_given:
      IF [2]<>' ' GOTO second_given
      MESSAGE 'Give ending ID'
      READ 2
second_given:
      ID=[1]
LOOP:
         HISTO/PLOT [ID]
         ID=ID+1
         IF ID<=[2] GOTO LOOP
   RETURN
```

which, if executed for example by EXEC PLOT 4 7, plots the histograms identified by 4 to 7 included. If one or both macro arguments are omitted when calling the macro, they will be asked for interactively. Note also that, while a macro variable of an IF statement may be without brackets, the positional arguments must always be enclosed in brackets, to avoid ambiguity with numeric values.

The right part of an assignment statement can be a vector name (with an optional subscript), as in the following:

```
   MACRO DOC3
      A=10
      IF $VEXIST(VV)=0 GOTO NOVEC
      VEC/DEL VV
NOVEC:
      VEC/CRE VV(5)
      VEC/INP VV 10 20 30 2*0
      VECVAR=VV
      MESSAGE First component of vector VV is [VECVAR]
      VECVAR=VV(2)
      MESSAGE Second component of vector VV is [VECVAR]
      VECVAR=VV($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
```

which produces:

```
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 useful 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 are defined by using the command ALIAS/CREATE. As an example consider the following PAW 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
```

As evident from the M7PP line in the example above, aliases need separators to be recognized in the input line. 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 also 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
```

Note

Another way to invoke a macro without typing the EXEC command in front of the macro name is by issuing once at the beginning of the PAW session (e.g. by entering it in your PAWLOGON) the command DEFAULTS -AUTO. Then, whenever a command is not found, a macro with the given name is searched.

A complex example of the use of aliases

Consider now the following macro file (containing three macros):

```
macro doc9
  message '   ...   Executing: MACRO DOC9'
return

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

macro m2
  message '   ...   Executing: MACRO DOC9#M2'
return
```

with the commands:

```
PAW > a/cre m1 'exec doc9#m1'
PAW > m1
   ...   Executing: MACRO DOC9#M1
   ...   Executing: MACRO DOC9#M2
PAW > a/cre m2 'exec doc9#m2'
PAW > m2
*** Unknown file EXEC.kumac
```

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

```
macro _doc9
  message '   ...   Executing: MACRO _DOC9'
return

macro _m1
  message '   ...   Executing: MACRO _DOC9#_M1'
  exec _m2
return

macro _m2
  message '   ...   Executing: MACRO _DOC9#_M2'
return
```

and then:

```
PAW > a/cr m1 'exec doc9new#_m1'
PAW > m1
   ...   Executing: MACRO _DOC9#_M1
   ...   Executing: MACRO _DOC9#_M2
PAW > a/cr m2 'exec doc9new#_m2'
PAW > m2
   ...   Executing: MACRO _DOC9#_M2
```

4.5 System functions

While aliases have a fixed value, system functions can be seen as aliases whose value depends on the function name and on its arguments (if any). Therefore, also system functions are literally replaced by their current value whenever encountered in a PAW command line. System functions, unlike aliases, do not need separators because they are pre-defined 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. System Functions

Function name and arguments	Returned value	Example
$DATE	Current date in format DD/MM/YY	`$DATE ==> 01/09/89`
$TIME	Current time in format HH.MM.SS	`$TIME ==> 15.49.22`
$CPTIME	CP time elapsed since last call (s)	`$CPTIME ==> 10.17811`
$RTIME	Real time elapsed since last call (s)	`$RTIME ==> 49`
$VDIM(VNAME,IDIM)	Physical length of vector VNAME on dimension IDIM (1..3)	`VECTOR/CREATE MYVEC(3,10)` `$VDIM(MYVEC,2) ==> 10`
$VLEN(VNAME,IDIM)	Logical length (stripping trailing 0) of vector VNAME	`VEC/CRE VEC(10) R 0 10 20 30` `$VLEN(VEC,1) ==> 4`
$NUMVEC	Current number of vectors	`$NUMVEC ==> 2`
$VEXIST(VNAME)	Index of vector VNAME (1..$NUMVEC or 0 if not existing)	`$VEXIST(ZZZ) ==> 0` `$VEXIST(VEC) ==> 2`
$SUBSTRING(STRING,IX,NCH)	STRING(IX:IX+NCH-1)	`$SUBSTRING(ABCDEF,3,2) ==> CD` `$SUBSTRING('ABCDEF',3,2) ==> CD`
$UPPER(STRING)	STRING changed to upper case	`$UPPER(ABCdef123) ==> ABCDEF123`
$LOWER(STRING)	STRING changed to lower case	`$LOWER(ABCdef123) ==> abcdef123`

$LEN(STRING)	Length of STRING, stripping leading/trailing blanks and single quotes	`$LEN(ABCdef123) ==> 9`
		`$LEN('ABCdef123') ==> 9`
		`$LEN('  ABCdef123  ') ==> 9`

$SIGMA(SIGMA_Expression)	Result, as computed by SIGMA, of SIGMA_Expression	`$SIGMA(1+0.5) ==> 1.5` `$SIGMA(SQRT(2)*PI) ==> 4.442883`

The $SIGMA system function in more detail

A SIGMA expression (see also “The PAW system function $SIGMA” on page 68) can involve scalar or vector types of operands, and, according to the type of the result, the string `$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 evaluating the `Sigma_expression`, e.g.

```
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.

```
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, e.g.

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

4.6 On aliases, system functions and macro variables

Alias and system function substitutions are made “literally,” i.e. “as a character string” and regardless the type of the parameter. For example, a system function resulting in a **C**haracter string value can be inserted in place of a numeric type parameter (**I**nteger or **R**eal). 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 **R**eal and the third one of type **C**haracter:

```
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 function resolution

- 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. `variable=value` was not executed within the macro nor `EXEC MACRO variable=value` was used as calling sequence
- The only exception where a variable may be put without brackets is on the right side of an assignement statement, while on the left side a name without brackets is always needed, e.g.

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

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

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

- System functions cannot be nested, e.g. the following is invalid:

  ```
  D=$LEN($SUBSTRING(ABC,2,1))
  ```

- Macro control statements cannot be aliased., e.g. the following code yields an error:

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

- Arguments of EXEC and GOTO may be alias or macro variables. Example:

  ```
  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 to:

- Enter the command LAST which writes all (or some) commands typed in the session into a disk file (by default LAST.KUMAC) and invokes the local editor on the 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 PAW session. At the beginning of another PAW session the old history file LAST.KUMAC is renamed into LAST.KUMACOLD and a new LAST.KUMAC is opened. In this way the user keeps always track of all the commands entered in the previous and in the current PAW sessions. The history files contain also heading and trailing comment lines, showing the date and time at which the PAW session was started/stopped.

  The command LAST 0 MYFILE may be put in the user's PAWLOGON.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[3], CMZ[5] , etc.)

- Use a Unix C-shell-like history mechanism, starting a command with an exclamation mark followed by:

  - An unsigned integer n, to re-execute the **n**-th command entered since the beginning of the session (e.g. !3)
  - A negative integer -n, 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 (e.g. !!)
  - 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 (e.g. just !)

  To have the command numbering the prompt string should be defined as containing an open and closed brackets (e.g. by the command SET/PROMPT 'MYPAW []'), inside which KUIP will put the command number.

4.8 Exception condition handling

User breaks (CTRL/C on VAX, CTRL/Q on APOLLO, etc.) are handled within PAW (IBM excluded): when a break is issued the execution of the current command is aborted and PAW is waiting again for the next command.

Nevertheless, one has to be careful in using this facility, because it requires that the code underneath must be reentrant (this is not the case, for example, for some GKS routines).

Program exception conditions (for example, floating-point overflow, negative square root, etc.) are also handled (IBM included): on the occurrence of the exception a warning is issued and the program flow is restarted again to wait 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 6.1, "Access to SIGMA" on page 68).

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.

Examples:

```
VECTOR/CREATE  X(100)       will create a 100-components vector, values = 0.
SIGMA X=ARRAY(100,1#100)    will create a 100-components vector and assign
                            to each element the values 1,2,...100
VECTOR X(100)               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:

VECTOR/INPUT vlist Input from the terminal values into the vector elements specified by the list vlist.

VECTOR/READ vlist Values can be read in from a file into the vector elements specified by the list vlist.

VECTOR/COPY v1 v2 Values in v1 are copied into v2.

VECTOR/WRITE vlist Values in the vector elements specified by the list vlist can be saved on a file.

VECTOR/PRINT vlist Values of the vector elements specified in vlist will be printed on the terminal.

VECTOR/LIST A **list** of existing vectors and their **characteristics** is printed on the terminal.

VECTOR/DELETE Allows global or selective deletion of vectors.

5.2 Vector addressing

Indexing of vectors is possible*. Examples are:

```
Vec           for all elements
Vec(13)       for element 13
Vec(12:)      for elements 12 up to the last
Vec(:10)      for elements 1 to 10
Vec(5:8)      for elements 5 to 8
```

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

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

5.3 Vector arithmetic operations

A number of basic vector arithmetic operations is available:

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

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.

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.

* Note 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.

5.6 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.

VECTOR/HFILL VNAME ID Each vector element of VNAME is used to fill the existing histogram ID.

GET_VECTOR/CONTENT Provides an interface between vectors and histograms.

PUT_VECTOR/CONTENT Provides an interface between histograms and vectors.

5.7 Graphical output of vectors

VECTOR/DRAW VNAME Interprets the content of the vector VNAME as a histogram contents and draw a `graph`.

VECTOR/PLOT VNAME 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 5 on page 17 in the tutorial section for an explanation of the difference between `vector/draw` and `vector/plot`.

A number of HIGZ[2] macroprimitives are available in PAW. Those directly related to the graphical output of vectors are:

GRAPH N X Y Draw a curve through a set of points defined by arrays X and Y.

HIST N X Y Draw an histogram defined by arrays X and Y.

PIE X0 Y0 RAD N VAL 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

This is shown in the example:*

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

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)
```

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

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 Chapter 5, “Vectors” on page 65). 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 COmponents (**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.

Basic operators

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

Note

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

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

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:

```
A=ARRAY (6,1#6)            1  2  3  4  5  6
A=ARRAY (4)                1  1  1  1
A=ARRAY (5,2&3&-1&2&1.2)   2  3  -1  2  1.2
A=ARRAY (3)*PI             3.1415927  3.1415927  3.1415927
A=ARRAY (1,123E4)          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 5 below.

Table 5. SIGMA functions

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.
LVMIN	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 COmponents** 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.

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,...) will produce

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**.

```
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 $X_1, X_2, \ldots, X_i, \ldots, X_n$ then the corresponding row of the result of the delta function operation will be $Z_1, Z_2, Z_3, \ldots, Z_n$ 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.

```
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 $X_1, X_2, \ldots, X_i, \ldots, X_n$ and creates an array with components equal to the forward difference of X $X_2 - X_1, X_3 - X_2, \ldots, X_{n-1}, X_0$, 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.

```
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+l)** shifts the N component rows of X by 1 to the left, and **LS(X,-l)** shifts the rows by N-1 to the left (or by 1 to the right). If **arg1** is a scalar, **R = arg1**.

```
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      10.00      11.00      12.00
  13.00      14.00      15.00      16.00
  17.00      18.00      19.00      20.00

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
  10.00      11.00      12.00      9.000
  14.00      15.00      16.00      13.00
  18.00      19.00      20.00      17.00

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
  10.00      11.00      12.00      9.000
  14.00      15.00      16.00      13.00
  18.00      19.00      20.00      17.00

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.

```
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.

```
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 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**.

```
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**.

```
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**.

```
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 $X_1, X_2, \ldots, X_n$ and creates an array with components equal to the running product of the component of the argument: $X_1, X_1 \times X_2, \ldots X_1 \times X_2 \times \cdots, X_n$

```
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:

$$R_1 = \int_{x_1}^{x_1} (\arg_1)\, dx = 0$$

for the second and third points

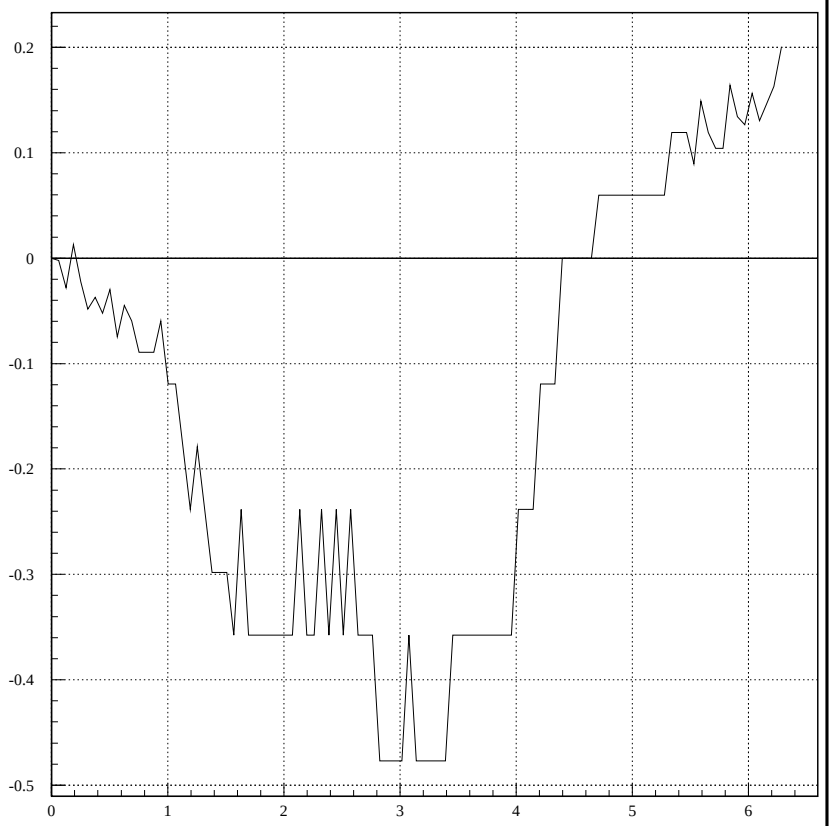
$$R_{i+1} = R_i + \frac{h}{24}(9f_i + 19f_{i+1} - 5f_{i+2} + f_{i+3})$$

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
```



R=SUMV(arg)

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

```
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 $X_1, X_2, \ldots, X_i, \ldots, X_n$ and creates a scalar whose value is equal to the sum of all the components of X: $X_1 + X_2 + X_2 + \cdots + X_n$.

```
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=vsum(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|)I0(x) (C313)
EBESI1 exp(-|x|)I1(x) (C313)
EBESK0 exp(x)K0(x) (C313)
EBESK1 exp(x)K1(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

Note

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

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 [7], 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** [6] 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.

HBOOK functionality:

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 histogramed 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 histogramed 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** data manager[4] to store its data elements in a COMMON block /PAWC/, shared with the **KUIP**[9] and **HIGZ**[2] 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 Figure 37 on page 83 and Figure 39 on page 84*.

In the /PAWC/ data store, the HBOOK, HIGZ and KUIP packages have all their own **division** (see [4] for more details on the notion of divisions) as is shown in Figure 34.

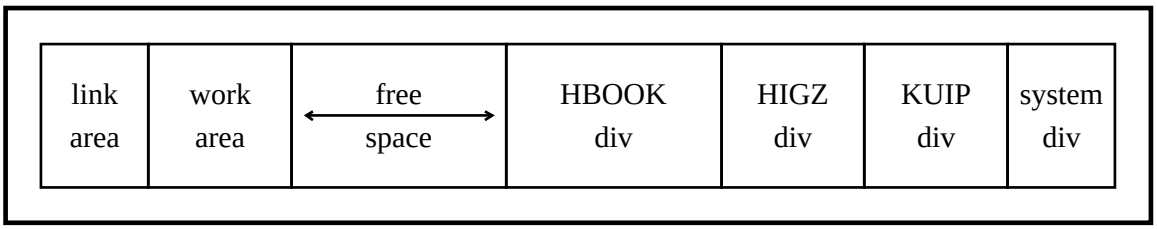


Figure 34. The layout of the /PAWC/ dynamic store

The use of ZEBRA

Inside the HBOOK division the various data elements are stored as a ZEBRA data structure, one for each "identifier". In fact all identifiers (histogram or Ntuple numbers) are stored in an ordered array in a ZEBRA bank and access to the information associated with the HBOOK data is via the **reference link** at the same offset as the identifier in the data part of the bank. The data structure for a given element depends on its characteristics. In any case the top bank for a given element contains the title and other constants, while the data themselves are stored in another bank hanging from the previous one. Sometimes other banks are created, e.g. for automatic binning, for storing the limits of the elements of a Ntuple and, when a Ntuple is kept in memory, for containing the overflow of the data, for **projections**, **slices** and **bands** in the 2-dim case of for containing the **errors** associated to a bin. This means that each HBOOK identifier has a whole set of **attributes** associated with its existence, and when a histogram or Ntuple is written to backup store and later reread, the **complete data structure**, containing all characteristics and attributes are retrieved. Figure 35 on page 80 shows the ZEBRA data structure for a two-dimensional histogram. The precise layout of this bank should be of no concern to the user. It is only shown here as an example of the underlying ZEBRA structure of HBOOK. Note the use of the **data** part of the bank for storing attributes (e.g. title, number of bins, number of entries) as well as of the **link** part for storing the addresses to access the associated data points (scatter plot contents, X and Y projections, slices and bands and their associated errors).

* This 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.

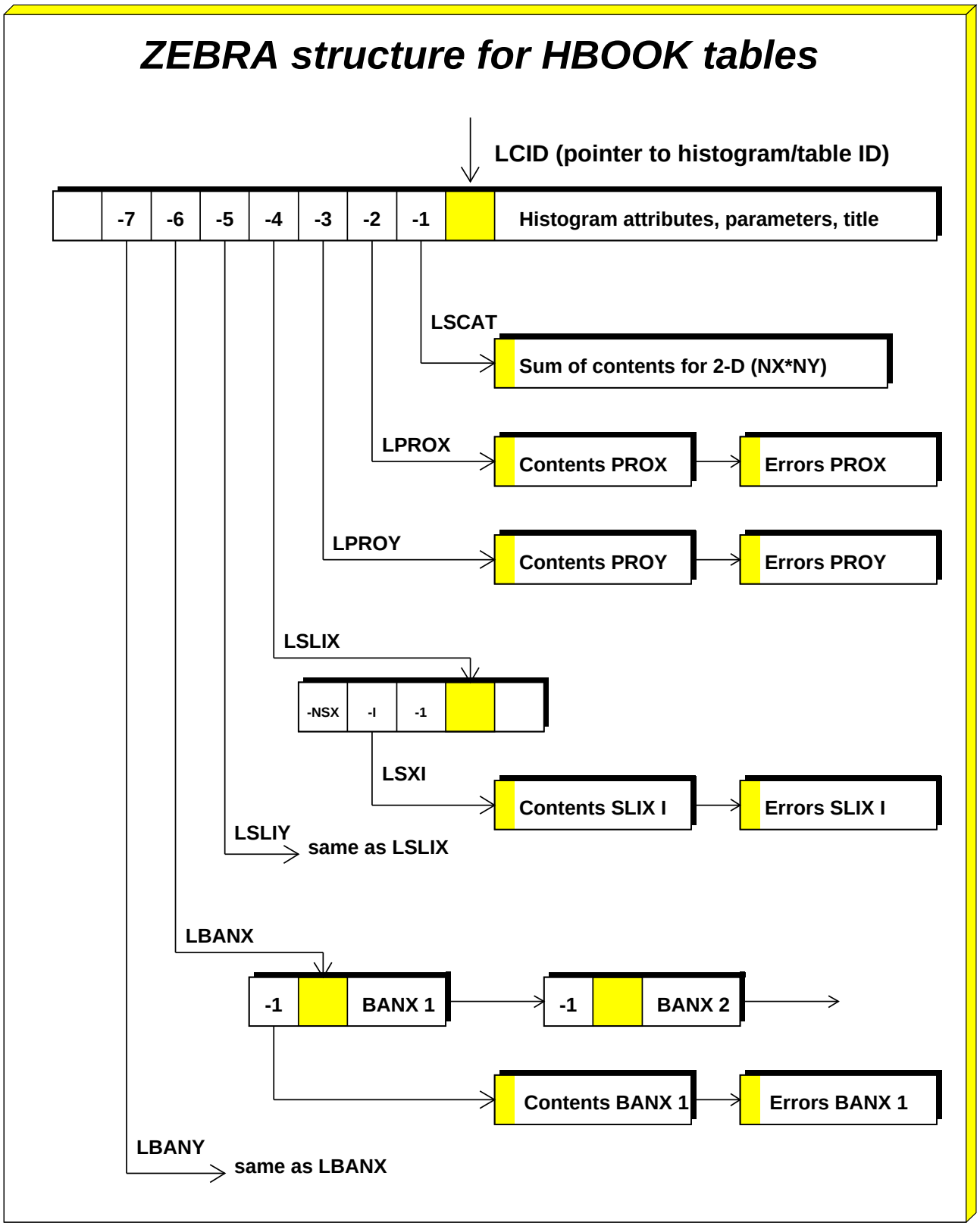


Figure 35. The ZEBRA data structure used for two-dimensional histograms

RZ directories and HBOOK files

Another advantage of the use of 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 HRFILE 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.

```
CALL HRFILE(1,'HISTO1',' ')    ! Open first  HBOOK RZ file (read only)
CALL HRFILE(2,'HISTO2','U')    ! Open second HBOOK RZ file (update)
CALL HCDIR('//HISTO1',' ')     ! Make HISTO1 current directory
CALL HRIN(20,9999,0)           ! Read ID 20 on file 1
  ....
CALL HCDIR('//HISTO2',' ')     ! Make HISTO2 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('HISTO1')           ! Close file 1
CALL HREND('HISTO2')           ! Close file 2
```

In the previous example (and also in figures 37 and 39) 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.

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 > CDIR //LUN1            | Set current directory to //LUN1
PAW > ADD 10 20 30           | Add histograms 10 and 20 into 30
                             | Histogram 30 is created in //PAWC
PAW > Histo/Plot //PAWC/30   | Show the result of the sum
PAW > CD //PAWC              | Set the current directory to memory
PAW > Histo/plot 30          | 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 > Histo/file 1 his.dat  | The file contains ID=10
PAW > Histo/Plot 10         | ID=10 read from file and plotted
PAW > H/plot 10             | ID=10 read again from file and plotted
PAW > H/fit 10 ! G          | Read from file, make a Gaussian fit on //PAWC/10
PAW > H/plot 10             | ID=10 read from memory since it changed
PAW > H/del 10              | Delete histogram 10 from memory
PAW > H/plot 10             | ID=10 read again from file and plotted
```

7.3 HBOOK batch as the first step of the analysis

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 **DST**s 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, 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.

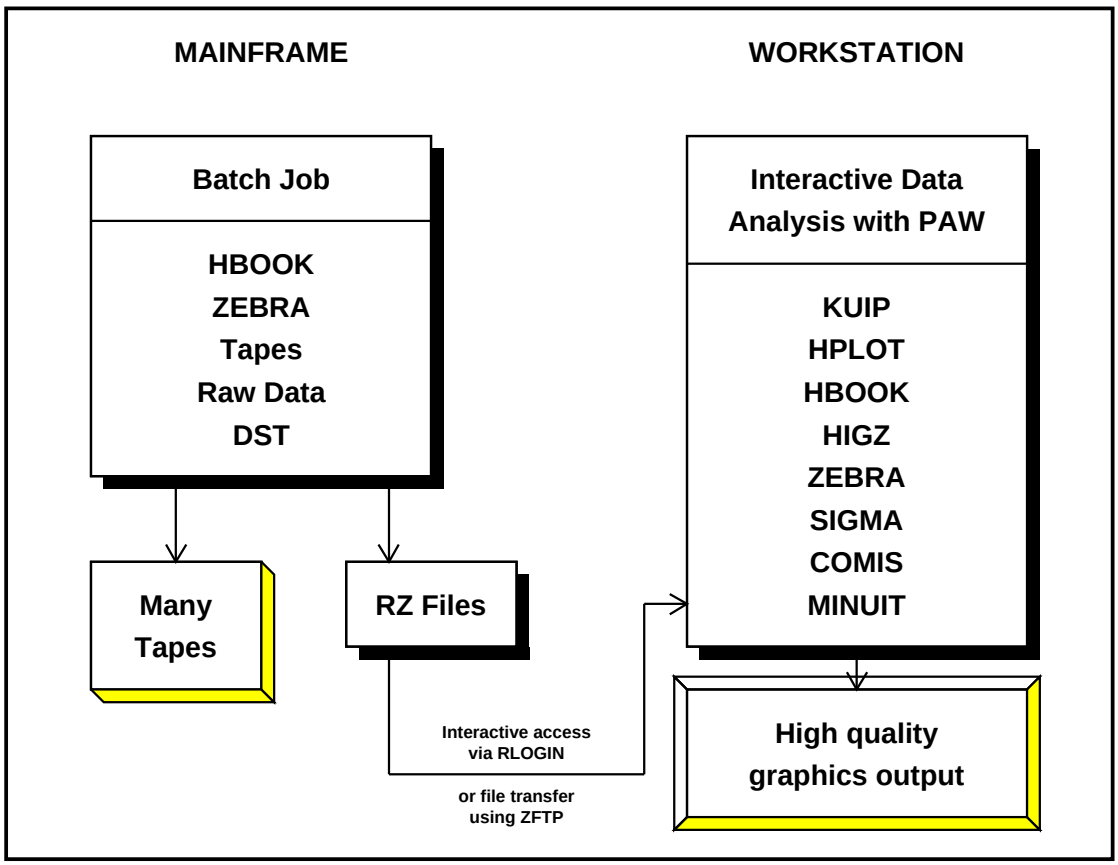


Figure 36. Schematic presentation of the various steps in the data analysis chain

A typical batch job in which data are analysed offline and some characteristics are stored in HBOOK is given in Figure 37 on page 83 After opening the RZ HBOOK file, HBOOK is initialised by a call to `HLIMIT`, which declares a length of 80000 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 38 on page 83.

```
      PROGRAM HTEST
      COMMON/PAWC/H(80000)
      EXTERNAL HTFUN1,HTFUN2
*.------------------------------------------------------------------
      CALL HLIMIT(80000)
*             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 HRPUT(0,'HTEST.DAT','N')
      CALL HDELET(100)
      CALL HDELET(200)
      CALL HPRINT(0)
      END

      FUNCTION HTFUN2(X,Y)

      HTFUN2=HTFUN1(X)*HTFUN1(Y)

      RETURN
      END

      FUNCTION HTFUN1(X)
      DATA C1,C2/1.,0.5/,XM1,XM2/0.3,0.7/,XS1,XS2/0.07,0.12/

      A1=-0.5*((X-XM1)/XS1)**2
      A2=-0.5*((X-XM2)/XS2)**2
      X1=C1
      X2=C2
      IF(ABS(A1).GT.0.0001)X1=C1*EXP(A1)
      IF(ABS(A2).GT.0.0001)X2=C2*EXP(A2)

      HTFUN1=X1+X2

      RETURN
      END
```

Figure 37. Writing data to HBOOK with the creation of a HBOOK RZ file

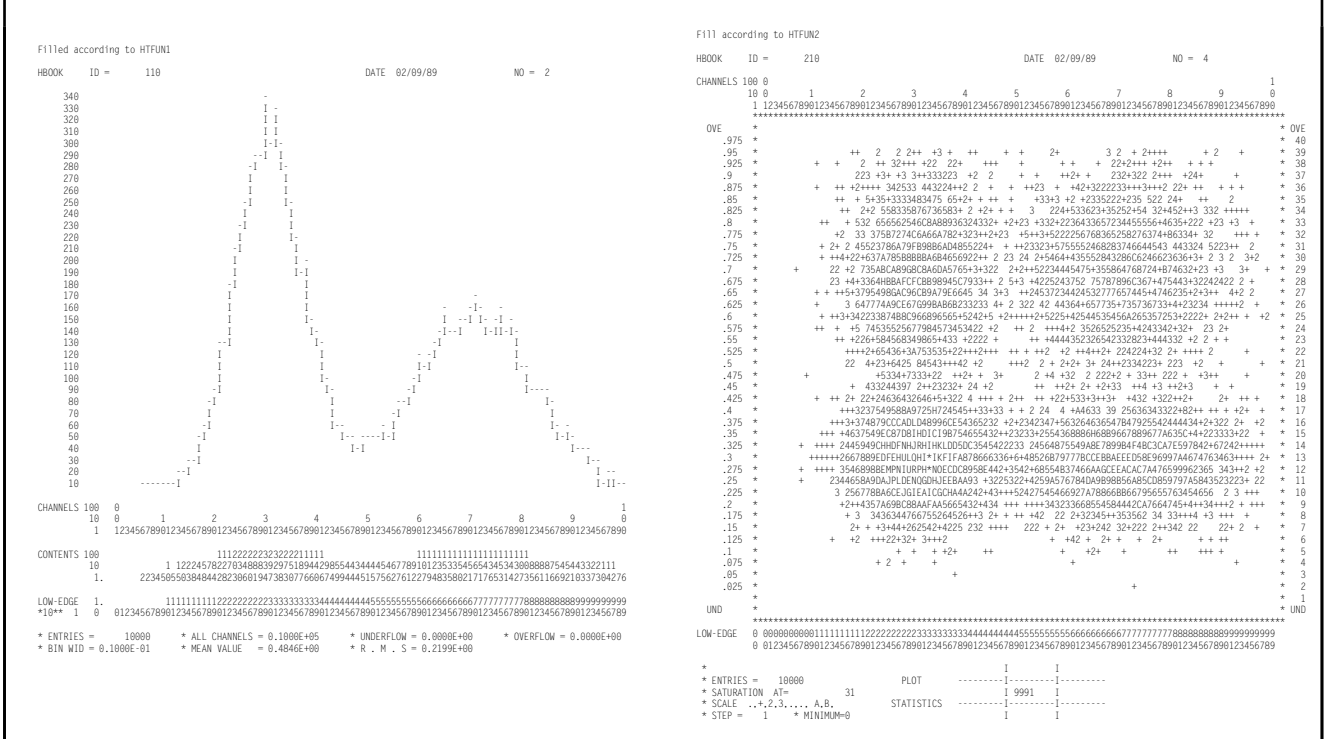


Figure 38. Output generated by job HTEST

Adding data to the RZ file

Program HTEST1 below shows how to add some data to the HBOOK RZ file created in 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 (see output of HLDIR command at end of 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 X versus Y 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 data written in the first job (HTEST) are indeed still present in the file under the top directory //EXAM2. Routine RZSTAT shows usage statistics about the RZ file.

```
      PROGRAM HTEST1
      COMMON/PAWC/H(80000)
      DIMENSION X(3)
      CHARACTER*8 CHTAGS(3)
      DATA CHTAGS/'   X   ','   Y   ','   Z   '/
*.------------------------------------------------------------
      CALL HLIMIT(80000)
*            Reopen data base
      CALL HROPEN(1,'EXAM2','HTEST.DAT','U',0,ISTAT)
      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
```

```
TEST1

HBOOK     ID =       10                              DATE  02/09/89            NO =  1

     280
     270                                              - -
     260                                             I I  -
     250                                          -  I I  I
     240                                       - I  I-I- I -
     230                                       I-I--I  I I-I-
     220                                      -I       I I  I-
     210                                      I        I I   I-
     200                                      I        I-I   I-
     190                                 - - --I               I --
     180                                 I-I-I                 I-II--
     170                                 I                           I
     160                                 I                          I--
     150                                - -I                          I --
     140                               -I-I                           I II
     130                               -I                             I-II-
     120                              -I                                  I-
     110                             --I                                   I--
     100                           --I                                      I
      90                          I                                         I
      80                          I                                         I----
      70                        -I                                              I-
      60                       -I                                               I--
      50                    ---I                                                   I--
      40                 -----I                                                      I--
      30                 I                                                           I-----
      20              - ----I                                                              I---
      10     --------I-I                                                                    I--------

CHANNELS 100   0                                                                          1
          10   0        1        2        3        4        5        6        7        8        9        0
           1   12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

CONTENTS 100                          1111111111111112222222222122222211111111111111
          10          1 1111333333444666980012343487888813252263749623319978877552442100777655443322222111
           1.  126648787712793258751606930343464432299094980936700403605684452524397532496351678256536531219485621 1

LOW-EDGE      ------------------------------------------------------
           1.  32222222222222221111111111111111                                  1111111111111111222222222222222
           0   0988776554432211099887665543322109998776654433211008112334456677899001223345566788990112233445567788
           0   0482604826048260482604826048260482604826048260482606284062840628406284062840628406284062840628406284

* ENTRIES =     10000     * ALL CHANNELS = 0.9969E+04     * UNDERFLOW = 0.1200E+02     * OVERFLOW = 0.1900E+02
* BIN WID = 0.6000E-01     * MEAN VALUE   =-0.3907E-02     * R . M . S = 0.9857E+00
```

```
TEST2

HBOOK     ID =       20                              DATE  02/09/89            NO =  2

CHANNELS  10 U 0        1        2        3 O
           1 N 12345678901234567890123456789 V
           ***********************************
  OVE      *        + ++ +232++2+ +++          * OVE
    2.8    *     ++ 2    +2 + 2  +             *  30
    2.6    *          2 2+  +34+++ ++   +      *  29
    2.4    *         2+ 3322343+ 3++ +         *  28
    2.2    *    + 2    247236663524+23++   +   *  27
    2      *    +    2+23769597A75 6+2+ 2      *  26
    1.8    *       + 5598576EBCDAA53357  2+ +  *  25
    1.6    *      ++3278CC9JFO8F98C86643+2+     *  24
    1.4    *      344686AAGJJMEMIDFG964232+   + *  23
    1.2    *    ++++44BBJGMOOPWNICCGI97322++  + *  22
    1      *     2+545BGOMTSX*VYTJMCFA755++2    *  21
     .8    *    2+4799DHSRUX****VXRQJC57635+    *  20
     .6    *   + +25CBEKLZ********MXGGCI4322  3 *  19
     .4    * 2   4+779BN*U*********YOIFB862     *  18
     .2    * 2 ++266CCLR************OIHA464+2 4 *  17
           * +  3238ECX*T***********YKPC772   + *  16
-    .2    * + +423D6LDS**X********ZUMGC543+  2 *  15
-    .4    * +  2347CAHSSX*********UMK75D2 3  + *  14
-    .6    *   2334AAKML*V**********IIH9773++ + *  13
-    .8    *   +22565CLJL*X******Z*TL9H948+ +   *  12
-   1      * 2 2 32666EMLN****Q*ULLQMABB342+  2 *  11
-   1.2    *   + 22377BDIUS*P***TTUNBDA545+2    *  10
-   1.4    * + + 2 +689E7KKNWUNRIHJCEA472+++  + *   9
-   1.6    *     2+3+74BCMJIGOIKEIAAD6643++   2 *   8
-   1.8    * + + +2222856AA8HGJACB6786+2+2++    *   7
-   2      *   +   2 +27359BEDC5977634++        *   6
-   2.2    * +   + ++2+2749775488 83+++2 +++    *   5
-   2.4    *         +  +3367558445+442+   +    *   4
-   2.6    *       +2 +  2224+6++7234 +    +    *   3
-   2.8    *          +  33+3+322++ +           *   2
-   3      *       ++ ++ 22 2 +4+2 2            *   1
  UND      *          + +  23 +2+++      +     * UND
           ***********************************
LOW-EDGE       ---------------
           1.  32222211111        1111122222
           0   0864208642086420246802468024 68

 *                                                I   19    I
 * ENTRIES =    10000                 PLOT       ---------I---------
 * SATURATION  AT=         255                     12    I 9936    I   19
 * SCALE  .,+,2,3,..., A,B,           STATISTICS  ---------I---------I---------
 * STEP =    1     * MINIMUM=0                    I   14    I
```

```
 ********************************************************
 * NTUPLE ID=   30  ENTRIES=  10000   N-TUPLE           *
 ********************************************************
 *  Var numb  *   Name    *    Lower     *    Upper     *
 ********************************************************
 *      1     *    X      * -.422027E+01 * 0.386411E+01 *
 *      2     *    Y      * -.411076E+01 * 0.378366E+01 *
 *      3     *    Z      * 0.485187E-04 * 0.179518E+02 *
 ********************************************************


 ===> Directory : //EXAM2/NTUPLE
         30 (N)   N-TUPLE
         10 (1)   TEST1
         20 (2)   TEST2

 ===> Directory : //EXAM2
        100 (1)   Test of HRNDM1
        110 (1)   Filled according to HTFUN1
        200 (2)   Test of HRNDM2
        210 (2)   Fill according to HTFUN2


      NREC    NWORDS    QUOTA(%)  FILE(%)   DIR. NAME
       34      34064       0.85     0.85   //EXAM2/NTUPLE
       41      40431       1.02     1.02   //EXAM2
```

Figure 39. 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 generated*. The PAW session below shows that the file HTEST.DAT is first opened via a call to HISTO/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 39 on page 84)

```
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 40. Reading a HBOOK direct access file

* In 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 Chapter 9, “Distributed PAW” on page 133 for more details.

Plot histogram data

The analysis of the data can now start and we begin by looking at the histograms in the top directory. Figure 41 shows the commands entered and the corresponding output plot. They should be compared with the lineprinter output in Figure 38 on page 83.

```
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
```

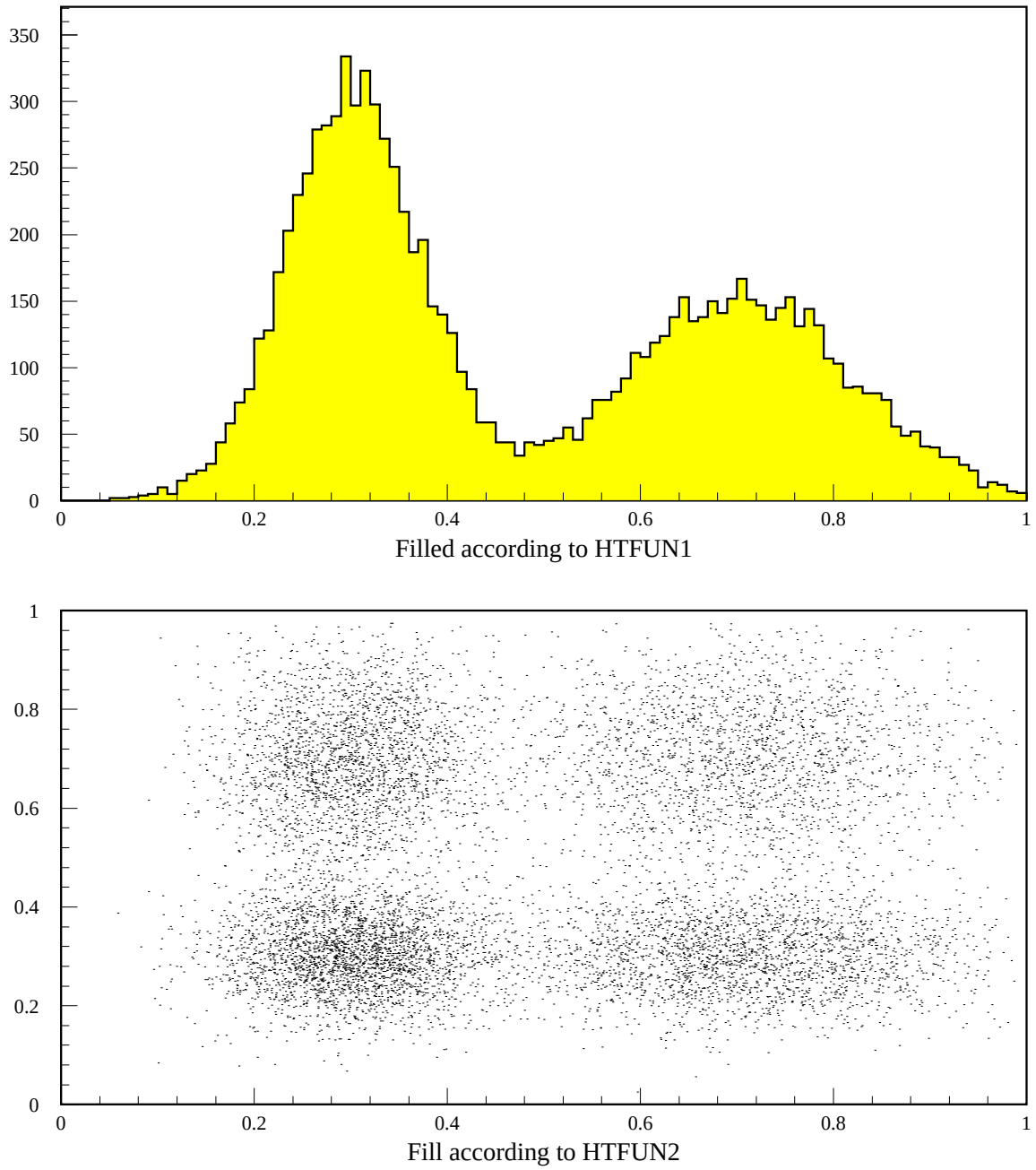


Figure 41. Plot of one- and two-dimensional histograms

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 42)

```
 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 42. Print and scan Ntuple elements

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 6)
UWFUNC Selection function (see Table 7) - 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)

Ntuple variable and selection function specification

Table 6. Syntax for specifying Ntuple variables

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 the 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. Syntax of a selection function used with a Ntuple

Format	Explanation	Example
"0" or missing	No selection is applied (weight is 1).	NT/PLOT 30.X
Combination of cuts	A CUT or combination of CUTs, each created by the command NTUPLE/CUTS	NT/PLOT 30.X 1 (use cut number 1) NT/PLOT 30.X 1.AND.2 NT/PLOT 30.X .NOT.(1.AND.3).OR.2
Combination of masks	A MASK or combination of MASKs, each created by the command NTUPLE/MASK	Assuming there exists a mask vector MSK NT/PLOT 30.X MSK(4) (bit 4) NT/PLOT 30.X MSK(1).OR.MSK(6)
Logical expression	Any **logical** combination of conditions between Ntuple variables, cuts and masks.	NT/PLOT 30.X X>3.14.AND.(Y<Z+5.) NT/PLOT 30.X 1.AND.MASK(3).OR.Z<10
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.	NT/PLOT 30.X Y weight X by Y NT/PLOT 30.X X**2+Y**2 weight X by $X^2 + Y^2$
Selection function	Name of a selection function in a text file of the form FUN.FTN, FUN FORTRAN, FUN.FOR (Apollo, IBM, VAX). The function value is applied as a **weight**	NTUPLE/PLOT 30.X SELECT.FOR 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	NT/PL 30.3%F1.FTN*SIN(X) 1.OR.F2.FTN

Ntuple selection mechanisms

With most Ntuple operations a selection "function" UWFUNC of a form described in Table 7 on page 88 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.

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.

```
 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.

```
 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. Examples are:

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

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

```
 PAW > NT/CUT 0 W cuts.dat                | write all cuts to file
 PAW > NT/CUT 4 R cuts.dat                | read cut 4 from file
 PAW > NT/CUT 4 P                         | 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 43).

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

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

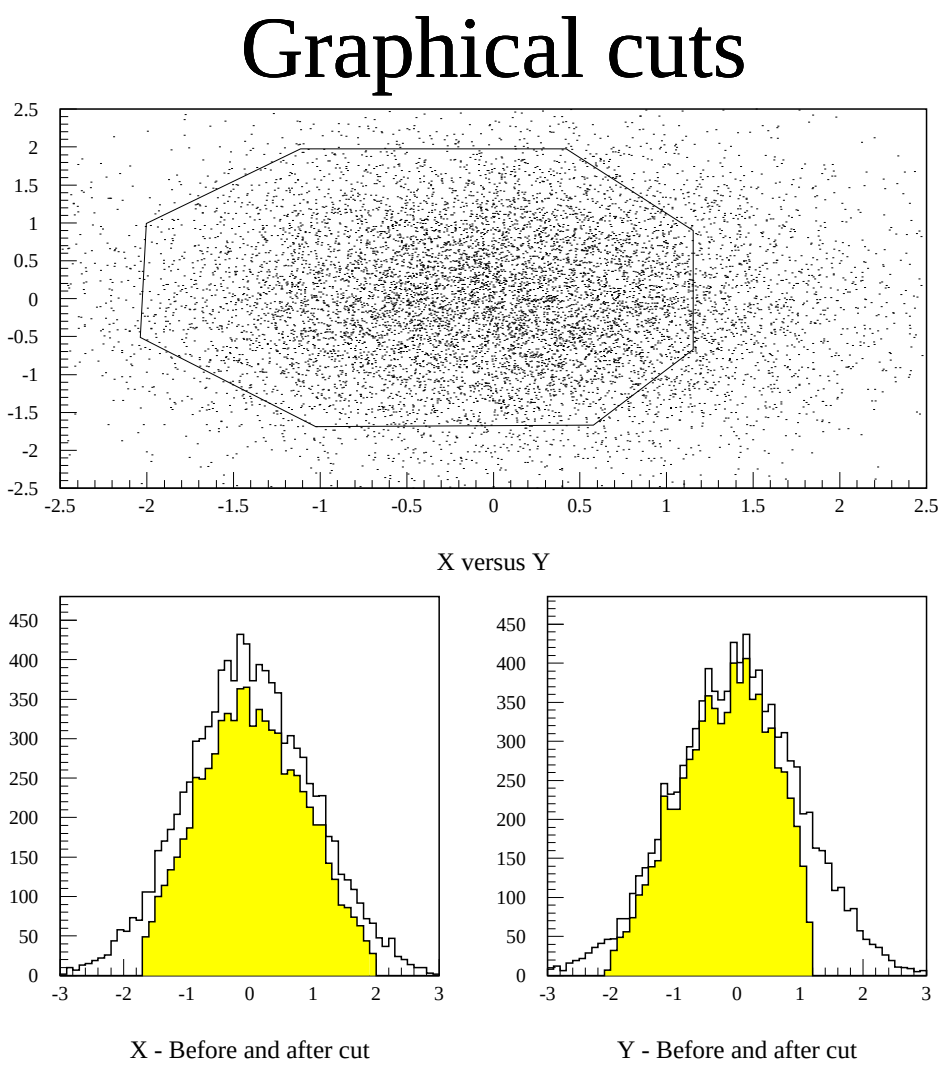


Figure 43. Graphical definition of cuts

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.

Table 8. Functions callable and common blocks which can be referenced from an external function with PAW

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 HPLOC HPLSET HPLGIV HPLOC
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 SBIT0 SBIT1 SBYT UCOPY UCTOH UHTOC VZERO
COMMON blocks	/PAWC/, /QUEST/, /KCWORK/, /PAWPAR/, /PAWIDN/

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.

```
 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    ,   Y    ,   Z
      COMMON/PAWIDN/IDNEVT,VIDN1,VIDN2,VIDN3,
     +    X    ,   Y    ,   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.

Examples

To put into practice the syntax explained above let us consider Figure 44. 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 39 on page 84. 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 44.

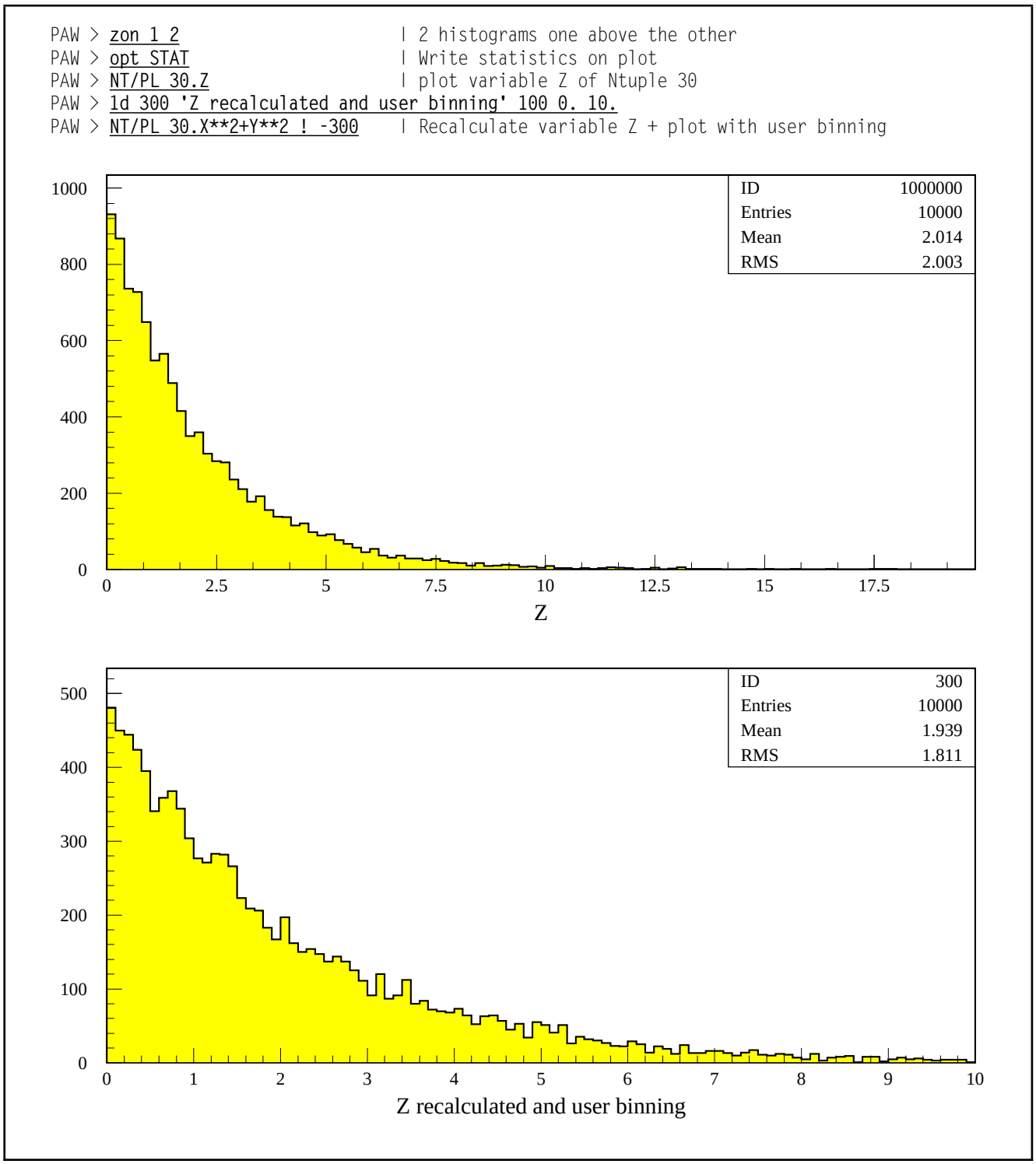
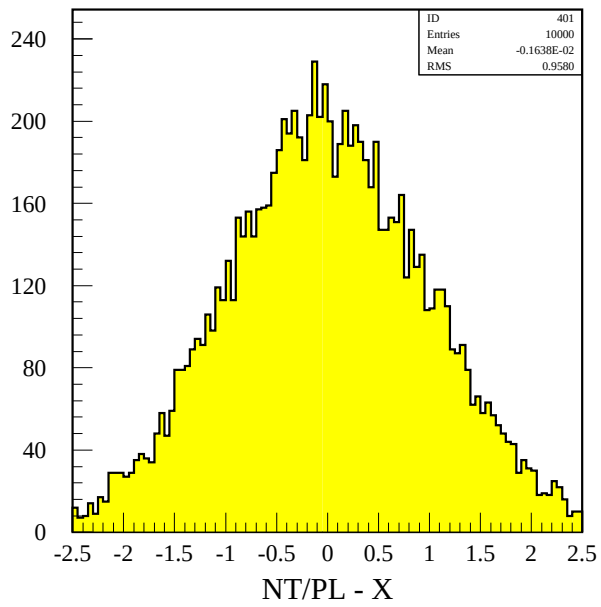
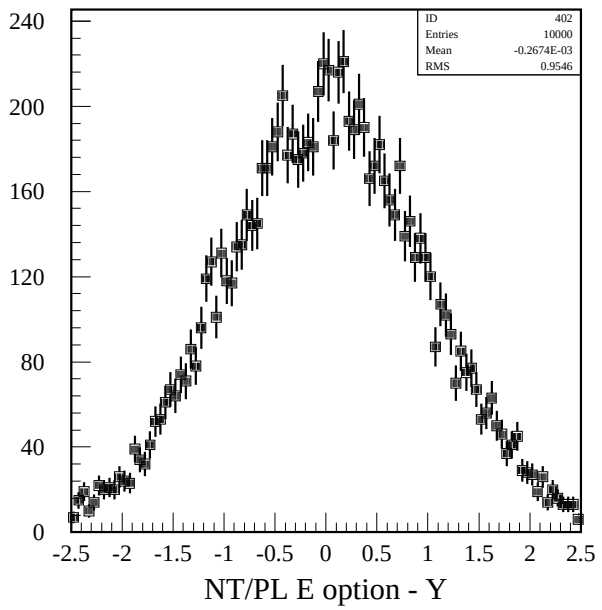
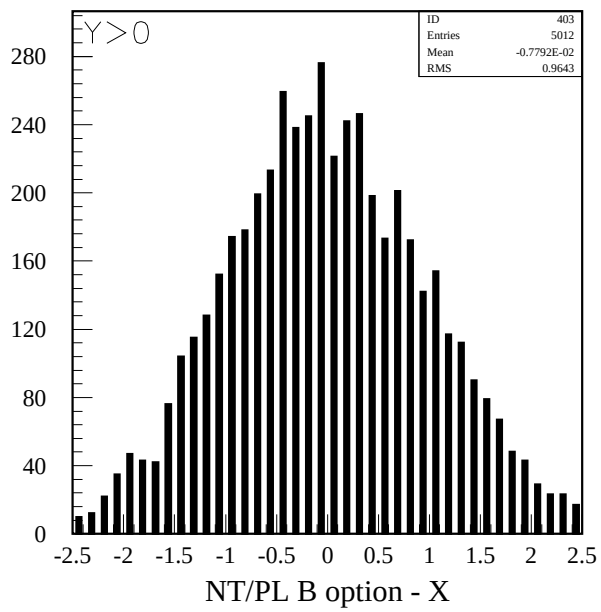


Figure 44. Read and plot Ntuple elements

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







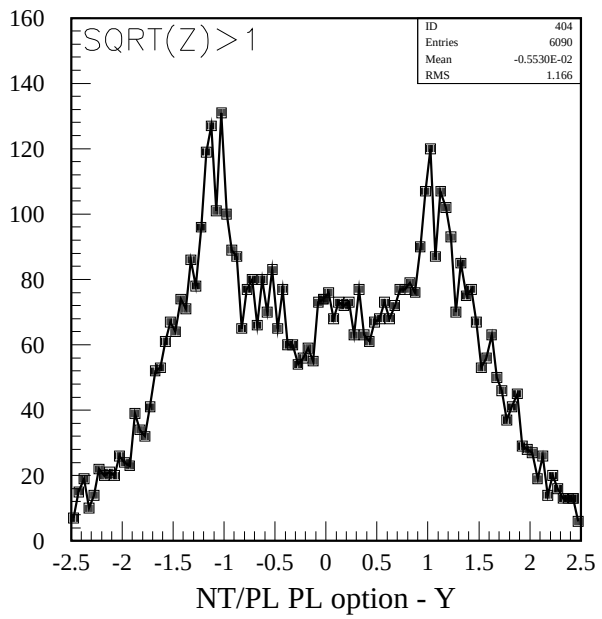


Figure 45. Selection functions and different data presentations

7.6 Fitting with PAW/HBOOK/MINUIT

MINUIT[20]* 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.

Basic concepts - The organization of MINUIT.

The MINUIT package acts on a multiparameter FORTRAN function to which we give the generic name **FCN**. In the PAW/HBOOK implementation, the function FCN is called `HFCNH` when the command `Histo/Fit` 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. The 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. The 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.

Basic concepts - The transformation for parameters with limits.

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

$$P_{int} = arc\sin(2\frac{P_{ext}-a}{b-a} - 1) \qquad P_{ext} = a + \frac{b-a}{2}(\sin P_{int} + 1)$$

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 $\partial P_{ext} / \partial P_{int} \approx 0$. 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.
- 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.

* The following information about MINUIT has been extracted from the MINUIT documentation

How to get the right answer from MINUIT.

MINUIT offers the user a choice of several minimization algorithms. The MIGRAD* 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 $\partial F/\partial P_{int}$ is zero no matter what the real derivative $\partial F/\partial P_{ext}$ is.

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

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.

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 discussuion of basic concepts, such as the meaning of the elements of the error matrix, or setting of exact confidence levels, see [11, 18, 19].

The 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.

---

* Other algorithms are available with Interactive MINUIT, as described in 7.7, “Doing more with MINUIT” on page 102

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:

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.

Fitting histograms

The general syntax of the command to fit histograms is:

HISTOGRAM/FIT 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 7.7, "Doing more with MINUIT" on page 102)
- **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 ($0 \le NP \le 34$)

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**.

A simple fit with a gaussian

```
PAW > opt stat                          | Select option to show histogram statsitics 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
```

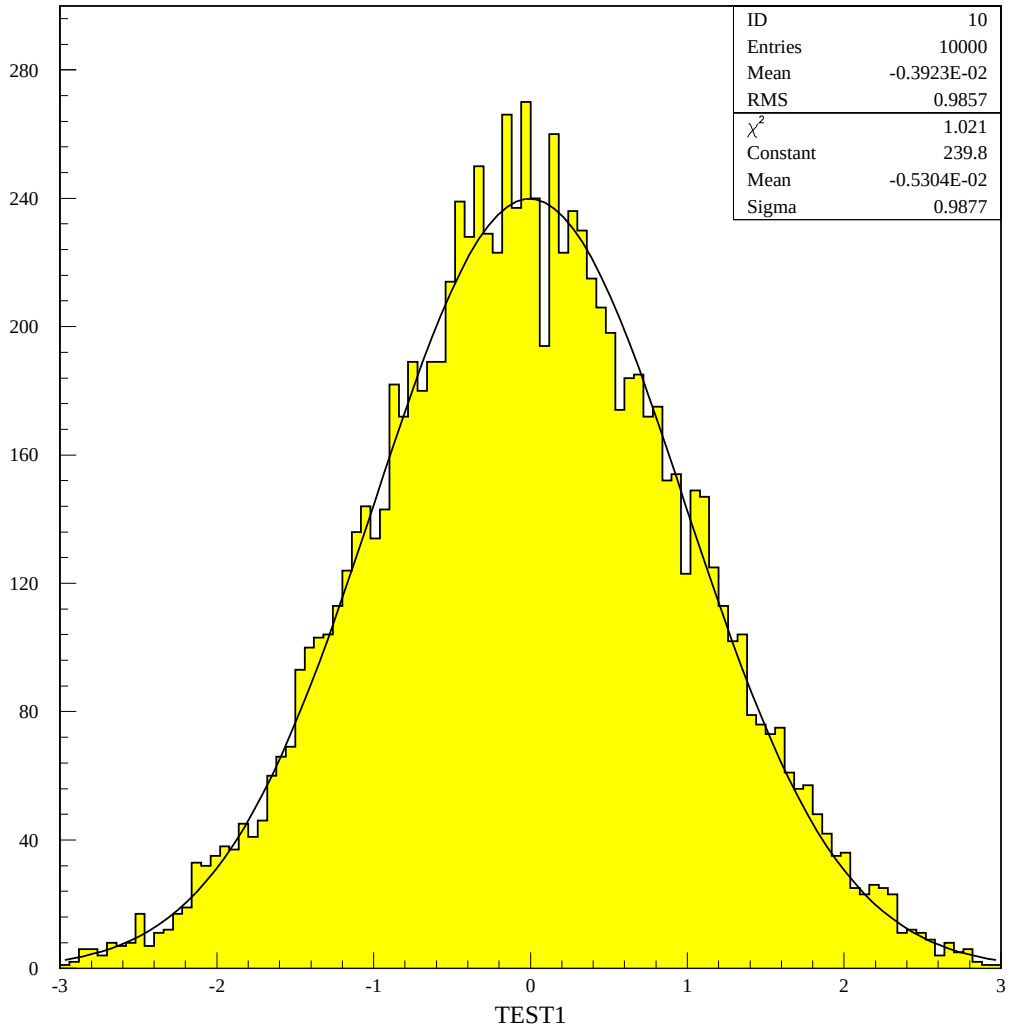


Figure 46. Example of a simple fit of a one-dimensional distribution

Fit parts of an histogram separately

```
PAW > opt NSTA                                    | Turn off option showing statistics on plot
PAW > ve/cr par(6)                                | Create a vector with 6 elements
PAW > set fit 111                                 | Show fitted parameters + errors on plot
PAW > hi/fit 110(1:50) G ! 0 par                  | Fit first half with a gaussian and plot

    *******************************************
    *                                         *
    * 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 > hi/fit 110(50:99) G 0 0 par(4)              | Fit second half with gaussian, do not plot

    *******************************************
    *                                         *
    * 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 > hi/plot 110 SFUNC                           | Plot result of fit on Same plot
PAW > ve/pr par(1:6)                              | 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
```

Figure 47 (Part 1 of 2). Example of a fit using sub-ranges bins

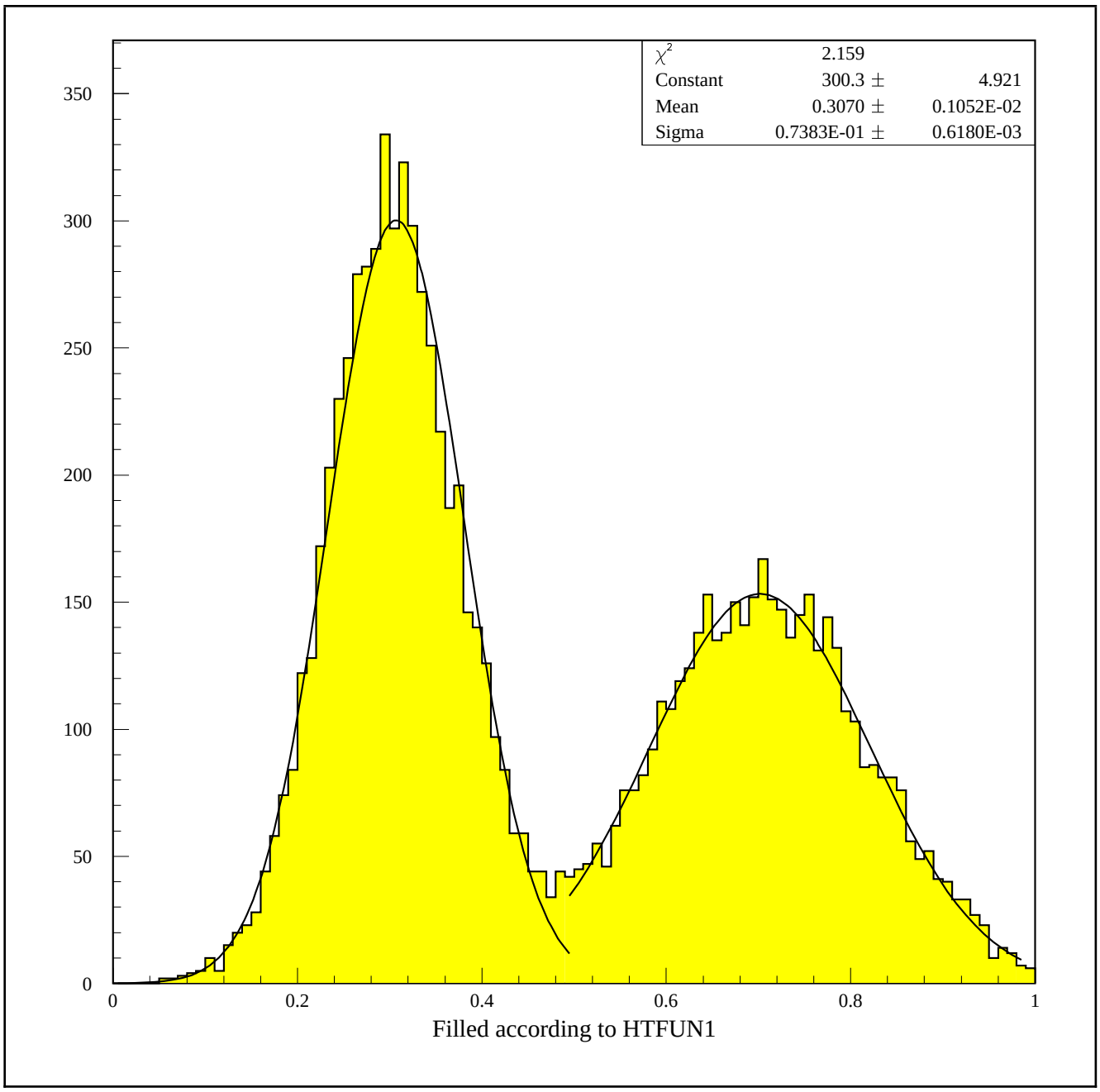


Figure 47 (Part 2 of 2). Example of a fit using sub-ranges bins

In Table 9 the results for the fitted parameters are compared with the initial values with which the gaussian distributions were generated in the "batch" job in Figure 37 on page 83. The table also includes the result of the double gaussian fit in the next section (Figure 48 on page 101).

Table 9. Comparison of the results of the fit for the double gaussian distribution

Parameter	Input value	Result of Figure 47	Result of Figure 48
First Gaussian:			
Height	1. (normalised)	300. ±5.	308. ±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
Second Gaussian			
Height	0.5 (normalised)	153. ±3	154. ±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

A more complex fit

```
PAW > * Create a vector of 6 elements and provide 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
```

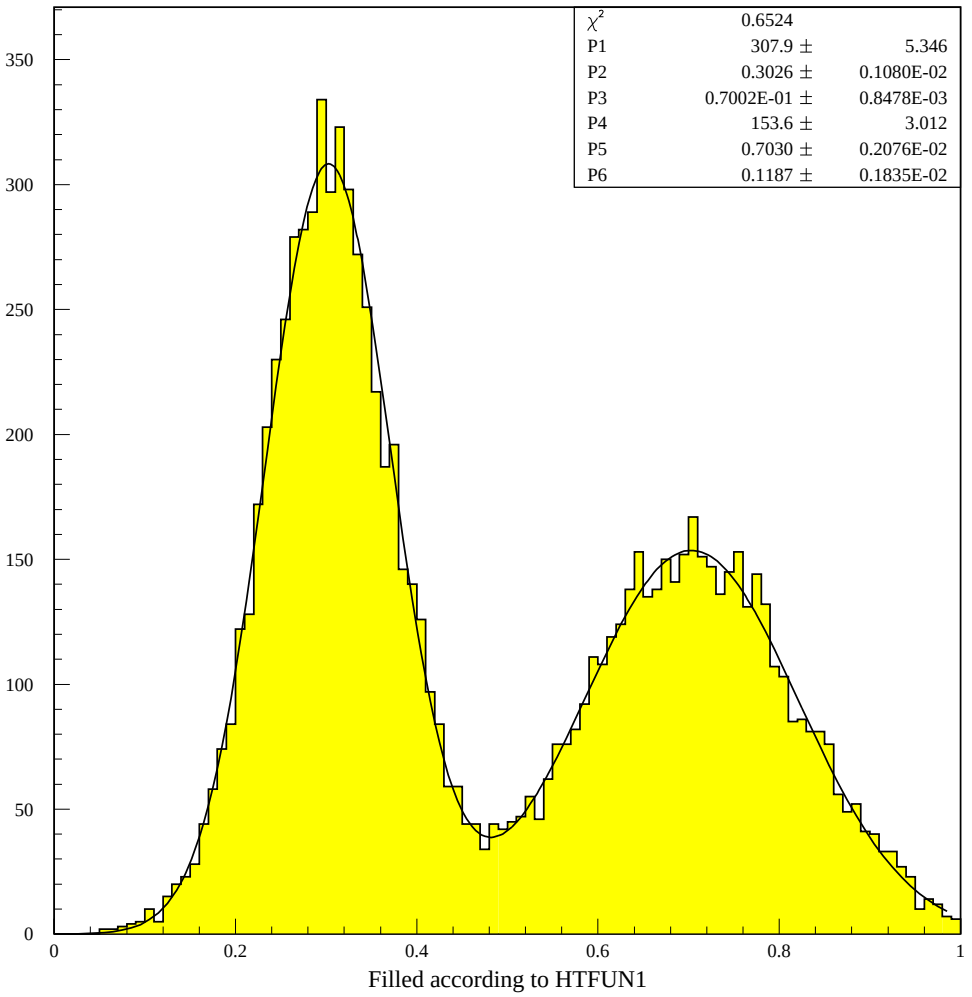


Figure 48. Example of a fit using a global double gaussian fit

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*[tolerance]*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.).

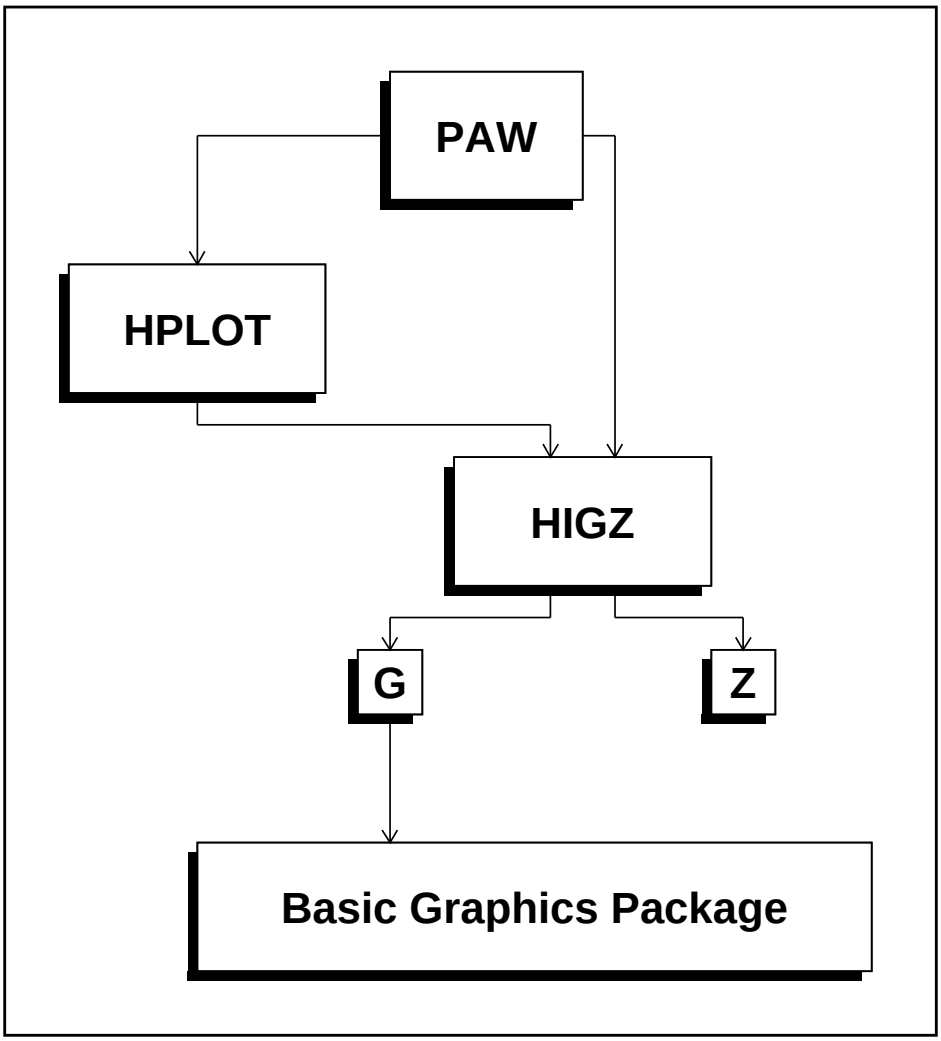


Figure 49. HPLOT and HIGZ in PAW

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/PLOT` 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:

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

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

In addition, more simple and 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 the 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 the **Z** on.

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

will display the picture PNAME. If the 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/PLOT ID` command a new HIGZ picture is created with an automatic naming:

```
PAW > HIST/PLOT 10
PAW > HIST/PLOT 110
PAW > HIST/PLOT 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, should we have 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 POST
FORTRAN/FILE 66 PAW.PS
META -66 -111
PICT/PLOT ' '
CLOSE 66
SHELL PRINT PAW.PS
RETURN
```

By typing `EXEC POST` the current HPLOT picture on the screen will be sent to the printer via the `SHELL` command which send to the local operating system the system-dependent **print** command (e.g. `prf -trans -pr lw` on APOLLO).

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.

Pictures in direct access files

HIGZ pictures are stored in 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 with 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 in ZEBRA files are shown below. Note that in the example the picture names could be '*', ' ' or a number.

```
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 9.2, "ZFTP" on page 136. As the size of the picture data base (see 8.3, "The HIGZ pictures" on page 106), 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 also 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.)

```
      PROGRAM HPICT
*.==========>
*.           HPLOT Program to demonstrate how to store HPLOT
*.           generated pictures into a direct access HIGZ picture file
*..=========>
      COMMON/PAWC/H(20000)
      DIMENSION SIG(2)
      CHARACTER*20 TITLE
*._____________________________________
*.
      CALL HLIMIT(20000)
*
*         Create histograms
*
      DO 10 ID=1,10
         WRITE(TITLE,1000)ID
 1000    FORMAT('Test number',I3)
         CALL HBOOK1(ID,TITLE,100,-3.,3.,0.)
  10  CONTINUE
*
*         Fill histograms
*
      DO 30 ID=1,10
         DO 20 I=1,1000
            CALL RANNOR(A,B)
            CALL HFILL(ID,A,0.,1.)
  20     CONTINUE
         CALL HFITGA(ID,COEFF,AV,SIGM,CHI2,2,SIG)
  30  CONTINUE
*
*         Initialize HPLOT. Set various graphics options.
*
      CALL HPLINT(0)
      CALL HPLZON(1,2,1,' ')
      CALL HPLOPT('ZFL',1)
      CALL HPLOPT('FIT',1)
      CALL HPLOPT('STAT',1)
      CALL HPLSET('STAT',1.)
      CALL HPLSET('HTYP',244.)
      CALL HPLSET('FWID',5.)
      CALL HPLSET('VFON',-40.)
      CALL HPLSET('TFON',-60.)
      CALL HPLSET('PWID',4.)
      CALL HPLSET('BCOL',1.01)
      CALL HPLSET('CSIZ',0.25)
      CALL HPLSET('CFON',-10.)
*
*         Open a picture file called "hpict.dat".
*         Option 'A' means "Automatic saving of pictures"
*         Option 'N' means "New file"
*         (Use option 'U' instead of 'N' to update an existing file)
*
      CALL IZOPEN(1,'Pictures','hpict.dat','AN',1024,ISTAT)
*
*         Select HIGZ option to store graphics in ZEBRA memory only
*         No calls to the local graphics package.
*
      CALL IGZSET('Z')
*
*         Plot all histograms
*
      CALL HPLOT(0,' ',' ',0)
      CALL HPLEND
*
      END
```

```
PAW > PICT/FILE 20 HPICT.DAT
PAW > LDIR
                        Directory ===> //LUN20 <===

     Created 891006/1026  Modified 891006/1026

 ===> List of objects
     PICTURE  NAME                          CYCLE
         PICT1                                1
         PICT2                                1
         PICT3                                1
         PICT4                                1
         PICT5                                1
PAW > META 10 -111
PAW > PICT/PLOT PICT2
PAW > CLOSE 10
PAW > * Print metafile
PAW > *  (see pages 106 and following)
PAW > SHELL print PAW.METAFILE
PAW > EXIT
```

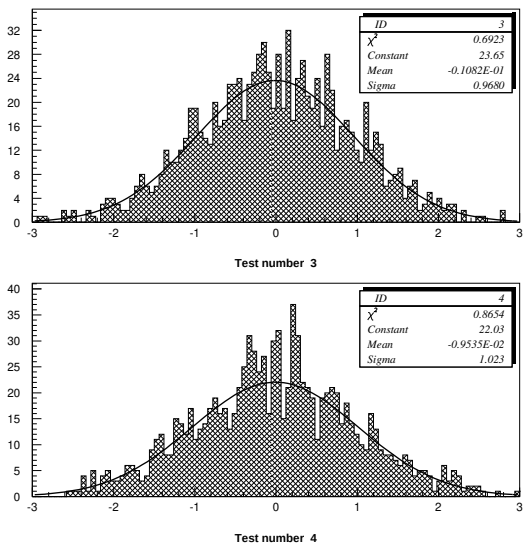


Figure 50. Visualising a HIGZ picture produced in a batch HPLOT program

8.5 Attributes setting

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

The IGSET command has two optional parameters:

CHNAME Attribute name (four characters).
VALUE Attribute value.

Examples

```
PAW > IGSET MTYP 20        | Change the marker type to 20
```

This new marker type will be used by all subsequently commands using the current marker type.

```
PAW > IGSET                | Display actual and default values of all HIGZ attributes
PAW > IGSET *              | Set ALL HIGZ attributes to their default values
PAW > IGSET LWID           | Set the GIVEN HIGZ attribute to its default value
```

Table 10. Parameters and default values in IGSET

Option	Default	Explanation
AURZ	0	Automatic saving flag
AWLN	0.0	Axis wire length. Default is length=0 (no grid)
BARO	0.25	Offset of bars chart representation (fraction between 0.0 and 1.0)
BARW	0.50	Width of bars chart representation (fraction between 0.0 and 1.0)
BASL	0.01	Basic segment length in NDC space (0-1) by (0-1) for dashed lines
BORD	0	Border flag. If equal 1, the border is drawn in Boxes, Pie charts, etc.
CHHE	0.01	Character height
CSHI	0.02	Shift between 2 passes of command TEXT
FACI	1	Fill area colour index
FAIS	0	Fill area interior style
FASI	1	Fill area style index
LAOF	0.013	Label axis offset
LASI	0.018	Label axis size
LTYP	1	Line type
LWID	1.00	Line width
MSCF	1.00	Marker scale factor
MTYP	1	Marker type
NCOL	8	Number of the colour in the colour table
PASS	1	Number of passes in command TEXT
PICT	1	Starting picture number for automatic naming
PLCI	1	Polyline colour index
PMCI	1	Polymarker colour index
TANG	0.00	Text angle
TMSI	0.019	Tick marks size
TXAL	0 0	Text alignment (horizontal and vertical)
TXCI	1	Text colour index
TXFP	1 0	Text font and precision (0: hard, 1: string, 2: soft)

OPTION

The OPTION command has one optional parameter:

CHOPT Option name (four characters).

```
PAW > OPTION              | Display actual and default values of all HPLOT options
PAW > OPTION *            | Set all HPLOT options to their default values
```

Table 11. Parameters and default values in OPTION

Option	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

SET

The SET command has two optional parameters:

CHATT Attribute name (four characters).
VALUE Attribute value.

```
PAW > SET                 | Display actual and default values of all HPLOT attributes
PAW > SET *               | Set ALL HPLOT attributes to their default values
PAW > SET STAT            | Set the GIVEN HPLOT attribute to its default value
```

Table 12. Parameters and default values in SET

Option	Default	Explanation
ASIZ	0.28	Axis label SIZe in cm
BARO	0.25	BAR histogram Offset (%) for bar charts representation
BARW	0.50	BAR histogram Width (%) for bar charts representation
BCOL	1.00	fill area COLour of the histogram rounding Box
BTYP	0	fill area TYPe of the histogram rounding Box
BWID	1.	line WIDth of the histogram rounding Box
CFON	02	Comment Font and Precision (10*font+precision)
CSHI	0.03	Character SHIft between 2 passes for the TEXT command
CSIZ	0.28	Comment and statistic SIZe (cf. OPTION STAT)
DASH	0.15	length of basic DASHed segment for line type 12-15
DATE	2	DATE position on each new plot (cf. OPTION DATE)
DMOD	1	Line type (Dash MODe)
ERRX	0.5	Error on X axis in % of bin width
FCOL	1	Function fill area COLour
FILE	1	FILE name position on each new plot (cf. OPTION FILE)
FIT	101	FIT values to be plotted (cf. OPTION FIT)
FPGN	1	First PaGe Number (cf. OPTION P)
FTYP	0	Function fill area TYPe
FWID	1.	Function line WIDth
GFON	02	Global title Font and Precision (10*font+precision)
GRID	3	GRID line type (cf. OPTION GRID)
GSIZ	0.28	Global title SIZe
HCOL	1	Histogram fill area COLour
HMAX	0.90	Histogram MAXimum for scale
HTYP	0	Histogram fill area TYPe
HWID	1	Histogram line WIDth
KSIZ	0.28	character Key SIZe in cm
LFON	02	axis Labels Font and Precision (10*font+precision)
NDVX	510.00	Number of DIVisions for X axis
NDVY	510.00	Number of DIVisions for Y axis
PASS	1	number of PASSes for software (TEXT) characters
PCOL	1	background fill area COLour of the Picture
PSIZ	0.28	Page number SIZe in cm
PTYP	0	background fill area TYPe of the Picture
PWID	1.	line WIDth of the box around the Picture
SSIZ	0.28	aSterisk SIZe (for functions)
STAT	1111	STAT values to be plotted (cf. OPTION STAT)
TFON	02	Text (and Title) Font and Precision (10*font+precision)
TSIZ	0.28	histogram Title SIZe in cm
VFON	02	axis Values Font and Precision (10*font+precision)
VSIZ	0.28	axis Values SIZe in cm
XCOL	1	X axis COLour
XLAB	1.40	distance y axis to LABel in cm
XMGL	2.00	size of the X MarGin Left in cm
XMGR	2.00	size of the X MarGin Right in cm
XSIZ	20.00	length of X axis in cm
XTIC	0.30	X axis TICk marks length in cm
XVAL	0.40	distance y axis to axis VALues in cm
XWIN	2.00	X space between WINdows (cf. command ZONE) in cm
YCOL	1	Y axis COLour
YGTI	1.50	Y position of Global TItle in cm
YHTI	1.20	Y position of Histogram TItle in cm
YLAB	0.80	distance x axis to LABel in cm
YMGL	2.00	size of the Y MarGin Low in cm
YMGU	2.00	size of the Y MarGin Up in cm
YNPG	0.60	Y position for PaGe Number in cm
YSIZ	20.00	length of Y axis in cm
YTIC	0.30	Y axis TICk marks length in cm
YVAL	0.20	distance x axis to axis VALues in cm
YWIN	2.00	Y space between WINdows (cf. command ZONE) in cm
2SIZ	0.28	scatter-plot and table character SIZe (cf. command H/PL id CHAR)

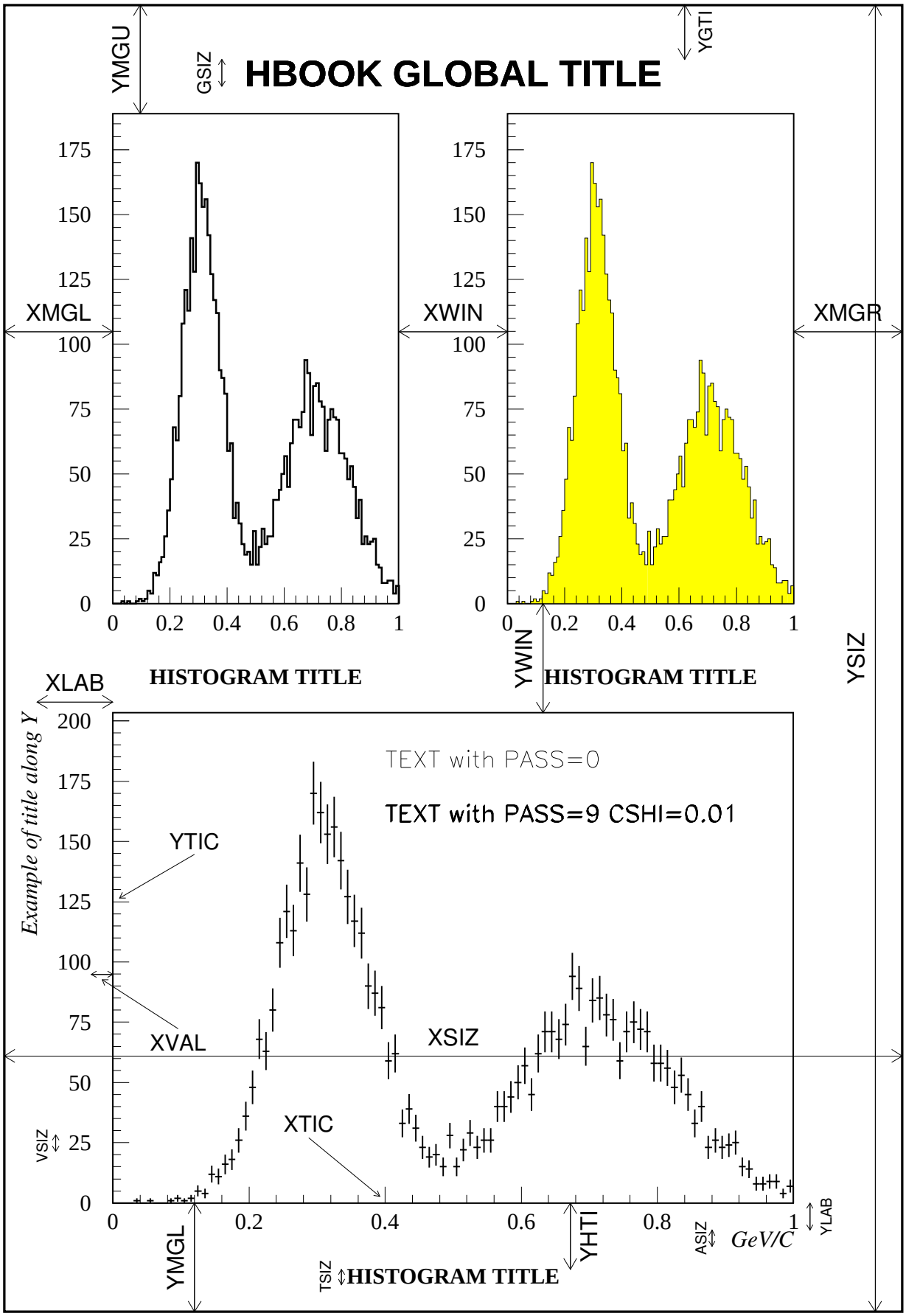


Figure 51. A graphical view of the SET parameters

8.6 More on labels

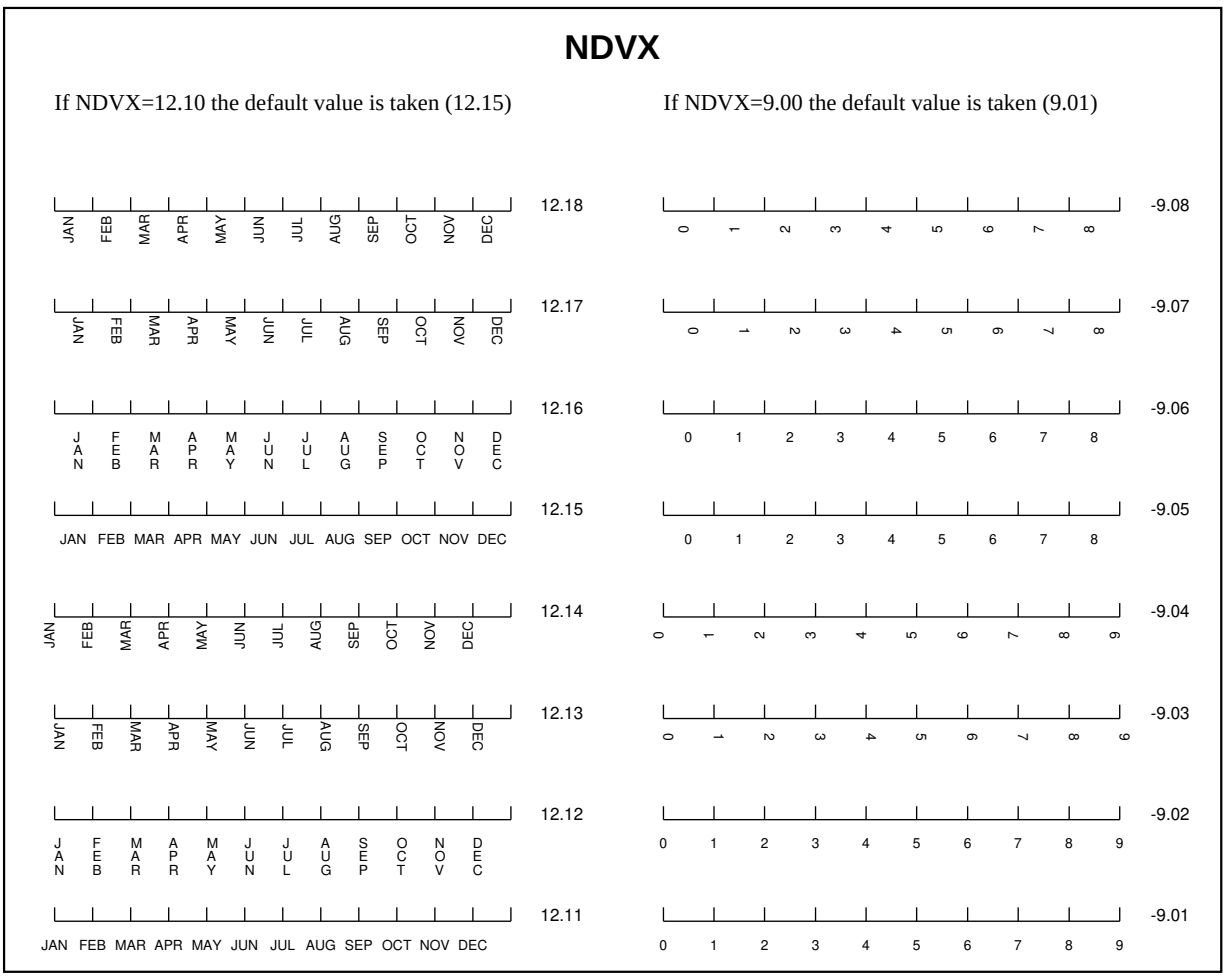


Figure 52. Example of labelling for horizontal axes

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 (for example `HISTO/PLOT`):

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 string 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:

```
    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 Figure 52 and Figure 53 for details. These two figures show that the labels could 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 will be used i.e.:

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

For example the commands:

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

set the labelling for each subsequent X axis to the months of the year centered in the middle of each bin.

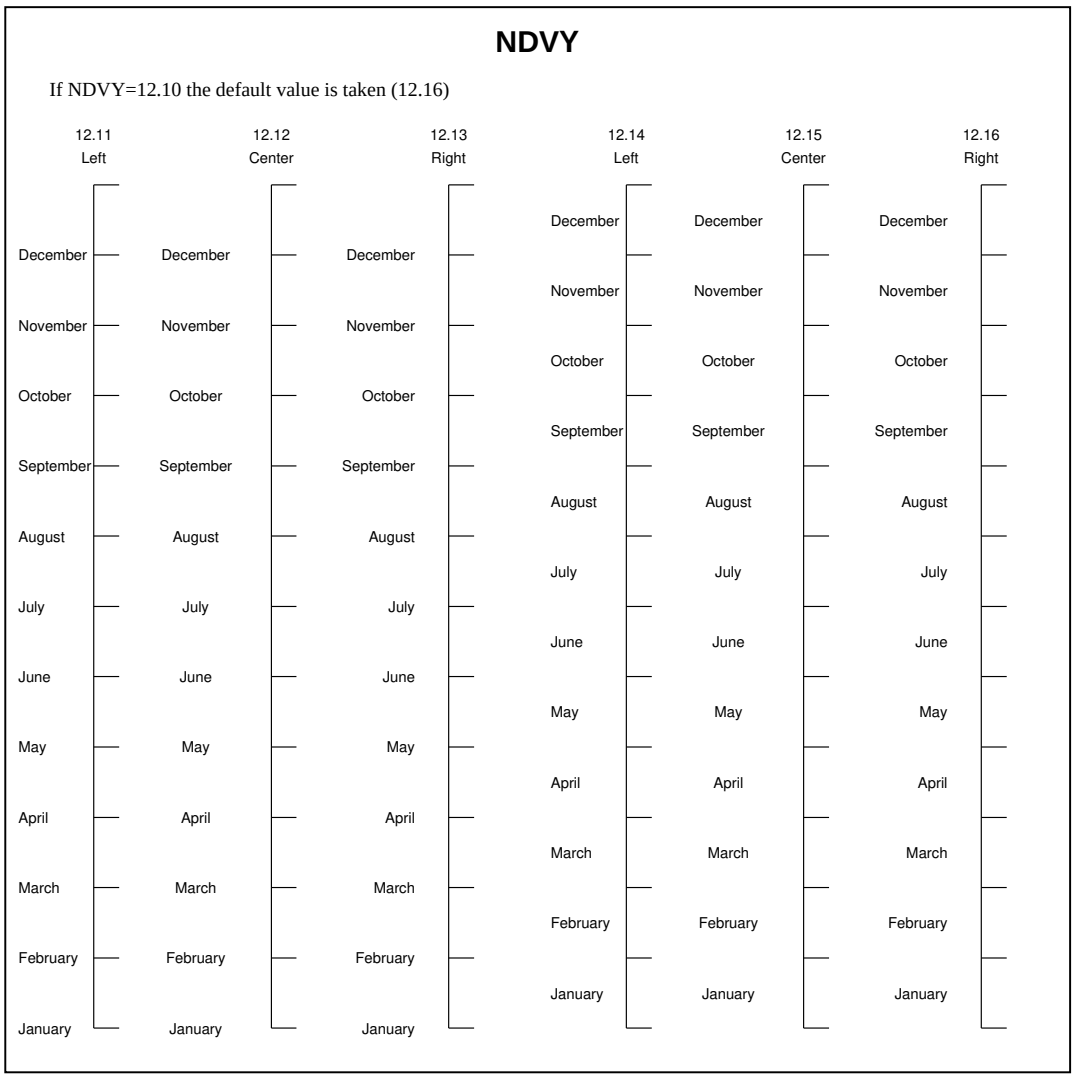


Figure 53. Example of labelling for vertical axes

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:

H Histogram
B zone Box
P Page
F Function

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: Figure 55 on page 121) should be used. The Figure 54 shows how to use the xTYP parameter. If HTYP equal 3001 a rainbow colour table (Figure C1 in colour section) is used on the Versacolor plotter at CERN for the 2D histogram plotted with option COL (see the top right picture on the cover).

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. For more details see Figure C2 in the colour section.

Figure C1 in the colour section shows on the first line the eight standard colours available on all kind of colour devices. The colours 9-256 shown on the same table are available only on Versacolor plotters and could be obtained via:

```
PAW > SET xTYP 2000+n          ! with 8 < n < 257
```

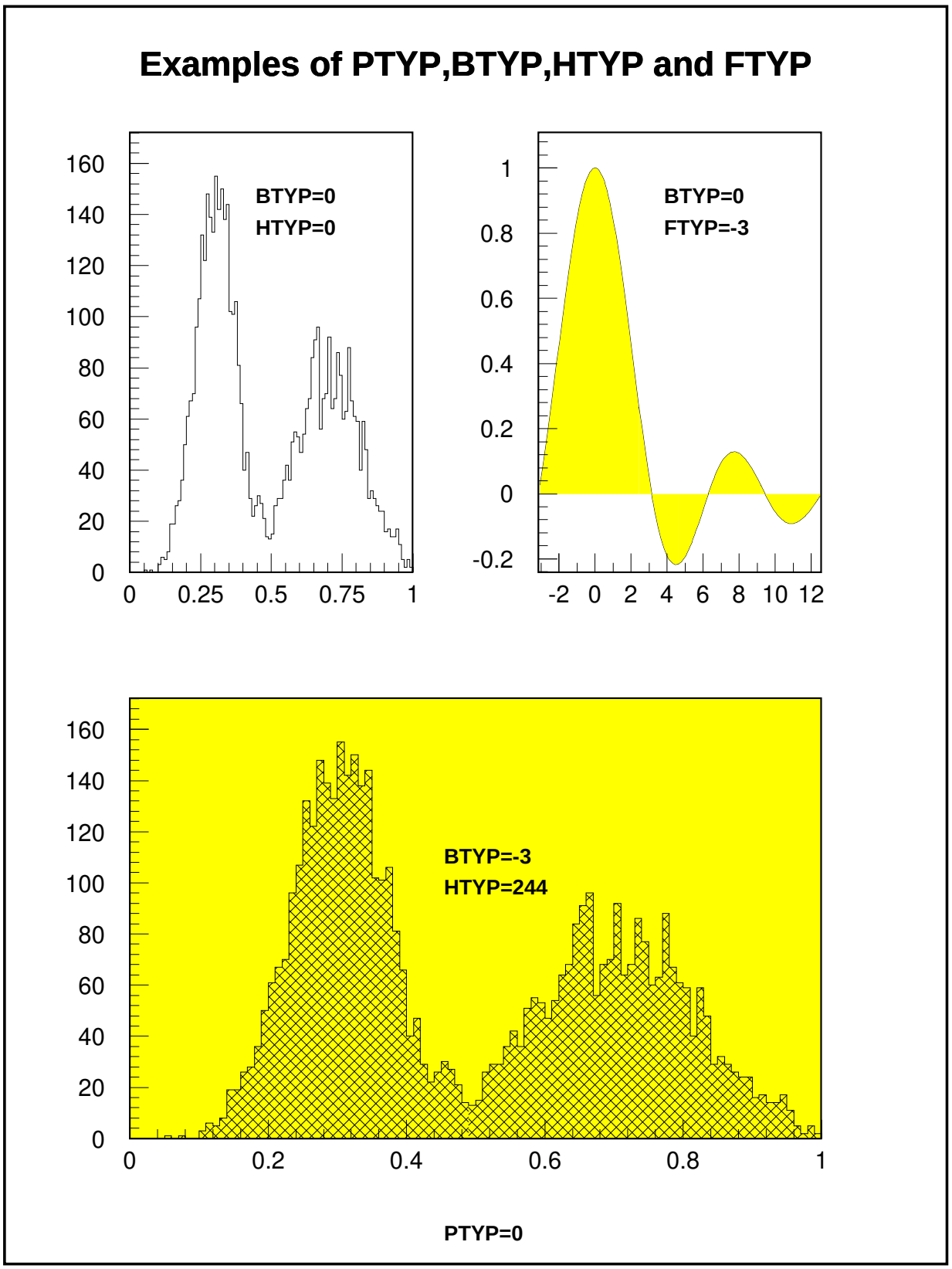


Figure 54. Example of fill area types in HPLOT

8.8 Information on 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 all these OPTIONs 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=OURMEIA.

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=CEP.

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

For example:

```
PAW > SET FIT 100
```

sets the fit informations to be only the χ^2.

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

- AURZ: If the AURZ mode is on, i.e the command:

```
PAW > IGSET AURZ 1
```

has been typed, all subsequent created pictures are stored automatically in the last open picture file via the command PICTURE/FILE. The following example explain how to use this facility:

```
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 contain 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
```

- TXFP: Sets text font and precision attributes for use by later invocations of ITX. The syntax is:

```
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, the font number 0, with precision 2, is always available, independently from the basic graphics package used (see Figure 59 on page 125). Hardware characters are available by specifying IGSET TXFP 10, i.e. font 1, precision 0

- TXAL: Sets the text alignment attributes for use by future invocations of ITX. The syntax is:

```
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. ITXALH must be in the range 0-3 and 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 13 on page 120).

Table 13. Text alignment parameters

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 ("g","y", etc.) plus any built in spacing.

• FAIS, FASI, MTYP and LTYPE: Set respectively the Fill Area Interior Style, The Fill Area Style Index, the Marker TYPe and the Line TYPe. 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
```

The Figure 56 gives the available GKSGRAL fill area styles index.

In addition, HIGZ provides some portable fill area styles index coded using three digits ijk:

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 14 and examples are shown in Figure 55 on page 121.

Table 14. Codification for the HIGZ portable fill area interior styles

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

For example:

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

set the fill area to be hatched with two sets of lines (90 and 0 degrees) 1mm spaced.

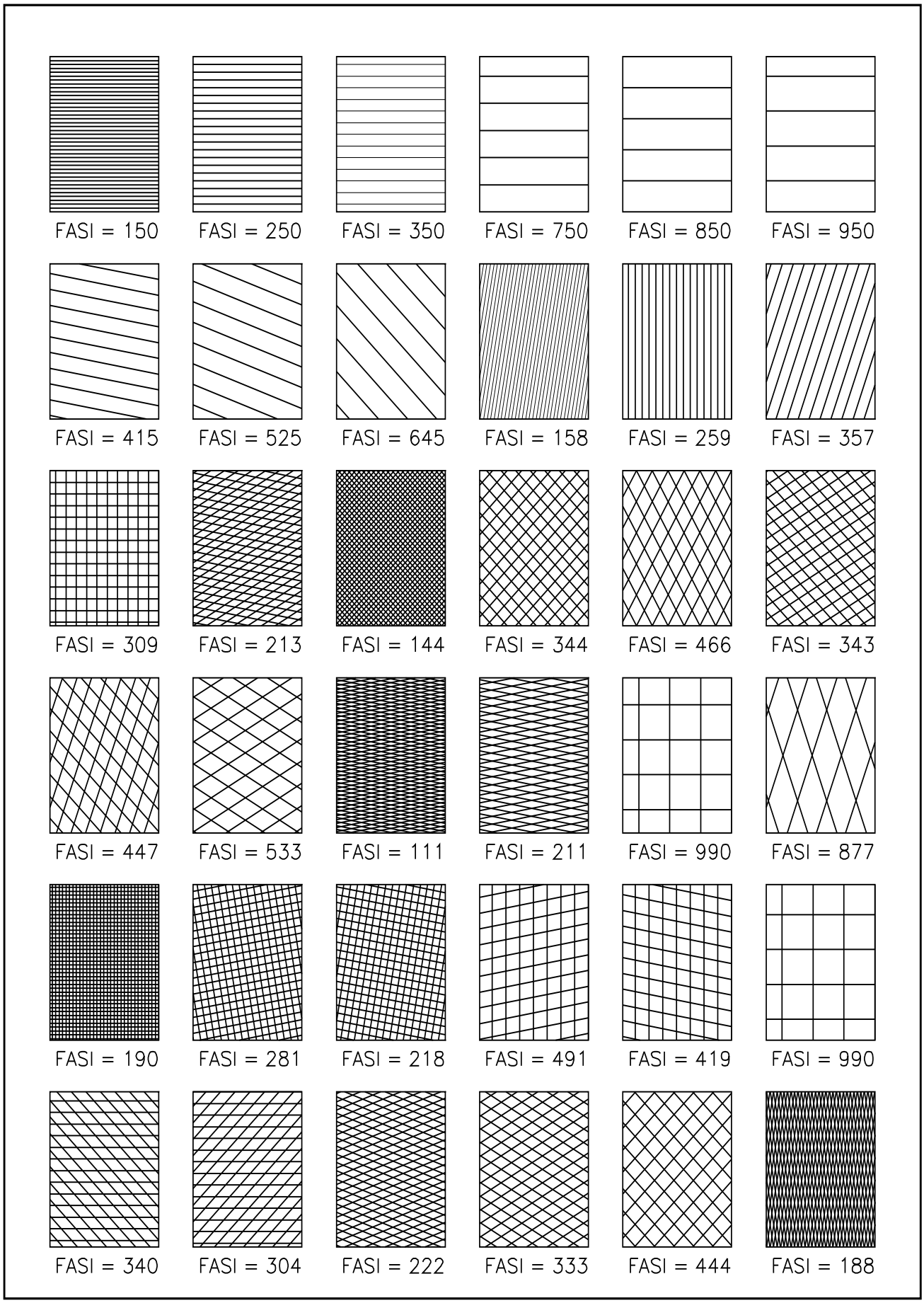


Figure 55. Examples of HIGZ portable hatch styles

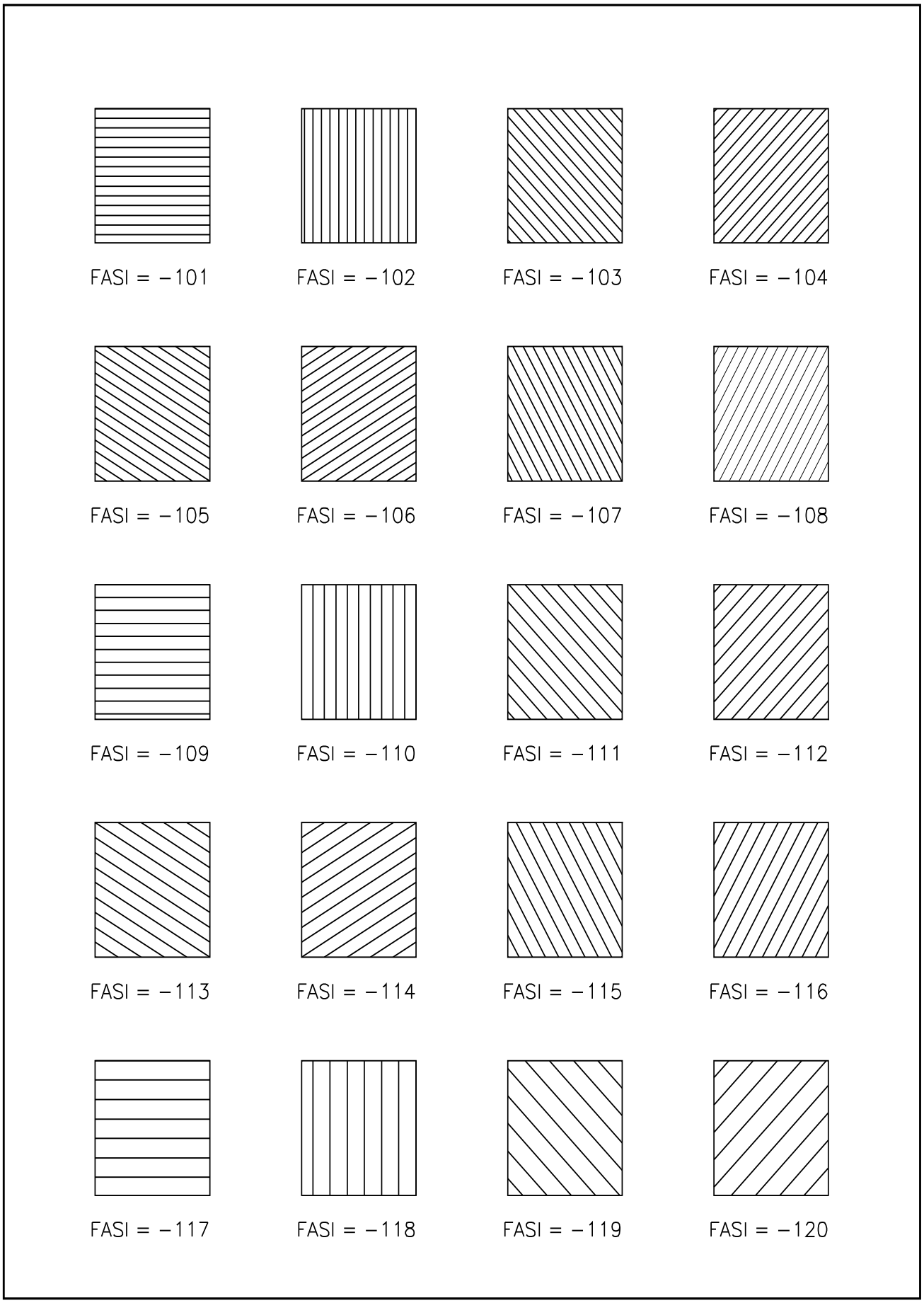


Figure 56. GKSGRAL Device independent hatch styles

Marker Index	Marker Type
31	
30	
29	
28	
27	
26	
25	
24	
23	
22	
21	
20	

Figure 57. HIGZ portable marker types

Line Index	Line Type
15	
14	
13	
12	

Figure 58. HIGZ portable line types

8.10 Text fonts

Text in PAW could 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 15. The repertoire of possible symbols is shown in Figure 59.

Table 15. List of HPLOT escape characters and their meaning

<	go to lower case*	**>**	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
$	termination character (optional)		

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 command:

```
PAW > IGSET TXFP fffp
```

where fff is the ***font identifier*** as defined in the figures on the next 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 62 and 63 using metafile types -111 and -112 are available. Figures 64, 65 and 66 give lists of PostScript characters accessible via an octal code preceded by a backslash, e.g. to print a "copyright" character © use:

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

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.

```
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 64 on page 129, e.g.:

```
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 l@'\372uvre on conna\333t l@'artisan\273'
PAW > ITX 1. 2  '\(proverbe fran\321ais\).' | quote escaped by @ character
PAW > ITX 1. 1. '\252\241Ma\337ana!\272, dit l@'\323l\325ve.'
```

would give

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

* Note that characters can also be entered directly in lower case.

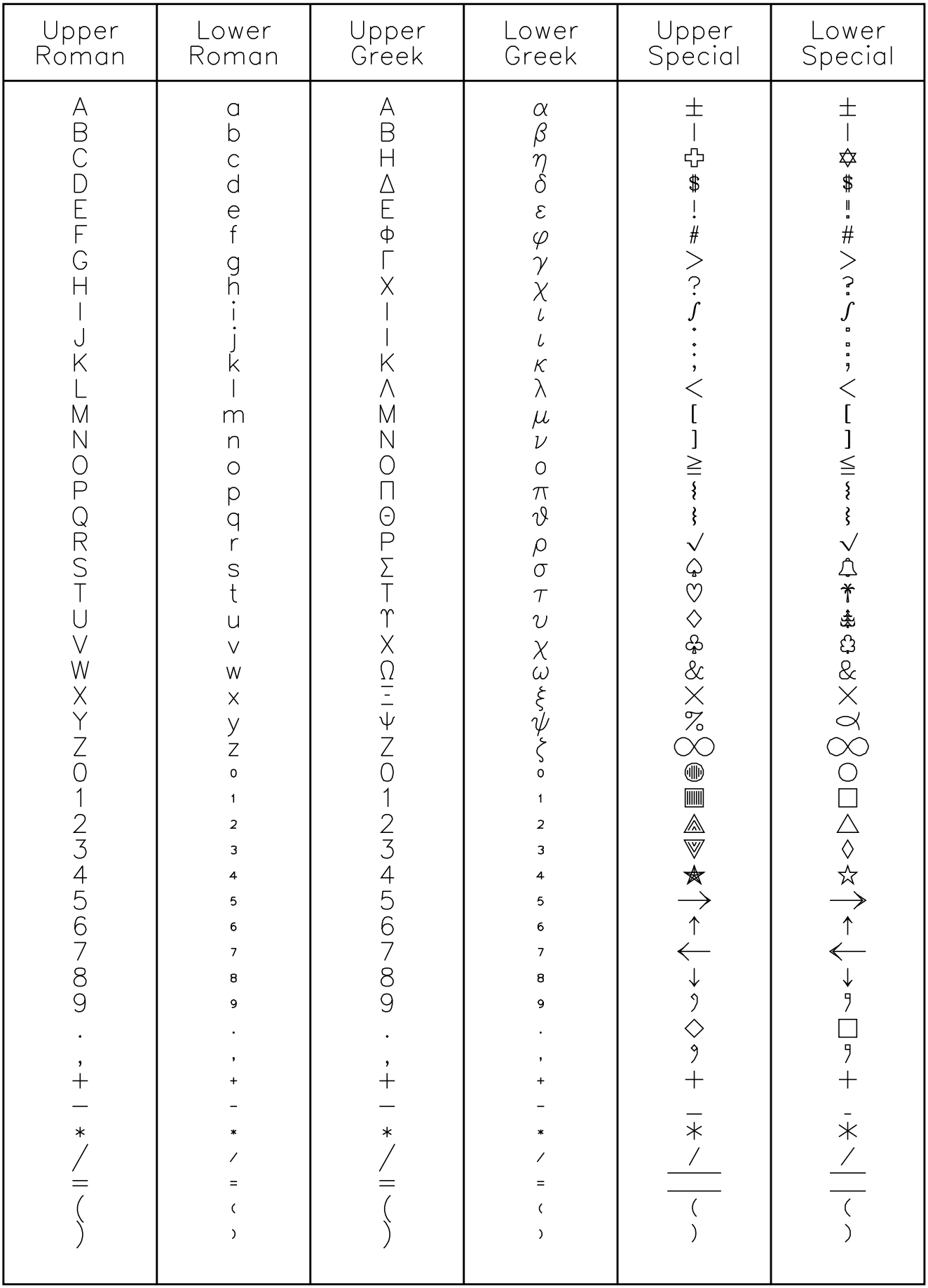


Figure 59. HIGZ portable software characters (Font 0, Precision 2)

GKS–GRAL Software Fonts							
Font/Prec	Style	Font/Prec	Style	Font/Prec	Style	Font/Prec	Style
–1/2	ABCabc	–101/2	*ABCabc*	–201/2	ABCabc	–301/2	*ABCabc*
–2/2	**ABCabc**	–102/2	***ABCabc***	–202/2	**ABCabc**	–302/2	***ABCabc***
–3/2	ABCabc	–103/2	*ABCabc*	–203/2	ABCabc	–303/2	*ABCabc*
–4/2	**ABCabc**	–104/2	***ABCabc***	–204/2	**ABCabc**	–304/2	***ABCabc***
–5/2	*ABCabc*	–105/2	*ABCabc*	–205/2	*ABCabc*	–305/2	*ABCabc*
–6/2	***ABCabc***	–106/2	***ABCabc***	–206/2	***ABCabc***	–306/2	***ABCabc***
–7/2	*ABCabc*	–107/2	*ABCabc*	–207/2	*ABCabc*	–307/2	*ABCabc*
–8/2	***ABCabc***	–108/2	***ABCabc***	–208/2	***ABCabc***	–308/2	***ABCabc***
–9/2	**ABCabc**	–109/2	***ABCabc***	–209/2	**ABCabc**	–309/2	***ABCabc***
–10/2	**ABCabc**	–110/2	***ABCabc***	–210/2	**ABCabc**	–310/2	***ABCabc***
–11/2	**ABCabc**	–111/2	***ABCabc***	–211/2	**ABCabc**	–311/2	***ABCabc***
–12/2	ABCabc	–112/2	ABCabc	–212/2	ABCabc	–312/2	ABCabc
–13/2	ABXαβχ	–113/2	*ABXαβχ*	–213/2	ABXαβχ	–313/2	*ABXαβχ*

Figure 60. GKSGRAL software fonts

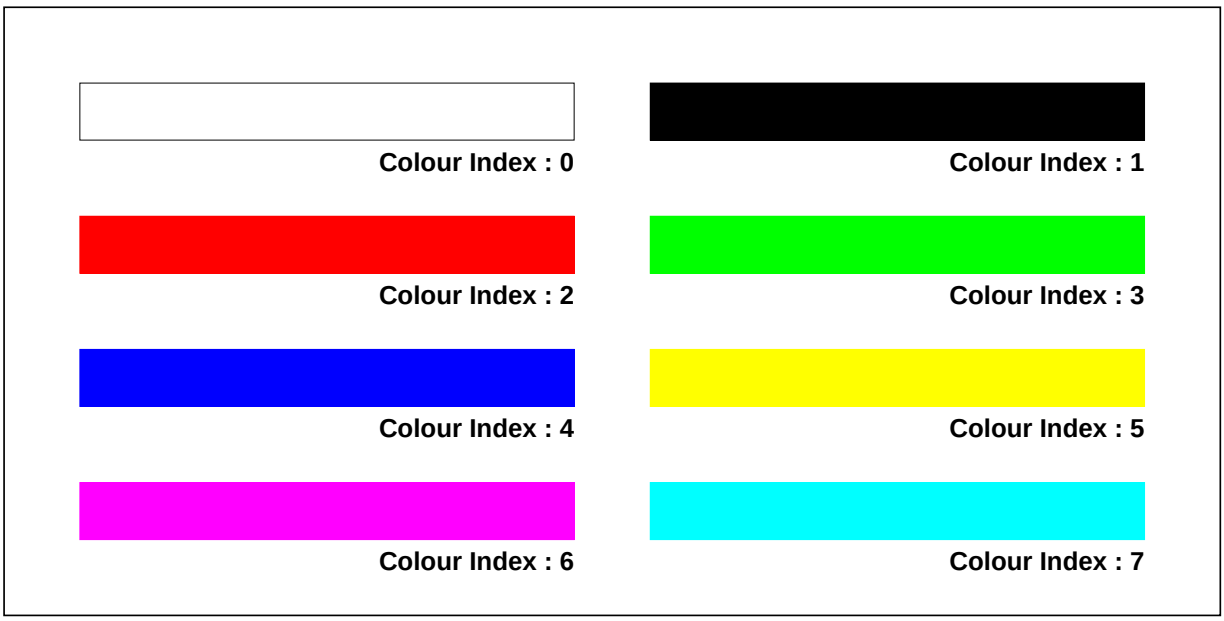


Figure 61. PostScript grey level simulation of the basic colours

Font/Prec	Font Style (HIGZ/PostScript Interface)	
−1/0	*ABCDEFGabcdef*	*Times-Italic*
−2/0	**ABCDEFGabcdef**	*Times-Bold*
−3/0	***ABCDEFGabcdef***	*Times-BoldItalic*
−4/0	ABCDEFGabcdef	*Helvetica*
−5/0	*ABCDEFGabcdef*	*Helvetica-Oblique*
−6/0	**ABCDEFGabcdef**	*Helvetica-Bold*
−7/0	***ABCDEFGabcdef***	*Helvetica-BoldOblique*
−8/0	ABCDEFGabcdef	*Courier*
−9/0	*ABCDEFGabcdef*	*Courier-Oblique*
−10/0	**ABCDEFGabcdef**	*Courier-Bold*
−11/0	***ABCDEFGabcdef***	*Courier-BoldOblique*
−12/0	ΑΒΧΔΕΦΓαβχδεφ	*Symbol*
−13/0	ABCDEFGabcdef	*Times-Roman*
−14/0	✡✢✂✇✣✥✧❀✲❀❄❄❄	*ZapfDingbats*
−15/0	*ABCDEFGabcdef*	*Times-Italic*
−16/0	ABCDEFGabcdef	*Times-Bold*
−17/0	*ABCDEFGabcdef*	*Times-BoldItalic*
−18/0	ABCDEFGabcdef	*Helvetica*
−19/0	*ABCDEFGabcdef*	*Helvetica-Oblique*
−20/0	ABCDEFGabcdef	*Helvetica-Bold*
−21/0	*ABCDEFGabcdef*	*Helvetica-BoldOblique*
−22/0	ΑΒΧΔΕΦΓαβχδεφ	*Symbol*
−23/0	ABCDEFGabcdef	*Times-Roman*
−24/0	✡✢✣✥✥✧✧❀✪❀❄❄❄	*ZapfDingbats*
0/2	ABCDEFGabcdef	*HIGZ-Software*

Figure 62. PostScript fonts

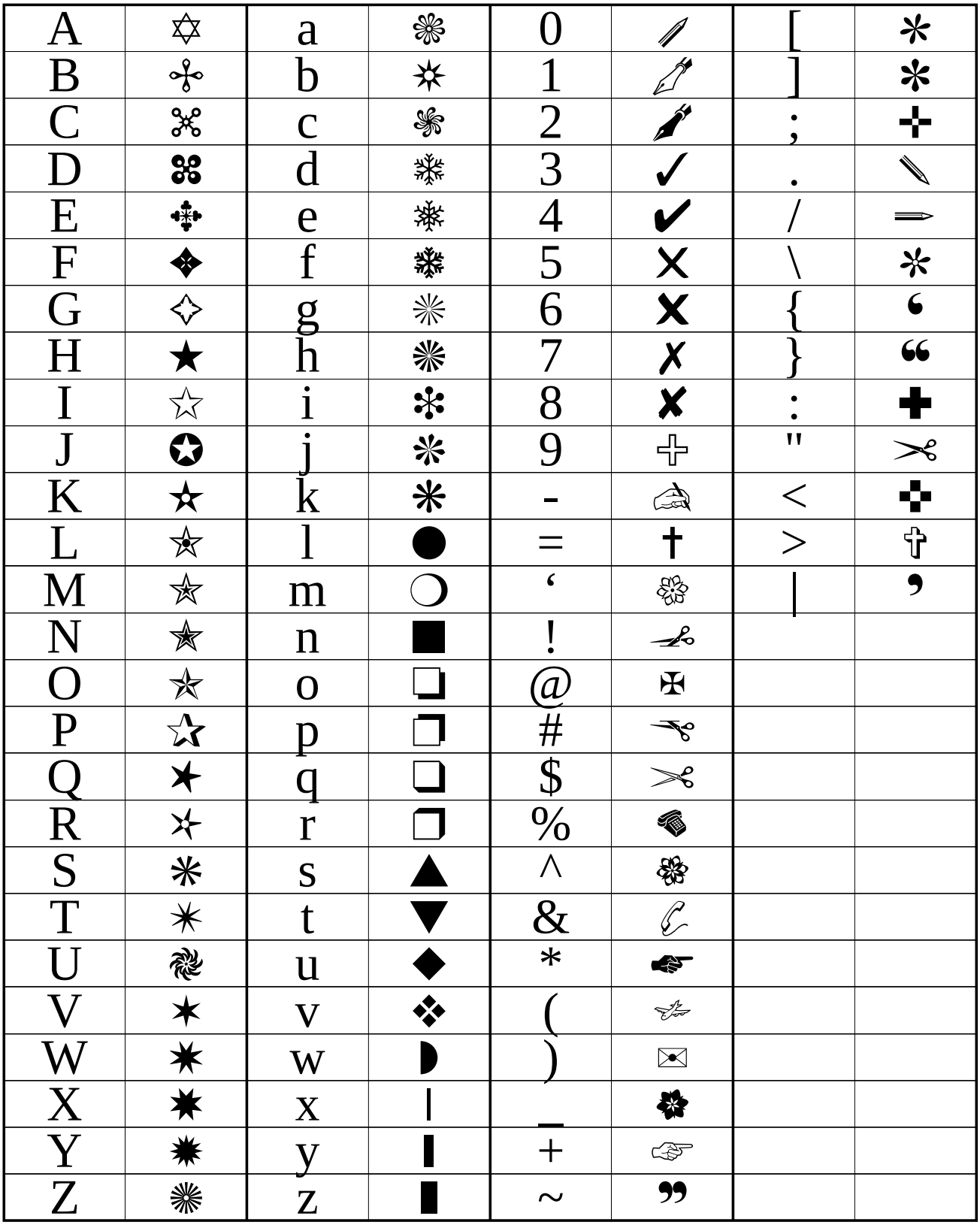


Figure 63. Correspondence between ASCII and ZapfDingbats font (-14)

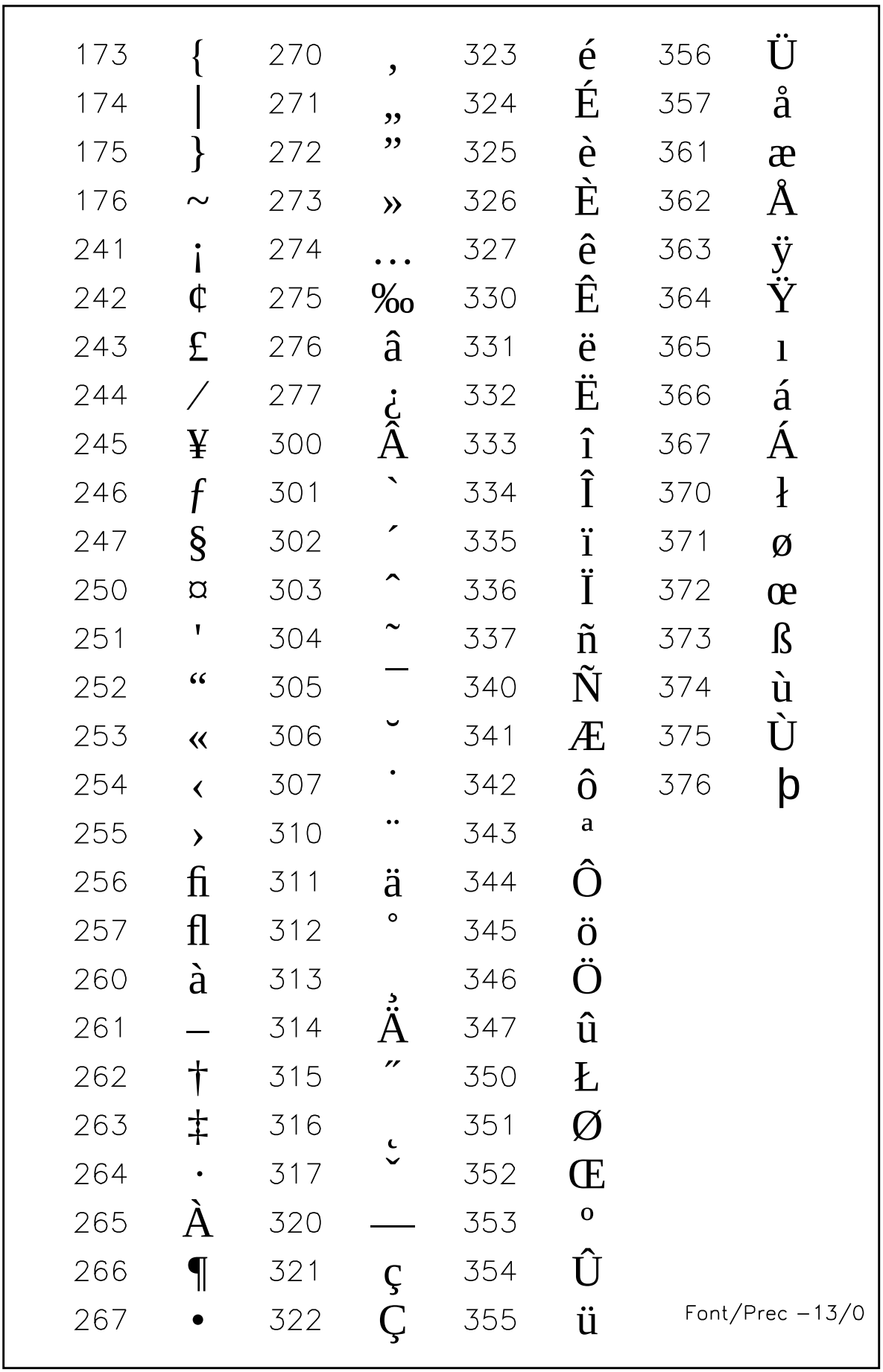


Figure 64. Octal PostScript characters in Times font

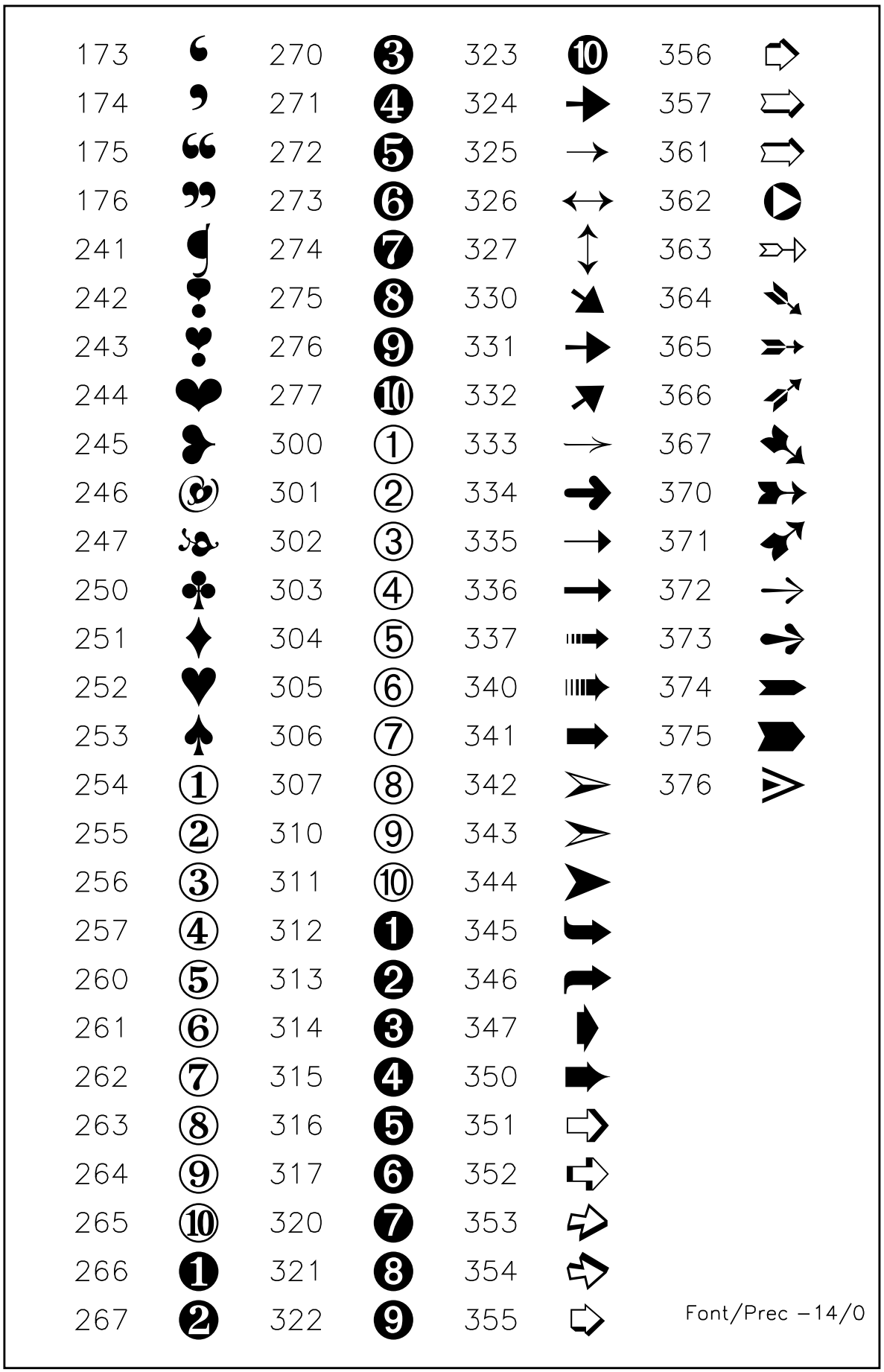


Figure 65. Octal PostScript characters in ZapfDingbats font (-14)

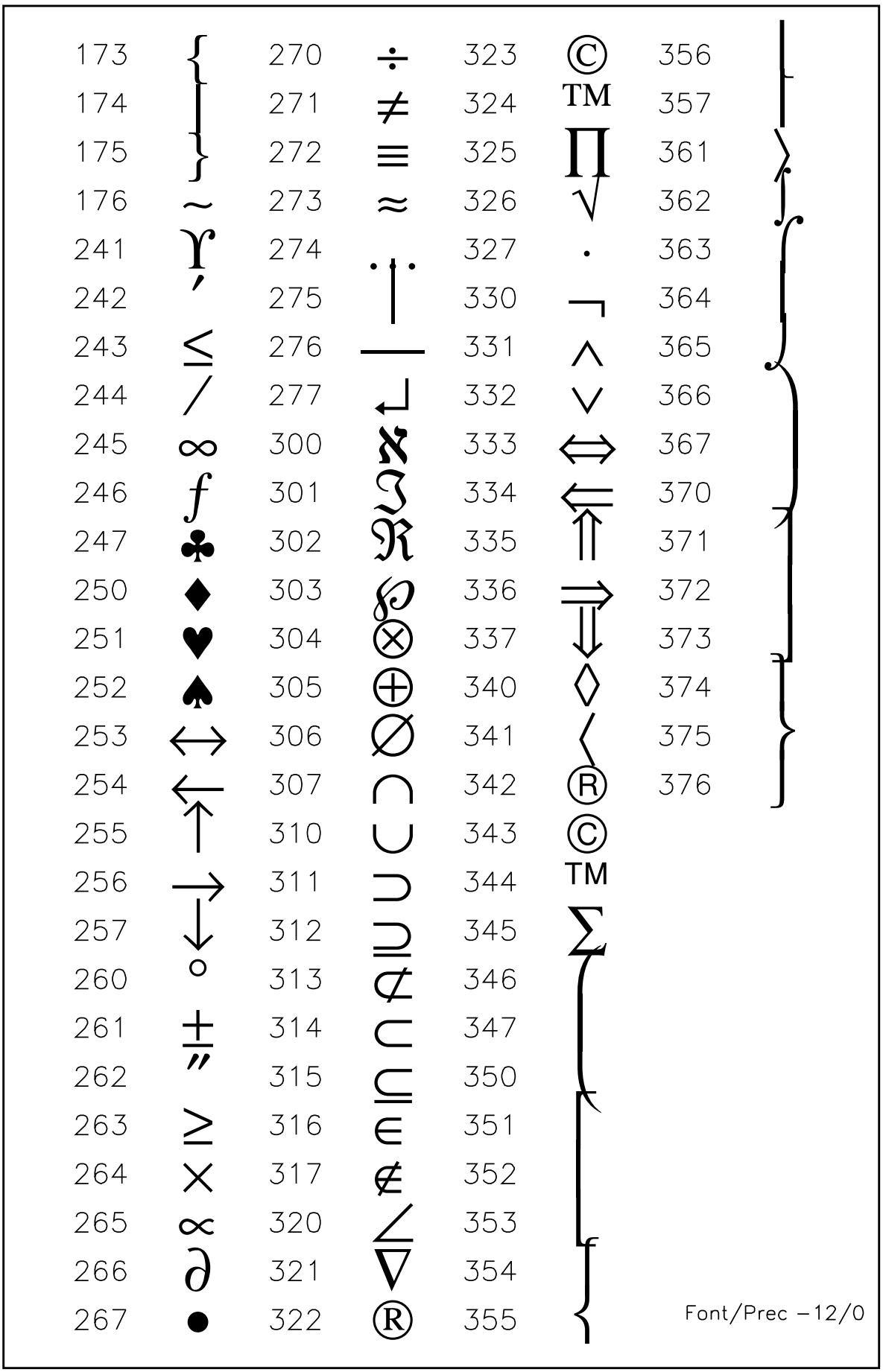


Figure 66. Octal PostScript characters in Symbol font

8.11 The HIGZ graphics editor

The HIGZ pictures in memory could be modified interactively with the HIGZ graphics editor. The command:

```
PAW > PICT/MODIFY PNAME
```

invokes the graphics editor. PNAME could be the complete name, the picture number in memory or ' '.

The following figure (produced by the graphics editor itself) gives more details on the graphics editor's functionality.

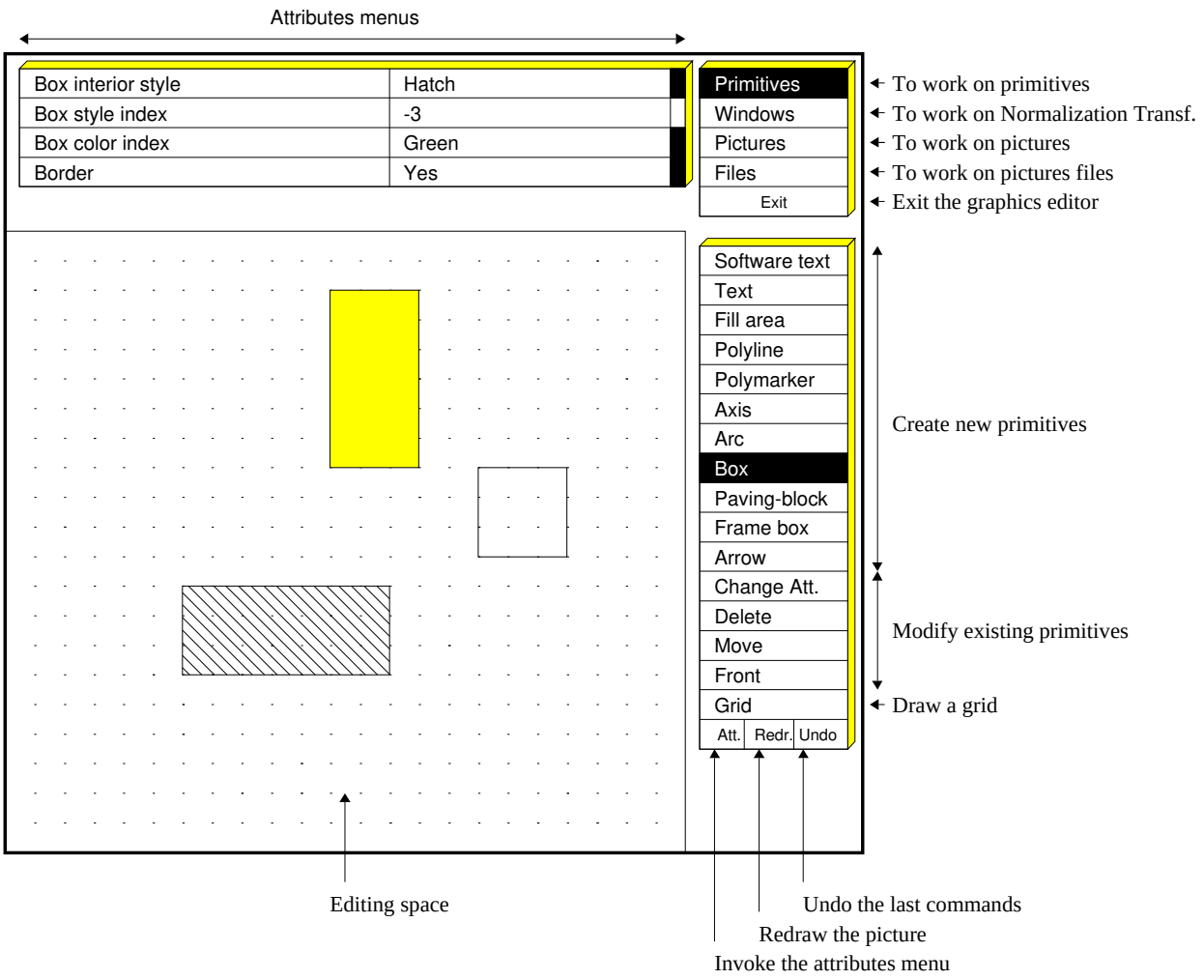


Figure 67. The HIGZ graphics editor

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 heteregoneous 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 [12] flag CZ). PAW and the PAW server must be linked with two special modules called **CZ** and **TCPAW** [8, 21].

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[21] 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.

In the relatively near future it is intended to extend its functionality to support DECNET as well as TCP/IP connections.

9.1 TELNETG and 3270G

Figure 69 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).

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):

```
$ 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
```

Figure 68. Example of a TELNETG session

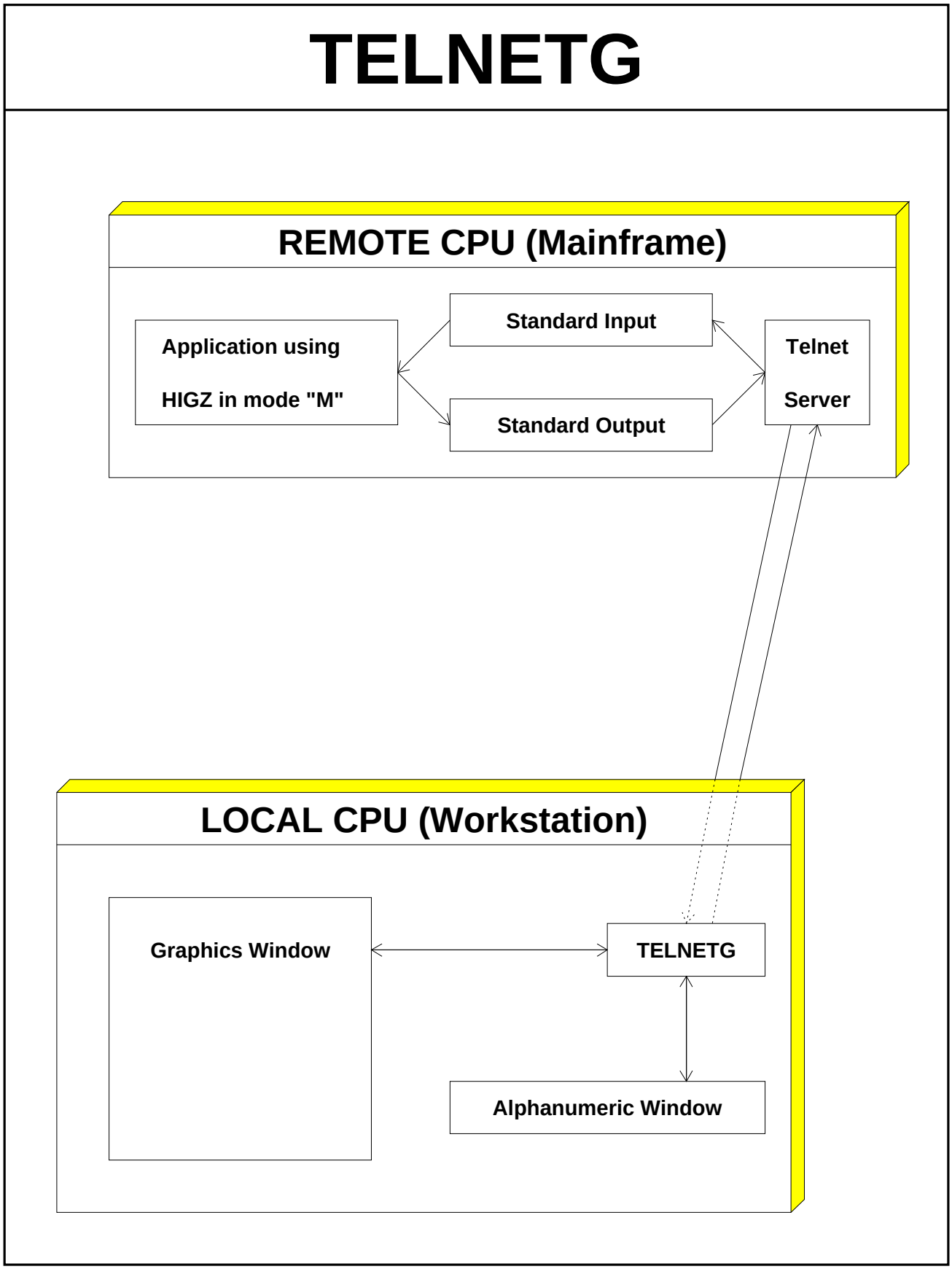


Figure 69. The TELNETG program

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.

```
# 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 D"
                                        |  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
```

Figure 70. Example of a ZFTP session

9.3 Access to remote files from a PAW session

When running PAW on a workstation, 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. But a more convenient mechanism for direct access is the possibility to access the file directly. In order to do that, PAW uses a conventional Client/Server model. The client (PAW) runs on the workstation. When the PAW command RLOGIN is invoked, a PAW server is automatically started on the remote machine. This server is watching for client messages.

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 following 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

```
PAW > rlogin CERNVM                          | connect to CERNVM
PAW > rshell file HRZTEST.DAT                | PAW server connects HRZTEST DAT A
                                             | to //LUN11
PAW > histo/plot 10                          | plot histogram 10 from CERNVM
PAW > histo/fit 20 G                         | fit histo 20 with a gaussian
                                             | and plot it
PAW > rlogin VXCRNA                          | connect to VXCRNA
PAW > rshell file DISK$DL:[PAW]HEXAM.DAT;3   | PAW server on VXCRNA connects file
                                             | to //LUN11
PAW > histo/plot 110                         | plot histogram 110 from VXCRNA
PAW > rshell file HRZTEST.DAT                | PAW server on VXCRNA connects file
                                             | to //LUN12
PAW > histo/plot 110 s                       | plot histogram 110 from HRZTEST.DAT
                                             | on VXCRNA on the existing picture
PAW > rshell ld //                           | list all files connected on VXCRNA
PAW > cdir //CERNVM                          | Change current PAW directory to CERNVM
PAW > histo/plot 110                         | plot histogram 110 from CERNVM
PAW > histo/plot //VXCRNA/110                | plot histogram 110 from VXCRNA
PAW > cdir //PAWC                            | current directory to local memory
PAW > histo/list                             | list all histograms in //PAWC
PAW > Histo/delete 0                         | delete all histograms in memory
PAW > hrin //VXCRNA/0                        | read all histograms from VXCRNA
                                             | file HRZTEST.DAT to //PAWC
PAW > cdir //CERNVM                          | change directory to CERNVM
PAW > rshell file NEW.DAT.D 1024 N           | creates a new file on the D disk
PAW > hrout 0                                | write all histograms from //PAWC
                                             | to CERNVM file NEW DAT D
```

Figure 71. Access to remote files from a workstation

9.4 Using PAW as a presenter on VMS systems

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.

```
      PROGRAM PRODUCE
*
*        Test program for Global Sections.
*
      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
```

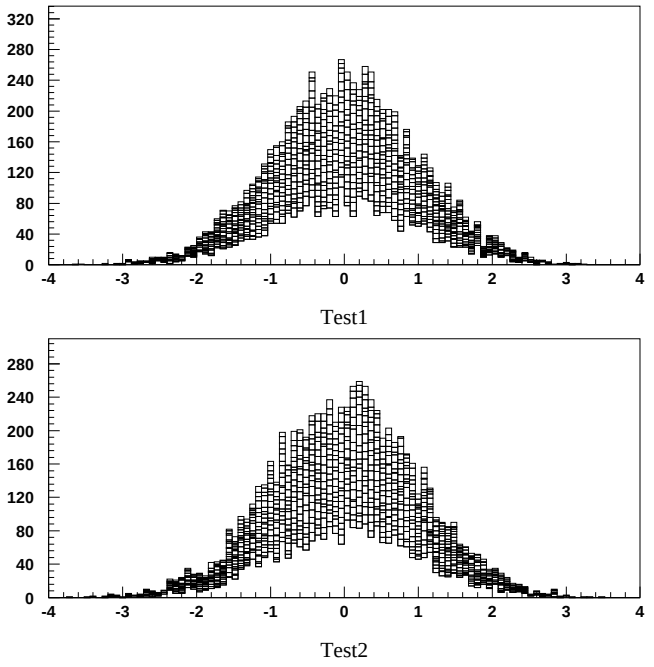


Figure 72. Visualising histograms in a global section

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
```

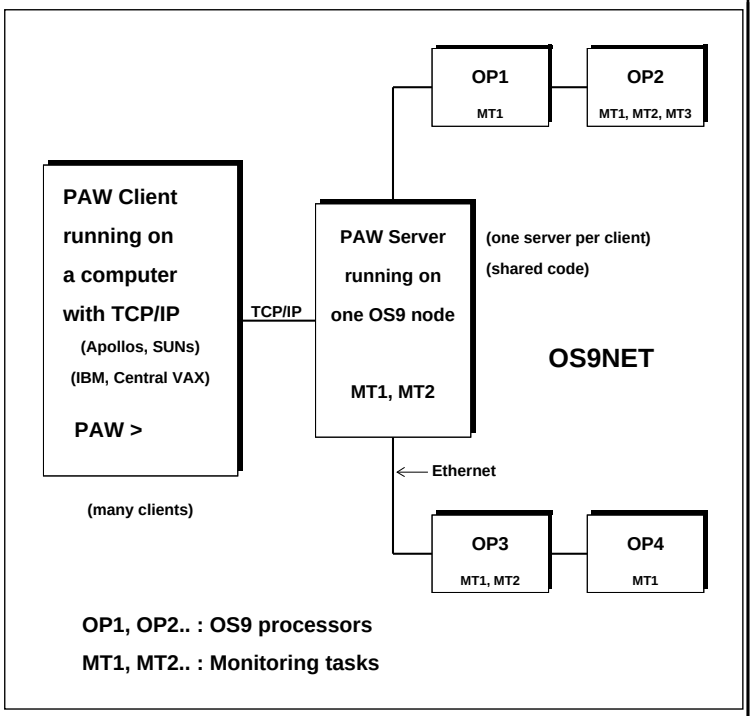
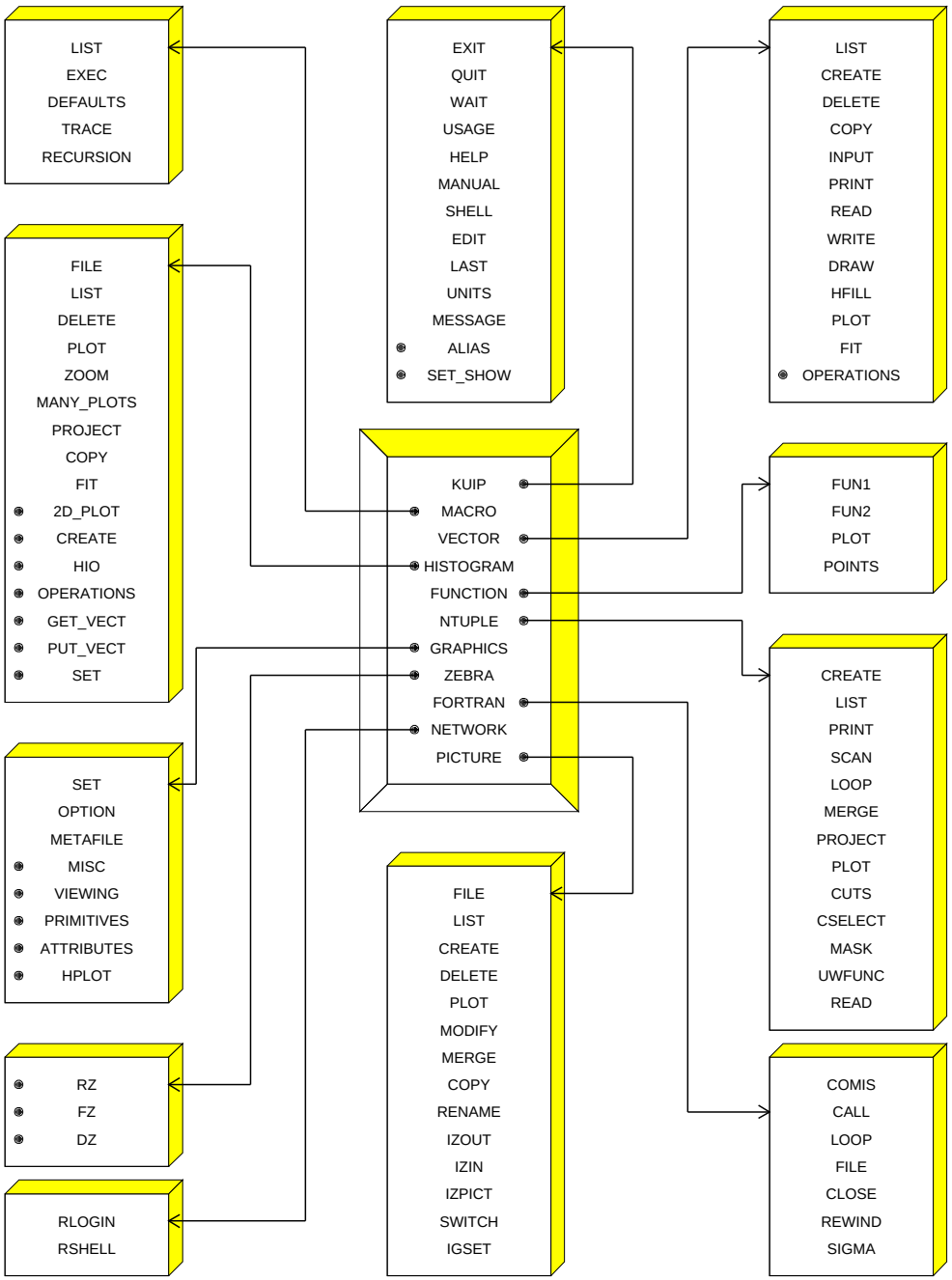


Figure 73. Visualising histograms in an OS9 module

```
PAW > rlogin O-OPAL01                  | connect to an OS9 machine
PAW > rshell module OP2/MT1            | PAW server connects to OP2/MT1
                                       | (Processor OP2, Monitoring Task MT1)
PAW > histo/plot 10                    | plot histogram 10
PAW > hrin 0                           | read all histograms into //PAWC
PAW > Histo/File 1 local.dat 1024 N    | create a new file local.dat
                                       | on the client machine
PAW > hrout 0                          | save all histograms from //PAWC
                                       | to the local file
PAW > rshell module OP3/MT2            | PAW server connects to another
                                       | OS9 monitoring task
PAW > Output 56 os9.listing            | Change output file on client
PAW > rshell ldir                      | list all histograms in MT2
                                       | on file os9.listing
PAW > Output -56                       | Change output file to default (unit 6)
                                       | file os9.listing is closed
```

Figure 74. Access to OS9 modules from PAW

Part 3. 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. FORTRAN

COMIS 143
CALL urout 144
LOOP ntimes urout 145
FILE lun fname 145
CLOSE lun 145
REWIND lun 145
SIGMA 145

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      C 'User routine'
```

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:

```
From HBOOK:
   HBOOK1,HBOOK2,HBOOKN,HFILL,HF1,HPRINT,HDELET,HRESET
   HFITGA,HFITPO,HFITEX,HPROJ1,HPROJ2,HFN
   HROPEN,PAOPEN,PACLOS,PAREAD,PAWRIT,HCDIR,HGIVEN
   HTITLE,HBFUN1,HBFUN2,HRNDM1,HRNDM2,HBARX,HBARY
   HPAK,HPAKE,HUNPAK,HGIVE,HGN,HGNF,HGNPAR,HF2,HFF1,HFF2
   HRIN,HROUT,HI,HIE,HIX,HIJ,HIF,HIDALL,HNOENT,HX,HXY
   HCOPY,HSTATI,HBPROF,HOPERA,HIDOPT,HDERIV
From HPLOT:
   HPLOT,HPLSYM,HPLERR,HPLEGO,HPLNT,HPLSUR,HPLSOF
   HPLSET,HPLGIV,HPLOC,HPLTOC
From ZEBRA:
   FZIN,FZOUT,FZFILE,FZENDI,FZENDO
   RZCDIR,RZLDIR,RZFILE,RZEND,RZIN,RZOUT,RZVIN,RZVOUT
From KUIP:
   KUGETV,KUDPAR,KUVECT,KILEXP,KUTIME,KUEXEL
From HIGZ:
   IPL,IPM,IFA,IGTEXT,IGBOX,IGAXIS,IGPIE,IGRAPH,IGHIST
   IGARC,IGLBL,IGRNG,IGMETA,IGSA,IGSET,IRQLC,IRQST,ISCR
   ISELNT,ISFAIS,ISFASI,ISLN,ISMK,ISVP,ISWN,ITX,ICLRWK
From KERNLIB:
   VZERO,UCOPY,RNDM,RANNOR,LENOCC,SBIT0,SBIT1,SBYT
   JBIT,JBYT,UCTOH,UHTOC

The following common blocks may be referenced:
  /PAWC/, /QUEST/, /KCWORK/, /PAWPAR/, /PAWIDN/
```

LOOP ntimes urout

```
NTIMES     I 'Number of calls' D=1
UROUT      C 'User routine'
```

The routine UROUT is called NTIMES times. See command CALL for explanation of UROUT.

FILE lun fname

```
LUN        I 'Logical unit number' R=1:128
FNAME      C 'File name'
```

Open a FORTRAN formatted text file.

CLOSE lun

```
LUN        I 'Logical unit number' 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 FZENDI(O), ICLWK, etc.

REWIND lun

```
LUN        I 'Logical unit number' 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 with parameter. 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 11. FUNCTION

FUN1 id ufunc ncx xmin xmax [chopt] 146
FUN2 id ufunc ncx xmin xmax ncy ymin ymax [chopt] 146
PLOT ufunc xlow xup [chopt] 147
POINTS npfunc 147

FUN1 id ufunc ncx xmin xmax [chopt]

```
ID        C 'Histogram Identifier'
UFUNC     C 'Name of the function' D='X'
NCX       I 'Number of channels' D=100
XMIN      R 'Low edge' D=0
XMAX      R 'Upper edge' D=100
CHOPT     C 'Options' 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        C 'Histogram (2-Dim) Identifier'
UFUNC     C 'Name of the function'
NCX       I 'Number of channels in X' D=40
XMIN      R 'Low edge in X' D=0
XMAX      R 'Upper edge in X' D=40
NCY       I 'Number of channels in Y' D=40
YMIN      R 'Low edge in Y' D=0
YMAX      R 'Upper edge in Y' D=40
CHOPT     C 'Options' 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.

PLOT ufunc xlow xup [chopt]

```
UFUNC      C 'Name of function'
XLOW       R 'Lower limit'
XUP        R 'Upper limit'
CHOPT      C 'Options' 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 npfunc

```
NPFUNC     I 'Number of points' D=100
```

Change the number of points to be used by FUN/PLOT.

Chapter 12. GRAPHICS

SET [chatt value] . . . 149
OPTION [choptn] . . . 149
METAFILE [lun metafl] . . . 149
12.1 MISC . . . 149
NEXT . . . 149
CLR . . . 149
LOCATE [ntpri] . . . 149
VLOCATE vecx vecy [chopt ntpri] . . . 149
HMOVE . . . 150
12.2 VIEWING . . . 150
ZONE [nx ny ifirst chopt] . . . 150
SIZE [xsize ysize] . . . 150
SVP nt x1 x2 y1 y2 . . . 150
SWN nt x1 x2 y1 y2 . . . 150
SELNT nt . . . 151
12.3 PRIMITIVES . . . 151
PLINE n x y . . . 151
LINE x1 y1 x2 y2 . . . 151
FAREA n x y . . . 151
PMARKER n x y . . . 151
BOX x1 x2 y1 y2 . . . 152
FBOX x1 x2 y1 y2 x3 x4 y3 y4 . . . 152
ARROW x1 x2 y1 y2 [size] . . . 152
ARC x1 y1 r1 [r2 phimin phimax] . . . 152
AXIS x0 x1 y0 y1 wmin wmax ndiv [chopt] . . . 153
PIE x0 y0 radius n values [chopt iao ias iac] . . . 154
TEXT x y text size [angle chopt] . . . 154
ITX x y text . . . 154
LABELS labnum nlabs chlabs . . . 154
HIST n x y [chopt] . . . 155
GRAPH n x y [chopt] . . . 156
PAVE x1 x2 y1 y2 [dz isbox isfram chopt] . . . 156
12.4 ATTRIBUTES . . . 157
SLN [iln] . . . 157
SFAIS [ints] . . . 157
SFASI [styli] . . . 157
SFACI [ifaci] . . . 157
SPLCI [iplci] . . . 157
SPMCI [ipmci] . . . 157
STXCI [itxci] . . . 157
STXFP [ifont iprec] . . . 157
SCHH [chh] . . . 157
SLWSC [lw] . . . 157
SMK [mkt] . . . 157
12.5 HPLOT . . . 158
SYMBOLS x y n [isymb ssize] . . . 158
ERRORS x y ex ey n [isymb ssize] . . . 158
KEY x y [isymb text] . . . 158
TICKS [chopt xval yval] . . . 158
ATITLE [xtit ytit] . . . 159
GRID . . . 159
NULL [xmin xmax ymin ymax chopt] . . . 159

SET [chatt value]

```
CHATT      C 'Attribute name' D='SHOW'
VALUE      R 'Attribute value' 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. See Table 12 on page 112.

OPTION [choptn]

```
CHOPTN     C 'Option name' D='SHOW'
```

Set general plotting options for HPLOT. If CHOPTN='SHOW' print all current and default options. If CHOPTN='*', restore all default options. See Table 11 on page 111.

METAFILE [lun metafl]

```
LUN        I 'Logical unit number' D=0
METAFL     I 'Metafile ID' D=4
```

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 initialisation time.

```
 METAFL=   4 Appendix E GKS.
 METAFL=-111 HIGZ/PostScript (Portrait).
 METAFL=-112 HIGZ/PostScript (Landscape).
```

12.1 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]

```
NTPRI      I 'Transformation with highest priority' D=-1
```

Locate a point on the screen using the graphics cursor and output coordinates on terminal. Several points may be located.

Control is returned when the point is outside picture limits or when the BREAK (right) mouse button is clicked (or CRTL/E). 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.

VLOCATE vecx vecy [chopt ntpri]

```
VECX       C 'Vector for coordinates X'
VECY       C 'Vector for coordinates Y'
CHOPT      C 'Options' D=' ' R='L,P,*,+,-'
NTPRI      I 'Transformation with highest priority' D=-1
```

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
```

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.

12.2 VIEWING

To define Normalization transformations. Either automatically (ZONE and SIZE) or 'by hand' (SVP, SWN and SELNT).

ZONE [nx ny ifirst chopt]

```
NX         I 'Number of divisions along X' D=1
NY         I 'Number of divisions along Y' D=1
IFIRST     I 'First division number' D=1
CHOPT      C 'Option' D=' ' R=' ,S'
```

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      R 'Size along X' D=20
YSIZE      R 'Size along Y' D=20
```

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 XSIZE cm. This command sets the parameters for the normalisation transformation number 1 to [0-XSIZE], [0-YSIZE]. This command is equivalent to a combination of the two commands `SET XSIZ` and `SET YSIZ` (see Table 12 on page 112).

SVP nt x1 x2 y1 y2

```
NT         I 'Normalization transformation number'
X1         R 'Low X  of viewport in NDC' D=0 R=0:1
X2         R 'High X of viewport in NDC' D=1 R=0:1
Y1         R 'Low Y  of viewport in NDC' D=0 R=0:1
Y2         R 'High Y of viewport in NDC' D=1 R=0:1
```

Set the viewport of the normalization transformation NT in the Normalized Device Coordinates (NDC).

SWN nt x1 x2 y1 y2

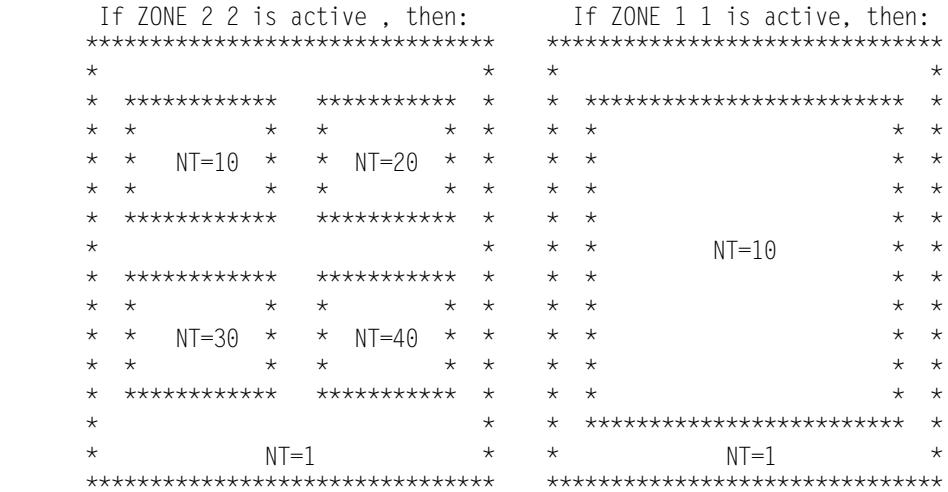
```
NT         I 'Normalize transformation number'
X1         R 'Low X  of window in WC' D=0
X2         R 'High X of window in WC' D=20
Y1         R 'Low Y  of window in WC' D=0
Y2         R 'High Y of window in WC' D=20
```

Set the window of the normalization transformation NT in World Coordinates (WC).

SELNT nt

```
NT         I 'Normalization transformation number'
```

Select a normalization transformation number.



12.3 PRIMITIVES

Call HIGZ drawing primitives

PLINE n x y

```
N          I 'Number of points'
X          C 'Vector name for X coordinates'
Y          C 'Vector name for Y coordinates'
```

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         R 'X first coordinate'
Y1         R 'Y first coordinate'
X2         R 'X second coordinate'
Y2         R 'Y second coordinate'
```

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          I 'Number of points'
X          C 'Vector name for X coordinates'
Y          C 'Vector name for Y coordinates'
```

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          I 'Number of points'
X          C 'Vector name for X coordinates'
Y          C 'Vector name for Y coordinates'
```

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        R 'X coordinate of first corner'
X2        R 'X coordinate of second corner'
Y1        R 'Y coordinate of first corner'
Y2        R 'Y coordinate of second corner'
```

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        R 'X coord of 1st corner of ext box'
X2        R 'X coord of 2nd corner of ext box'
Y1        R 'Y coord of 1st corner of ext box'
Y2        R 'Y coord of 2nd corner of ext box'
X3        R 'X coord of 1st corner of int box'
X4        R 'X coord of 2nd corner of int box'
Y3        R 'Y coord of 1st corner of int box'
Y4        R 'Y coord of 2nd corner of int box'
```

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        R 'X coordinate of start point'
X2        R 'X coordinate of end point'
Y1        R 'Y coordinate of start point'
Y2        R 'Y coordinate of end point'
SIZE      R 'Arrow size' 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.

ARC x1 y1 r1 [r2 phimin phimax]

```
X1        R 'X coordinate of centre'
Y1        R 'Y coordinate of centre'
R1        R 'Inner radius'
R2        R 'Outer radius' D=-1
PHIMIN    R 'Minimum angle' D=0
PHIMAX    R 'Maximum angle' 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.

AXIS x0 x1 y0 y1 wmin wmax ndiv [chopt]

```
X0         R 'X axis origin in WC'
X1         R 'X end axis in WC'
Y0         R 'Y axis origin in WC'
Y1         R 'Y end axis in WC'
WMIN       R 'Lowest value for labels'
WMAX       R 'Highest value for labels'
NDIV       I 'Number of divisions' D=510
CHOPT      C 'Options' D=' '
```

Draw an axis in the current Normalization transformation.

```
       NDIV=N1 + 100*N2 + 10000*N3
       N1, N2 and N3 are the number of 1st, 2nd and 3th level divisions, e.g.
           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= '0' : 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 bining optimization. By default the axis bining is optimized .
       CHOPT='N': No bining optimization
       CHOPT='I': Integer labelling
```

PIE x0 y0 radius n values [chopt iao ias iac]

```
X0          R 'X coordinate of centre of the pie'
Y0          R 'Y coordinate of centre of the pie'
RADIUS      R 'Radius of the pie chart'
N           I 'Number of values'
VALUES      C 'Vector name for N values'
CHOPT       C 'Options' D=' ' R=' ,P,N,L'
IAO         C 'Name of vector with offsets' D=' '
IAS         C 'Name of vector with styles' D=' '
IAC         C 'Name of vector with colors' 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           R 'X coordinate'
Y           R 'Y coordinate'
TEXT        C 'Text to be drawn'
SIZE        R 'Text size' D=0.3
ANGLE       R 'Comment angle' D=0
CHOPT       C 'Justification option' 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           R 'X coordinate'
Y           R 'Y coordinate'
TEXT        C 'Text to be drawn'
```

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      I 'Label identifier' D=1 R=1:9
NLABS       I 'Number of labels' D=0 R=0:50
CHLABS      C 'List of labels' 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).

HIST n x y [chopt]

```
N          I 'Number of values'
X          C 'Vector name for X coordinates'
Y          C 'Vector name for Y coordinates'
CHOPT      C 'Options' D='AHW' R=' ,W,R,N,H,F,C,L,*,P,B,A'
```

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 BARO 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          I 'Number of values'
X          C 'Vector name for X coordinates'
Y          C 'Vector name for Y coordinates'
CHOPT      C 'Options' D='ALW' R=' ,W,L,C,A,*,P,R,B'
```

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.
 '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 BARO 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 'AHW' in order to start a new zone
or/and draw the axes.
```

PAVE x1 x2 y1 y2 [dz isbox isfram chopt]

```
X1         R 'X bottom left corner of box'
X2         R 'X top right corner of box'
Y1         R 'Y bottom left corner of box'
Y2         R 'Y top right corner of box'
DZ         R 'Box width' D=0.4
ISBOX      I 'Box style' D=0
ISFRAM     I 'Frame style' D=5
CHOPT      C 'Option' 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
```

12.4 ATTRIBUTES

SLN [iln]

```
ILN        I 'Line style' D=1 R=1:
```

Set the line style.

SFAIS [ints]

```
INTS       I 'Fill area interior style' D=0 R=0:3
```

Set the fill area interior style : Hollow=0, Solid=1, Pattern=2, Hatch=3.

SFASI [styli]

```
STYLI      I 'Fill area style index' D=1
```

Set the fill area style index.

SFACI [ifaci]

```
IFACI      I 'Fill area color index' D=1
```

Set the fill area color index.

SPLCI [iplci]

```
IPLCI      I 'Polyline color index' D=1
```

Set the polyline color index.

SPMCI [ipmci]

```
IPMCI      I 'Polymarker color index' D=1
```

Set the polymarker color index.

STXCI [itxci]

```
ITXCI      I 'Text color index' D=1
```

Set the text color index.

STXFP [ifont iprec]

```
IFONT      I 'Font number' D=0
IPREC      I 'Font precision' D=2
```

Set text font and precision.

SCHH [chh]

```
CHH        R 'Character height' D=0.28
```

Set the character height.

SLWSC [lw]

```
LW         I 'Line width' D=1 R=1:
```

Set the line width.

SMK [mkt]

```
MKT        I 'Marker type' D=1
```

Set the marker type.

12.5 HPLOT

Draw various HPLOT objects (symbols, errors, key, etc.).

SYMBOLS x y n [isymb ssize]

```
X          C 'Vector of X coordinates'
Y          C 'Vector of Y coordinates'
N          I 'Number of points' D=1
ISYMB      I 'Symbol number' D=24
SSIZE      R 'Symbol size' 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          C 'Vector of X coordinates'
Y          C 'Vector of Y coordinates'
EX         C 'Vector of X error bars'
EY         C 'Vector of Y error bars'
N          I 'Number of points' D=1
ISYMB      I 'Symbol number' D=24
SSIZE      R 'Symbol size' 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          R 'X coordinate of comment'
Y          R 'Y coordinate of comment'
ISYMB      I 'Symbol number' D=24
TEXT       C 'Legend' D=' '
```

Draw one symbol and its explanation (legend) at a point x,y in the current normalization transformation.

TICKS [chopt xval yval]

```
CHOPT      C 'Options' D=' '
XVAL       R 'X position' D=9999
YVAL       R 'Y position' D=9999
```

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       C 'X Axis title' D=' '
YTIT       C 'Y Axis title' 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       R 'Low range in X' D=0
XMAX       R 'High range in X' D=1
YMIN       R 'Low range in Y' D=0
YMAX       R 'High range in Y' D=1
CHOPT      C 'Options' 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 13. HISTOGRAM

DELETE id . . . 161
PLOT [id chopt] . . . 161
FILE lun fname [lrecl chopt] . . . 162
LIST [chopt] . . . 162
ZOOM [id chopt icmin icmax] . . . 162
MANY_PLOTS idlist . . . 162
PROJECT id . . . 162
COPY id1 id2 [title] . . . 162
FIT id func [chopt np par step pmin pmax errpar] . . . 163
13.1 2D_PLOT . . . 164
LEGO [id theta phi] . . . 164
SURFACE [id theta phi] . . . 164
CONTOUR [id nlevel iflag param] . . . 164
13.2 CREATE . . . 164
1DHISTO id title ncx xmin xmax [valmax] . . . 164
PROFILE id title ncx xmin xmax ymin ymax [chopt] . . . 164
BINS id title ncx xbins [valmax] . . . 165
2DHISTO id title ncx xmin xmax ncy ymin ymax [valmax] . . . 165
PROX id . . . 165
PROY id . . . 165
SLIX id nslices . . . 165
SLIY id nslices . . . 165
BANX id ymin ymax . . . 165
BANY id xmin xmax . . . 165
TITLE_GLOBAL [chtitl] . . . 166
13.3 HIO . . . 166
HRIN id [icycle iofset] . . . 166
HROUT id [chopt] . . . 166
HSCRATCH id . . . 166
HFETCH id fname . . . 166
HREAD id fname . . . 166
PRINT id [chopt] . . . 166
DUMP id . . . 167
OUTPUT_LP [lun fname] . . . 167
GLOBAL_SECT gname . . . 167
GRESET id . . . 167
13.4 OPERATIONS . . . 167
ADD id1 id2 id3 [c1 c2] . . . 167
SUBTRACT id1 id2 id3 [c1 c2] . . . 167
MULTIPLY id1 id2 id3 [c1 c2] . . . 167
DIVIDE id1 id2 id3 [c1 c2] . . . 168
RESET id [title] . . . 168
DIFF id1 id2 [chopt] . . . 168
SMOOTH id [isel] . . . 168
SPLINE id [isel knotx kx] . . . 168
PARAM id [isel r2min maxpow] . . . 169
HSETPR param value . . . 169
13.5 GET_VECT . . . 170
CONTENTS id vname . . . 170
ERRORS id vname . . . 170
FUNCTION id vname . . . 170
ABSCISSA id vname . . . 170
REBIN id x y ex ey [n ifirst ilast] . . . 170
13.6 PUT_VECT . . . 171

CONTENTS id vname 171
ERRORS id vname 171
13.7 SET 171
MAXIMUM id vmax 171
MINIMUM id vmin 171
NORMALIZE_FACTOR id [xnorm] 171
SCALE_FACTOR_2D id [xscale] 171
IDOPT id option 172

DELETE id

```
ID        C 'Histogram Identifier'
```

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        C 'Histogram Identifier'
CHOPT     C 'Options' 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.
 '-'     Substract 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.
 '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.
 '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
```

FILE lun fname [lrecl chopt]

```
LUN        I 'Logical unit number' R=1:128
FNAME      C 'File name'
LRECL      I 'Record length in words' D=1024
CHOPT      C 'Options' 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      C 'Options' D=' ' R=' ,I'
```

List histograms and Ntuples in the current directory. If CHOPT='I' a verbose format is used (HINDEX).

ZOOM [id chopt icmin icmax]

```
ID         C 'Histogram Identifier'
CHOPT      C 'Options' D=' ' R=' ,C,S,+,B,L,P,*'
ICMIN      I 'First channel' D=1
ICMAX      I 'Last channel' D=9999
```

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     C 'List of histogram Identifiers'
```

Plot one or several histograms into the same plot. Plotted histograms are superimposed on the same zone of the picture.

PROJECT id

```
ID         C 'Histogram Identifier'
```

Fill all booked projections of a 2-Dim histogram. Filling is done using the 2-D contents of ID.

COPY id1 id2 [title]

```
ID1        C 'First histogram Identifier'
ID2        C 'Second histogram Identifier'
TITLE      C 'New title' 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        C 'Histogram Identifier'
FUNC      C 'Function name' D=' '
CHOPT     C 'Options' D=' ' R=' ,0,N,Q,V,B,L,D,W,M,*'
NP        I 'Number of parameters' D=0 R=0:34
PAR       C 'Vector of parameters'
STEP      C 'Vector of steps size'
PMIN      C 'Vector of lower bounds'
PMAX      C 'Vector of upper bounds'
ERRPAR    C 'Vector of errors on parameters'
```

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 ! 2 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 '0NQVBLDWM' + 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 centre 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.
```

13.1 2D_PLOT

Plotting of 2-Dim histograms in various formats.

LEGO [id theta phi]

```
ID         C 'Histogram Identifier'
THETA      R 'Angle THETA in degrees' D=30 R=0:90
PHI        R 'Angle PHI in degrees' D=30 R=0:90
```

Draw a lego plot (Manhattan plot) from a 2-Dim histogram.

SURFACE [id theta phi]

```
ID         C 'Histogram Identifier'
THETA      R 'Angle THETA in degrees' D=30 R=0:90
PHI        R 'Angle PHI in degrees' D=30 R=0:90
```

Draw a surface plot from a 2-Dim histogram.

CONTOUR [id nlevel iflag param]

```
ID         C 'Histogram Identifier'
NLEVEL     I 'Number of contour lines' D=10
IFLAG      I 'Rendering flag' D=1
PARAM      C 'Vector of containing contour level values'
```

Draw a contour plot from a 2-Dim histogram.

```
 IFLAG = 0 - use colour to distinguish contours
       = 1 - use line style to distinguish contours
       = 2 - line style and colour are the same for all contours
```

If PARAM is not given, contour levels are equidistant. If given, the vector PARAM can contain up to 50 values corresponding to the levels at which contours have to be drawn.

13.2 CREATE

Creation ("booking") of HBOOK objects in memory.

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 'Maximum bin content' 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         C 'Histogram Identifier'
TITLE      C 'Histogram title' D=' '
NCX        I 'Number of channels' D=100
XMIN       R 'Low edge in X' D=0
XMAX       R 'Upper edge in X' D=100
YMIN       R 'Low edge in Y' D=-0.1E+31
YMAX       R 'Upper edge in Y' D=0.1E+31
CHOPT      C 'Options' 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        C 'Histogram Identifier'
TITLE     C 'Histogram title' D=' '
NCX       I 'Number of channels' D=100
XBINS     C 'Vector of NCX+1 low-edges'
VALMAX    R 'Maximum bin content' 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        C 'Histogram Identifier'
TITLE     C 'Histogram title' D=' '
NCX       I 'Number of channels in X' D=40
XMIN      R 'Low edge in X' D=0
XMAX      R 'Upper edge in X' D=40
NCY       I 'Number of channels in Y' D=40
YMIN      R 'Low edge in Y' D=0
YMAX      R 'Upper edge in Y' D=40
VALMAX    R 'Maximum bin content' D=0
```

Create a two dimensional histogram. The contents are set to zero. See 1DHISTO for VALMAX.

PROX id

```
ID        C 'Histogram (2-Dim) Identifier'
```

Create the projection onto the x axis. The projection may be filled with Histo/Project (see also “PLOT [id chopt]” on page 161).

PROY id

```
ID        C 'Histogram (2-Dim) Identifier'
```

Create the projection onto the y axis. The projection may be filled with Histo/Project (see also “PLOT [id chopt]” on page 161).

SLIX id nslices

```
ID        C 'Histogram (2-Dim) Identifier'
NSLICE    I 'Number of slices'
```

Create projections onto the x axis, in y-slices. The projection may be filled with Histo/Project.

SLIY id nslices

```
ID        C 'Histogram (2-Dim) Identifier'
NSLICE    I 'Number of slices'
```

Create projections onto the y axis, in x-slices. The projection may be filled with Histo/Project.

BANX id ymin ymax

```
ID        C 'Histogram (2-Dim) Identifier'
YMIN      R 'Low edge in Y'
YMAX      R 'Upper edge in Y'
```

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        C 'Histogram (2-Dim) Identifier'
XMIN      R 'Low edge in X'
XMAX      R 'Upper edge in X'
```

Create a projection onto the y axis, in a band of x. The projection may be filled with Histo/Project.

TITLE_GLOBAL [chtitl]

```
CHTITL     C 'Global title' D=' '
```

Set the global title. The global title is plotted at the top of each picture. The size and the Y position of the global title may be changed by the commands SET GSIZ and SET YGTI respectively.

13.3 HIO

Input/Output operations of histograms.

HRIN id [icycle iofset]

```
ID         C 'Histogram Identifier'
ICYCLE     I 'Cycle number' D=999
IOFSET     I 'Offset' 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         C 'Histogram Identifier'
CHOPT      C 'Options' 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         C 'Histogram Identifier'
```

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         C 'Histogram Identifier'
FNAME      C 'File name'
```

Fetch histogram ID from file FNAME. FNAME has been created by the old version of HBOOK3 (Unformatted).

HREAD id fname

```
ID         C 'Histogram Identifier'
FNAME      C 'File name'
```

Read histogram ID from file FNAME. FNAME has been created by the old version of HBOOK3 (Formatted).

PRINT id [chopt]

```
ID         C 'Histogram Identifier'
CHOPT      C 'Options' 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        C 'Histogram Identifier'
```

Dump the histogram ZEBRA data structure on the terminal.

OUTPUT_LP [lun fname]

```
LUN       I 'Logical unit number' D=6
FNAME     C 'File name' 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     C 'Global section name' D=' '
```

Map the global section GNAME (VAX only). The current directory is changed to //GNAME.

GRESET id

```
ID        C 'Histogram Identifier'
```

Reset histogram ID in the global section.

13.4 OPERATIONS

Histogram operations and comparisons.

ADD id1 id2 id3 [c1 c2]

```
ID1       C 'First histogram Identifier'
ID2       C 'Second histogram Identifier'
ID3       C 'Result histogram Identifier'
C1        R 'Scale factor for ID1' D=1
C2        R 'Scale factor for ID2' D=1
```

Add histograms: ID3 = C1*ID1 + C2*ID2. Applicable to 1-Dim and 2-Dim histograms. See command HRIN to automatically add histograms with the same ID from different files.

SUBTRACT id1 id2 id3 [c1 c2]

```
ID1       C 'First histogram Identifier'
ID2       C 'Second histogram Identifier'
ID3       C 'Result histogram Identifier'
C1        R 'Scale factor for ID1' D=1
C2        R 'Scale factor for ID2' D=1
```

Subtract histograms: ID3 = C1*ID1 - C2*ID2. Applicable to 1-Dim and 2-Dim histograms.

MULTIPLY id1 id2 id3 [c1 c2]

```
ID1       C 'First histogram Identifier'
ID2       C 'Second histogram Identifier'
ID3       C 'Result histogram Identifier'
C1        R 'Scale factor for ID1' D=1
C2        R 'Scale factor for ID2' 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        C 'First histogram Identifier'
ID2        C 'Second histogram Identifier'
ID3        C 'Result histogram Identifier'
C1         R 'Scale factor for ID1' D=1
C2         R 'Scale factor for ID2' D=1
```

Divide histograms: ID3 = C1*ID1 / C2*ID2. Applicable to 1-Dim and 2-Dim histograms.

RESET id [title]

```
ID         C 'Histogram Identifier'
TITLE      C 'New title' D=' '
```

Reset contents and errors of an histogram. Bin definition is not modified.

DIFF id1 id2 [chopt]

```
ID1        C 'First Histogram Identifier'
ID2        C 'Second Histogram Identifier'
CHOPT      C 'Options' D='D' R=' ,N,D,O,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.
  O Overflow, requests that overflow bins be taken into account.
  U Underflow, requests that underflow bins be taken into account.
```

SMOOTH id [isel]

```
ID         C 'Histogram Identifier'
ISEL       I 'Option flag' 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         C 'Histogram Identifier'
ISEL       I 'Option flag' D=2
KNOTX      I 'Number of knots' D=10
KX         I 'Degree of the spline' 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         C 'Histogram Identifier'
ISEL       I 'Control word' D=11
R2MIN      R 'Min correlation coefficient' D=1
MAXPOW     I 'Max degree of polynomials' 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      C 'Parameter name' D='FEPS'
VALUE      R 'Parameter value' 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         C 'Histogram Identifier'
VNAME      C 'Vector name'
```

Get contents of histogram ID into vector VNAME.

ERRORS id vname

```
ID         C 'Histogram Identifier'
VNAME      C 'Vector name'
```

Get errors of histogram ID into vector VNAME.

FUNCTION id vname

```
ID         C 'Histogram Identifier'
VNAME      C 'Vector name'
```

Get function associated to histogram ID into vector VNAME.

ABSCISSA id vname

```
ID         C 'Histogram Identifier'
VNAME      C 'Vector name'
```

Get values of center of bins abscissa into vector VNAME.

REBIN id x y ex ey [n ifirst ilast]

```
ID         C 'Histogram Identifier'
X          C 'Name of vector X'
Y          C 'Name of vector Y'
EX         C 'Name of vector EX'
EY         C 'Name of vector EY'
N          I 'Number of elements to fill' D=100
IFIRST     I 'First bin' D=1
ILAST      I 'Last bin' D=100
```

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        C 'Histogram Identifier'
VNAME     C 'Vector name'
```

Replace contents of histogram with values of vector VNAME.

ERRORS id vname

```
ID        C 'Histogram Identifier'
VNAME     C 'Vector name'
```

Replace errors of histogram with values of vector VNAME.

13.7 SET

Set histogram attributes.

MAXIMUM id vmax

```
ID        C 'Histogram Identifier'
VMAX      R 'Maximum value'
```

Set the maximum value on the Y axis. To select again an automatic scale, just set VMAX less then the minimum.

MINIMUM id vmin

```
ID        C 'Histogram Identifier'
VMIN      R 'Minimum value'
```

Set the minimum value on the Y axis. To select again an automatic scale, just set VMIN greater then the maximum.

NORMALIZE_FACTOR id [xnorm]

```
ID        C 'Histogram Identifier'
XNORM     R 'Normalization factor' 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        C 'Histogram Identifier'
XSCALE    R 'Scale factor' D=0
```

Set the scale factor for histograms (2-Dim).

IDOPT id option

```
ID         C 'Histogram Identifier'
OPTION     C 'Options'
```

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 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
NPCO  Do not print bin contents
PLOW* Print values of low edge of the bins
NPLO  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
```

Figure 75. Description of the IDOPT parameters

Chapter 14. KUIP

HELP [item] . . . 174
USAGE [cmnd] . . . 174
MANUAL [item outfil docsys] . . . 174
EDIT fname . . . 174
LAST [n fname] . . . 174
MESSAGE [string] . . . 175
SHELL [cmd] . . . 175
WAIT [string sec] . . . 175
UNITS . . . 175
EXIT . . . 175
QUIT . . . 175
14.1 ALIAS . . . 175
CREATE aname string . . . 175
LIST . . . 176
DELETE alist . . . 176
TRANSLATION [option] . . . 176
14.2 SET_SHOW . . . 176
STYLE [option sgylen sgsize sgyspa sgbord wktype] . . . 176
PANEL line [gkey] . . . 176
COMMAND [chpath] . . . 177
APPLICATION [path cmdex] . . . 177
ROOT [path] . . . 177
TIMING [option] . . . 178
PROMPT [option] . . . 178
BREAK [option] . . . 178
COLUMNS [ncol] . . . 178
RECORDING [nrec] . . . 178
HOST_EDITOR [option top left width height dxpad dypad npads] . . . 178

HELP [item]

```
ITEM       C 'Command name' 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       C 'Command name' D=' '
```

Give the syntax of a command. If CMND='/' the syntax of all commands is given.

MANUAL [item outfil docsys]

```
ITEM       C 'Command name or help (key)word' D=' '
OUTFIL     C 'Output file name' D=' '
DOCSYS     C 'Text formatting system' 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.

EDIT fname

```
FNAME      C 'File name'
```

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          I 'N last commands to be saved' D=-1
FNAME      C 'File name' 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     C 'Message string' 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        C 'Shell command' 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     C 'Message string' D=' '
SEC        R 'Number of seconds' 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.

14.1 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

```
ANAME      C 'Alias name'
STRING     C 'Equivalent string'
```

Create an alias named ANAME corresponding to STRING. Also switch ON the alias translation, i.e. ALIAS/TRANSLATION ON. 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
```

LIST

List all aliases (name, equivalent string).

DELETE alist

```
ALIST      C 'Alias list'
```

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     C 'Option' 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.

14.2 SET_SHOW

Set or show various KUIP parameters and options.

STYLE [option sgylen sgsize sgyspa sgbord wktype]

```
OPTION     C 'Option' D='?' R='?,C,AN,AL,G,GW,GS,GP'
SGYLEN     R 'max Y LENgth of each menu item box' D=0.025 R=0.005:0.25
SGSIZE     R 'space available for the application' D=0.8 R=0:0.9
SGYSPA     R 'max Y length of space between menus' D=0.02 R=-0.5:0.5
SGBORD     R 'X or Y border for menus' D=0.015 R=0:0.25
WKTYPE     I 'Graphics workstation type' 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 inizialized).

PANEL line [gkey]

```
LINE       R 'Line number' D=0
GKEY       C 'Graphics key value(s)' 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     C 'Path name for command line' 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 CHPATH 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       C 'Application name' D=' '
CMDEX      C 'Exit command' 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).

ROOT [path]

```
PATH       C 'Root directory' 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     C 'Option' 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     C 'Prompt string' D=' '
```

Set the prompt string for the command mode dialogue. If OPTION is blank the current prompt is left unchanged.

BREAK [option]

```
OPTION     C 'Option' 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       I 'Number of columns for terminal output' 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       I 'Rate for recording on history file' 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     C 'Name of default host editor' D='?' R='EDT,TPU,WINDOW,PAD'
TOP        I 'Top position of the edit window' D=20
LEFT       I 'Left position of the edit window' D=20
WIDTH      I 'Width of the edit window' D=0
HEIGHT     I 'Height of the edit window' D=0
DXPAD      I 'X offset for help PAD windows' D=30 R=0:
DYPAD      I 'Y offset for help PAD windows' D=20 R=0:
NPADS      I 'Maximum number of shifted pads' D=4 R=1:
```

Set the default host editor (only on the VAX). If OPTION='?' the current host editor is shown. The EDIT command will invoke this editor. The startup value is EDIT/EDT.

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 meaningful on the VAX only: on the Apollo the default host editor is always the screen editor and on IBM/VM-CMS it is always XEDIT.

Chapter 15. MACRO

EXEC mname 179
LIST [mname] 179
TRACE [option level prompt] 179
DEFAULTS [chpath] 180
RECURSION [option] 180

EXEC mname

```
MNAME      C 'Macro name'
```

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      C 'Macro name restrictions' 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     C 'Option' D='ON' R='ON,OFF'
LEVEL      C 'Level' D=' ' R=' ,WAIT,FULL'
PROMPT     C 'Prompt string' D=' '
```

Set ON/OFF the trace of commands during macro execution. 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 TRACE='ON' the command being executed is written on the terminal, after the prompt defined in the parameter PROMPT; 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     C 'Path name for macro files' 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     C 'Option' D='ON' R='ON,OFF'
```

Set ON/OFF the option to execute macros recursively. The startup value is OFF.

Chapter 16. NETWORK

RLOGIN host 181
RSHELL message 181

RLOGIN host

```
HOST       C 'Host name' D=' '
```

Start a communication with a remote machine HOST. Current Directory will be changed to //HOST.

RSHELL message

```
MESSAG     C 'Message to remote host' 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.

Chapter 17. NTUPLE

CREATE idn title nvar chrzpa nprime varlist 183
LOOP idn uwfunc [nevent ifirst] 183
MERGE idn1 idn2 [uwfunc nevent ifirst] 183
PRINT idn 183
PROJECT idh idn [uwfunc nevent ifirst] 184
READ idn fname [format chopt nevent] 184
UWFUNC idn fname [chopt] 184
PLOT idn [uwfunc nevent ifirst nupd chopt] 185
CUTS icut [chopt fname] 186
SCAN idn [chfunc nevent ifirst nvars varlis] 186
CSELECT [chopt csize] 187
MASK mname [chopt number] 187
LIST 187

CREATE idn title nvar chrzpa nprime varlist

```
IDN        C 'Ntuple Identifier'
TITLE      C 'Ntuple title' D=' '
NVAR       I 'Number of variables' D=1 R=1:512
CHRZPA     C 'RZ path' D=' '
NPRIME     I 'Primary allocation' D=1000
VARLIS     C 'Names of the NVAR variables'
```

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.

LOOP idn uwfunc [nevent ifirst]

```
IDN        C 'Identifier of Ntuple'
UWFUNC     C 'Selection function or cut number' D=' '
NEVENT     I 'Number of events' D=999999
IFIRST     I 'First event' D=1
```

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       C 'Identifier of first Ntuple'
IDN2       C 'Identifier of second Ntuple'
UWFUNC     C 'Selection function or cut number' D=' '
NEVENT     I 'Number of events' D=999999
IFIRST     I 'First event' D=1
```

Merge two Ntuples. Invoke the selection function UWFUNC for each of the NEVENT events starting at event IFIRST of Ntuple IDN1. 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.

As an example, suppose we have four files containing Ntuple `ID=10` and we want to merge the four files into file 4. Then the following sequence of commands should be used:

```
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 .........
```

PRINT idn

```
IDN        C 'Ntuple Identifier'
```

Print a summary about Ntuple IDN. Number of entries, variables names and limits are listed.

PROJECT idh idn [uwfunc nevent ifirst]

```
IDH        C 'Identifier of histogram to fill'
IDN        C 'Identifier of Ntuple'
UWFUNC     C 'Selection function or cut number' D=' '
NEVENT     I 'Number of events' D=999999
IFIRST     I 'First event' D=1
```

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        C 'Ntuple Identifier'
FNAME      C 'File name'
FORMAT     C 'Format' D='*'
CHOPT      C 'Options' D=' '
NEVENT     I 'Number of events' 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.

UWFUNC idn fname [chopt]

```
IDN        C 'Ntuple Identifier'
FNAME      C 'File name'
CHOPT      C 'Options' 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.
```

PLOT idn [uwfunc nevent ifirst nupd chopt]

```
IDN       C 'Ntuple Identifier'
UWFUNC    C 'Selection function' D='0'
NEVENT    I 'Number of events' D=999999
IFIRST    I 'First event' D=1
NUPD      I 'Frequency to update histogram' D=1000000
CHOPT     C 'Options' 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 command HISTO/PLOT for explanation of CHOPT.

```
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/PLOT 30.X Y  weight of each event is variable Y
         NT/PLOT 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/PLOT 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/PLOT 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/PLOT 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/PLOT 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/PLOT 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 CHOPT. Every NUPD events, the current status of the histogram is displayed.

CUTS icut [chopt fname]

```
ICUT       I 'Cut number' R=0:100
CHOPT      C 'Options' D='P' R=' ,G,P,S,-,R,W,D'
FNAME      C 'File name' D=' '
```

Define cut number ICUT for an Ntuple. This cut can then be used in subsequent commands NTUPLE/PLOT, PROJECT.

```
  CHOPT='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 delimitate
            the selected area. The polygon will automatically
            be closed by PAW.
  CHOPT='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.
  CHOPT='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.

SCAN idn [chfunc nevent ifirst nvars varlis]

```
IDN        C 'Ntuple Identifier'
CHFUNC     C 'User cut function' D='0'
NEVENT     I 'Number of events' D=999999
IFIRST     I 'First event' D=1
NVARS      I 'Number of variables to scan' D=8 R=0:8
VARLIS     C 'Names of the NVARS variables to scan'
```

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.

CSELECT [chopt csize]

```
CHOPT      C 'Options' D='N' R=' N,R,B,C'
CSIZE      R 'Comment size' 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.
 CSEL  ' '     Reset CSEL mechanism
```

MASK mname [chopt number]

```
MNAME      C 'Mask name'
CHOPT      C 'Options' D=' ' R=' ,U,N,P,C,R'
NUMBER     I 'Bit number' 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
```

LIST

List all Ntuples in the Current Directory. Note that the command HISTO/LIST lists all histograms and Ntuples in the Current Directory.

Chapter 18. PICTURE

FILE lun fname [lrecl chopt] . . . 188
LIST . . . 188
CREATE pname . . . 188
DELETE pname . . . 188
SCRATCH pname . . . 188
PLOT [pname] . . . 189
MODIFY [pname chopt] . . . 189
MERGE pname [x y scale chopt] . . . 189
COPY pname1 pname2 . . . 189
RENAME pname1 pname2 . . . 189
IZOUT [pname] . . . 189
IZIN pname [icycle] . . . 189
IZPICT pname [chopt] . . . 190
SWITCH [chopt] . . . 190
IGSET [chatt value] . . . 190

FILE lun fname [lrecl chopt]

```
LUN       I 'Logical unit number' R=1:128
FNAME     C 'File name'
LRECL     I 'Record length in words' D=1024
CHOPT     C 'Options' 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     C 'Picture name'
```

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     C 'Picture name' D=' '
```

Delete the picture PNAME from memory. PNAME='*' means all pictures.

SCRATCH pname

```
PNAME     C 'Picture name' D=' '
```

Delete the picture PNAME from current directory on disk.

PLOT [pname]

```
PNAME      C 'Picture name' D=' '
```

Plot the picture PNAME. PNAME=' ' means the current picture. PNAME='*' means all pictures.

MODIFY [pname chopt]

```
PNAME      C 'Picture name' D=' '
CHOPT      C 'Options' 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      C 'Picture name'
X          R 'X coord(NDC) where to draw PNAME' D=0
Y          R 'Y coord(NDC) where to draw PNAME' D=0
SCALE      R 'Scale factor' D=1
CHOPT      C 'Options' D=' ' R=' ,D'
```

Add the picture PNAME to the current picture.

```
 CHOPT='D' Picture PNAME is displayed during merging.
```

COPY pname1 pname2

```
PNAME1     C 'Picture name'
PNAME2     C 'New picture name'
```

Copy a picture.

RENAME pname1 pname2

```
PNAME1     C 'Old picture name'
PNAME2     C 'New picture name'
```

Rename a picture.

IZOUT [pname]

```
PNAME      C 'Picture name' 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      C 'Picture name'
ICYCLE     I 'Cycle number' D=9999
```

Read picture into memory from a direct access picture file. (see command PICTURE/FILE). PNAME='*' means all pictures.

IZPICT pname [chopt]

```
PNAME      C 'Picture name'
CHOPT      C 'Options' 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      C 'Options' 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      C 'Attribute name' D='SHOW'
VALUE      R 'Attribute value' 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. See Table 10 on page 110.

Chapter 19. VECTOR

CREATE vname [type] . . . 192
LIST . . . 192
DELETE vlist . . . 192
COPY vnam1 vnam2 . . . 192
PRINT vname . . . 192
READ vlist fname [format chopt match] . . . 193
WRITE vlist [fname format chopt] . . . 193
INPUT vname . . . 193
19.1 OPERATIONS . . . 194
VBIAS vnam1 bias vnam2 . . . 194
VSCALE vnam1 scale vnam2 . . . 194
VADD vnam1 vnam2 vnam3 . . . 194
VMULTIPLY vnam1 vnam2 vnam3 . . . 194
VSUBTRACT vnam1 vnam2 vnam3 . . . 194
VDIVIDE vnam1 vnam2 vnam3 . . . 194
HFILL vname id . . . 194
DRAW vname [id chopt] . . . 195
PLOT vname [id chopt] . . . 195
FIT x y ey func [chopt np par step pmin pmax errpar] . . . 195

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. Is 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      C 'Vector name(length)'
TYPE       C 'Vector type' 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      C 'Vector list' 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      C 'Source vector name'
VNAM2      C 'Destination vector name'
```

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.

PRINT vname

```
VNAME      C 'Vector name'
```

Write to the terminal the content of a vector. Enter HELP VECTOR for more information on vector addressing.

READ vlist fname [format chopt match]

```
VLIST      C 'Vector list'
FNAME      C 'File name' D=' '
FORMAT     C 'Format' D=' '
CHOPT      C 'Options' D='OC' R='OC,O, ,C'
MATCH      C 'Matching pattern' 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.

CHOPT is used to select between the following options:

```
 'OC'   file is Open, read and then Closed (default case)
 'O'    file is Open 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 CHOPT, 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
```

Enter HELP VECTOR for more information on vector addressing.

WRITE vlist [fname format chopt]

```
VLIST      C 'Vector list'
FNAME      C 'File name' D=' '
FORMAT     C 'Format' D='5(1X,G13.7)'
CHOPT      C 'Options' D='OC' R='OC,O, ,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 Open, written and then Closed (default case)
 'O'    file is Open 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.

INPUT vname

```
VNAME      C 'Vector name'
```

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.

19.1 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      C 'Source vector name'
BIAS       R 'Bias value'
VNAM2      C 'Destination vector name'
```

VNAM2(I) = BIAS + VNAM1(I)

VSCALE vnam1 scale vnam2

```
VNAM1      C 'Source vector name'
SCALE      R 'Scale factor'
VNAM2      C 'Destination vector name'
```

VNAM2(I) = SCALE * VNAM1(I)

VADD vnam1 vnam2 vnam3

```
VNAM1      C 'First source vector name'
VNAM2      C 'Second source vector name'
VNAM3      C 'Destination vector name'
```

VNAM3(I) = VNAM1(I) + VNAM2(I)

VMULTIPLY vnam1 vnam2 vnam3

```
VNAM1      C 'First source vector name'
VNAM2      C 'Second source vector name'
VNAM3      C 'Destination vector name'
```

VNAM3(I) = VNAM1(I) * VNAM2(I)

VSUBTRACT vnam1 vnam2 vnam3

```
VNAM1      C 'First source vector name'
VNAM2      C 'Second source vector name'
VNAM3      C 'Destination vector name'
```

VNAM3(I) = VNAM1(I) - VNAM2(I)

VDIVIDE vnam1 vnam2 vnam3

```
VNAM1      C 'First source vector name'
VNAM2      C 'Second source vector name'
VNAM3      C 'Destination vector name'
```

VNAM3(I) = VNAM1(I) / VNAM2(I) (or 0 if VNAM2(I)=0)

HFILL vname id

```
VNAME      C 'Vector name'
ID         C 'Histogram Identifier'
```

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.

DRAW vname [id chopt]

```
VNAME      C 'Vector name'
ID         C 'Histogram Identifier' D='12345'
CHOPT      C 'Options' 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.
```

PLOT vname [id chopt]

```
VNAME      C 'Vector name'
ID         C 'Histogram Identifier' D='12345'
CHOPT      C 'Options' 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          C 'Vector of X coordinates'
Y          C 'Vector of Y coordinates'
EY         C 'Vector of errors on Y' D='?'
FUNC       C 'Function name'
CHOPT      C 'Character options' D=' ' R=' ,0,N,Q,V,B,D,W,M,*'
NP         I 'Number of parameters' D=0 R=0:20
PAR        C 'Vector of parameters'
STEP       C 'Vector of steps size'
PMIN       C 'Vector of lower bounds'
PMAX       C 'Vector of upper bounds'
ERRPAR     C 'Vector of errors on parameters'
```

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.

Chapter 20. ZEBRA

20.1 RZ 197
FILE lun fname [lrecl chopt] 197
MDIR chdir [nwkey chform chtags] 197
DDIR chdir 197
LDIR [chpath chopt] 197
CDIR [chpath chopt] 197
PURGE [keep] 197
LOCK [chlock] 197
FREE [chlock] 198
STAT chpath 198
20.2 FZ 198
FILE lun fname [lrecl chopt] 198
TOFZ lun [chopt] 198
FRFZ lun [chopt] 198
TOALPHA fname 198
FRALPHA fname 198
20.3 DZ 199
SHOW name [number chopt] 199
SURV name [number] 199
SNAP [idiv chopt] 199
VERIFY [idiv chopt] 199
STORE [ixstor] 199

20.1 RZ

ZEBRA/RZ package: direct access Input/Output.

FILE lun fname [lrecl chopt]

```
LUN        I 'Logical unit number' R=1:128
FNAME      C 'File name'
LRECL      I 'Record length in WORDS' D=1024
CHOPT      C 'Options' D=' ' R=' ,U'
```

Open an existing direct access file.

```
 CHOPT=' ' readonly mode
 CHOPT='U' update mode
```

MDIR chdir [nwkey chform chtags]

```
CHDIR      C 'Directory name'
NWKEY      I 'Number of words per Key' D=1
CHFORM     C 'CHFORM' D='I'
CHTAGS     C 'List of Tags' D='HBOOK-ID'
```

Create a new RZ directory below the current directory.

DDIR chdir

```
CHDIR      C 'Directory name'
```

Delete the directory CHDIR from the current directory.

LDIR [chpath chopt]

```
CHPATH     C 'Path name' D=' '
CHOPT      C 'Options' D=' ' R=' ,A'
```

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.
```

CDIR [chpath chopt]

```
CHPATH     C 'Path name' D=' '
CHOPT      C 'Options' 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       I 'Number of cycles to be kept' D=1
```

Purge an RZ directory.

LOCK [chlock]

```
CHLOCK     C 'Lock identifier' D='RZFILE'
```

Lock an RZ directory.

FREE [chlock]

```
CHLOCK     C 'Lock identifier' D='RZFILE'
```

Free an RZ directory.

STAT chpath

```
CHPATH     C 'Name of top directory'
```

Print space statistics for an RZ file.

20.2 FZ

ZEBRA/FZ package: sequential access Input/Output.

FILE lun fname [lrecl chopt]

```
LUN        I 'Logical unit number' R=1:128
FNAME      C 'File name'
LRECL      I 'Record length in words' D=900
CHOPT      C 'Options' D='IX' R='I,O,X,A,IX,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        I 'Logical unit number of FZ file' R=1:128
CHOPT      C 'Options' D=' '
```

Copy the current directory tree onto an FZ file.

FRFZ lun [chopt]

```
LUN        I 'Logical unit number of FZ file' R=1:128
CHOPT      C 'Options' D=' '
```

Copy the FZ file into the current directory tree.

TOALPHA fname

```
FNAME      C 'Name of the FZ text file'
```

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      C 'Name of the FZ text file'
```

Copy the FZ alphanumeric file into the current directory.

20.3 DZ

SHOW name [number chopt]

```
NAME       C 'Bank name'
NUMBER     I 'Bank number' D=1
CHOPT      C 'Options' 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       C 'Bank name'
NUMBER     I 'Bank number' D=1
```

Print a survey of the structure identified by NAME, NUMBER.

SNAP [idiv chopt]

```
IDIV       I 'Division number' D=2 R=0:24
CHOPT      C 'Options' 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       I 'Division number' D=0 R=0:24
CHOPT      C 'Options' 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 in the banks (implies 'C')
 CHOPT='S' Check the store parameters
 CHOPT='U' Check the validity of the up and origin links in the banks (implies 'C')
 CHOPT='F' Errors are considered fatal and generate a call to ZFATAL
```

STORE [ixstor]

```
IXSTOR     I 'Store number' 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.

Appendix A. PAW commands and parameters

Table 16 (Page 1 of 5). PAW commands and parameters

Page	PAW command with parameters
	FORTRAN
144	CALL urout
145	CLOSE lun
143	COMIS
145	FILE lun fname
145	LOOP ntimes urout
145	REWIND lun
145	SIGMA
	FUNCTION
146	FUN1 id ufunc ncx xmin xmax [chopt]
146	FUN2 id ufunc ncx xmin xmax ncy ymin ymax [chopt]
147	PLOT ufunc xlow xup [chopt]
147	POINTS npfunc
	GRAPHICS
	ATTRIBUTES
157	SCHH [chh]
157	SFACI [ifaci]
157	SFAIS [ints]
157	SFASI [styli]
157	SLN [iln]
157	SLWSC [lw]
157	SMK [mkt]
157	SPLCI [iplci]
157	SPMCI [ipmci]
157	STXCI [itxci]
157	STXFP [ifont iprec]
	HPLOT
159	ATITLE [xtit ytit]
158	ERRORS x y ex ey n [isymb ssize]
159	GRID
158	KEY x y [isymb text]
159	NULL [xmin xmax ymin ymax chopt]
158	SYMBOLS x y n [isymb ssize]
158	TICKS [chopt xval yval]
149	METAFILE [lun metafl]
	MISC
149	CLR
150	HMOVE
149	LOCATE [ntpri]
149	NEXT
149	VLOCATE vecx vecy [chopt ntpri]
149	OPTION [choptn]

Table 16 (Page 2 of 5). PAW commands and parameters

Page	PAW command with parameters
	PRIMITIVES
152	ARC x1 y1 r1 [r2 phimin phimax]
152	ARROW x1 x2 y1 y2 [size]
153	AXIS x0 x1 y0 y1 wmin wmax ndiv [chopt]
152	BOX x1 x2 y1 y2
151	FAREA n x y
152	FBOX x1 x2 y1 y2 x3 x4 y3 y4
156	GRAPH n x y [chopt]
155	HIST n x y [chopt]
154	ITX x y text
154	LABELS labnum nlabs chlabs
151	LINE x1 y1 x2 y2
156	PAVE x1 x2 y1 y2 [dz isbox isfram chopt]
154	PIE x0 y0 radius n values [chopt iao ias iac]
151	PLINE n x y
151	PMARKER n x y
154	TEXT x y text size [angle chopt]
149	SET [chatt value]
	VIEWING
151	SELNT nt
150	SIZE [xsize ysize]
150	SVP nt x1 x2 y1 y2
150	SWN nt x1 x2 y1 y2
150	ZONE [nx ny ifirst chopt]
	HISTOGRAM
	2D_PLOT
164	CONTOUR [id nlevel iflag]
164	LEGO [id theta phi]
164	SURFACE [id theta phi]
162	COPY id1 id2 [title]
	CREATE
164	1DHISTO id title ncx xmin xmax [valmax]
165	2DHISTO id title ncx xmin xmax ncy ymin ymax [valmax]
165	BANX id ymin ymax
165	BANY id xmin xmax
165	BINS id title ncx xbins [valmax]
164	PROFILE id title ncx xmin xmax ymin ymax [chopt]
165	PROX id
165	PROY id
165	SLIX id nslices
165	SLIY id nslices
166	TITLE_GLOBAL [chtitl]
161	DELETE id
162	FILE lun fname [lrecl chopt]
163	FIT id func [chopt np par step pmin pmax errpar]
	GET_VECT
170	ABSCISSA id vname
170	CONTENTS id vname
170	ERRORS id vname
170	FUNCTION id vname
170	REBIN id x y ex ey [n ifirst ilast]

Table 16 (Page 3 of 5). PAW commands and parameters

Page	PAW command with parameters
	HIO
167	DUMP id
167	GLOBAL_SECT gname
167	GRESET id
166	HFETCH id fname
166	HREAD id fname
166	HRIN id [icycle iofset]
166	HROUT id [chopt]
166	HSCRATCH id
167	OUTPUT_LP [lun fname]
166	PRINT id [chopt]
162	LIST [chopt]
162	MANY_PLOTS idlist
	OPERATIONS
167	ADD id1 id2 id3 [c1 c2]
168	DIFF id1 id2 [chopt]
168	DIVIDE id1 id2 id3 [c1 c2]
169	HSETPR param value
167	MULTIPLY id1 id2 id3 [c1 c2]
169	PARAM id [isel r2min maxpow]
168	RESET id [title]
168	SMOOTH id [isel]
168	SPLINE id [isel knotx kx]
167	SUBTRACT id1 id2 id3 [c1 c2]
161	PLOT [id chopt]
162	PROJECT id
	PUT_VECT
171	CONTENTS id vname
171	ERRORS id vname
	SET
172	IDOPT id option
171	MAXIMUM id vmax
171	MINIMUM id vmin
171	NORMALIZE_FACTOR id [xnorm]
171	SCALE_FACTOR_2D id [xscale]
162	ZOOM [id chopt icmin icmax]
	KUIP
	ALIAS
175	CREATE aname string
176	DELETE alist
176	LIST
176	TRANSLATION [option]
174	EDIT fname
175	EXIT
174	HELP [item]
174	LAST [n fname]
174	MANUAL [item outfil docsys]
175	MESSAGE [string]
175	QUIT

Table 16 (Page 4 of 5). PAW commands and parameters

Page	PAW command with parameters
	SET_SHOW
177	APPLICATION [path cmdex]
178	BREAK [option]
178	COLUMNS [ncol]
177	COMMAND [chpath]
178	HOST_EDITOR [option top left width height dxpad dypad npads]
176	PANEL line [gkey]
178	PROMPT [option]
178	RECORDING [nrec]
177	ROOT [path]
176	STYLE [option sgylen sgsize sgyspa sgbord wktype]
178	TIMING [option]
175	SHELL [cmd]
175	UNITS
174	USAGE [cmnd]
175	WAIT [string sec]
	MACRO
180	DEFAULTS [chpath]
179	EXEC mname
179	LIST [mname]
180	RECURSION [option]
179	TRACE [option level prompt]
	NETWORK
181	RLOGIN host
181	RSHELL message
	NTUPLE
183	CREATE idn title nvar chrzpa nprime varlist
187	CSELECT [chopt csize]
186	CUTS icut [chopt fname]
187	LIST
183	LOOP idn uwfunc [nevent ifirst]
187	MASK mname [chopt number]
183	MERGE idn1 idn2 [uwfunc nevent ifirst]
185	PLOT idn [uwfunc nevent ifirst nupd chopt]
183	PRINT idn
184	PROJECT idh idn [uwfunc nevent ifirst]
184	READ idn fname [format chopt nevent]
186	SCAN idn [chfunc nevent ifirst nvars varlis]
184	UWFUNC idn fname [chopt]
	PICTURE
189	COPY pname1 pname2
188	CREATE pname
188	DELETE pname
188	FILE lun fname [lrecl chopt]
190	IGSET [chatt value]
189	IZIN pname [icycle]
189	IZOUT [pname]
190	IZPICT pname [chopt]
188	LIST
189	MERGE pname [x y scale chopt]
189	MODIFY [pname chopt]
189	PLOT [pname]
189	RENAME pname1 pname2
188	SCRATCH pname
190	SWITCH [chopt]

Table 16 (Page 5 of 5). PAW commands and parameters

Page	PAW command with parameters
	VECTOR
192	COPY vnam1 vnam2
192	CREATE vname [type]
192	DELETE vlist
195	DRAW vname [id chopt]
195	FIT x y ey func [chopt np par step pmin pmax errpar]
194	HFILL vname id
193	INPUT vname
192	LIST
192	OPERATIONS
194	VADD vnam1 vnam2 vnam3
194	VBIAS vnam1 bias vnam2
194	VDIVIDE vnam1 vnam2 vnam3
194	VMULTIPLY vnam1 vnam2 vnam3
194	VSCALE vnam1 scale vnam2
194	VSUBTRACT vnam1 vnam2 vnam3
195	PLOT vname [id chopt]
192	PRINT vname
193	READ vlist fname [format chopt match]
193	WRITE vlist [fname format chopt]
	ZEBRA
	DZ
199	SHOW name [number chopt]
199	SNAP [idiv chopt]
199	STORE [ixstor]
199	SURV name [number]
199	VERIFY [idiv chopt]
	FZ
198	FILE lun fname [lrecl chopt]
198	FRALPHA fname
198	FRFZ lun [chopt]
198	TOALPHA fname
198	TOFZ lun [chopt]
	RZ
197	CDIR [chpath chopt]
197	DDIR chdir
197	FILE lun fname [lrecl chopt]
198	FREE [chlock]
197	LDIR [chpath chopt]
197	LOCK [chlock]
197	MDIR chdir [nwkey chform chtags]
197	PURGE [keep]
198	STAT chpath

Bibliography

[1] Berezhnoi V., Brun R., Nikitin S., Petrovykh Y., and Sikolenko V..
CERN Program Library L210.
COMIS, Compilation and Interpretation System.
1988.

[2] Bock R., Brun R., Couet O., Nierhaus R., Cremel N., Vandoni C., and Zanarini P..
CERN Program Library Q120.
HIGZ - High level Interface to Graphics and ZEBRA.
1988.

[3] Brun R., Bruyant F., Maire M., McPherson A.C., and Zanarini P..
CERN Data Handling Division, DD/EE/84-1.
GEANT3.
1987.

[4] Brun R. and Zoll J..
CERN Program Library Q100.
ZEBRA - Data Structure Management System.
1989.

[5] Brun M., Brun R., and Rademakers F.
CMZ - Code Management system using Zebra.
Proceedings of Conference on Computing in High Energy Physics, Oxford April 1989,
1989.

[6] Brun R., Couet O., and Cremel N..
CERN Program Library Y251.
HPLOT User Guide - Version 5.
1988.

[7] Brun R. and Lienart D..
CERN Program Library Y250.
HBOOK User Guide - Version 4.
1988.

[8] Brun R. and Segal B..
CERN/DD report. To be published.
A distributed Physics Analysis workbench.
1989.

[9] Brun R. and Zanarini P..
CERN Program Library I202.
KUIP - Kit for a User Interface Package.
1988.

[10] Myers D.R..
CERN DD/US/111.
Guide to Computer Graphics at CERN.
1988.

[11] Eadie et al.
Statistical Methods in Experimental Physics.
North Holland, 1971.

[12] Klein H.J. and Zoll J..
CERN Program Library L400.
PATCHY Reference Manual.
1988.

[13] IBM Corporation.
DCF SCRIPT/VS Language Reference - Release 3.2, SH35-0070-05.
1988.

[14] IBM Corporation.
DCF SCRIPT/VS User's Guide - Release 3.2, S544-3191-00.
1988.

[15] IBM Corporation.
SGML Translator DCF Edition - User's Guide, SC34-5074-00.
1988.

[16] IBM Corporation.
Bookmaster Release 2.0 - User's Guide, SC34-5009-02.
1989.

[17] Adobe Systems Incorporated.
PostScript Language - Reference Manual.
Addison-Wesley Publishing Company, 1988.

[18] James F..
CERN Yellow Report 81-03 and DD Report DD/81/02.
Determining the Statistical Significance of Experimental Results.
1981.

[19] James F..
CERN Program Library D506 - Supplement.
MINUIT - Interpretation of the Errors on Parameters.
1988.

[20] James F. and Roos M..
CERN Program Library D506.
MINUIT - Function Minimization and Error Analysis.
1988.

[21] Segal B..
Internal DD report. To be published.
The TCPAW Package.
1989.

Index of entries

Special Characters

_ continuation character 15
: subarray separation character 15
//LUN1 81
//PAWC 81
$SIGMA 68
* comment character 14
| comment character 14

Numerics

1DHISTO 164
2D_PLOT 164
2DHISTO 165
3270G 133, 134

A

abbreviation 4, 9, 15, 52, 60
ABSCISSA 170
accents 124
active picture 107
ADD 167
addressing of vectors 66
alias 4, 35, 60, 63
ALIAS 175
alignment
  horizontal 110, 119
  vertical 110, 119
alphanumeric
  labels 37, 114
  menu mode 53
Apollo 7
APPLICATION 177
application SIGMA 68
ARC 152
array
  filling 69
  in SIGMA 69
ARROW 152
asterisk size (for functions) 112
ATITLE 159
attribute 79, 110
ATTRIBUTES 157
automatic
  naming of pictures 110
  storage of pictures 119
axis
  divisions 114
  labels
    font and precision 112
    size 112
  values
    font and precision 112
    size 112
AXIS 153

B

band 3, 79
BANX 165
BANY 165
bar
  chart 37, 111
  histogram
    offset 112
    width 112
basic operator in SIGMA 69
batch 4, 8, 31
binning for Ntuple plots 34
BINS 165
block (lego) plot 28
book histogram 3
boolean value in SIGMA 69
border 110
box
  around picture 111
  fill area
    colour 112
    type 112
  line width 112
BOX option 24
BOX 152
break 64
BREAK 178

C

CALL 144
case sensitivity 14
CDF Command Definition File 4
CDIR 197
change directory 81
character
  height 110
  key size 112
  shift 112
chisquare 6
client 139
CLOSE 145
CLR 149
CMS 7
colon as separation character in array 15
colours 116
COLUMNS 178
COMIS 6, 21, 24, 25, 65, 66, 91, 143
command
  abbreviation 4, 9, 15, 18
  definition file (CDF) 4
  mode 13, 52
  parameter 9
    list 52

command *(continued)*
parameter *(continued)*
mandatory 9
optional 9
search path 7
structure 9, 51
COMMAND 177
comment 14
and statistic size 112
font and precision 112
star 14
vertical bar 14
components
of PAW 4
of vector 59
CONTENTS
(GET_VECT) 170
(PUT_VECT) 171
continuation character 14
underscore 15
CONTOUR 164
contour plot 23, 28, 29
control operator in SIGMA 69
COPY
(HISTOGRAM) 162
(PICTURE) 107, 192
(VECTOT) 189
correlation 6
CPTIME KUIP system function 62
create
histogram 25
Ntuple 31
vector 65
CREATE
(HISTOGRAM) 164
(KUIP) 175
(NTUPLE) 183
(PICTURE) 188
(VECTOR) 192
cross-wires 111
CSELECT 187
current
directory 81
picture 106
cut 3, 35, 37, 88, 89
graphical 90
CUTS 89, 186
CZ 134

D

dash mode for lines 112
data
structure 79
data presentation
one-dimensional 27
two-dimensional 28, 38
date 118
and hour on pictures 111, 118
position 112
DATE KUIP system function 62
DDIR 197
DECNET 7, 133
default setting 4, 110, 111
DEFAULTS 180
DELETE
(HISTOGRAM) 161
(KUIP) 176
(PICTURE) 107, 188
(VECTOR) 192
delta function 71
DI3000 5
dialogue style 4, 52
DIFF 72, 168
directory 81
change 81
current 81
display 7
distance
x axis
to axis values 112
to labels 112
y axis
to axis values 112
to labels 112
DIVIDE 168
divisions 114
Domain 7
DRAW 195
driver 7
DST 3, 78, 82
DUMP 167
DZ 199

E

EDIT 174
editor 132
error 79
bars 111
ERRORS
(GET_VECT) 170
(GRAPHICS) 158
(PUT_VECT) 171
errors on fitted parameters 95
event 3, 30
exception handling 64
exclamation mark character
place-holder 9, 15, 53
exec 56
EXEC 179
EXIT 175
expression in SIGMA 62

F

FAREA 151
FBOX 152
FILE
(FORTRAN) 145
(FZ) 198

FILE *(continued)*
(HISTOGRAM) 162
(PICTURE) 188
(RZ) 197
file name
on pictures 111, 118
position 112
fill
area 116
colour index 110
interior style 110
style index 110, 120
histogram 3
vector 65
first page number 112
fit 3, 6, 19, 94
parameters on pictures 111, 118
values to be plotted 112
vector 67
FIT
(HISTOGRAM) 163
(VECTOR) 195
font
PostScript 124
text 124
fonts 126, 127, 128, 129, 130
FORTRAN 143—145
interpreter 24
FRALPHA 198
FREE 198
French accented character 124
FRFZ 198
FTP 136
FUN1 146
FUN2 23, 146
function 4, 21, 25, 63
fill area
colour 112
type 112
in SIGMA 70
line width 112
one-dimensional 21
two-dimensional 21, 23
FUNCTION 146—147
(GET_VECT) 170
FZ 198

G

GDDM 7
GDDM (IBM) 5
German umlauted character 124
GET_VECT 170
GKS 4, 5, 7, 12
workstation type 8
GKSGRAL
implementation 8
GL (Silicon Graphics) 5
global
section 81, 133, 138
global *(continued)*
title
font and precision 112
size 112
GLOBAL_SECT 167
GMR3D (Apollo) 5
GOTO macro
control 58
statement 56
GPR 7
GPR (Apollo) 5
GRAPH 67, 156
graphical
cut 90
menu mode 53
output 67
graphics
editor 132
modes 53
terminal 7
GRAPHICS 148—159
Greek 124
GRESET 167
grid 111
line type 112
GRID 159
GRPLOT 106

H

hardware characters 111
hatch style 13, 18, 27, 120, 121
HBOOK 5, 31, 78
batch 82
histogram file 32
Title 111
HBOOKN 31
HELP 7, 10, 174
HESSE 95
HFCNF 94
HFCNV 94
HFETCH 166
HFILL 194
HIGZ 5, 12, 79, 105, 106, 109, 110
HIO 166
HIST 67, 155
histogram 3, 17, 18, 25, 27, 78
bar option 17
creation 17
file 32
fill area
colour 112
type 112
hatch style 18
line width 112
maximum for scale 112
operations 27
presentation 116
title size 112

HISTOGRAM 160—172
history file 4, 64
HLIMIT 79
HMOVE 150
host 7
HOST_EDITOR 178
HPLOT 5, 78, 105, 109, 110, 158
HREAD 166
HRIN 166
HROUT 166
HSCRATCH 166
HSETPR 169

I

IBM 7
IBM 3192G graphics terminal 7
IDOPT 172
IF macro
  control 58
  statement 56
IGSET 110, 190
initialisation 8
INPUT 193
input histograms 26
integer or real divisions on axis 111
interactive mode 4
interrupt 64
ITX 154
IZIN 189
IZOUT 189
IZPICT 106, 190

K

KEY 158
KUIP 4, 51, 62, 79, 173—178
  STYLE 53
  system functions 62
    CPTIME 62
    DATE 62
    LEN 62
    LOWER 62
    NUMVEC 62
    RTIME 62
    SIGMA 62
    SUBSTRING 62
    TIME 62
    UPPER 62
    VDIM 62
    VEXIST 62
    VLEN 62
  vector 66

L

label 22, 114
  in KUIP macro 56
  offset 110
  size 110
  text justification 114
LABELS 154
LAST 64, 174
LDIR 197
LEGO 164
  plot 38
LEN KUIP system function 62
length of
  basic dashed segment 112
  X axis 112
  Y axis 112
library functions in SIGMA 77
limits on fitted parameters 95
line
  type 110, 123
  width 110, 116
linear scale 111
LINE 151
LIST
  (HISTOGRAM) 162
  (KUIP) 176
  (MACRO) 179
  (NTUPLE) 187
  (PICTURE) 188
  (VECTOR) 192
list histograms 26
LOCATE 149
LOCK 197
logarithmic scale 111
logical operator in SIGMA 69
loop 19, 22, 36
  goto 19
  if 19
  label 19
LOOP
  (FORTRAN) 145
  (NTUPLE) 183
LOWER KUIP system function 62
LS 72
LVMAX 73
LVMIN 73

M

macro 3, 4, 56
  GOTO 56
  IF 56
  label 56
  ON ERROR GOTO 56
  parameter 22, 57
  READ 56
  RETURN 56
  statements 56
  variable 58, 63
MACRO 179—180
mandatory parameter 9
MANUAL 174
MANY_PLOTS 162
marker 16
  scale factor 110
  type 110, 122

mask 3, 30, 36, 88, 89
  loop 36
  mask 36
MASK 187
MAX 73
MAXIMUM 171
MAXV 73
MDIR 197
menu 4, 9, 53
  alphanumeric 53
  graphical 53
    panel 53
    pull-down 53
  mode 53
MERGE
  (NTUPLE) 183
  (PICTURE) 189
merge pictures 42
MESSAGE 175
metafile 4, 12, 24, 106
METAFILE 149
MIGRAD 95
MIN 73
minimisation 6, 94
MINIMUM 171
MINUIT 6, 94, 102
MINV 73
MIPS 3
MISC 149
mode Z 107
MODIFY 132, 189
multiple dialogue 52
MULTIPLY 167

N

national character 124
NCO 73
NETWORK 181
NEXT 149
Nntuple 3
NORMALIZE_FACTOR 171
Ntuple 33, 35, 78, 87, 88
  create 31, 33
  cut 88
  examples 30
  mask 88
  plot 33
  print 33
  read 34
    text file 33
  weight 88
  write Ntuple RZ file 33
NTUPLE 182—187
NULL 159
number of
  divisions for
    X axis 112
    Y axis 112
  passes for software characters 112
NUMVEC KUIP system function 62

O

ON ERROR GOTO macro statement 56
one-dimensional
  data presentation 27
  function 21
online help 4
operation on vectors 14, 66
OPERATIONS
  (HISTOGRAM) 167
  (VECTOR) 194
operator in SIGMA 69
OPTION 110, 111, 149
optional parameter 9
ORDER 74
OS9 139
  module 81, 133
OSI 133
output histogram 25
OUTPUT_LP 167
overflow on RZ file 32

P

page
  format 111
  number 111
  number size 112
PAMAIN 79
panel 53
  menu 9, 55
PANEL 176
paper orientation 111
PARAM 169
parameter 4
  errors (fit) 95
  list 52
path 7
PAVE 156
PAW
  access 7
  command structure 51
  entities 12
  examples 13
  initialisation 8
  object 12
  server 134, 139
  structure 4
  terminology 3
PAWC common 79
PAWINT 79
PAWLOGON 7, 8
PG terminal type 7
picture 4, 106, 111
  fill area
    colour 112
    type 112
  line width 112

PICTURE 107, 132, 188—190
PIE 67, 154
pie chart 43
place-holder
  exclamation mark character 9, 15
PLINE 151
PLOT 88
  (FUNCTION) 147
  (HISTOGRAM) 161
  (NTUPLE) 88, 185
  (PICTURE) 189
  (VECTOR) 195
  commands 12
plotting 27
  E option 27
  P and L options 27
PMARKER 151
POINTS 147
polyline colour index 110
polymarker 27
  colour index 110
PostScript 4, 12, 24, 124
  fonts 124
prefix SIGMA 68
presenter 138, 139
PRIMITIVES 151
PRINT
  (HISTOGRAM) 166
  (NTUPLE) 183
  (VECTOR) 192
  commands 12
PROD 74
PROFILE 164
Program Library
  NEW 7
  OLD 7
  PRO 7
PROJECT
  (HISTOGRAM) 162
  (NTUPLE) 184
projection 3, 79
PROMPT 178
PROX 165
PROY 165
PTO (Please Turn Over) 111
pull-down menu 9, 53, 54
PURGE 197
PUT_VECT 171

Q

QUAD 75
QUIT 175

R

READ
  (NTUPLE) 184
  (VECTOR) 193
READ macro statement 56
real time 81
REBIN 170
recalling previous commands 64
RECORDING 178
RECURSION 180
remote
  access 85, 133
  file 137
  login 137, 139
  shell 137, 139
RENAME 107, 189
RESET 168
RETURN macro statement 56
REWIND 145
RLOGIN 85, 137, 139, 181
ROOT 177
RSHELL 137, 139, 181
RTIME KUIP system function 62
RZ 197
  file 32

S

SCALE_FACTOR_2D 171
scan 35
SCAN 186
scatter plot 26
  (BOX option) 28, 29
  and table character size 112
  table 78
SCHH 157
SCRATCH 188
selection 35
  function 88, 89, 91
SELNT 151
server 139
SET 110, 111, 149, 171
SET NDVX 38
SET NDVY 38
SET_SHOW 176
SFACI 157
SFAIS 157
SFASI 157
SHELL 175
SHOW 199
SIGMA 6, 39, 40, 65, 66, 68
  $SIGMA 68
  access 68
  APPLication SIGMA 68
  array 69
    filling 69
    structure 69
  basic operator 69
  boolean value 69
  control operator 69
  expression 62
  function 70
  library functions 77
  logical operator 69

SIGMA *(continued)*
prefix SIGMA 68
system function in KUIP 62
vector 69
SIZE 150
slice 3
slides 46
SLIX 165
SLIY 165
SLN 157
SLWSC 157
SMK 157
SMOOTH 168
SNAP 199
software
characters 111, 124
fonts 124
special symbols 11, 124
SPLCI 157
SPLINE 168
SPMCI 157
STAT 198
statistic
analysis 6
parameters on pictures 111, 118
values to be plotted 112
STORE 199
structure of PAW 4
STXCI 157
STXFP 157
STYLE 176
alphanumeric mode 53
AN 53
G 53, 54
GP 53, 55
subarray uses colon as separation character 15
subrange 29
subscript 124
SUBSTRING KUIP system function 62
SUBTRACT 167
SUMV 76
superscript 124
SURFACE 164
surface plot 23, 28, 29
SURV 199
SVP 150
SWITCH 190
Z 107
SWN 150
symbols 11
SYMBOLS 158
system function 62, 63

T

TCP/IP 7, 85, 133, 139
TCPAW 133, 134
Tektronix 7
TELNET 133, 136
TELNETG 133, 134
text
(and title) font and precision 112
angle 110
data 12
font 110, 119, 124
precision 110, 119
width 110
TEXT 154
tick marks 22, 115
size 110
TICKS 158
TIME KUIP system function 62
TIMING 178
title font and precision 112
TITLE_GLOBAL 166
tn3270 7
TOALPHA 198
TOFZ 198
TRACE 179
TRANSLATION 176
two-dimensional
data presentation 28, 38
function 21

U

umlauts 124
UNITS 175
unix 7
UPPER KUIP system function 62
USAGE 174
USAGE command 11
user
interface 13
routine 25
title 111
UWFUNC 91, 184

V

VADD 194
VAX 7, 133
VAX/VMS 138
Vaxstation 7
VBIAS 194
VDIM KUIP system function 62
VDIVIDE 194
vector 3, 14, 17, 65
address 66
arithmetic 66, 69
component 59
create 65
draw 17
fill 65
GET_VECT 170, 171
in SIGMA 69
operations 18, 69
plot 17
print 19

VECTOR 191—195
VERIFY 199
version 7
VEXIST KUIP system function 62
VIEWING 150
VLEN KUIP system function 62
VLOCATE 149
VM-CMS 7
VMAX 76
VMIN 76
VMS 7, 138
VMULTIPLY 194
VSCALE 194
VSUBTRACT 194
VSUM 76

W

WAIT 175
weight 88
weighting factor 89
wildcard 18
workstation 3, 7
  type 8
WRITE 193

X

X axis
  colour 112
  tick marks length 112
X margin
  left 112
  right 112
X space between windows
X windows 5, 7
X11 7

Y

Y axis
  colour 112
  tick marks length 112
Y margin
  low 112
  up 112
Y position of
  global title 112
  histogram title 112
  page number 112
Y space between windows

Z

ZEBRA 6, 79, 133, 196—199
  FRALFA 12
  FZ file 12
  RZ file 12, 136
  TOALFA 12
ZFL (option) 107
ZFL1 (option) 107
ZFTP 133, 136
zone 16
ZONE 150
ZOOM 162