PAW tutorial

Physics Analysis Workstation

Olivier Couet

Application Software Group

Computing and Networks Division

CERN Geneva, Switzerland


```
Olivier Couet
CERN-CN Division
CH-1211 Geneva 23
Switzerland
Tel.  +41 22 767 4886
Fax.  +41 22 767 7155
Bitnet:  COUET@CERNVM.CERN.CH
Internet:  couet@hphigz.cern.ch
```


Informations about **PAW** can be found in:

`http://asdwww.cern.ch/pl/paw/index.html`

Table of contents

Basic Principles

- **PAW** (Physics Analysis Workstation) is an INTERACTIVE SYSTEM designed for data analysis and data presentation.
- **PAW** provides a set of COMMANDS acting on specific objects. The main objects or data type are: VECTORS, HISTOGRAMS, and NTUPLES. The aim of the tutorial is to illustrate with examples how to work with these objects in a data analysis process.
- The **PAW** commands are organized into a TREE structure.
- The general structure of the tree is:

```
OBJECT/ACTION
```

Example:

```
NTUPLE/PLOT
HISTOGRAM/PROJECT
VECTOR/DRAW
```

- The usual user interface is a "command line interface": commands are typed on keyboard and executed after `<CR>`. Commands parameters are separated with the blank character.
- Commands editing and retrieving is also possible. It is controlled via the command `RECALL_STYLE` (p 16) (ksh, DCL etc ..).
- Commands can be grouped into "Macros". Macros are files with the extension `.kumac` containing several commands with eventually construction like "do loop", "if endif", etc .. . To execute a macro it is enough to type `EXEC macroname` (p 21) if the macro is in the file `macroname.kumac`.
- It is possible to have online help on commands with the command `HELP` (p 2) which gives the full description of a command, and with the command `USAGE` (p 2) which gives the command syntax.
- A printable version of the reference manual can be obtain with the command `MANUAL` (p 2) .
- **PAW**++ provides a Motif based User Interface to **PAW**.
- **PAW** and **PAW**++ have the SAME basic functionality.

Starting PAW Tutorial

```
MACRO PAWLOGON
Mess '***********************************************************'
Mess '*                                                         *'
Mess '*                  Starting PAW tutorial                  *'
Mess '*                                                         *'
Mess '***********************************************************'
```

This tutorial present the basic principles of **PAW** using a set of examples (**PAW** macros). It try to cover the most frequently used basic functions of **PAW**. Some of the examples are extracted from the **PAW** manual (Edition - October 1993).

All the references (page numbers) point to the PAW reference manual given with this tutorial.

In the examples, the highlighted points are written in UPPERCASE with a reference in the left margin. This reference point to a comment after the listing of the macro.

If the example produce a graphics output, it is given on the page behind the example. Under each figure, the name of the corresponding macro is given.

This example shows what could be the `MACRO PAWLOGON` (in the file `PAWLOGON.KUMAC`) which is automatically executed (if it exists) at the beginning of each **PAW** session.

In this tutorial we assume that the macro `ALLDEF` is executed before each example.

alldef.kumac

```
MACRO ALLDEF
Size 18 24
Next
Set * ; Option *
Size 18 24
Histogram/Delete * ; Vector/Delete *
Title_global ' '
Title_global ' ' U
Option NBOX
Option NGRI
Set HWID 1
Set FWID 1
Set BWID 1
Set PWID 1
Set LWID 1
Set CSIZ 0.25
Set VSIZ 0.25
Set TSIZ 0.32
Set XMGL 1.2
Set XMGR 1.2
Set YMGU 0.5
Set YMGL 1.5
Set GSIZ 0.1
Set YHTI 0.7
Set KSIZ 0.15
Set MTYP 1
Zone 1 1
Next
Return
```

Starting with vectors

```
❻     * Starting with vectors
❻     VECTOR/CREATE VECT1(10)  | Create a vector of length 10
❶     VECTOR/INPUT VECT1 10 8 6 4 2 3 5 7 9 11
❼ ❶   VECTOR/CRE VX(20) R 1. 2. 3. 4. 5. 6. 7. 8. 9. _
      10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20.
      v/cr vy(20) r 1.1 3.2 5.3 7.4 7.5 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
❽     ZON 1 2
❷     VECTOR/DRAW VECT1
❹ ❷   GRAPH 20 VX VY
❹ ❺   graph 20 VX VY *
❺     gra 20 VX VY C
❸     VECT/DEL *
```

❶ Here we see two ways to fill a vector:

(a) V/CREATE (p 41) : create a vector and, optionally, fill it.

(b) V/INPUT (p 42) : allows to fill an existing vector.

We will see other ways later.

❷ Graphic representations of vectors : VECTOR/DRAW (p 44) and GRAPH (p 119) .

❸ VECT/DELETE allows to delete a vector from memory. "*" means delete all vectors in memory. Very often in **PAW** a command acting on a specific kind of objects (vectors, histogram, pictures) can access the complete object set with "*".

<u>Note also:</u>

❹ The **PAW** commands are case insensitive.

❺ Command abbreviations are permitted.

❻ The character "*" and "|" are used for comments.

❼ The character "_" is used to indicate a continuation line.

❽ The command ZONE (p 102) subdivides the graphical area.

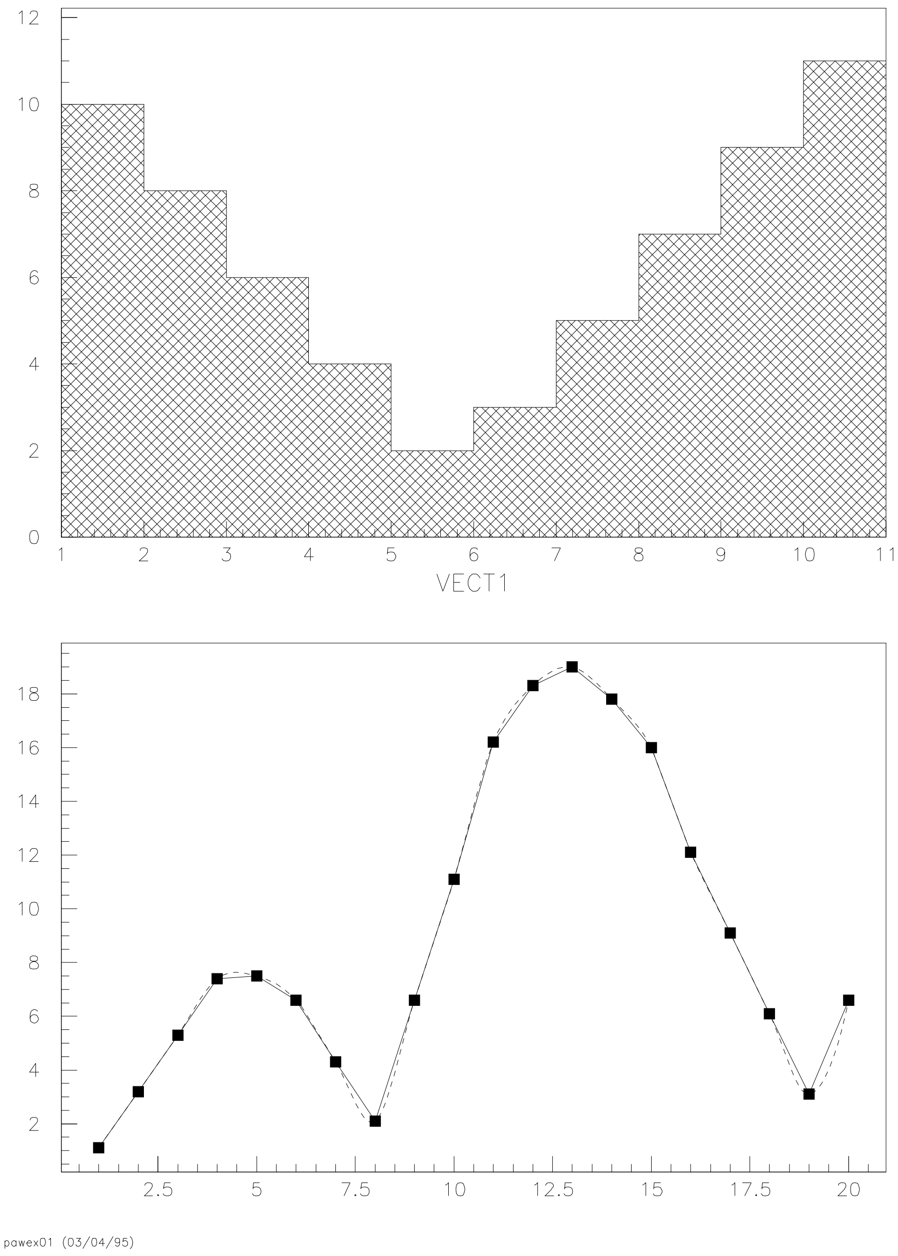


Figure 1: Exec pawex01.kumac

Some more vector commands

```
    vector/create VECT(10,3) R _
    1. 2. 3. 4. 5. 6. 7. 8. 9. 10. _
    9.1 8.1 7.1 6.1 5.1 4.1 3.1 2.1 1.1 0.1 _
    6.2 4.2 3.2 2.2 1.2 1.2 2.2 3.2 4.2 5.2
    vector/create VECT1(10) R _
    1.1 2.2 3.3 4.4 5.5 6.6 5.5 4.4 3.3 2.2
❺   SET HTYP 244 ; VE/DR VECT(1:10,3)
❶❷  VECTOR/DRAW VECT(1:10,3) ! SC
❹❻  VECTOR/DRAW VECT1 ! L*S
    ve/list
❸   VE/WRITE VECT 'vector.data' '(3(10f5.0,/))'
```

❶ A vector can have up to three dimensions. Dimensions which are not specified are taken as 1, for example `VEC(10)` → `VEC(10,1,1)` and `VEC` → `VEC(1,1,1)`.

❷ It is possible to access a subrange of a vector, for example: `V(2:3)`, `V(3:)` or `V(:5)`.

❸ The command `VECT/WRITE` (p 44) creates the file `vector.data` as follows:

```
1.   2.   3.   4.   5.   6.   7.   8.   9.  10.
9.   8.   7.   6.   5.   4.   3.   2.   1.   0.
6.   4.   3.   2.   1.   1.   2.   3.   4.   5.
```

<u>Note also:</u>

❹ The character "!" means default value of a parameter.

❺ It is possible to have several commands, separated with ";", on the same line.

❻ Many commands have a parameter which defines options. Such parameters (often called `CHOPT` or `OPTION`) have the attribute "Option" (see the help). Each option is a character string. It is possible to mix several options, e.g. "SC" or "L*S".

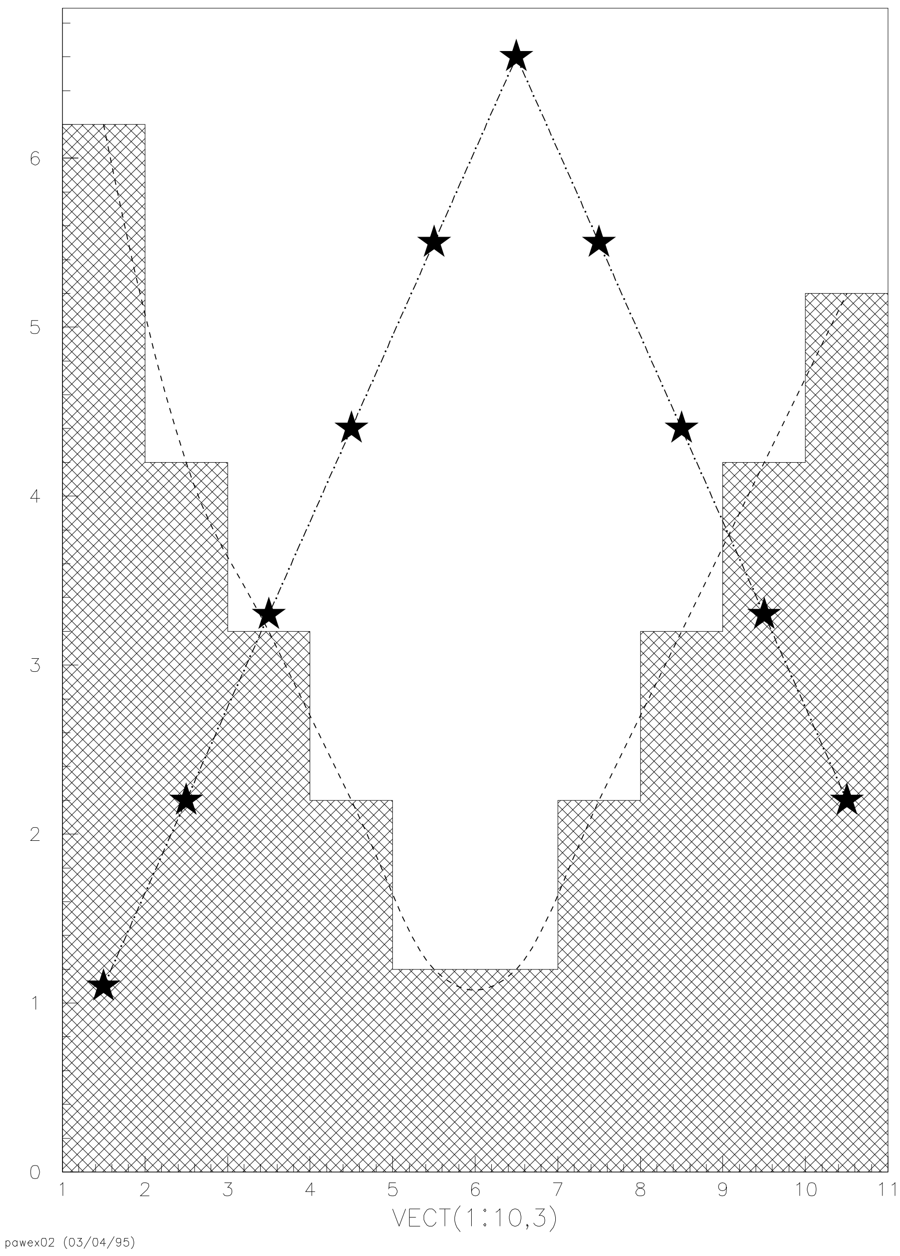


Figure 2: Exec pawex02.kumac

Some possible data representations with VECTOR/DRAW

```
    zone 2 3
    ve/create v(10) R 5 1 3 2 4 1 3 1 8 6
❶   SET HTYP 244
    ve/draw v
    ve/draw v ! b
    ve/draw v ! l
❸   VE/DRAW V CHOPT=L*
    ve/draw v ! bl*
❷   SET MTYP 21
    ve/draw v ! e
    ve/de V
❹   RETURN
```

❶ The command SET (p 94) defines some high level graphics attributes for commands like VECT/DRAW or HIST/PLOT. Here the HTYP (Histogram hatch TYPe) is defined.

❷ Set the marker type.

❸ By default the parameters of a command are positional but it is possible to assign values by name, i.e. PARAMETER=value. For example we have here CHOPT=L*. In this case the intermediate "!" can be suppressed.

Note also:

❹ The statement RETURN is not mandatory in a macro except if there are several macros in the same file. In this case, a macro within a file can be executed by: EXEC FILENAME#MACRONAME (p 21) .

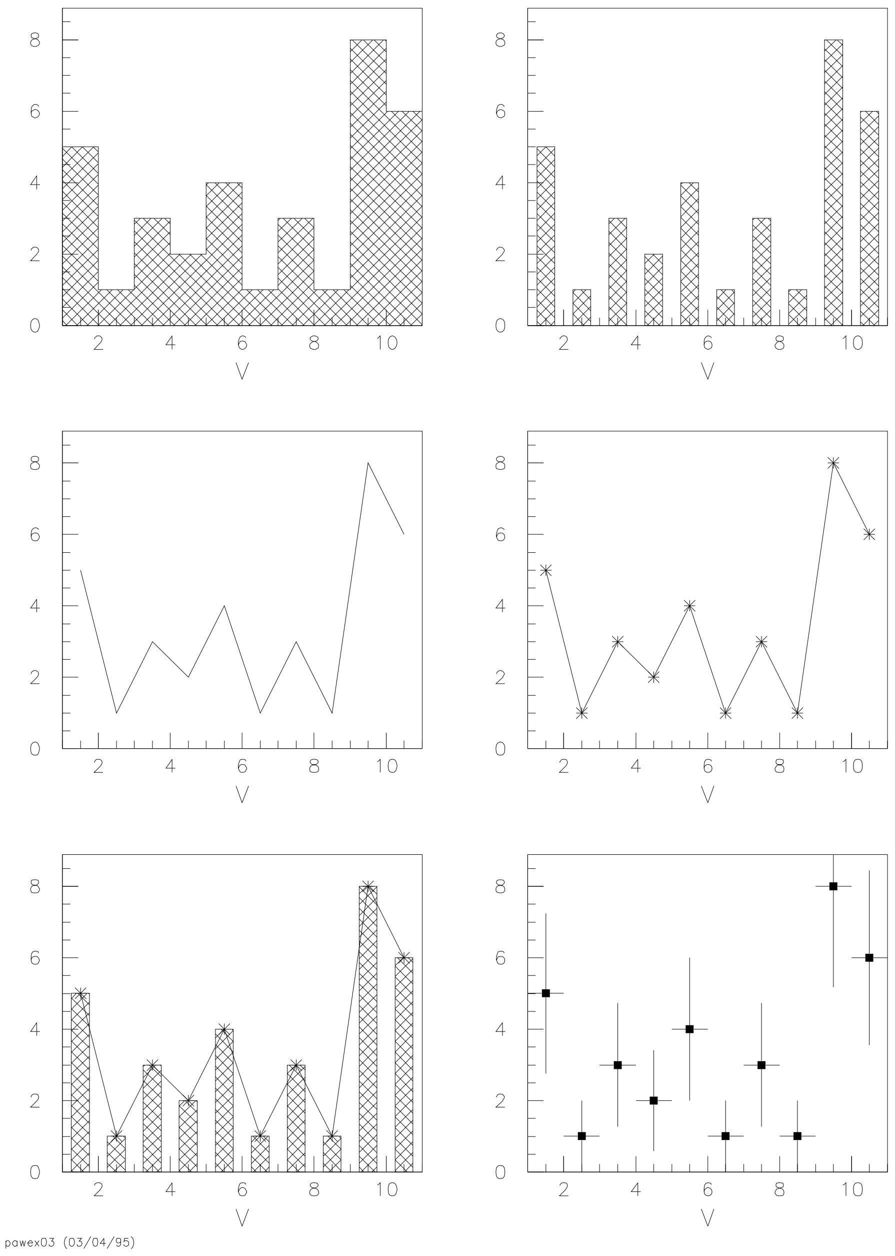


Figure 3: Exec pawex03.kumac

Functionality of VECT/DRAW, VECT/PLOT, VECT/HFILL and PUT/CONT

```
   zone 2 2
   ve/create VECT1(10) R 1 2 3 4 5 5 4 3 2 1
   *
   ve/draw VECT1
❶  VE/PLOT VECT1
   *
❹  CREATE/1DHISTO 100 'test vector/hfill' 5 1. 6.
   max 100 2.5
❷  VE/HFILL VECT1 100
   histo/plot 100 b
   hi/de 100
   *
   create/1dhisto 100 'test put/contents' 10 1. 11.
❺  MAX 100 5.5
❺  MIN 100 0.5
❸  PUT/CONTENTS 100 VECT1
   histo/plot 100
```

❶ `VECT/PLOT` (p 45) draws the statistic of the given vector.

❷ `VECT/HFILL` (p 44) fills an existing histogram (create with `1DHIST`) with the values taken from a vector. Note that the command `VECTOR/PLOT` can automatically book an histogram and fill it with the vector content.

❸ `PUT/CONT` (p 72) replaces the content of an histogram with the values of a vector.

Note also:

❹ Histograms are **HBOOK** objects. They can be created, like here, interactively in **PAW** or in a batch **HBOOK** program.They can be stored in direct access files (we will see examples later).

❺ `MIN` and `MAX` (p 74) define the minimum and maximum of an histogram. By default they are computed automatically.

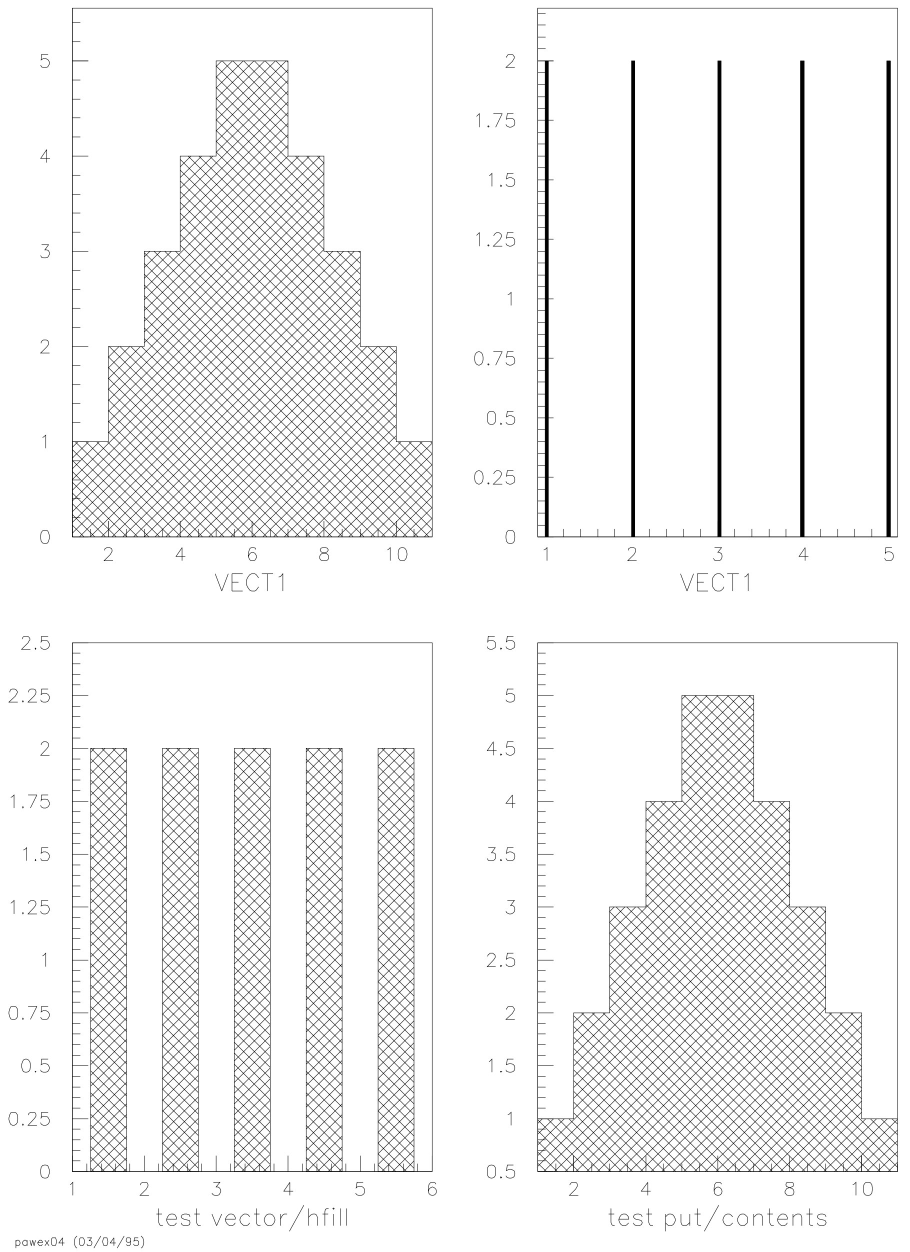


Figure 4: Exec pawex04.kumac

Vector operations

```
zone 1 2
ve/create V1(10) R 1 2 3 4 5 5 4 3 2 1
vector/operations/vscale V1 0.5   V12
VE/OP/VSCALE             V1 0.25  V14
ve/dr V1
ve/dr V12 ! S
ve/dr V14 ! S
VSUB                      V1 V14 V14M
ve/dr V1
set htyp 344
ve/dr V14M ! S
set htyp 144
ve/dr V12  ! S
```

❶ Some simple operations are possible on vectors (p 48) .

```
VBIAS     : Y(i) = a + X(i)
VSCALE    : Y(i) = a * X(i)
VADD      : Z(i) = X(i) + Y(i)
VMULTIPLY : Z(i) = Z(i) * Y(i)
VSUBSTRACT: Z(i) = X(i) - Y(i)
VDIVIDE   : Z(i) = X(i) / Y(i)
```

In these operations the resulting vectors are created automatically. Note that for operations with mathematical functions like SQRT or trigonometric functions etc... , **SIGMA** must be used (we will see examples later).

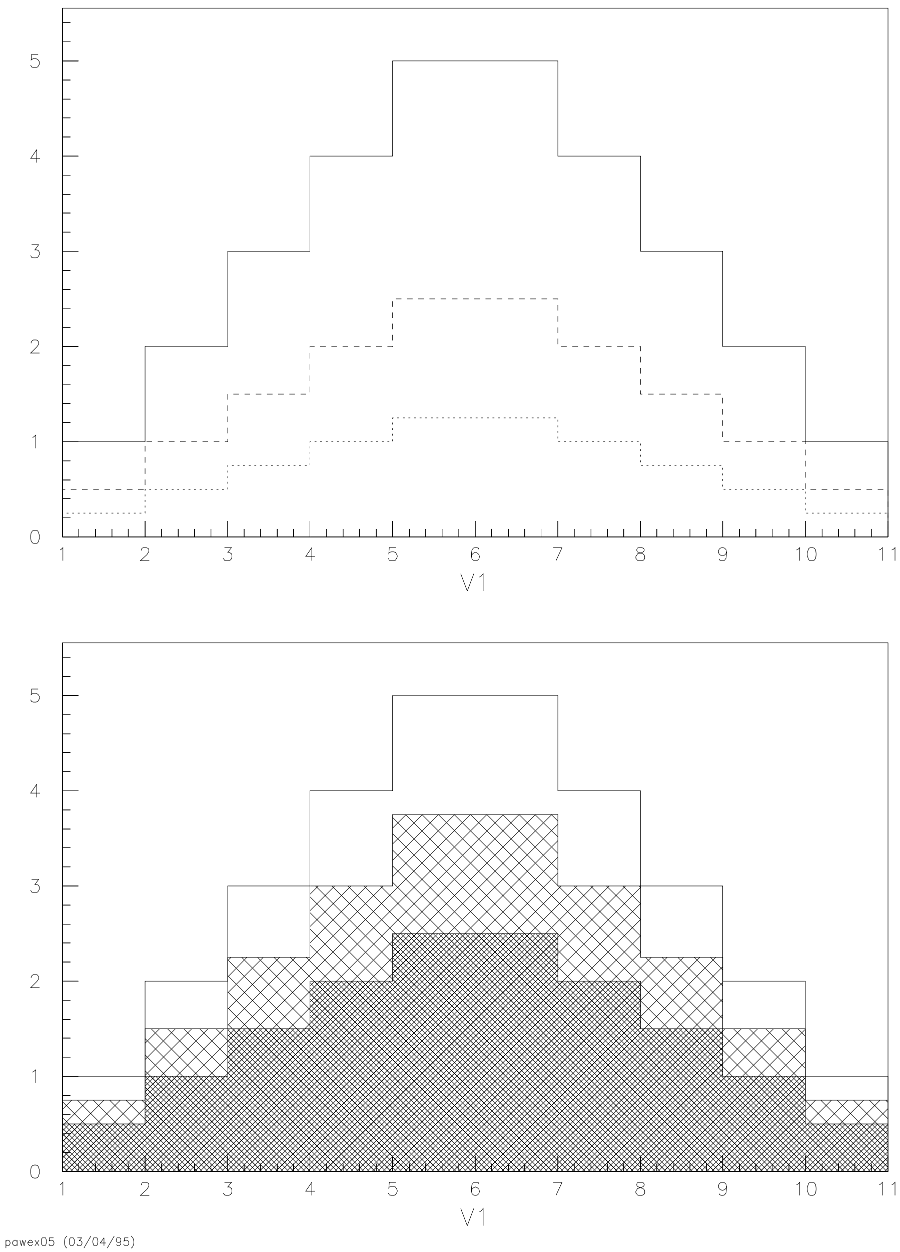


Figure 5: Exec pawex05.kumac

Simple macro, with a loop and a VECTOR fit

```
      ve/create VECT(10,3)
❶     VE/READ VECT 'vector.data'
      *
      ve/print VECT(1:10,3)
      vbias vect(1:10,1) 0.5 vect(1:10,1)
      zon 1 2
      *
❹     DO IP = 2,3
        ve/draw vect(1:10,[ip])
❷❸      ORDER = [IP] - 1
❺       VECT/FIT VECT(1:10,1) VECT(1:10,[IP]) ! P[order] WS
❹     ENDDO
      ve/delete VECT
```

❶ The file `vector.data` previously created is read again in this example via the command `VECT/READ` (p 43) . Note that it is not necessary to specify the format.

❷ This example shows the usage of variables in the macros (`IP`). The content of a variable can be accessed via:

`[variable]`

Note that the name of a variable in not case sensitive.

❸ Simple computations on variables are possible, like `i=[i]+1` or `a=[b]+2`. However it is not possible to do complex operations on variables. For this kind of computation vectors and **SIGMA** (or **COMIS**) must be used.

❹ Some controls statements are available in macros (see the complete list in the next example).

❺ It is possible to fit the vectors with functions (p 46) . Here the function used for the fit is a polynome. The fitting mechanisms are very complete in **PAW** and simple to use. All the details useful to use the commands `HIST/FIT` and `VECT/FIT` are given in the section *Fitting with PAW/HBOOK/MINUIT* of the chapter *HBOOK* in the **PAW** manual.

Simple macro, with a loop and a VECTOR fit

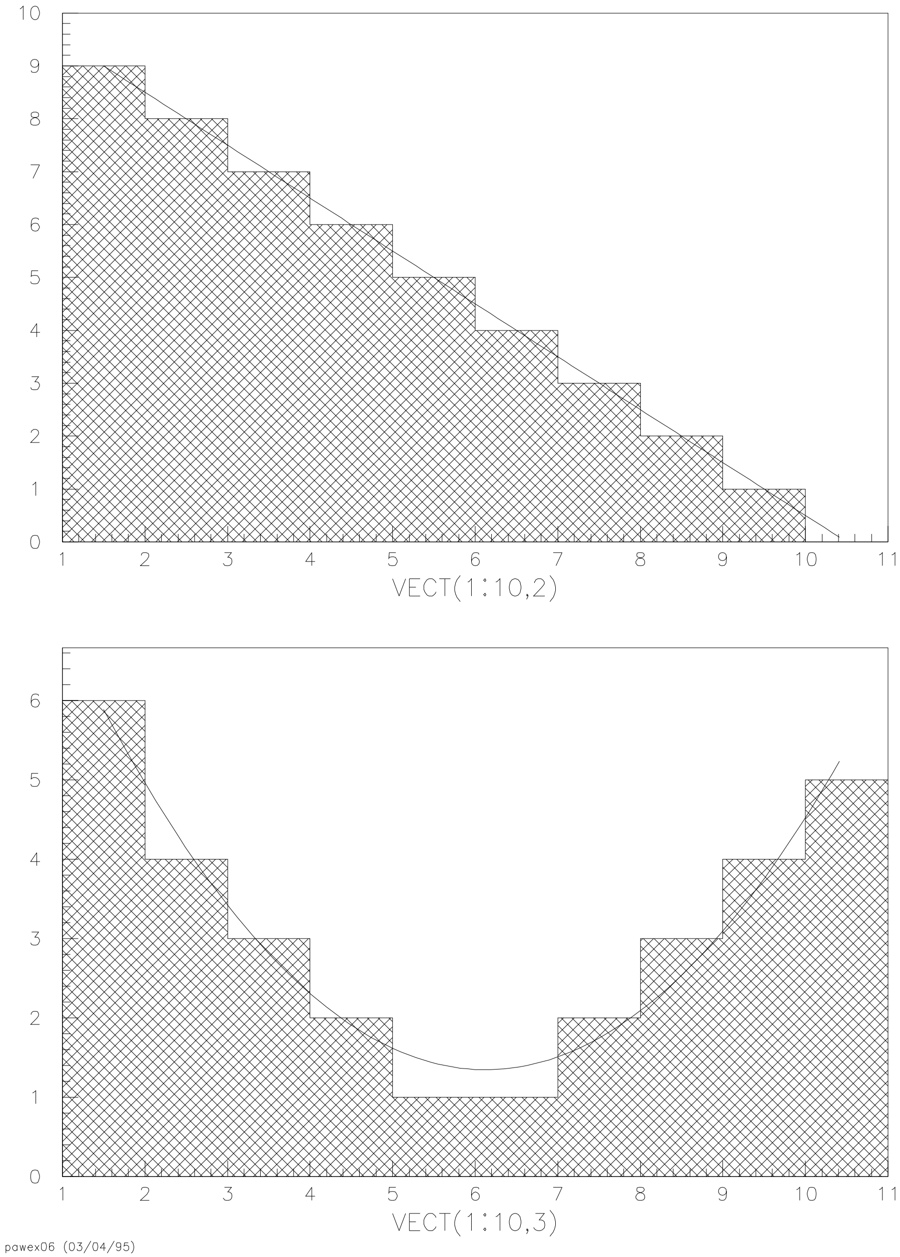


Figure 6: Exec pawex06.kumac

There are several constructs available for controlling the flow execution, which include conditional statement blocks, several looping constructs and variable assignation. Some example are given in this tutorial. For a complete description, see the section **Macros** in the chapter *The KUIP interface* in the **PAW** reference manual.

Macro Statements

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

Conditional statement

```
MACRO PAWEX06A
  A = 10
  NN = 1.5
  TOT = [A]+[NN]
  IF [TOT] > 11 THEN
    MESSAGE Sum of [A] and [NN] is [TOT]
    AOK = correct
  ELSE
    AOK = wrong
  ENDIF
  MESSAGE KUIP arithmetic is [AOK].
RETURN
```

```
PAW > exe pawex06a
Sum of 10 and 1.5 is 11.5
KUIP arithmetic is correct.
```


Unassigned variables cannot be substituted by their values.

```
MACRO PAWEX06B
   A = 10
   NN = 1.5
   TOT = [A]+[XX]
   MESSAGE Result of sum is [TOT]
RETURN
```

```
PAW > exe pawex06b
Result of sum is 10+[XX]
```

Example for CASE labels with wildcards

```
MACRO PAWEX06C
CASE [1] IN
(*.f, *.for) TYPE = FORTRAN
(*.c)        TYPE = C
(*.p)        TYPE = PASCAL
(*.tex)      TYPE = LaTex
(*)          TYPE = UNKNOWN
ENDCASE
MESSAGE [1] is a [TYPE] file.
RETURN
```

```
PAW > exe pawex06c paw.tex
paw.tex is a LaTex file.
```

Fit the function sin between 0 and 2π

```
❹    APPLICATION SIGMA
❹       alpha=array(100,0#2*PI)
❹       sina=sin(alpha)+rndm(alpha)*0.1
❹       err=array(100,0.1#0.1)
❹    EXIT
     zone 2 2
❶    V/FIT ALPHA(1:50) SINA(1:50) ERR(1:50) G
❶    V/FIT ALPHA SINA ERR P3
❶    V/FIT ALPHA SINA ERR P5
     v/create par(1) r 10.
❷    V/FIT ALPHA SINA ERR SINFIT.F ! 1 PAR
❸    V/PRI PAR
```

❶ In this macro two different types of predefined fits are used: Gaussian, Polynomial. As we will see later, the histograms fitting command HISTO/FIT has exactly the same syntax except that the 3 vectors are replaced by an unique parameter: the histogram identifier. On histograms some other minimization mechanisms are available via the commands SPLINE, SMOOTH, etc.. .

❷ It is also possible to defined specific functions. Here the function SINFIT is defined as follow:

The function SINFIT

```
function sinfit(x)
common /pawpar/ par(1)
sinfit=par(1)*sin(x)
end
```

❸ This VECT/PRI shows that now PAR(1) is close to 1.

```
PAR(1) = 0.994221
```

❹ Vector initialization with **SIGMA**. We will see other **SIGMA** examples later.

❺ Try to modify the macro and the COMIS program sinfit.f to have a fit with two parameters in order to improve the quality of the fit.

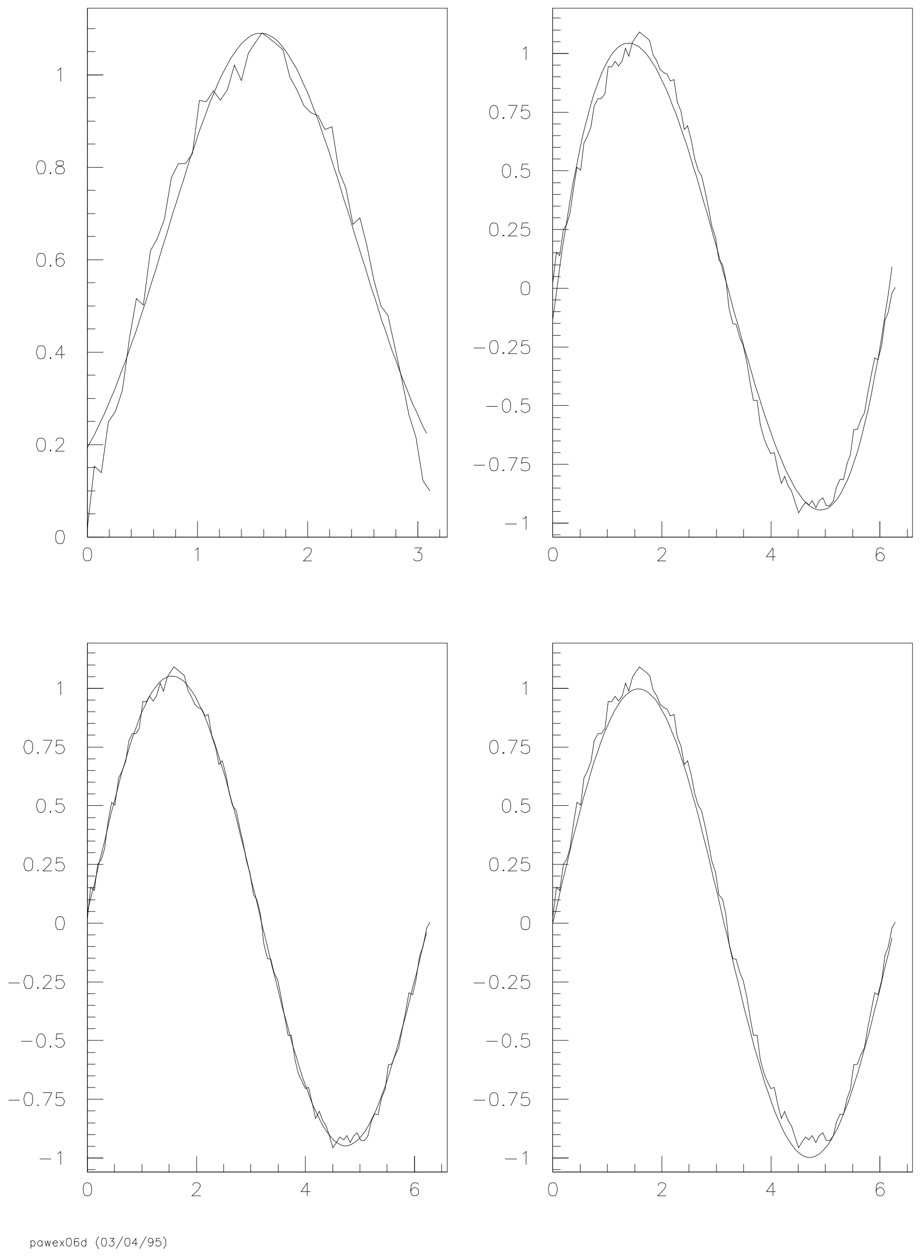


Figure 7: Exec pawex06d.kumac

Output of the Gaussian fit

```
 **********************************************
 *                                            *
 * Function minimization by SUBROUTINE HFITV  *
 * Variable-metric method                     *
 * ID =          0  CHOPT =                   *
 *                                            *
 **********************************************
Convergence when estimated distance to minimum (EDM) .LT.   .10E-03

FCN=   2221.676     FROM MIGRAD    STATUS=CONVERGED    239 CALLS      240 TOTAL

                    EDM=   .85E-05    STRATEGY= 1      ERROR MATRIX ACCURATE

 EXT PARAMETER                                 STEP         FIRST
 NO.   NAME        VALUE          ERROR         SIZE      DERIVATIVE
  1      P1        1.1316        .24808E-01    .64412E-03    .15289
  2      P2        1.5419        .21417E-01    .62018E-03    .42301E-01
  3      P3       -.76813        .17032E-01    .43531E-03   -.25527
```

Output of the Polynomial fit (P3)

```
CHISQUARE =  .2290E+02  NPFIT =  100
 **********************************************
 *                                            *
 * Function minimization by SUBROUTINE HFITV  *
 * Variable-metric method                     *
 * ID =          0  CHOPT =                   *
 *                                            *
 **********************************************
Convergence when estimated distance to minimum (EDM) .LT.   .10E-03

FCN=   49.31862     FROM MIGRAD    STATUS=FAILED       90 CALLS       91 TOTAL
                    EDM=   .79E-01  STRATEGY=1  ERROR MATRIX UNCERTAINTY= 70.2%

 EXT PARAMETER                APPROXIMATE       STEP         FIRST
 NO.   NAME        VALUE          ERROR         SIZE      DERIVATIVE
  1      P1       -.13523        .34965E-02    .00000E+00    5.6896
  2      P2        1.8729        .53793E-02    .00000E+00   -6.8643
  3      P3       -.86391        .32623E-03    .00000E+00    94.054
  4      P4        .91424E-01    .23105E-03    .00000E+00    6.6564

CHISQUARE =  .5137E+00  NPFIT =  100
```


Output of the Polynomial fit (P5)

```
 **********************************************
 *                                            *
 * Function minimization by SUBROUTINE HFITV  *
 * Variable-metric method                     *
 * ID =          0  CHOPT =                   *
 *                                            *
 **********************************************
Convergence when estimated distance to minimum (EDM) .LT.   .10E-03

FCN=   7.164283     FROM MIGRAD    STATUS=FAILED       240 CALLS      241 TOTAL
                    EDM=   .19E+01    STRATEGY= 1      ERR MATRIX NOT POS-DEF

 EXT PARAMETER                APPROXIMATE       STEP         FIRST
 NO.   NAME        VALUE          ERROR         SIZE      DERIVATIVE
  1      P1        .46785E-01    .20704E-03    .68172E-07    32.993
  2      P2        .93224        .10038E-02    .74579E-06   -551.05
  3      P3        .20962        .33827E-03    .16770E-06   -3073.1
  4      P4       -.36899        .32674E-03    .29519E-06   -1084.4
  5      P5        .82836E-01    .19712E-04    .66269E-07    821.80
  6      P6       -.52834E-02    .12561E-05    .42267E-08   -5204.8

CHISQUARE =  .7622E-01  NPFIT =  100
```

Output of the "**COMIS**" fit

```
 **********************************************
 *                                            *
 * Function minimization by SUBROUTINE HFITV  *
 * Variable-metric method                     *
 * ID =          0  CHOPT =                   *
 *                                            *
 **********************************************
Convergence when estimated distance to minimum (EDM) .LT.   .10E-03

FCN=   32.13273     FROM MIGRAD    STATUS=CONVERGED     21 CALLS       22 TOTAL
                    EDM=   .92E-05    STRATEGY= 1      ERROR MATRIX ACCURATE

 EXT PARAMETER                                 STEP         FIRST
 NO.   NAME        VALUE          ERROR         SIZE      DERIVATIVE
  1      P1        .99811        .13752E-01    .51510E-04   -.31172

CHISQUARE =  .3246E+00  NPFIT =  100
```

<* VECTOR/READ VLIST FNAME [FORMAT OPT MATCH]>

```
❶    V/READ X,Y,Z match.dat 6x,3(F4.1) ! /Data/
     v/draw X
     v/draw Y ! S
     v/draw Z ! S
```

match.dat

```
❷    Title: File used for tests of the MATCH parameter in V/READ
     Data : 1.0 2.0 3.0
     Data : 2.0 3.0 4.0
     Data : 3.0 4.0 5.0
     Data : 4.0 5.0 6.0
❷    This line will be ignored by a V/READ with MATCH
     Data : 5.0 6.0 7.0
     Data : 6.0 7.0 8.0
     Data : 7.0 8.0 9.0
     Data : 8.0 9.0 1.0
     Data : 9.0 1.0 2.0
❷    End
```

This example shows how the MATCH parameter can be used in order to read only a subset of a file. MATCH is used to specify a pattern string, restricting the vector filling only to the records in the file which verify the pattern. Example of patterns:

- /string/ match a string (starting in column 1)
- -/string/ do not match a string (starting in column 1)
- /string/(n) match a string, starting in column n
- /string/(*) match a string, starting at any column

❶ When the MATCH parameter is used, the command V/READ reads the file in two passes:

(a) to find how many lines should be read in order to create vectors with the proper length.

(b) to read the lines where the MATCH parameter is found.

❷ these lines are skipped during the reading pass.

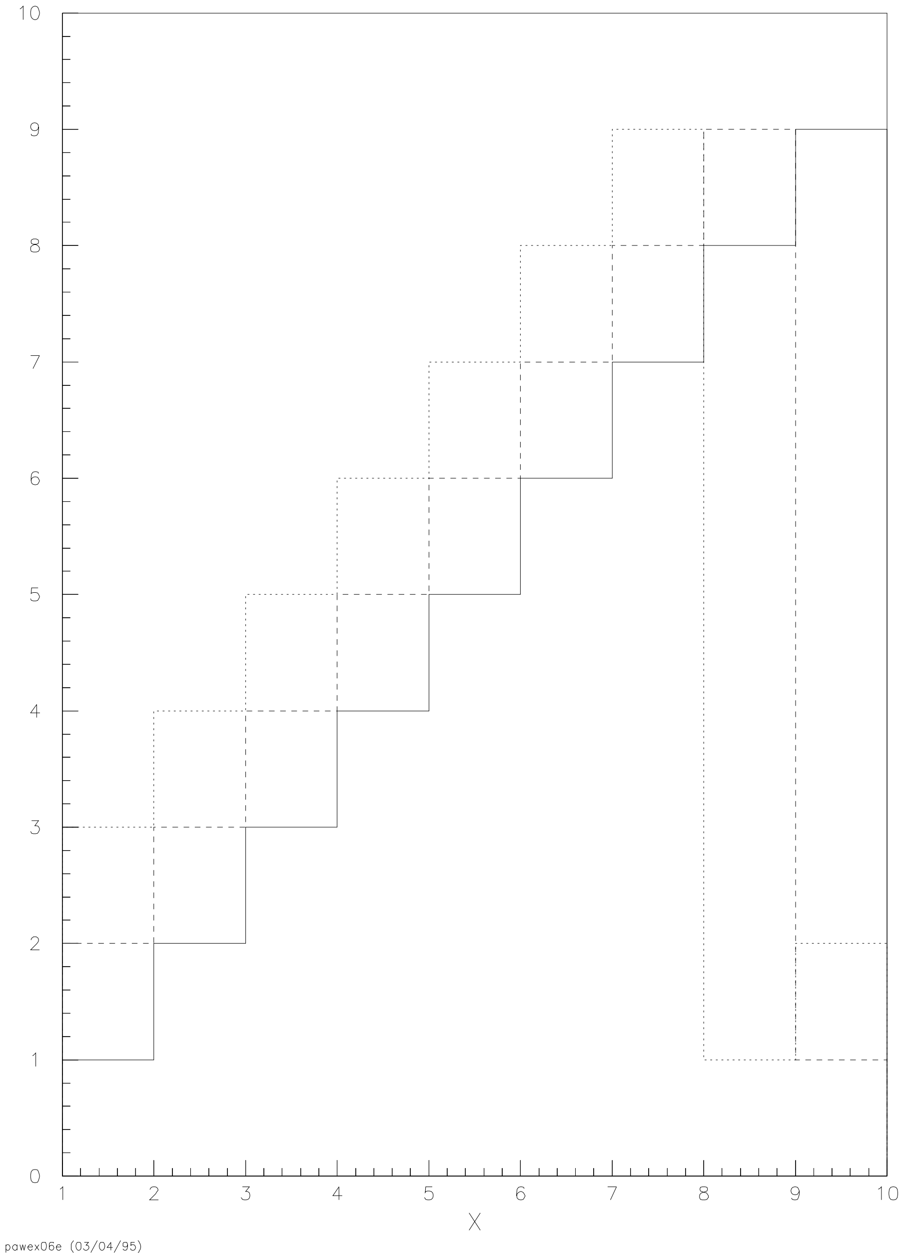


Figure 8: Exec pawex06e.kumac

Application DATA usage

```
❶    Application DATA tmp.dat
      1 1 5
      2 2 4
      3 3 3
      4 4 2
      5 5 1
      6 5 1
      7 4 2
      8 3 3
      9 2 4
     10 1 5
❶   tmp.dat
❸    V/READ X,Y,Z tmp.dat
     graph 10 X Y CW*
     graph 10 X Z C*
❸    SHELL rm tmp.dat
```

❶ Application data allows to create an ASCII file. The end of the data set is marked by the file name. This is a convenient way to keep the data set and the macro together.

❷ The file `tmp.dat` can be read immediately after the `Application DATA`.

❸ The command `SHELL` (p 3) allows to pass a command to the local operating system. Here the temporary file `tmp.dat` is removed (UNIX syntax).

Data embedded in a macro

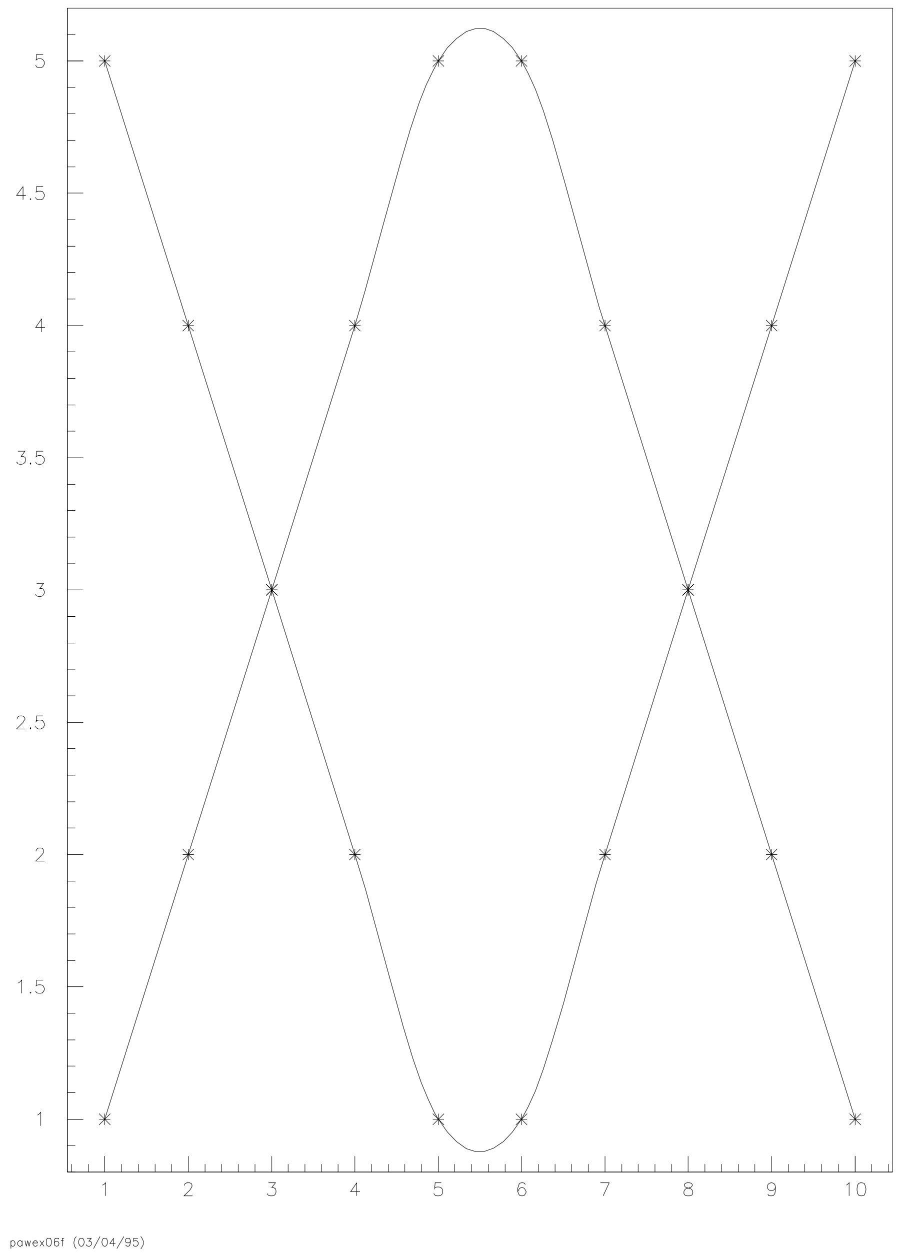


Figure 9: Exec pawex06f.kumac

Plot a few one-dimensional functions

* FUNCTION/PLOT UFUNC XLOW XUP [CHOPT]

```
❸     OPT GRID
❶     FUNC/PLOT X*SIN(X)*EXP(-0.1*X)  -10. 10.
❷     SET DMOD 2
      func/plot (sin(x)+cos(x))**5    -10. 10. s
      set dmod 3
      func/plot (sin(x)/(x)-x*cos(x)) -10. 10. s
```

❶ FUN/PLOT (p 79) allows to plot 2D functions. The character "x" or "X" is used as the variable name. The command FUN1 (p 77) is analog to FUNC/PLOT (p 79) but it produces also an histogram with the value of the function. The number of steps used to compute the function along the X axis can be defined via the command POINTS (p 79).

Note also:

❷ SET DMOD allows to define the line type for the drawing the function. Note that SET LTYP (the generic attribute for lines) cannot be used is this case because in the command FUN/PLOT many different lines are drawn (axes, boxes, etc ..). So a specific attribute must be used (DMOD) for the line type of a function or an histogram.

❸ OPTION GRID allows to have a grid on the subsequent plots.

Plot a few one-dimensional functions

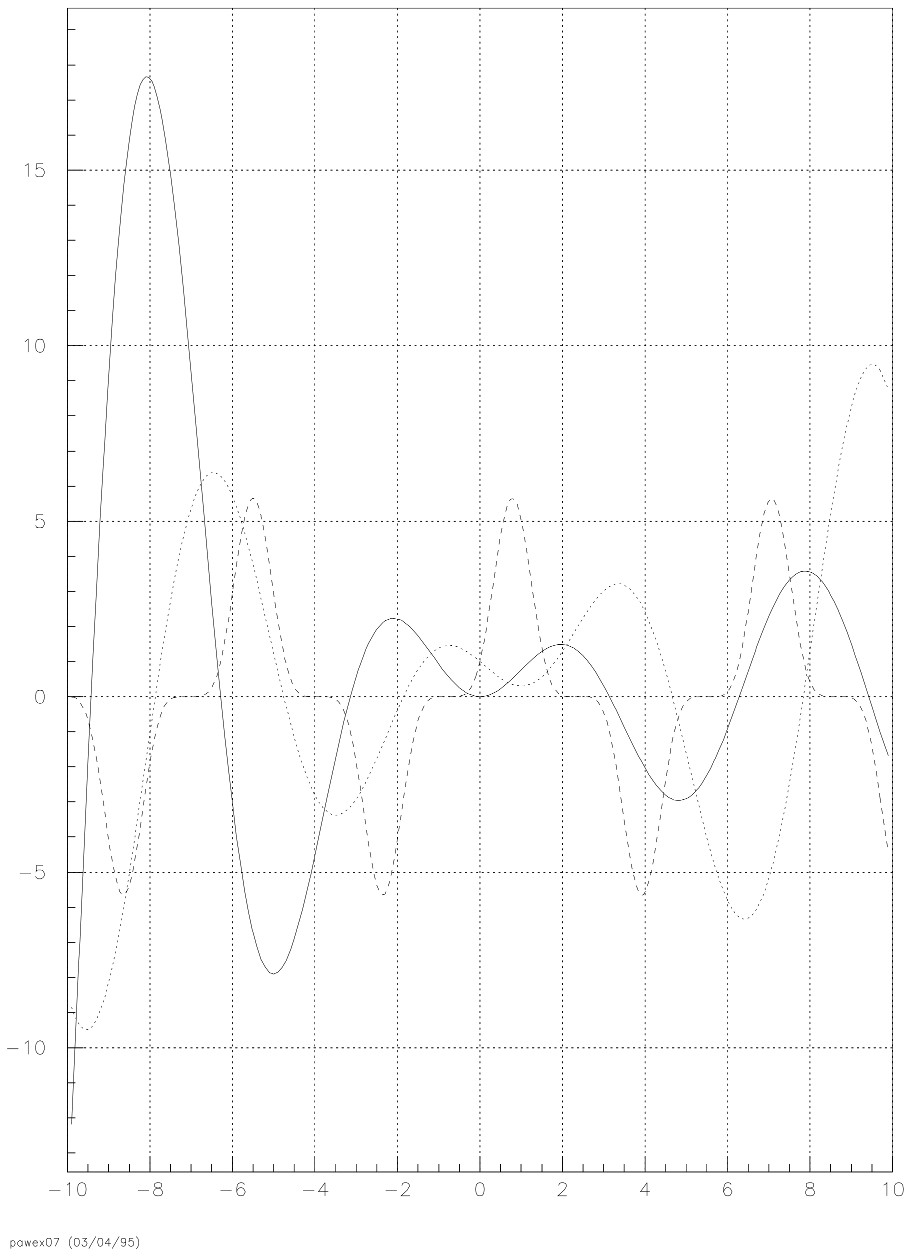


Figure 10: Exec pawex07.kumac

Plot a one-dimensional function and loop

```
❶❷  MACRO PLOT  1=8
    * The Macro parameter is the number of plots to be drawn.
    * the defaults is 8.
    set dmod 1
❸   SET XTIC 0.0001
❸   SET YTIC 0.0001
    set xval 100.
    set yval 100.
    opt utit
    fun/plot x*sin(x) -10 10
    fun/plot x*cos(x)*sin(x) -10 10 s
    a=[1]-1
    do i=[a],1,-1
      fun/plot x*sin(x)*[i]/[1] -10 10 s
      fun/plot x*cos(x)*sin(x)*[i]/[1] -10 10 s
    enddo
```

❶ In this example we can see that macros can have input parameters. These parameters can be positional, and they can be accessed in the macro via `[n]`, where `n` is the parameter number in the input list, or they can be specified by name and they are accessed like variables. The next example gives more details on the input parameters management.

❷ If one parameter (positional or not) needs to have a default value, the value can be specified on the `MACRO` line. At execution time this default value is taken if no value is given. Note that for parameters given by name, the default value on the line `MACRO` is mandatory.

❸ It is possible to define the geometry of a picture via the `SET` parameters described in the section *setting attributes* of the chapter *Graphics (HIGZ and HPLOT)* in the **PAW** manual. In this example the size of the tick marks is set to 0 (`XTIC` and `YTIC`). But it is not possible to specify: `SET XTIC 0` as, for the `SET` command, 0 means default value.

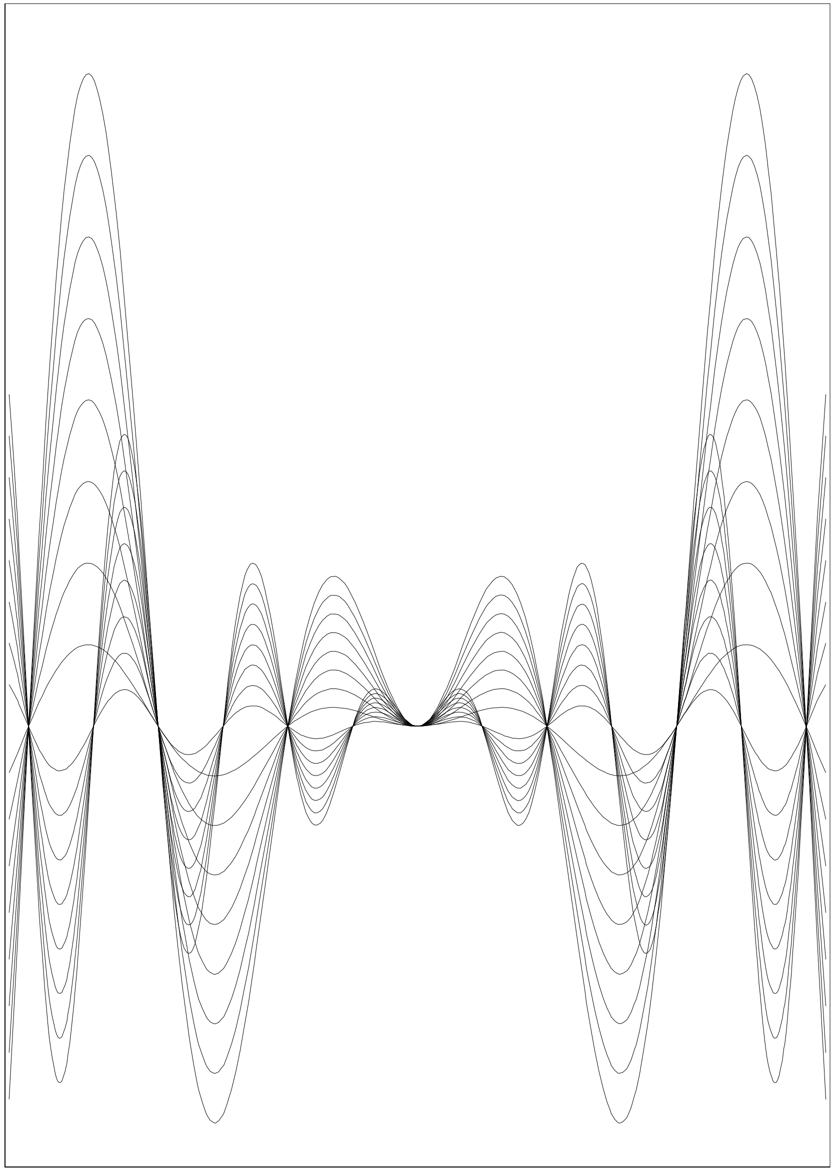


Figure 11: Exec pawex08.kumac

More on macro input parameters and return code

Access to the parameter list

```
MACRO PAWEX08A                        PAW > exe pawex08a 23 9
i = 10                                      23 squared is 529
❶ FOR p IN [*] [i] 1 2                      9 squared is 81
  sq = [p] * [p]                            10 squared is 100
  message [p] squared is [sq]               1 squared is 1
ENDFOR                                      2 squared is 4
```

Indexed positional parameters

```
MACRO PAWEX08B                        PAW > exe pawex08b 23 9 48
❷ DO i = 1, [#]                            parameter 1 is 23
❸  message parameter [i] is [%i]            parameter 2 is 9
ENDDO                                       parameter 3 is 48
```

❶ The * sign allows to access the list of input parameters.

❷ The # sign allows to access the number of input parameters.

❸ % allows to have indexed positional parameters.

EXITM return code

```
MACRO PAWEX08C                        PAW > exe pawex08c
  MESSAGE At first, '[@]' = [@]             At first, [@] = 0
  EXEC EXIT2                                Macro EXIT2: NUM ? 25
  IF [@] = 0 THEN                           Number too large
     MESSAGE Macro EXIT2 successful         Error in EXIT2 : 5
  ELSE                                PAW > exe pawex08c
     MESSAGE Error in EXIT2 : [@]           At first, [@] = 0
  ENDIF                                     Macro EXIT2: NUM ? 16
RETURN                                      Macro EXIT2 successful

MACRO EXIT2
  READ NUM
  IF [NUM] > 20 THEN
     MESSAGE Number too large
     EXITM [NUM]-20
  ELSE
     VEC/CRE VV([NUM])
  ENDIF
RETURN
```


Global variables

The global variables behave exactly like the normal variables. They can be set with =, their content is accessed via [].

They are valid in all macros and in command mode. The following examples illustrate how to manipulate them.

Global variables manipulation

```
MACRO PAWEX08D
❶ G/CREATE PI 3.14159              Approximate value of PI
G/CREATE GV This_is_a_text         Text global variable
❷ G/LIST
❸ G/DELE GV
G/LIST
RETURN


PAW > exe pawex08d
 @ = 0 | macro return value
 GV = This_is_a_text | Text global variable
 PI = 3.14159 | Approximate value of PI
 @ = 0 | macro return value
 PI = 3.14159 | Approximate value of PI
❸ PAW > MESS [PI]
 3.14159
```

❶ Global variables are created with a name, a value and an associated comment.

❷ This command allows to list all the global variables. Note that @ (macro return code) is defined as a global variable.

❸ Delete global variables.

❹ Global variables can de accessed outside macros.

Plot two-dimensional functions

*** FUNCTION/FUN2 ID UFUNC NCX XMIN XMAX NCY YMIN YMAX [CHOPT]**

```
   zone 2 2
❶  FUN2 10 ABS(SIN(X)/X)*(COS(Y)*Y) 40 -6 6 40 -6 6
   contour 10 40 0
   hi/de 10
   fun2 10 x*sin(x)*y*sin(y) 40 -10. 10. 40 -10. 10. CONT
   h/pl 10 surf4
```

❶ The command FUN2 allows to plot 2D functions and fill an histogram. The variables names are X and Y.

❷ It is possible to represent a 2D histogram in several ways :

(a) As a scatter plot.

(b) With proportional boxes.

(c) With a color table.

(d) As a surface plot.

(e) As a surface with color levels.

(f) As a surface with a contour plot on top.

(g) As a surface with Gouraud shading.

(h) As a lego plot.

(i) As a lego plot with colours or shading.

(j) As a line contour plot.

(k) As a table.

(l) As an arrows plot.

Plot two-dimensional functions

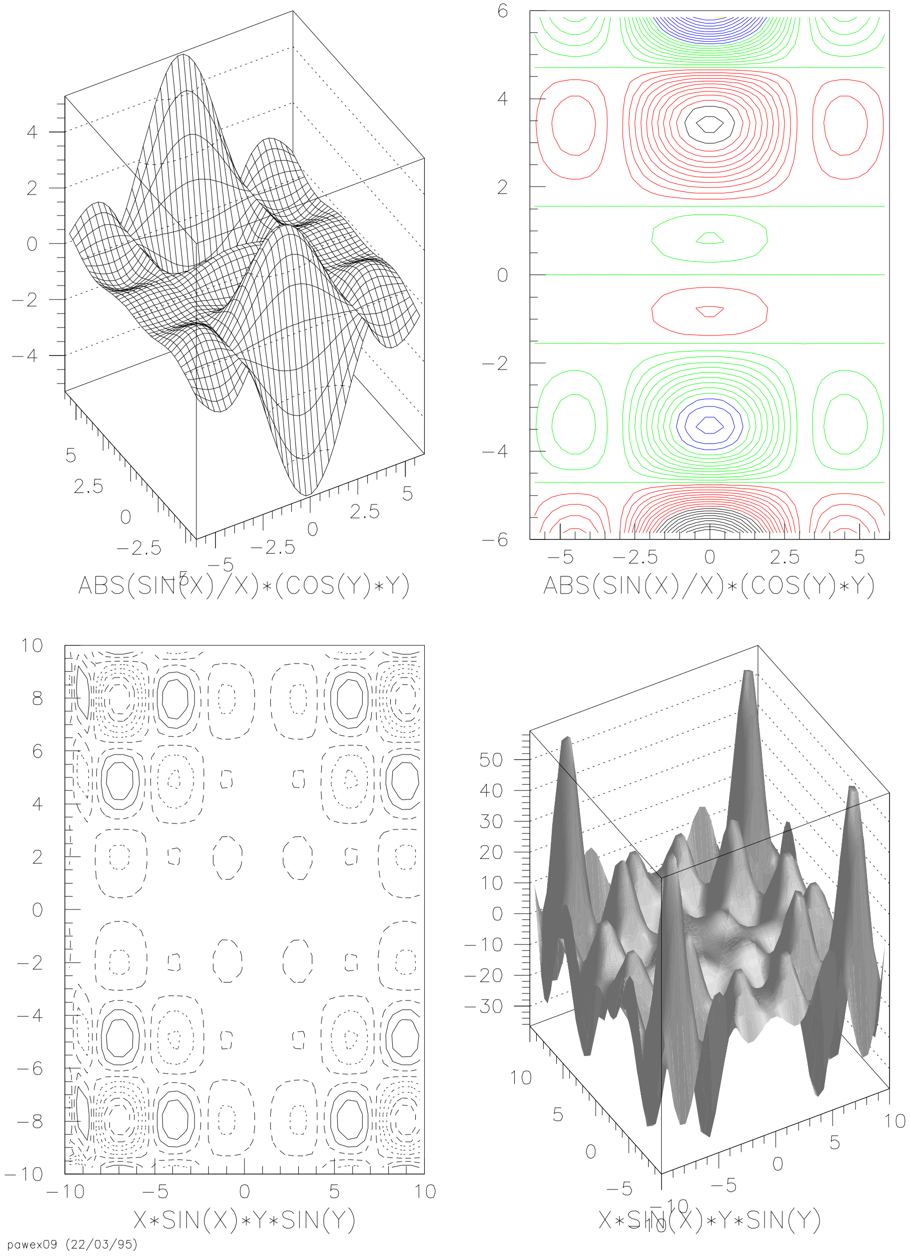


Figure 12: Exec pawex09.kumac

Calculate and plot (BOX option) the Mandelbrot distribution

```
❶     FUN2 10 mandel.f [1] -2.4 .8 [1] -1.2 1.2 ' '
❷     HI/PL 10 BOX
```

FORTRAN Routine MANDEL

```
      real function mandel(xp)
❸     dimension xp(2)
      data nmax/30/
      x=xp(1)
      y=xp(2)
      xx=0.
      yy=0.
      do 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 20
      enddo
   20 mandel=float(n)/float(nmax)
      end
```

❶ This example shows one of the usages of COMIS. In this case, the name of the function to be plotted by `FUN2` is replaced by a COMIS FORTRAN function.

❷ `CHOPT=' '` in the command `FUN2` means to fill only the histogram without producing the plot which is by default a surface. The plot is produced by the command `HIST/PLOT`.

❸ The vector `XP` is an input parameter given by `FUN2`, for each cell, to the FORTRAN program. `XP` contains the `X` and `Y` coordinates of each cell. You can try to insert:

```
print*, XP
```

in `mandel.f` to see the values changing (in this case it is better to set the input parameter of the macro to 10).

The Mandelbrot distribution

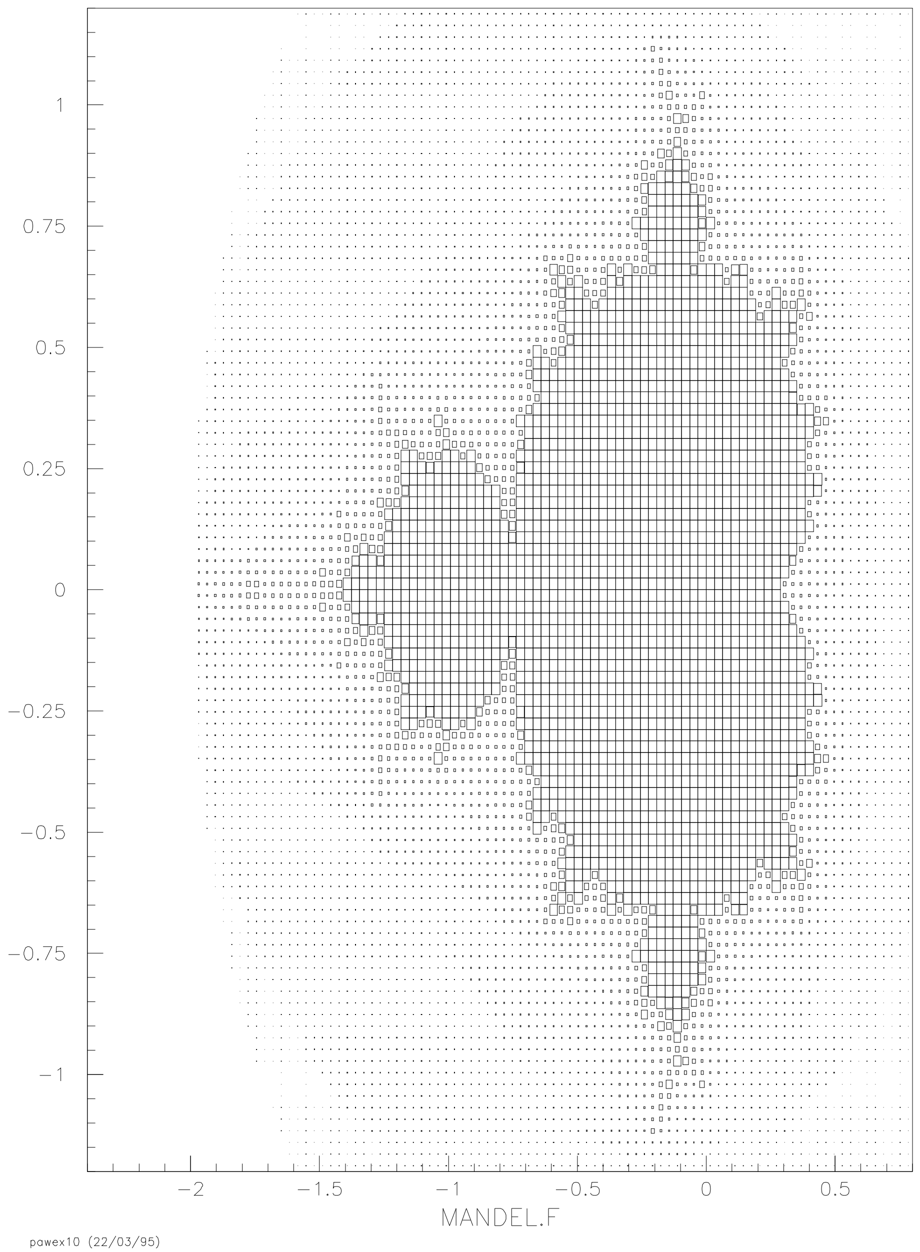


Figure 13: Exec pawex10.kumac

3D functions drawing

FUNCTION/DRAW and RANGE

```
   zon 2 2
❶  FUN/DRAW X**2+Y**2+Z**2=1
❷  RANGE 0 1
❶  FUN/DRAW X**2+Y**2+Z**2=1
❷  RANGE 0 1 0 1
❶  FUN/DRAW X**2+Y**2+Z**2=1
❷  RANGE 0 1 0 1 0 1
❶  FUN/DRAW X**2+Y**2+Z**2=1
```

❶ This command draws a sphere of radius 1. The function can be also a COMIS program.

❷ The command `RANGE` (p 79) modify the X, Y and Z range in which the function is drawn.

3D functions drawing

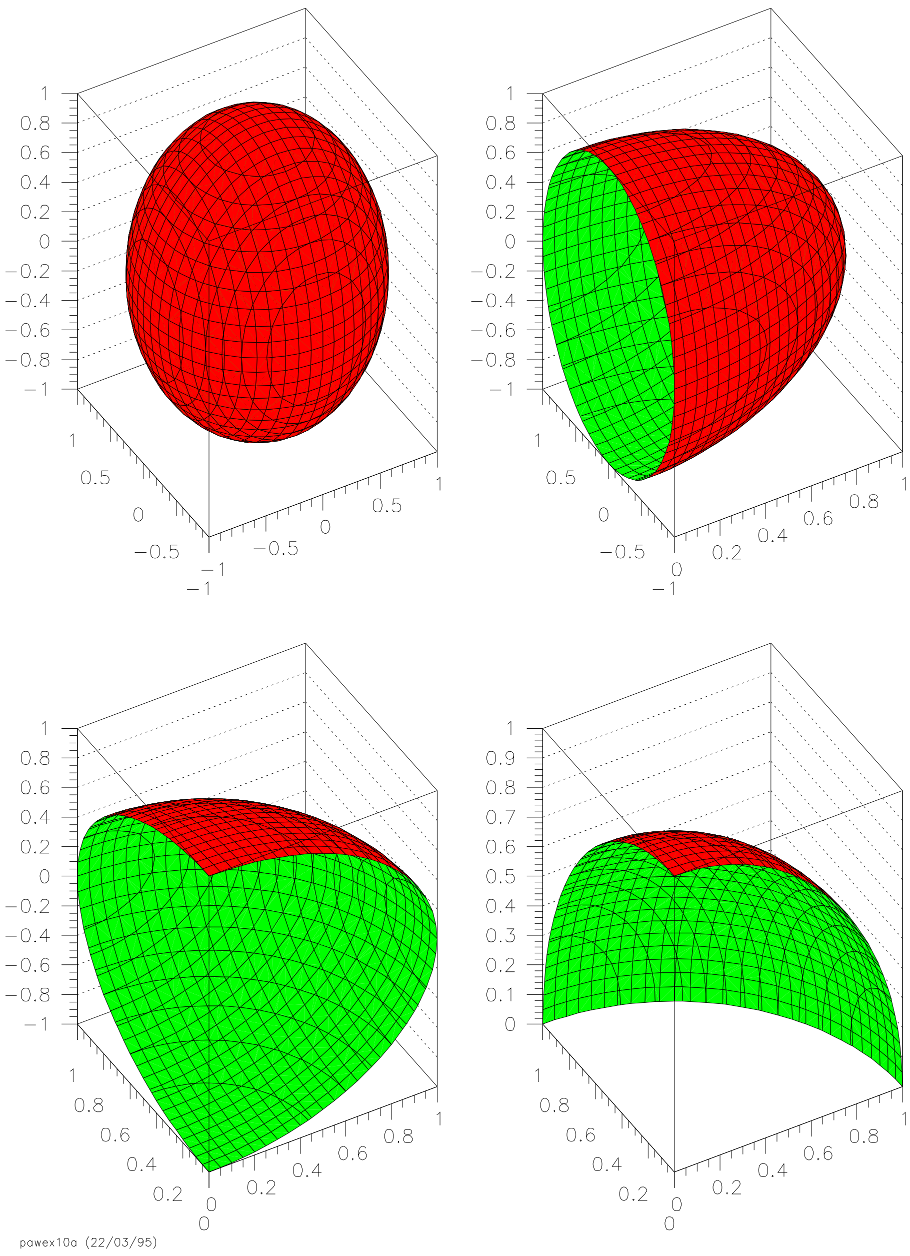


Figure 14: Exec pawex10a.kumac

Creation of one and two dimensional histograms

```
        zon 1 2
        function/fun1 100 htfun1.f 100. 0. 1.
        1dh  110 'Test 1-dim Histo' 100 0. 1. 1000.
❶       CALL UROUT.F(5000)
❻       FUN/FUN2 200 HTFUN2 25. 0. 1. 25. 0. 1. CONT
        hi/li
❸ ❺     HISTOGRAM/FILE 1 PAWHISTS.HBOOK 1024 N
❹       HROUT 0
```

The FORTRAN Routines HTFUN1,HTFUN2 and UROUT

```
      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.0.0001)x1=c1*exp(a1)
      if(abs(a2).gt.0.0001)x2=c2*exp(a2)
      htfun1=x1+x2
      end
```

```
❻     function htfun2(x,y
      htfun2=100*htfun1(x)*htfun1(y)
      end


      subroutine urout(nev)
      do i=1,nev
❷       x=HRNDM1(100,I)
❷       CALL HFILL(110,X,0.,1.)
      enddo
      end
```

❶ In this example **COMIS** is used in the simplest way, via the command `CALL` (p 142) (`CALL UROUT.F`). This command just calls the FORTRAN routine given as parameter and executes it.

❷ It is possible to call several routines of the CERN library. `HELP CALL` gives the list of available routines (see next page). Here the routines `HRNDM1` and `HFILL` (to fill an histogram) are called by `UROUT`.

❸ It is possible to store the histograms in memory into a direct access file opened via the command `HIST/FILE`. Here `CHOPT=N` means: "create a New **HBOOK** file". If the first parameter (`LUN`) is 0 the next free logical unit will be used.

❹ To store an histogram in a file it is enough to execute the command `HROUT` (p 62) . `HROUT 0` (or `HROUT *`) stores all the histograms currently in memory.

❺ Several files can be attached via `HIST/FILE` during a **PAW** session. To change the current file it is enough to execute `CD //LUNn` (p 134) where "n" is the first parameter given to `HI/FILE`. Note that the command `LD //` (p 19) gives the list of all the files currently attached. Each attached direct access file is similar to a directory (cf UNIX).

❻ `HTFUN2` is in the file `htfun1.f`. That is why it can be invoked without the extension `.f` because it has been compiled during the `CALL` to `htfun1`.

Most of the time, the histograms are created and filled outside **PAW** in batch programs calling **HBOOK** directly, and after interactively analyzed with **PAW**.

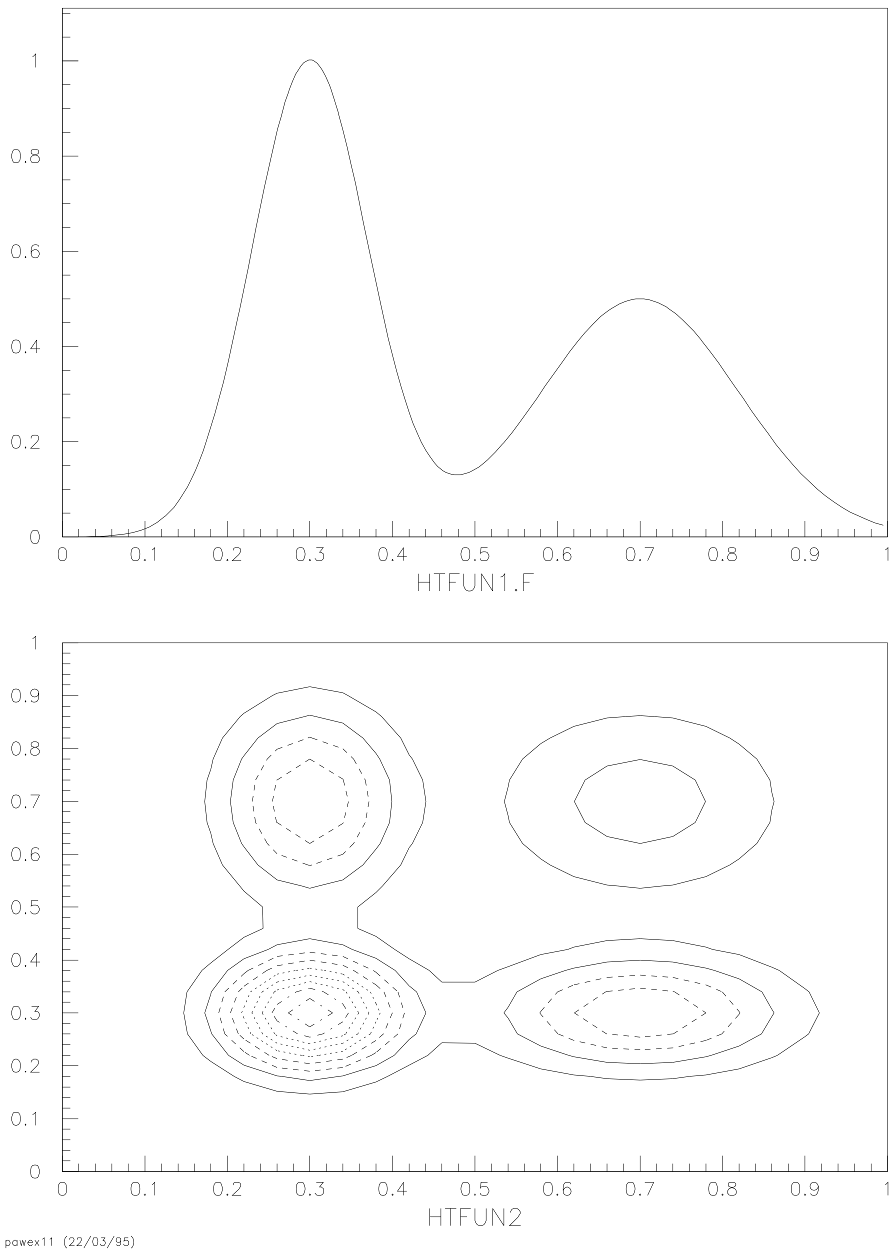


Figure 15: Exec pawex11.kumac

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,HGFIT,HRENID
HROPEN,PAOPEN,PACLOS,PAREAD,PAWRIT,HCDIR,HGIVEN,HKIND
HTITLE,HBFUN1,HBFUN2,HRNDM1,HRNDM2,HBARX,HBARY,HDIFFB
HPAK,HPAKE,HUNPAK,HGIVE,HGN,HGNF,HGNPAR,HF2,HFF1,HFF2
HRIN,HROUT,HI,HIE,HIX,HIJ,HIF,HIDALL,HNOENT,HX,HXY
HTITLE,HCOPY,HSTATI,HBPROF,HOPERA,HIDOPT,HDERIV,HBAR2
HMAXIM,HMINIM,HMAX,HMIN,HSUM,HNORMA,HMCINI,HMCMLL
HEXIST,HREND,HRGET,HRPUT,HSCR,HFIND,HCX,HCXY,HLABEL
HBPROX,HBPROY,HBANDX,HBANDY,HBSLIX,HBSLIY,HPROF2
HBOOKB,HBSTAT,HDIFF,HUNPKE,HREBIN,HERROR,HGNTB,HSTAF
HOUTPU,HERMES,HISTDO,HFUNC,HXI,HIJXY,HXYIJ,HLPOS,HFC1
HSPLI1,HSPLI2,HMDIR,HLDIR,HLOCAT,HFITH,HFITV,HFINAM
HBNT,HBNAME,HBNAMC,HFNT,HFNTB,HGNT,HGNTF,HGNTV,HBSET
HRENAME,HNTDUP
```

From HPLOT

```
HPLOT,HPLSYM,HPLERR,HPLEGO,HPLNT,HPLSUR,HPLSOF,HPLFRA
HPLABL,HPLSET,HPLGIV,HPLOC,HPLTOC,HPLNEW,HPLOPT
```

From ZEBRA

```
MZSTOR,MZDIV,MZLINK,MZWORK,MZBOOK,MZDROP,MZPUSH
MZWIPE,MZGARB,MZFORM,LZFIND,LZFID,DZSHOW,DZVERI
FZIN,FZOUT,FZFILE,FZENDI,FZENDO
RZCDIR,RZLDIR,RZFILE,RZEND,RZIN,RZOUT,RZVIN,RZVOUT
RZOPEN,RZIODO,RZCLOS,RZQUOT
```

From KUIP

```
KUGETV,KUDPAR,KUVECT,KILEXP,KUTIME,KUEXEL,KUPROS
KUNWG,KUCMD,KUGUID,KUNDPV,KUPAR,KUPVAL,KUACT
```


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
IGPAVE,IGTERM,ISFACI,IGHTOR,IGONT
```

From KERNLIB

```
VZERO,UCOPY,RANNOR,LENOCC,SBIT0,SBIT1,SBYT
JBIT,JBYT,UCTOH,UHTOC,CLTOU,CUTOL,ERF,ERFC,FREQ,GAMMA
PROB,DENLAN,DSTLAN,DIFLAN,XM1LAN,XM2LAN,RANLAN
RNDM,RDMIN,RDMOUT,SORTZV,CSF77
```

The following common blocks may be referenced

```
/PAWC/, /QUEST/, /KCWORK/, /PAWPAR/, /PAWIDN/
/HCFITS/, /HCFITD/, /RZCLUN/
```

Read histograms from file and plot

Read histograms from file and plot

```
❶    HISTOGRAM/FILE 1 PAWHISTS.HBOOK
❷    HRIN  *
❸    LDIR
❸    HI/LIST
❺    ZON 2 2
     hi/pl 100
     set htyp 244
     hi/pl 110
❺    ZONE 1 2 2 'S'
     hi/plot 200
❹    CLOSE 1
❻    HI/DEL 100,200
```

❶ In this example the existing file `PAWHISTS.HBOOK` is attached in `READ-ONLY` mode.

❷ The command `HRIN *` (or `HRIN 0` (p 62)) gets all the histograms from the file `PAWHISTS.HBOOK` into the memory. Note that commands like `HIST/PLOT` (p 51) take automatically the histogram from the file if it is not already in memory.

❸ Both `LDIR` and `HI/LIST` give the list of the histograms. `LDIR` is the generic command to list the content of a **ZEBRA** file. It has no knowledge about the objects stored in the file that's why it cannot retrieve the histogram names. The **HBOOK** specific command `HIST/LIST` (p 50) is able to find informations on the histogram like the histogram title and the histogram type. On the next page is given the output of these two commands.

❹ To release an histogram file it is enough to do `CLOSE n` (p 143) where "n" is the logical unit number used by the command `HIST/FILE` (p 50) .

Note also:

❺ The usage of the command `ZONE` (p 102) . It is used two times to define zones with different sizes.

❻ In some commands, some parameters have the attributes Loop. This is visible in the help:

HELP of HISTOGRAM/DELETE

```
* HISTOGRAM/DELETE ID

   ID         C 'Histogram Identifier' Loop

   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.
```

When a list of parameters (separated by ",") is given **PAW** execute the invoked command for each parameter in the list.

Read histograms from file and plot

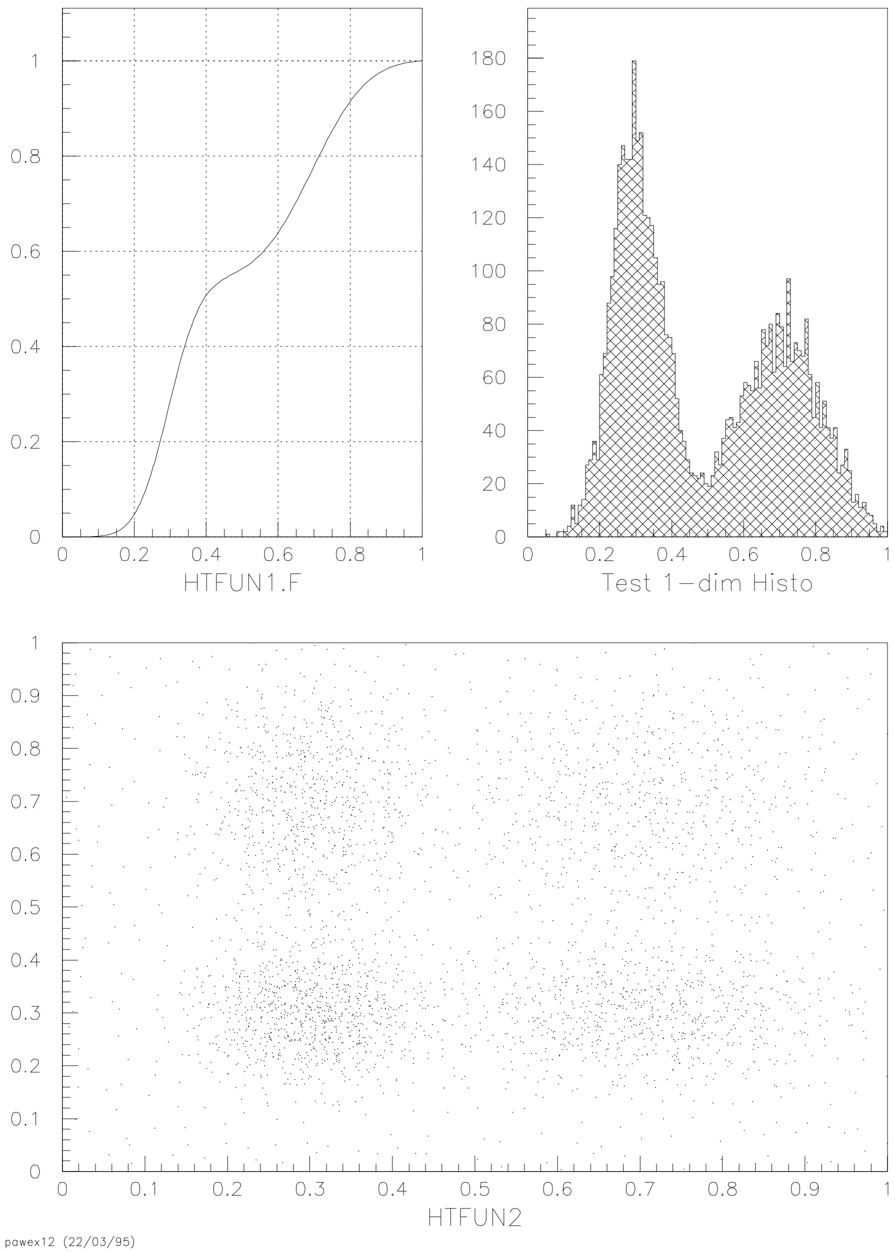


Figure 16: Exec pawex12.kumac

Read histograms from file and plot

Output of LDIR

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

                 Created 941121/1116  Modified 941208/1413

===> List of objects
    HBOOK-ID  VARIABLE                             CYCLE    DATE/TIME   NDATA
       100         0                                   2   941121/1116    151
       110         0                                   2   941121/1116     85
       200         0                                   2   941121/1116    780

 Number of records =     5  Number of megawords =  0 +  4250 words
 Per cent of directory quota used =    .016
 Per cent of file used            =    .016
 Blocking factor                  =  23.164
```

Output of HIST/LIST before HISTOGRAM/DELETE

```
===> Directory :
       100 (1)   HTFUN1.F
       110 (1)   Test 1-dim Histo
       200 (2)   HTFUN2
```

Output of HIST/LIST after HISTOGRAM/DELETE

```
===> Directory :
       110 (1)   Test 1-dim Histo
```

In this example, the histograms in an existing **HBOOK** file are moved in a new **HBOOK** file in two separated directories according to their type.

Histogram archiving and directories into **HBOOK** files

```
❶    HISTOGRAM/FILE 0 pawtut.hbook
     hi/li
     hrin  *
     close 1
❷    HISTOGRAM/FILE 0 pawtutnew.hbook ! N
❸    MDIR 1Dhistograms
❸    MDIR 2Dhistograms
     ldir
     cd 1Dhistograms
❹    HROUT 514,30001,60202
     ldir
     cd //LUN1/2Dhistograms
❹    HROUT 8001,1103,11,12
     ldir
     close 1
```

❶ Attach an existing **HBOOK** file.

❷ Create a new **HBOOK** file.

❸ Create two subdirectories in the file `pawtutnew.hbook`.

❹ Store the 1d and 2d histograms in two separated directories. Note that the "Loop" facility is used again here.

Output of LDIR

```
===> Directory :
        10 (N)   CERN Population
       514 (1)   Angular density
     30001 (1)   mix
     60202 (1)   p dy like
      8001 (2)   Data (gluino)
      1103 (2)   Charged particle theta vs. phi
        11 (2)   PHI VS. Y  +VE    WEIGHTED
        12 (2)   PHI VS. Y  +VE    WEIGHTED

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

                 Created 930602/1428  Modified 930602/1428


===> List of subdirectories
1DHISTOGRAMS     Created 930602/1428 at record     3
2DHISTOGRAMS     Created 930602/1428 at record     4

===> List of objects
    HBOOK-ID  VARIABLE                            CYCLE    DATE/TIME   NDATA

************** Directory ===> //LUN1/1DHISTOGRAMS <===

                 Created 930602/1428  Modified 930602/1428


===> List of objects
    HBOOK-ID  VARIABLE                            CYCLE    DATE/TIME   NDATA
       514         0                                  1   930602/1428    153
     30001         0                                  1   930602/1428    200
     60202         0                                  1   930602/1428    152

************** Directory ===> //LUN1/2DHISTOGRAMS <===

                 Created 930602/1428  Modified 930602/1428


===> List of objects
    HBOOK-ID  VARIABLE                            CYCLE    DATE/TIME   NDATA
      8001         0                                  1   930602/1428    537
      1103         0                                  1   930602/1428   5361
        11         0                                  1   930602/1428    444
        12         0                                  1   930602/1428  13114
```

Fit of the histogram 110 with two Gaussians

```
      histogram/File 1 pawhists.hbook
      hrin  *
❶     VECT/CREATE PAR(6)
      histo/plot 110
❺     SET FWID 6
❺     SET DMOD 2
❷     HISTO/FIT 110(1:50) G QS 0 PAR(1:3)
❸     HISTO/FIT 110(50:100) G QS 0 PAR(4:6)
❺     SET DMOD 1
❹     HISTO/FIT 110 G+G QS 6 PAR
```

❶ The vector PAR will be used to get the initial values of the fit parameters.

❷ Compute a gaussian fit on the first 50 channels. After this command the gaussian parameters are stored in PAR(1:3).

❸ Compute a gaussian fit on the last 50 channels. After this command the gaussian parameters are stored in PAR(4:6).

❹ Compute the global fit using PAR for initial values.

Note also:

❺ The first two gaussian fits are drawn with dashed lines and the third one with a solid line.

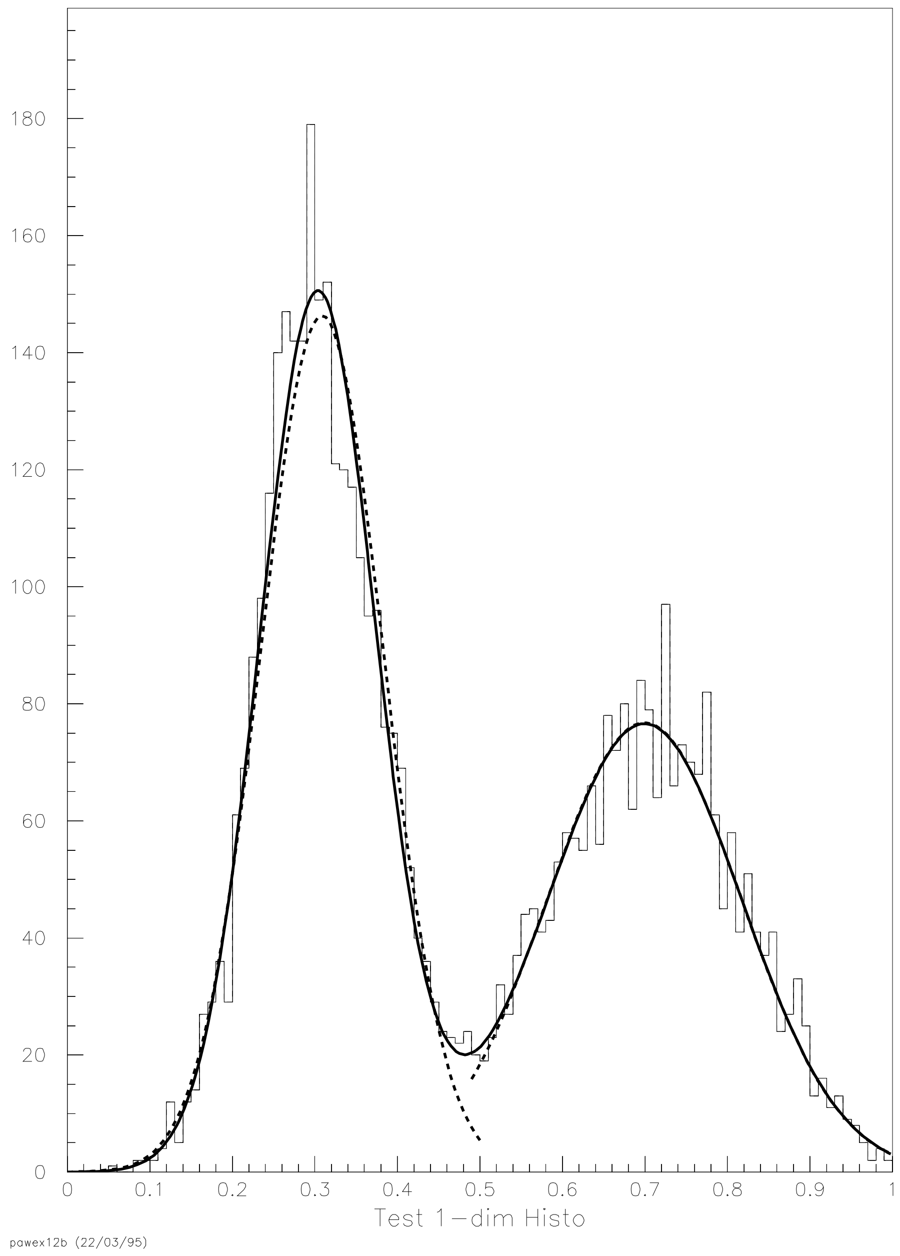


Figure 17: Exec pawex12b.kumac

Perform operations on histograms read from file and save results

```
❶   HISTOGRAM/FILE 1 PAWHISTS.HBOOK ! U
    hrin  *
    zon 2 2
    opt grid
    set mtyp 26
    hi/pl 110 e
    hi/pl 110 pl
    zon 1 2 2 s
❷   HI/OP/ADD 110 110 120 0.5 0.
    hi/op/add 110 110 130 0.25 0.
    set htyp 245
    hi/pl 110
    set htyp 254
❸   HI/PL //PAWC/120 s
    set htyp 253
    hi/pl //PAWC/130 s
    text 0.55 95. 'LEP Very Preliminary' 0.35 25.
    hrout 0
```

❶ The option "U" (for Update) in the command `HIST/FILE`, is used when the user wants to change the content of an existing histogram file by adding a new histogram (`HROUT` (p 62)) or deleting an histogram (`HSCRATCH` (p 62)).

❷ It is possible to perform operations between histograms like addition with the commands in the menu `HISTOGRAM/OPERATIONS` (p 65) .

❸ The memory, like the attached files, can be considered as a directory. This is the current directory by default and `//PAWC` is its name. The command `HI/PL //PAWC/id` plots the histogram "id" in memory while the current directory is `//LUN1`.

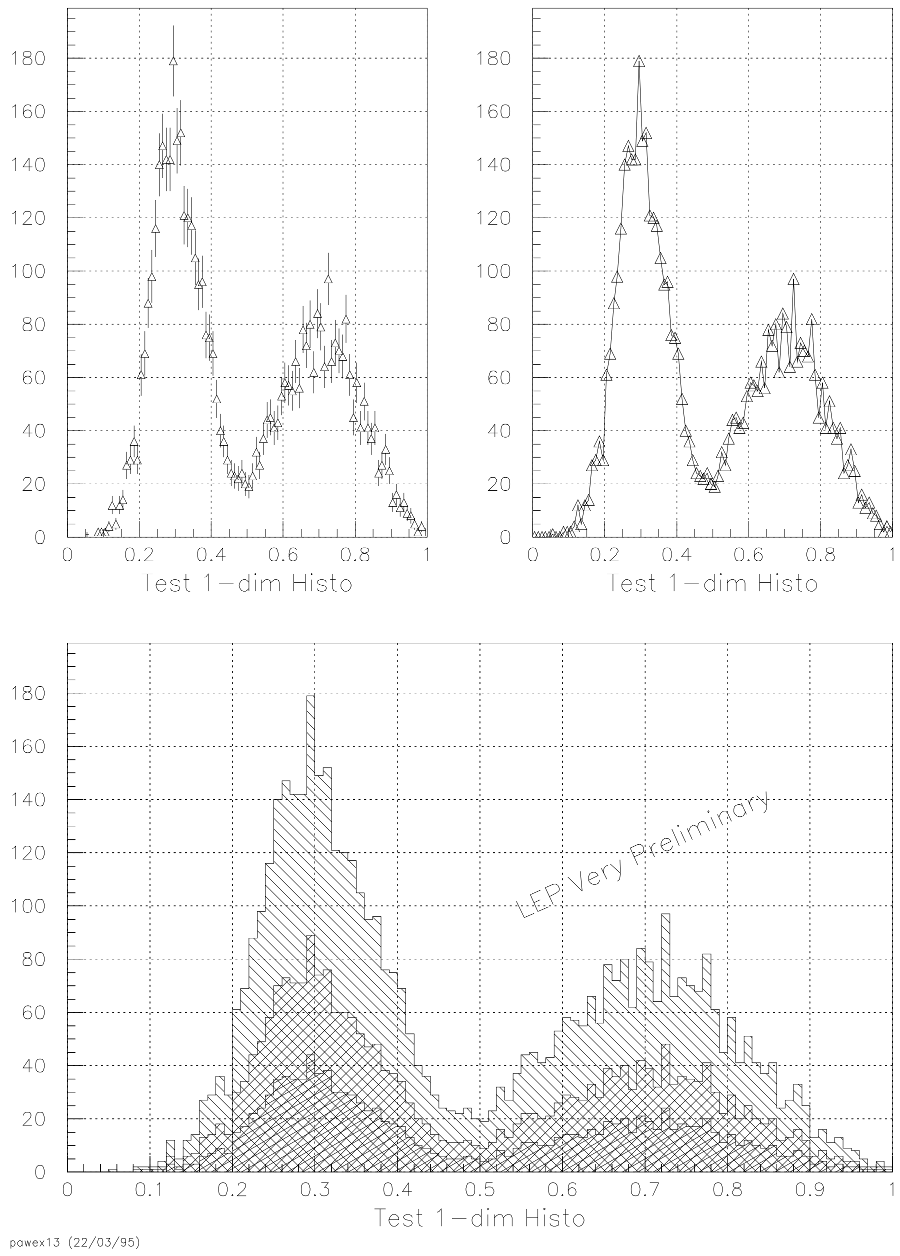


Figure 18: Exec pawex13.kumac

Some "cosmetic modifications" on PAWEX13

```
   histogram/file 1 pawhists.hbook ! u
   hrin  0
   zon 2 2
   opt grid
❶  SET *FON -60
❷  SET BWID 4
❸  SET BCOL 1.5
   set mtyp 26
   hi/pl 110 e
   hi/pl 110 pl
   zon 1 2 2 s
   hi/op/add 110 110 120 0.5 0.
   hi/op/add 110 110 130 0.25 0.
   set htyp 245
   hi/pl 110
   set htyp 254
   hi/pl //pawc/120 s
   set htyp 253
   hi/pl //PAWC/130 s
❹  SET CHHE .35
❹  SET TANG 25.
❹  ITX 0.55 95. 'LEP Very Preliminary'
   hrout 0
```

❶ All the text fonts used for `HISTO/PLOT` are set to -60.

❷ The line width for the boxes around the histograms is set to 4 pixels. Like for the fonts it is possible to do `SET *WID` to set all the width available in the `SET` command.

❸ The color of the shadow around the histograms is set to 5 (Yellow), it appears grey on black and white PostScript printers.

❹ To access hardware fonts (ie PostScript fonts) the command `ITX` and its related attributes should be used.

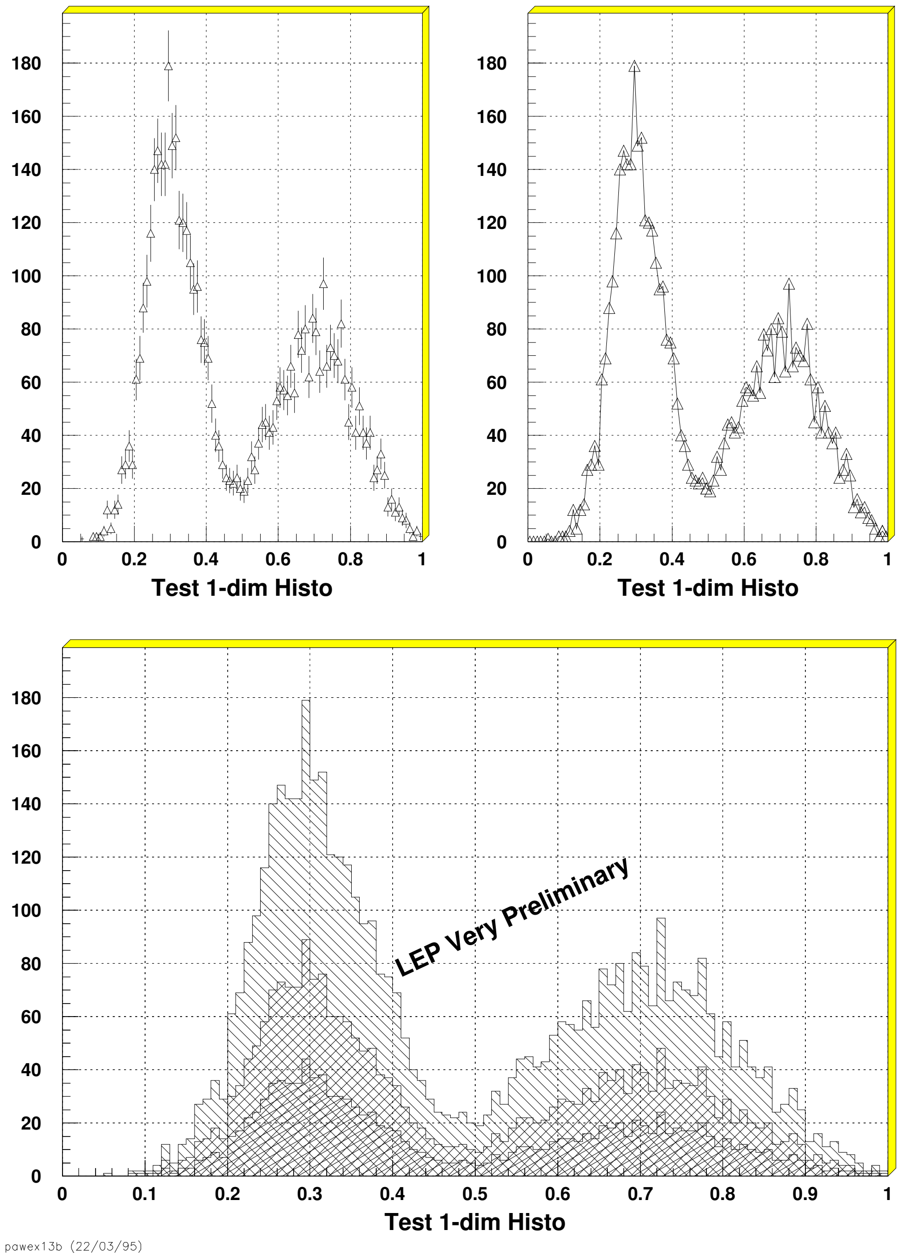


Figure 19: Exec pawex13b.kumac

Histograms can have attributes. They are stored in the histogram data structure.

IDOPT usage

```
    histogram/file 1 pawhists.hbook
    zone 1 3
    hist/plot 110
    close 1
❶   IDOPT 110 LOGY
❶   IDOPT 110 ERRO
    h/pl 110
❷   HISTOGRAM/FILE 2 PAWHISTSNEW.HBOOK ! N
❷   HROUT 110
❷   CLOSE 2
❷   H/DEL *
❷   HISTOGRAM/FILE 1 PAWHISTSNEW.HBOOK
❷   HISTO/PLOT 110
```

❶ The command `IDOPT` (p 75) , allows to set attributes on a given histogram. Here, the histogram 110 will have logarithmic scale on the Y axis and will be drawn with error bars. These options are independant from the global settings define via the command `OPTION` (p 95) .

❷ Here we show that the options set via `IDOPT` are stored in the histogram data stucture.

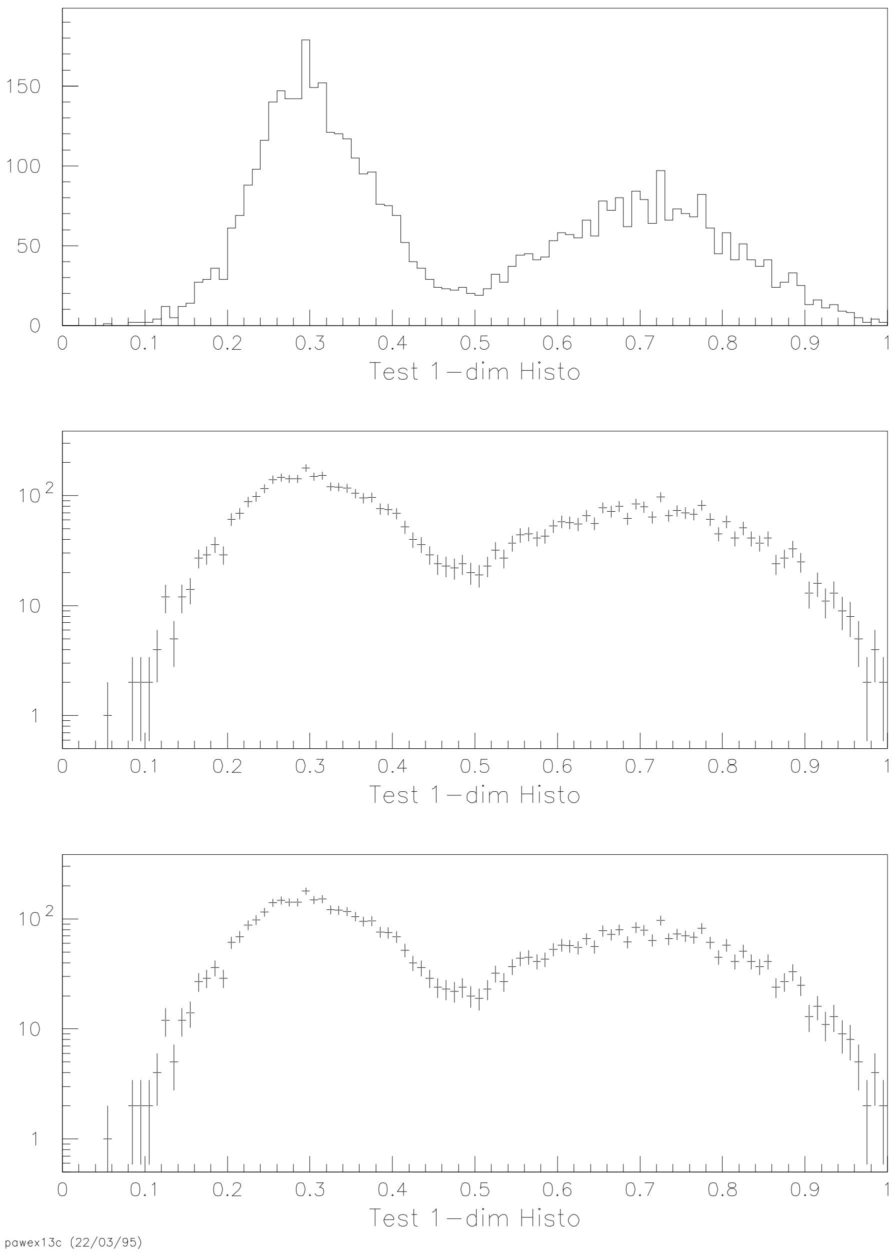


Figure 20: Exec pawex13c.kumac

Different representations of two-dimensional histograms

```
   histogram/file 1 pawhists.hbook
   zon 2 2
❶  HI/PL 200 BOX
❶  CONTOUR 200 20 0
❶  LEGO 200
❶  SURFACE 200
   hi/del *
```

❶ As we have already seen, the command H/PL allows to draw 2D histograms in different ways. Three additional commands are also available:

```
* /HISTOGRAM/2D_PLOT/CONTOUR  [ ID NLEVEL CHOPT PARAM ]
* /HISTOGRAM/2D_PLOT/SURFACE  [ ID THETA PHI CHOPT ]
* /HISTOGRAM/2D_PLOT/LEGO     [ ID THETA PHI CHOPT ]
```

These commands have more parameters than HIS/PLOT. For example CONTOUR (p 57) allows to specify a set of levels to be drawn via the parameter PARAM (see next example).

❷ Note that it is also possible to have 1D histograms represented as lego or surface plots. For example you can do: HI/PLOT 110 LEGO.

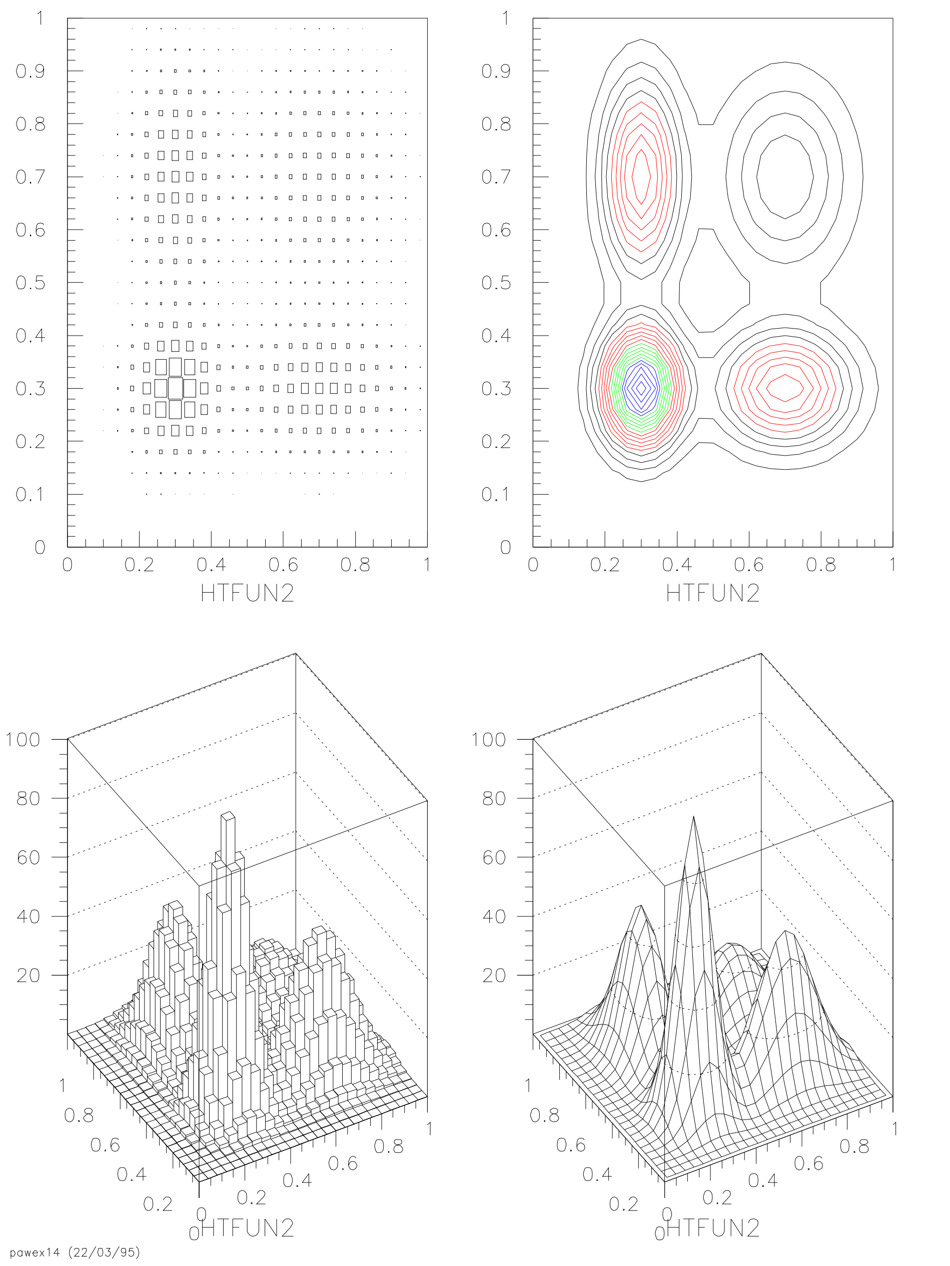


Figure 21: Exec pawex14.kumac

User defined non equidistant contour plots

```
      histogram/file 1 pawhists.hbook
❶     VECTOR/CREATE LEVEL(8) R  10 11 12 13 14 15 90 99
      zone 1 2
❷     CONTOUR 200 8 2 LEVEL
      arrow .8 .75 .5 .54 .2
❺     ARROW .8 .75 .5 .44 .2
❹     SET CHHE .28
❸     ITX .81 .5 '10.0'
      Arrow .5 .32 .1 .28 .2
      Itx .51 .1 '100.0'
      option LOGY
      h/plot 200 BOX
❻     ARROW .5 .32 .1 .28 .2
❻     ITX .51 .1 '100.0'
      close 1
```

The command CONTOUR allows to draw user defined levels.

❶ The vector LEVEL contains the list of 8 levels to be drawn.

❷ Only the levels specified in the the vector LEVEL are drawn.

Note also:

❸ Some comments can be drawn with the command ITX.

❹ The size of the text is in centimeters even if the position is in histogram coordinates (current normalization transformation).

❺ The position of the arrow is in the current normalization transformation (here histogram coordinates), but its size is in centimeters (last parameter. Here 0.2).

❻ Arrows and text can be drawn in logarithmic coordinates. For lines the logarithm should be computed with **SIGMA**.

Non equidistant contour plots

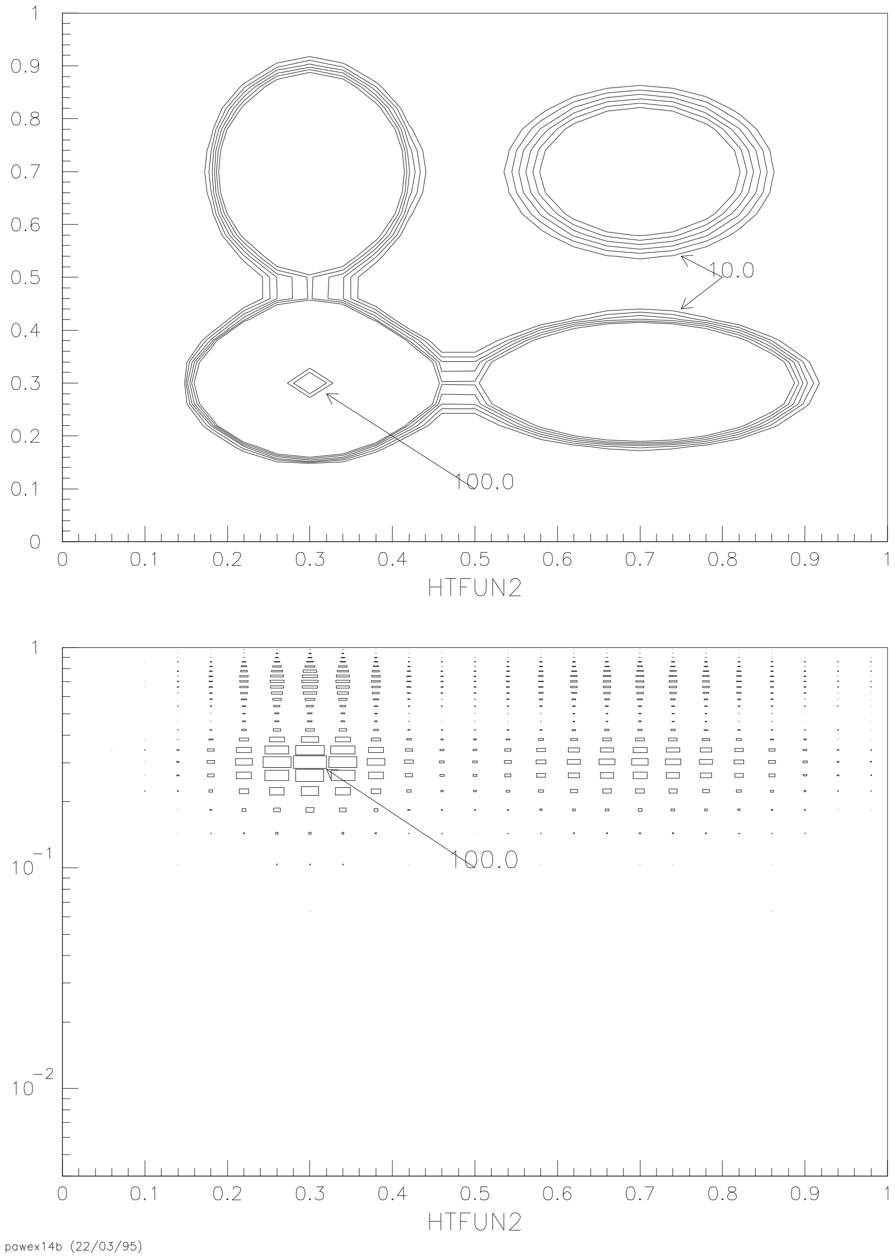


Figure 22: Exec pawex14b.kumac

Coordinate systems with legos and surfaces

```
      histogram/file 1 pawhists.hbook
      zon 2 2
❷     OPT UTIT
❷     TITLE 'Polar coordinates' U
❶     HI/PL 200 LEGO,POL
      title 'Cylindrical coordinates' U
❶     HI/PL 200 LEGO,CYL
      title 'Spherical coordinates' U
❶     HI/PL 200 LEGO,SPH
      title 'Pseudo rapidy coordinates' U
❶     HI/PL 200 LEGO,PSD
      close 1
```

❶ Legos and Surfaces plot can be drawn in Polar, Cylindrical, Spherical and Pseudo rapidity coordinates.

<u>Note also:</u>

❷ The option `UTIT` allows to use "user title" on histogram. To define the title itself, the command `TITLE` should be used with the option `U`. Without this option `TITLE` define the global title.

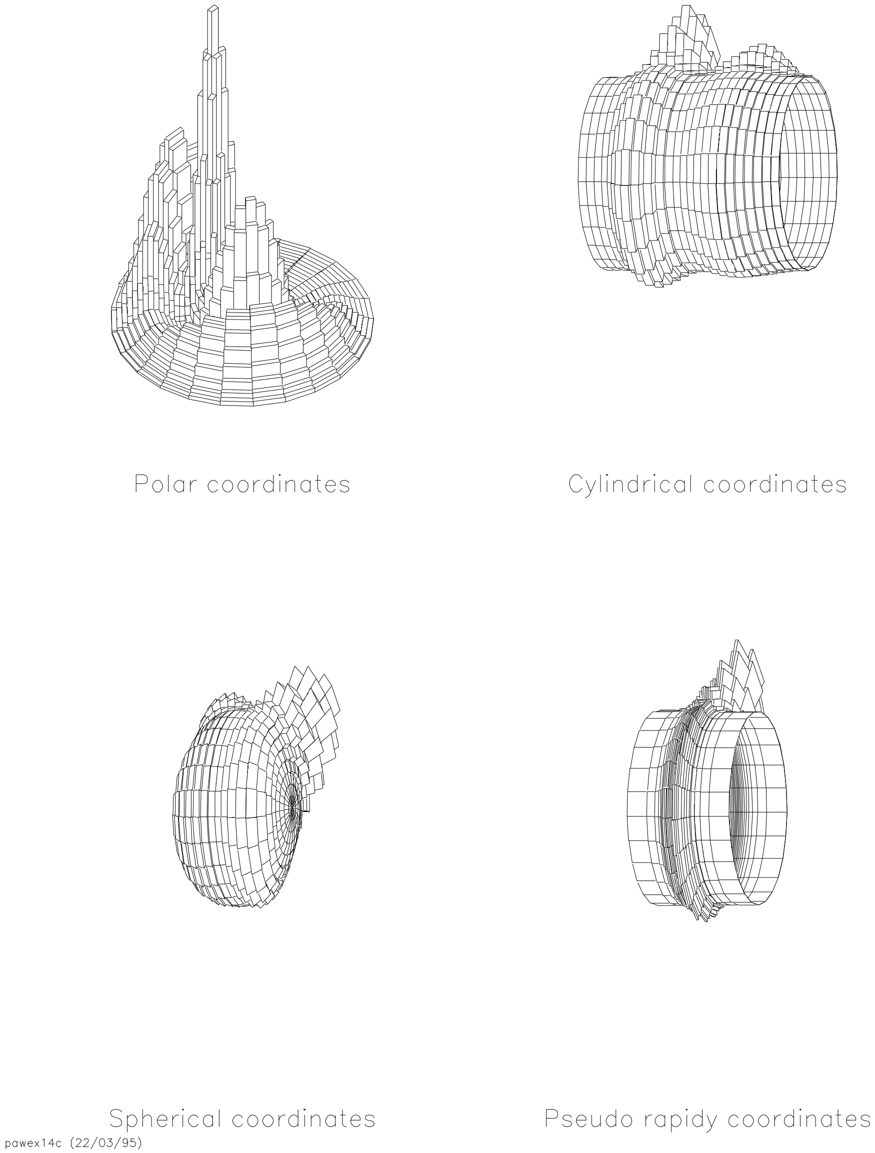


Figure 23: Exec pawex14c.kumac

Logarithmic scales on lego plots and surfaces plot

```
    histogram/file 1 pawhists.hbook
    zon 2 2
    opt utit
    hi/pl 200 lego
❶   OPT LOGX
    hi/pl 200 lego
❶   OPT LOGY
    hi/pl 200 lego
❶   OPT LOGZ
    hi/pl 200 lego
    close 1
```

❶ Logarithmic are possible on Legos and Surfaces plot. It works also in Polar, Cylindrical, Spherical and Pseudo rapidity coordinates.

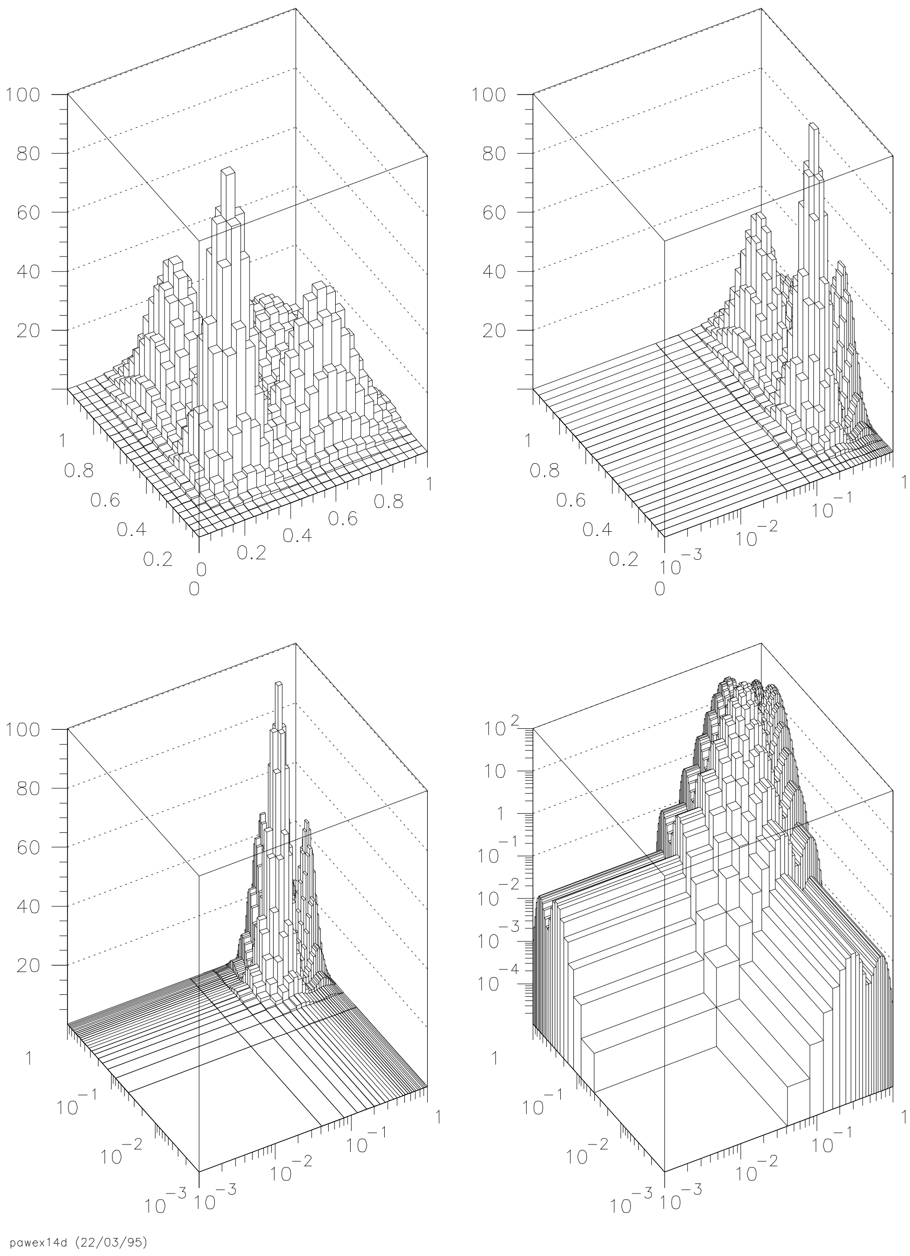


Figure 24: Exec pawex14d.kumac

Usage of subranges in histogram identifiers

```
        histogram/file 1 pawhists.hbook
        hrin  0
        close 1
❹       TRACE ON
        zon 2 2
❶       HI/PL 110(56:95) E
❹       * Comments are not printed in TRACE mode
        hi/pl 200(8:8,)      box
❸       HI/PL 200(3:15,3:15) CONT
❹       TRACE OFF
❷       hi/pl 200(0.:12,0.1:0.5) LEGO
```

❶ This example shows how to plot subranges of 1D or 2D histograms. The different possibility to give the range are the following:

(a) `id(n1:n2)` with $\texttt{n1} \leq \texttt{n2}$.

(b) `id(n1:)` in this case `n2` = `number of bins`.

(c) `id(:n2)` in this case `n1` = `1`.

❷ If `n1` or `n2` are integer they are consider as bin numbers. But if they are real they are consider axis values. Note that bin values and axis values can be mixed inside the same range definition.

❸ In case of 2D histograms, the two ranges are separate with ",".

Note also:

❹ The `TRACE` (p 21) command sets `ON` or `OFF` the trace mode. When this mode is on, all the command executed inside macros are displayed on the standard output.

Ouput of the TRACE mode

```
>>>>> zon 2 2
>>>>> HI/PL 110(56:95) E
>>>>> hi/pl 200(8:8,)      box
>>>>> HI/PL 200(3:15,3:15) CONT
>>>>> TRACE OFF
```

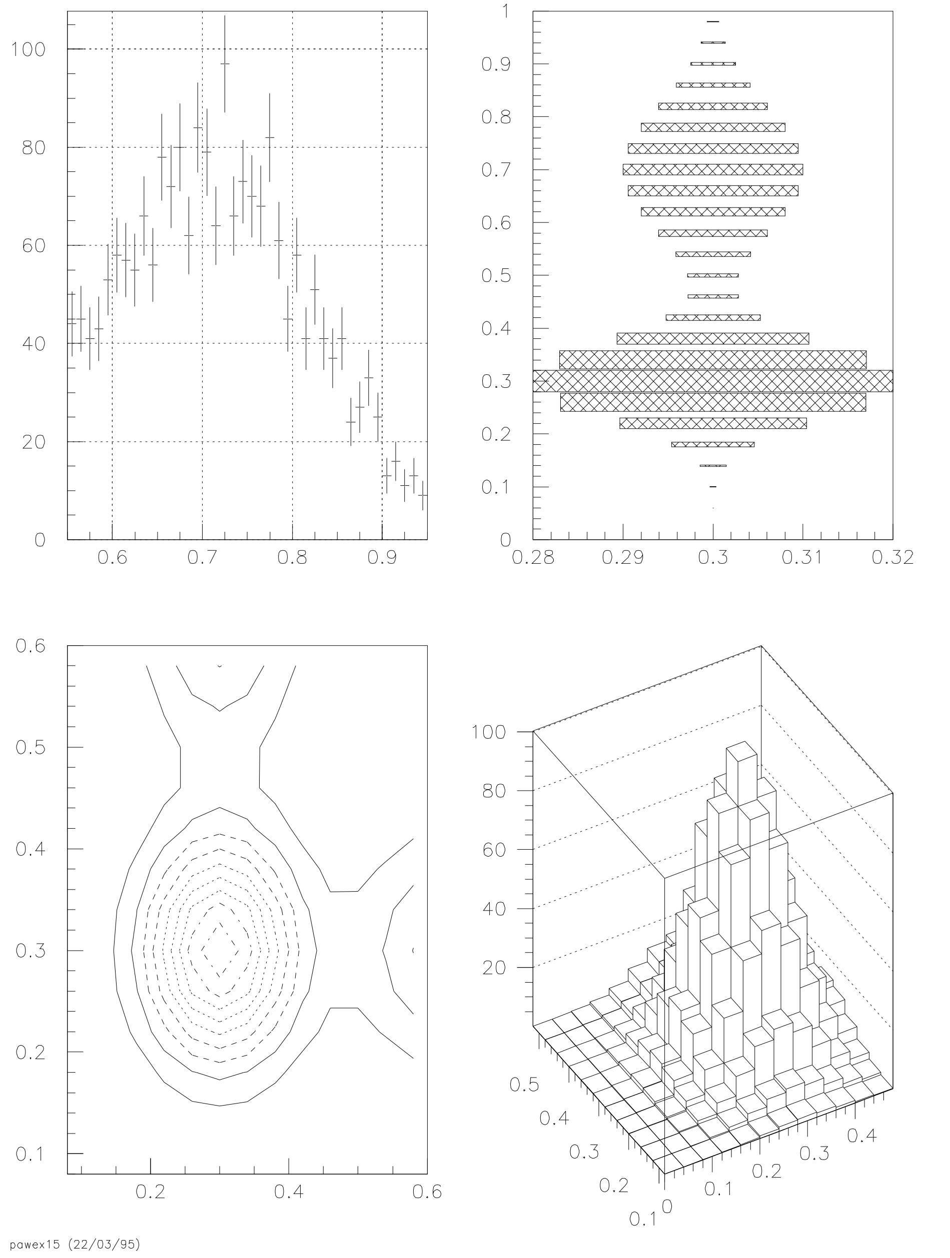



Figure 25: Exec pawex15.kumac

Stacked Lego plots and subranges

```
         hi/file 1 pawhists.hbook
         zon 2 2
❶        HIST/PLOT 200(0.:0.5,0.:0.5) LEGO1
❶        HIST/PLOT 200(0.5:1.,0.5:1.) LEGO1
         zon 1 2 2 s
❸        OPTION BAR
❷        HIST/PLOT 200(0.:0.5,0.:0.5)+200(0.5:1.,0.5:1.) LEGO1
         close 1
```

❶ The two commands draw submatrices of the histogram 200 as Lego plots.

❷ The submatrices previously drawn are now stacked.

❸ The option BAR is active on Lego plots.

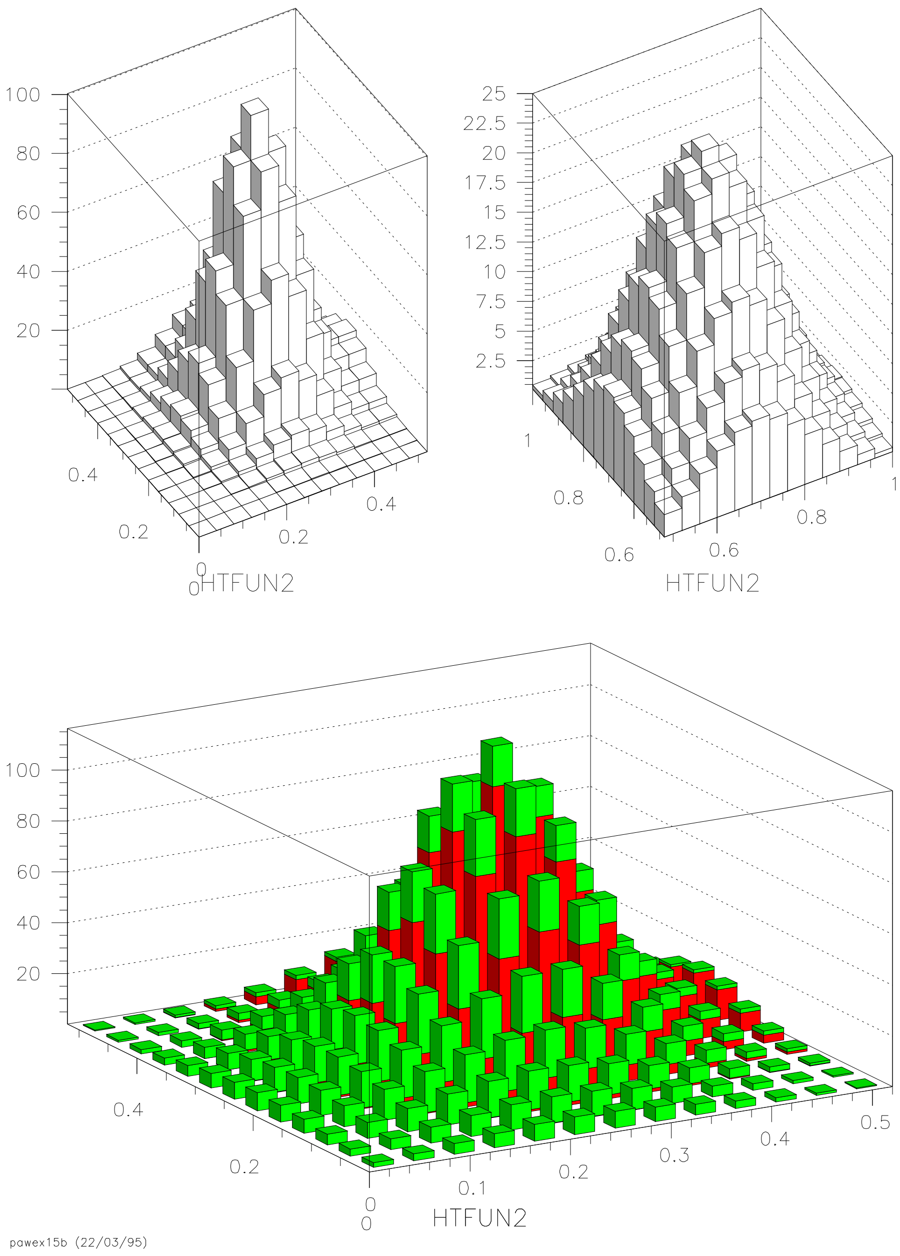


Figure 26: Exec pawex15b.kumac

Representation and definition of errors

```
      histogram/file 1 pawhists.hbook
      hrin  *
      zone 2 3
❶     H/PL 110(1:50) E
❷     H/PL 110(1:50) E1
❸     H/PL 110(1:50) E2
      set htyp 245
❹     H/PL 110(1:50) E3
      set htyp 234
❺     H/PL 110(1:50) E4
      sigma err=array(100,0#200)
❻     PUT/ERR 110 ERR
❻     H/PL 110(1:50) E
      close 0
```

❶ Defaults error bars drawing.

❷ Draw small lines at the end of the error bars. Note that the size of the tick marks at the end of the error bars is the minimum of the error on X and the marker size. If the errors on X are 0 no tick marks are drawn.

❸ Draw error rectangles.

❹ Draw a filled area through the end points of the vertical error bars.

❺ Draw a smoothed filled area through the end points of the vertical error bars.

❻ By default the errors on a histogram are the square root of the content. It is possible to define the errors with the command PUT/ERR (p 72) .

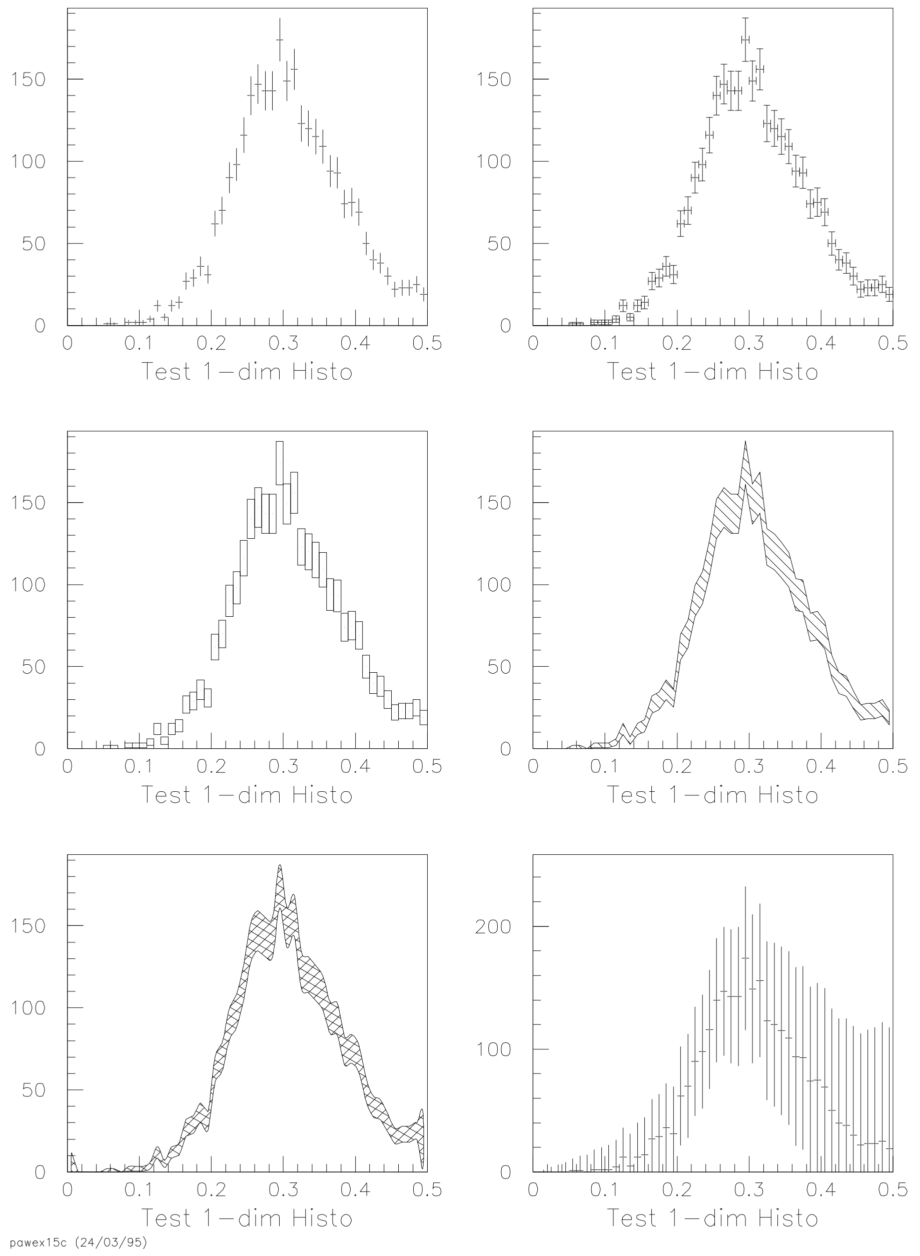


Figure 27: Exec pawex15c.kumac

Errors can be mapped on the colours

```
     zone 2 2
     fun2 2 x**2*y 20 -1 1 20 -1 1 ' '
     fun2 3 sin(x)*cos(y) 20 -6 6 20 -6 6 ' '
     h/plot 2 surf1
❶    H/PLOT 2 SURF1,E
     zone 1 2 2 s
❷    V/CR ERR(400)
❷    GET/CONT 3 ERR
❷    PUT/ERR 2 ERR
❷    H/PLOT 2 SURF1,E,Z
```

❶ 2D histograms can be plotted with errors. With options, like SURF1, in which colors are involved, the errors are mapped onto the color map.

❷ By default the errors on a histogram are the square root of the content. It is possible to define the errors with the command PUT/ERR (p 72) .

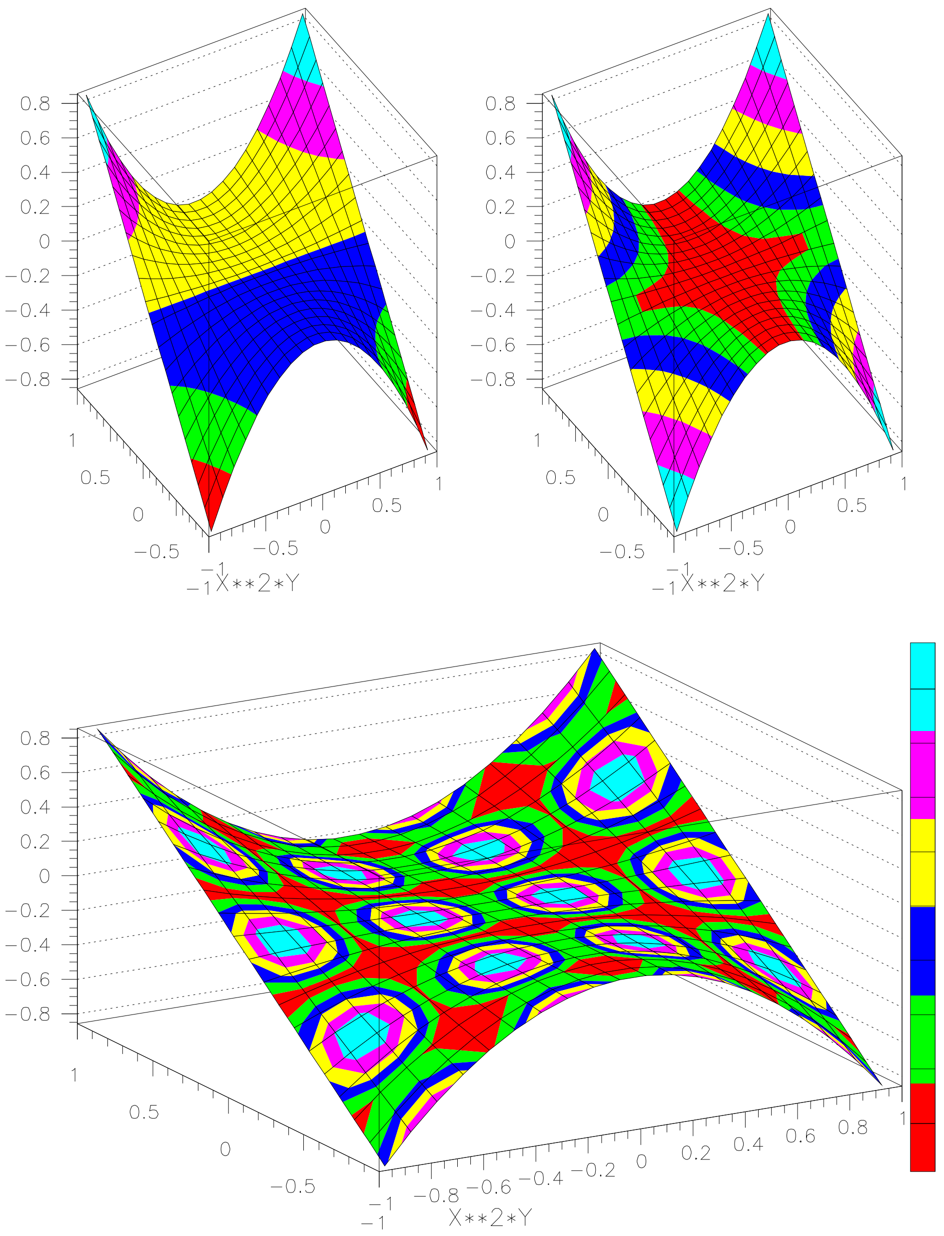


Figure 28: Exec pawex15d.kumac

This example shows how to define a new histogram representation with a **COMIS** routine.

The routine derr.f

```
    Subroutine Derr(Id)
    Character*80 Chtitl
    Real Errx(2)
    Real Erry(2)
    Real Errz(2)
❶   Call Hgive(Id,Chtitl,Nx,Xmin,Xmax,Ny,Ymin,Ymax,Nwt,Idb)
    If (Ny.Eq.0) Return
    Zmin = Hcxy(1,1,1)
    Zmax = Zmin
❷   Do i=1,Nx
       Do j=1,Ny
          Z   = Hcxy(i,j,1)
          E   = Hcxy(i,j,2)
          Ze1 = Z-E
          Ze2 = Z+E
          If (Ze1.Lt.Zmin) Zmin=Ze1
          If (Ze1.Gt.Zmax) Zmax=Ze1
          If (Ze2.Lt.Zmin) Zmin=Ze2
          If (Ze2.Gt.Zmax) Zmax=Ze2
       Enddo
    Enddo
❸   Call Hplfr3(Xmin,Xmax,Ymin,Ymax,Zmin,Zmax,30.,30.,'FWB')
    Dx = (Xmax-Xmin)/Nx
    Dy = (Ymax-Ymin)/Ny
    X  = Xmin+Dx/2.
❹   Do i=1,Nx
       Y = Ymin+Dy/2.
       Do j=1,Ny
          Z       = Hcxy(i,j,1)
          E       = Hcxy(i,j,2)
          Errx(1) = X
          Errx(2) = Errx(1)
          Erry(1) = Y
          Erry(2) = Erry(1)
          Errz(1) = Z-E
          Errz(2) = Z+E
          Call Ipm3(1,X,Y,Z)
          Call Ipl3(2,Errx,Erry,Errz)
          Y = Y+Dy
       Enddo
       X = X+Dx
    Enddo
    End
```


An other way of drawing errors for 2D histograms

How to use derr.f

```
    hi/file 0 pawhists.hbook
    hrin *
❺   SET MTYP 20
❺   CALL DERR.F(200)
```

❶ Retrieve informations about the histogram.

❷ Loop over all the bins to find the minimum and the maximum taking care of the errors.

❸ Draw the 3D frame.

❹ Loop over all the bins and draw the error bars.

❺ Set the marker type and invoke the `derr` routine.

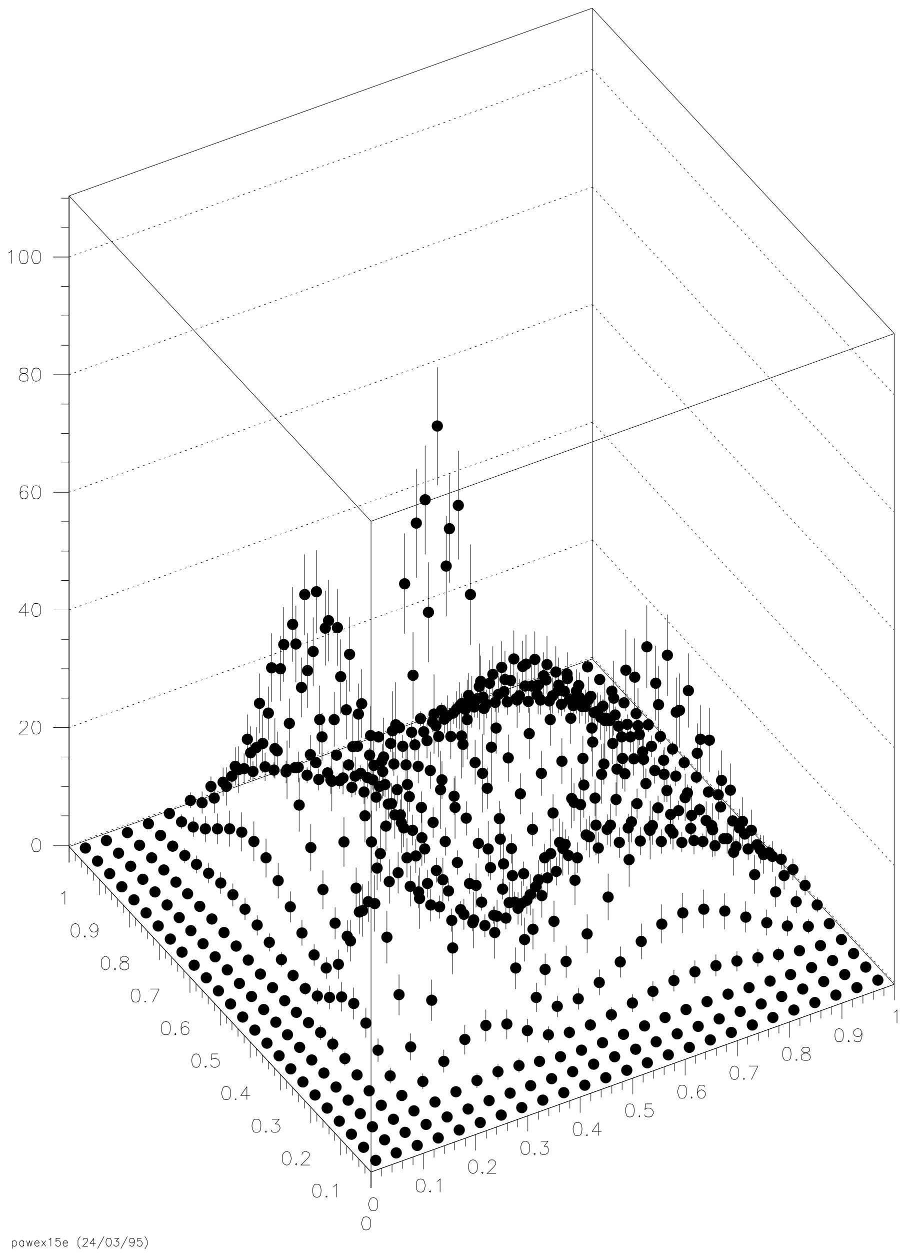


Figure 29: Exec pawex15e.kumac

Fit a background with a P3

```
     Macro PAWEX15A ID=30001 IP1=40 IP2=111 IZ1=33 IZ2=150 LOOP=20
*
     Set 2BUF 1
     Hi/file 1 pawtut.hbook ; Hrin [ID]
     Set FWID 6 ; Set DMOD 1
❶    NBIN = $HINFO([ID],'XBINS')
     Vector/Create  FUNC([NBIN])
     Vector/Create  Y([NBIN])
     Vector/Create  S([NBIN])
     Vector/Create  X([NBIN],[LOOP])
     Histogram/Copy [ID] 1
     Histogram/Copy [ID] 2
*
❷    Do i=1,[LOOP]
        Histogram/Plot 1
❸       Histogram/Fit  1([IZ1]:[IZ2]) P3 0q
❸       Get/Func 1 FUNC ; Put/Cont 2 FUNC
❸       Sub 1 2 3
❹       Histogram/Fit  3([IP1]:[IP2]) G 0q
        Histogram/Plot 3([IP1]:[IP2]) FUNCS
❹       Get/Func 3 FUNC ; Put/Cont 2 FUNC
❹       Sub 1 2 1
        Get/Func 3 X(1:[NBIN],[i])
        Call igterm
     Enddo
*
     Get/Func 1 FUNC ; Put/Cont 2 FUNC
     Sub [id] 2 3
     Zone 1 2
     Histogram/Plot [ID]
     Histogram/Plot 1 FUNCS
❺    Do i=1,[LOOP]
        Vector/Copy X(1:[NBIN],[i]) Y
        SIGMA S = S + Y
        SIGMA Y = Y+FUNC
        Put/Cont 2 Y
        Histogram/Plot 2([IP1]:[IP2]) SL
     Enddo
     Histogram/Plot 3([IP1]:[IP2]) HIST
     Put/Cont 2 S
     Histogram/plot 2([IP1]:[IP2]) Sl
*
     Close 1
     V/Del FUNC,X,Y,S ; H/Del 1,2,3
```


❶ This system function allows to retrieve informations on an histogram. Note that in a **COMIS** program information about histograms can be retrieve with the routine `HGIVE` (see example below).

❷ This loop try to find a `P3` background.

❸ After a `P3` fit, a new histogram is booked with the fit value at each channel. This new histogram is consider as an approximation of the background and is removed from the original histogram.

❹ A gaussian fit allows to remove the pick.

❺ This loop produce the two final plots.

How to retrieve histogram informations in a COMIS program

```
      subroutine hinfo(id)
      character*32 chtitl
      vector hid(6)
      call hgive(id,chtitl,ncx,xmin,xmax,ncy,ymin,ymax,nwt,loc)
      hid(1) = ncx
      hid(2) = xmin
      hid(3) = xmax
      hid(4) = ncy
      hid(5) = ymin
      hid(6) = ymax
      end
```

A more complex example

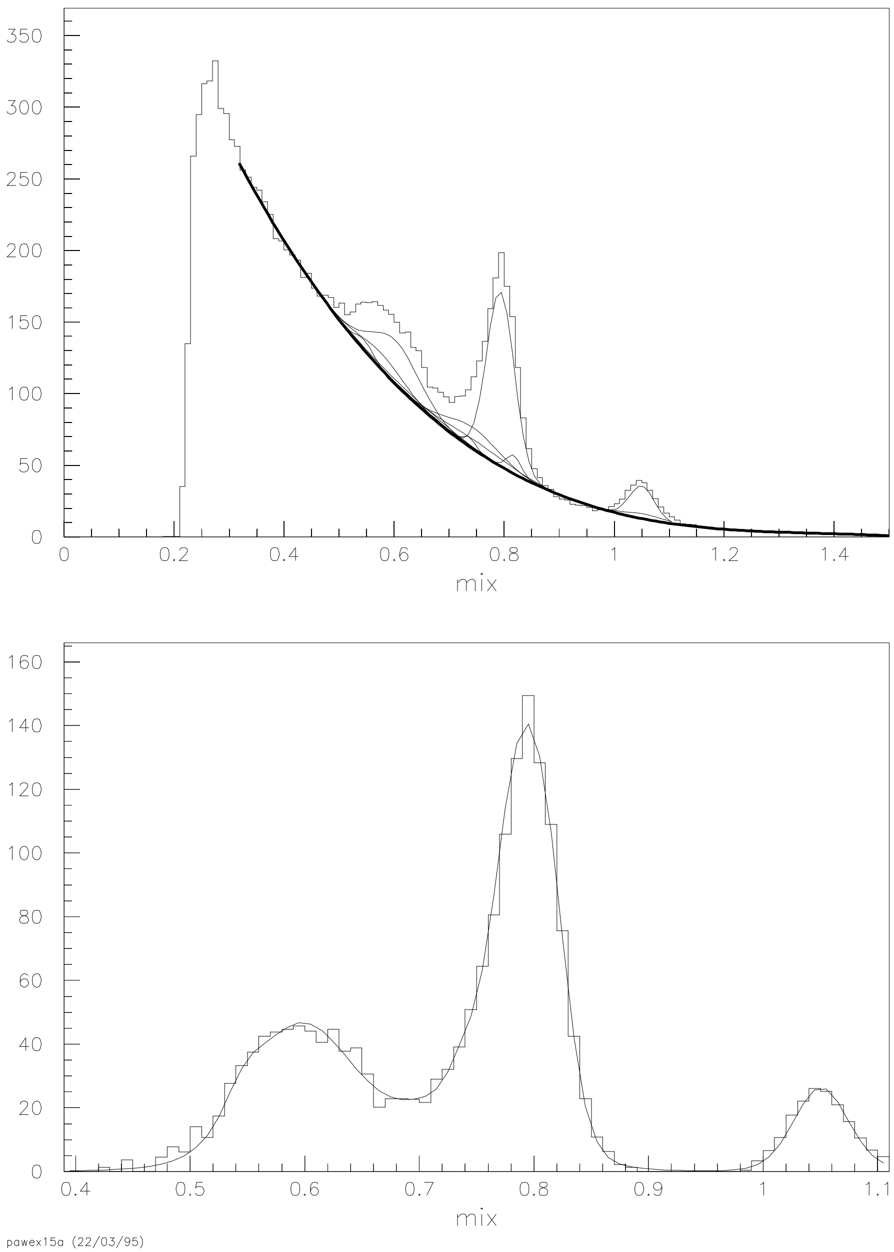


Figure 30: Exec pawex15a.kumac

Creation of an Row-Wise Ntuple (RWN) and first look at its contents

```
❶   NTUPLE/CREATE 10 'CERN Population' 11 ' ' 3500  _
    Category Division Flag Age Service Children Grade _
    Step Nation Hrweek Cost
    *
❷   NTUPLE/READ 10 APTUPLE.DAT
❸   HISTO/FILE 1 RWN_APTUPLE.HBOOK 1024 N
❸   HROUT 10
❹   NTUPLE/PRINT 10
    zone 1 2
❻   OPT STAT
❻   SET STAT 110
❹   NTUPLE/PLOT 10.Age
    ntuple/plot 10.Division
```

❶ `/NTUPLE/CREATE IDN TITLE NVAR CHRZPA NPRIME VARLIST` (p 81) : Allows to create an Ntuple. An Ntuple is a matrix of n columns. Each line of the matrix is often called an "event". Internally there is two different way to access the data: by rows (Row-Wise Ntuple) or by columns (Column-Wise Ntuple). The Ntuple may be created either in memory or, if necessary, using an automatic overflow to an histogram file.

❷ `NT/READ` (p 85) allows to fill an RW/Ntuple with numeric values read from an existing `ASCII` file.

❸ Like histograms, Ntuples are **HBOOK** objects and can be stored into histogram files opened via the command `HIST/FILE`.

❹ The command `NT/PRINT` (p 81) gives the description of the Ntuple (see next page).

❺ `NT/PLOT` (p 85) allows to plot an Ntuple. The syntax is:

`NT/PLOT nid.n ......`

where "`nid`" is the Ntuple identifier (a number) and "`n`" is the number or the name of one of the variable in the Ntuple. By default, if "`n`" is not specified, the first variable of the Ntuple is ploted.

Note also:

❻ `OPT STAT` and `SET STAT` are used to plot some statistical informations.

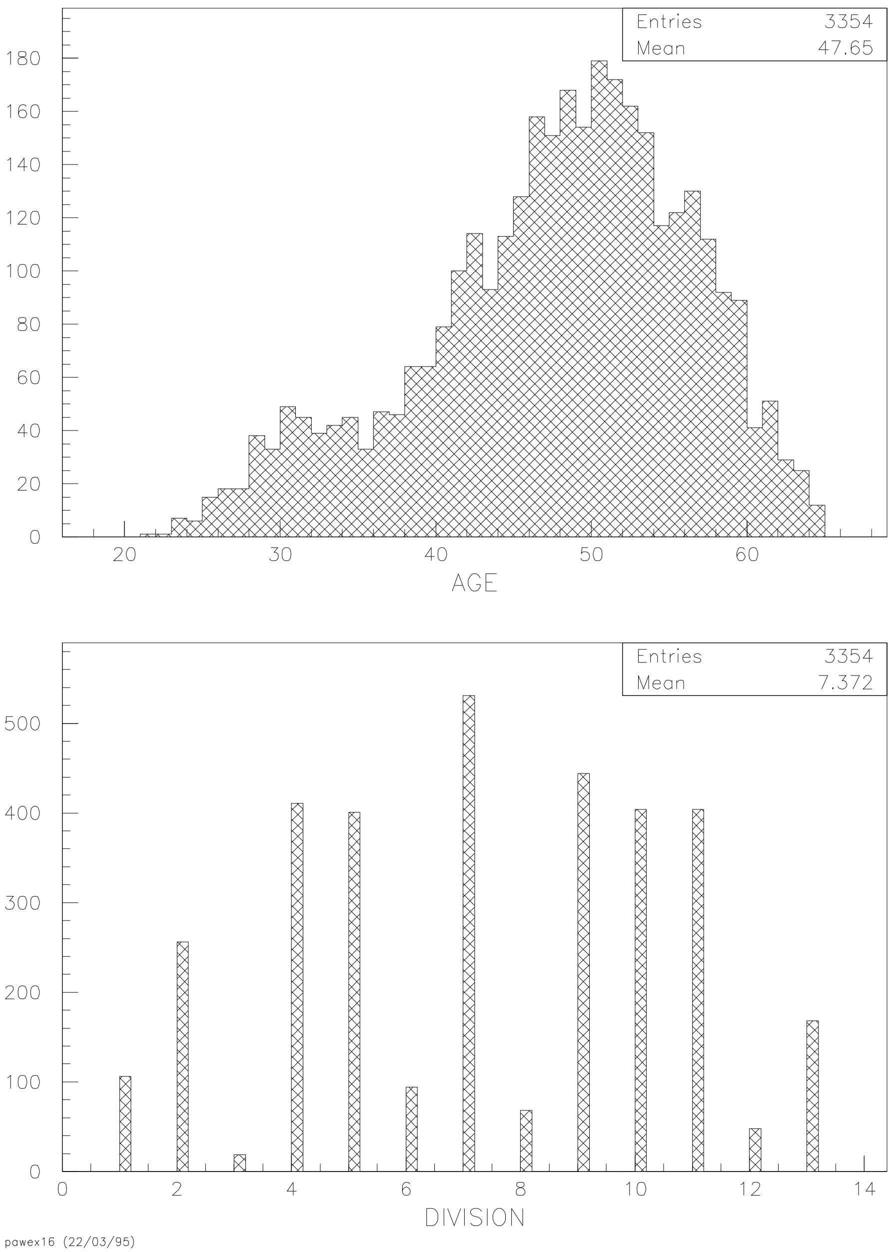


Figure 31: Exec pawex16.kumac

Ntuple creation

Creation of Column-Wise Ntuple (CWN)

```
❶    HISTO/FILE 1 CWN_APTUPLE.HBOOK 1024 N
❷    CALL CERNPOP.F
     hrout 11
     ntuple/print 11
     zone 1 2
     opt stat
     set stat 110
     ntuple/plot 11.Age
❸    NTUPLE/PLOT 11.Division
```

❶ A new **HBOOK** file is open. If the Ntuple created after doesn't fit in memory, it will be automatically write on this file.

❷ This command create and read a CW/Ntuple. It is the equivalent of the `/NTUPLE/CREATE` and `/NTUPLE/READ` commands in the previous example (for the time being these commands work only with the RWN format). For more details on the CW/Nutples management see the **HBOOK** manual.

The main advantages of the CW/Nutples compare to the RW/Nutples are:

(a) The speed: only the used data are read.

(b) The possibility to have typed variable (not only real).

(c) The compactness.

(d) The "Array variables" with a fixed or variable length.

❸ The axis are directly drawn with character labels.

Ntuple creation

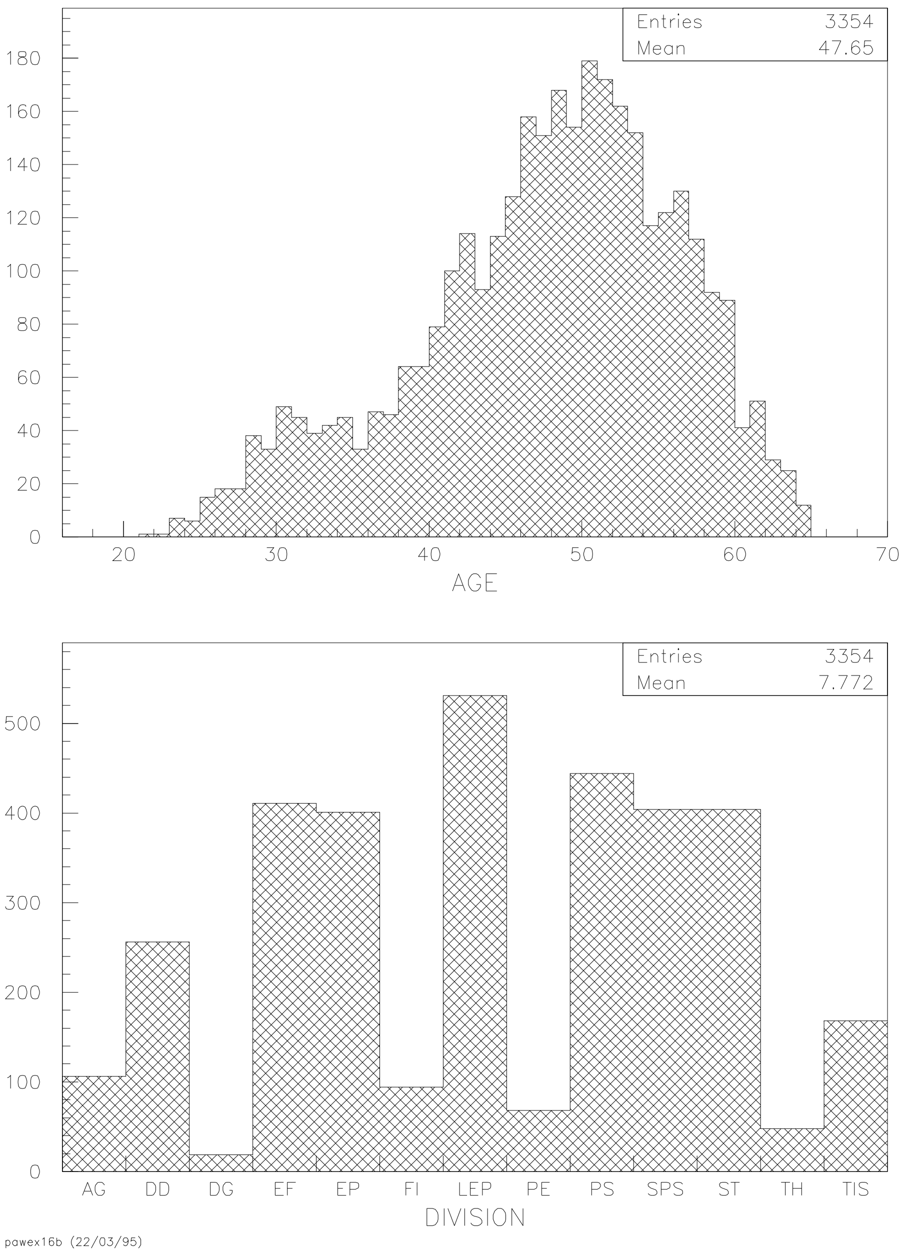


Figure 32: Exec pawex16b.kumac

Ntuple creation

COMIS routine used to create a CW/Ntuple

```
      Subroutine cernpop
*
      integer category, flag, age, service, children, grade, step,
     +        hrweek, cost
      common /cern/ category, flag, age, service, children, grade,
     +              step, hrweek, cost
      character*4   division, nation
      common /cernc/ division, nation
*
      character*132 chform
      dimension    rdata(11)
      character*4 divs(13), nats(15)
      data divs /'AG', 'DD', 'DG', 'EF', 'EP', 'FI', 'LEP', 'PE',
     +           'PS', 'SPS', 'ST', 'TH', 'TIS'/
      data nats /'AT', 'BE', 'CH', 'DE', 'DK', 'ES', 'FR', 'GB',
     +           'GR', 'IT', 'NL', 'NO', 'PT', 'SE', 'ZZ'/
*
      open(unit=41,file='aptuple.dat',status='old')
*
      call hbnt(11,'CERN Population (CWN)',' ')
      chform = ' CATEGORY[100,600]:I, FLAG:U:6, AGE[1,100]:I,'//
     +         ' SERVICE[0,60]:I, CHILDREN[0,10]:I, GRADE[3,14]:I,'//
     +         ' STEP[0,15]:I, HRWEEK:I, COST:I'
      call hbname(11, 'CERN', category, chform)
      chform = 'DIVISION:C,           NATION:C'
      call hbnamc(11, 'CERN', division, chform)
*
10    read(41, '(10F4.0, F7.0)', end=20) rdata
      category = rdata(1)
      division = divs(int(rdata(2)))
      flag     = rdata(3)
      age      = rdata(4)
      service  = rdata(5)
      children = rdata(6)
      grade    = rdata(7)
      step     = rdata(8)
      nation   = nats(int(rdata(9)))
      hrweek   = rdata(10)
      cost     = rdata(11)
      call hfnt(11)
      goto 10
*
  20  close (41)
      end
```


Ntuple creation

RWN NT/PRINT output

```
 ********************************************************
 * 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      * 0.686000E+03 * 0.188530E+05 *
 ********************************************************
```

CWN NT/PRINT output

```
 ******************************************************************
 * Ntuple ID = 11     Entries = 3354      CERN Population (CWN)
 ******************************************************************
 * Var numb * Type * Packing *    Range     *  Block   *  Name    *
 ******************************************************************
 *      1   * I*4  *    11   * [100,600]    * CERN     * CATEGORY
 *      2   * U*4  *    6    *              * CERN     * FLAG
 *      3   * I*4  *    8    * [1,100]      * CERN     * AGE
 *      4   * I*4  *    7    * [0,60]       * CERN     * SERVICE
 *      5   * I*4  *    5    * [0,10]       * CERN     * CHILDREN
 *      6   * I*4  *    5    * [3,14]       * CERN     * GRADE
 *      7   * I*4  *    5    * [0,15]       * CERN     * STEP
 *      8   * I*4  *         *              * CERN     * HRWEEK
 *      9   * I*4  *         *              * CERN     * COST
 *     10   * C*4  *         *              * CERN     * DIVISION
 *     11   * C*4  *         *              * CERN     * NATION
 ******************************************************************
 *  Block   * Unpacked Bytes * Packed Bytes *   Packing Factor    *
 ******************************************************************
 * CERN     *    44          *    22        *    2.000            *
 * Total    *    44          *    22        *    2.000            *
 ******************************************************************
 * Number of blocks = 1     Number of columns = 11                *
 ******************************************************************
```

Read an Ntuple from a histogram file. Automatic and user binning

```
    hi/file 2 rwn_aptuple.hbook
    zon 2 2
    ntuple/pl 10.age
    1dhisto 11 'Age - User binning' 45 20. 65.
❷   SET NDVX -509
❶   NTUPLE/PROJECT 11 10.AGE
    hi/plot 11
    1dhisto 12 'Cost - User binning' 50 0. 20000.
❷   SET NDVX
    ntuple/plot 10.cost
    set ndvx -504
❸   NTUPLE/PLOT 10.COST IDH=12
```

❶ `NT/PROJECT` (p 85) Project an Ntuple onto a 1-Dim or 2-Dim histogram. The histogram is not reset before the projection. This allows several PROJECTs from different Ntuples.

❷ By default the labeling on the axis is automatic. It possible to change the number of division via the commands `SET NDVX`, `SET NDVY` and `SET NDVZ`. The number of divisions (`NDIV`) is calculated according to the following convention:

`(NDIV = N1 + 100*N2 + 10000*N3)`

Where `N1` is the number of primary divisions, `N2` is the number of second order divisions and `N3` is the number of third order divisions.

The sign of `NDIV` is also used to control the labeling:

(a) If `NDIV` is positive, it is taken as a maximum number and the binning is optimized.

(b) If `NDIV` is negative, its absolute value is taken as the exact number of division without optimization.

(c) If `NDIV` equal zero is given the default (510. i.e. 10 primary divisions and 5 secondary) is taken.

The number of primary divisions is also optimized according the number of zones (p 102) . i.e : along the X direction the number of primary divisions is divided by `the number of X zones` along the Y direction the number of primary divisions in divided by `(the number of Y zones)/2`

❸ The variable `COST` is ploted according to the binning defined by the histogram 12. When the parameter `IDH` is not used in the command `NTUPLE/PLOT`, a histogram is automatically created with the identifier 1000000.

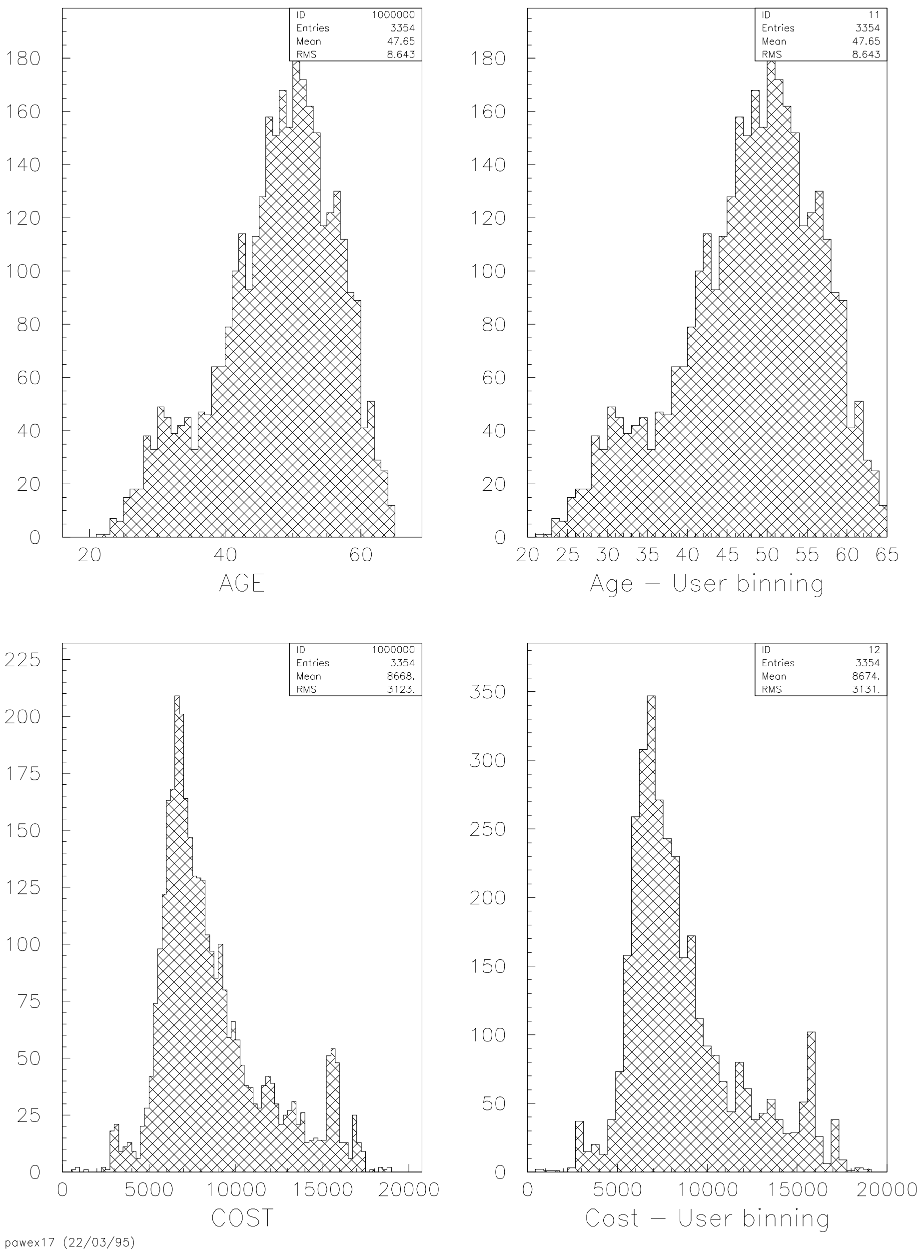


Figure 33: Exec pawex17.kumac

Ntuple SCAN and the use of simple selection criteria

```
      hi/file 2 rwn_aptuple.hbook
      hi/file 3 cwn_aptuple.hbook
❹     ALIAS/CREATE DIVEP 5
      alias/create NATFR 7
      cd //pawc
      *
❶❷    NT/SCAN //LUN2/10 nation=NATFR.and.division=DIVEP _
      ! ! ! age service children grade nation
❶❷    NT/SCAN //LUN3/11 nation='FR'.and.division='EP'  _
      ! ! ! age service children grade nation
      *
      hi/cr/1d 200 'Number of years at CERN' 35 0. 35.
      max 200 250
      set ndvx 507
      set htyp 235
❸     NT/PL //LUN2/10.SERVICE IDH=200
❺     ATITLE 'Years at CERN' 'Number of staff'
      set htyp 253
❷❸    NT/PL //LUN3/11.SERVICE NATION='FR' OPTION=S IDH=200
      set htyp 250
      nt/pl //LUN3/11.Service division='EP'.and.nation='FR' OPTION=S IDH=200
```

❶ `NT/SCAN` (p 84) prints in an alphanumeric way the content of an Ntuple. On the next page is given the output of this command both for RW and CW Ntuples.

❷ In the commands `NT/PLOT` and `NT/SCAN`, the second parameter is the selection criteria. Only the events satisfying this selection are taken into account. Note that with CWN character strings can be used.

❸ By default `NT/PLOT` fill an histogram with the indentifier `IDH=1000000`. The next invocation of this command will overwrite the content of this histogram. IDH may have been created with `H/CREATE`. Before filling `IDH`, the contents of `IDH` are reset if `IDH` already exists. Note that `IDH` not equal to 1000000 is a convenient way to force user binning. This is used here.

We'll see later another way to fill an histogram with data read in an Ntuple.

Note also:

❹ The aliases allow to define shortcut abbreviations. The aliases are known globally e.g. in all macros and in command mode.

❺ `ATITLE` (p 125) allows to define the title on the axis.

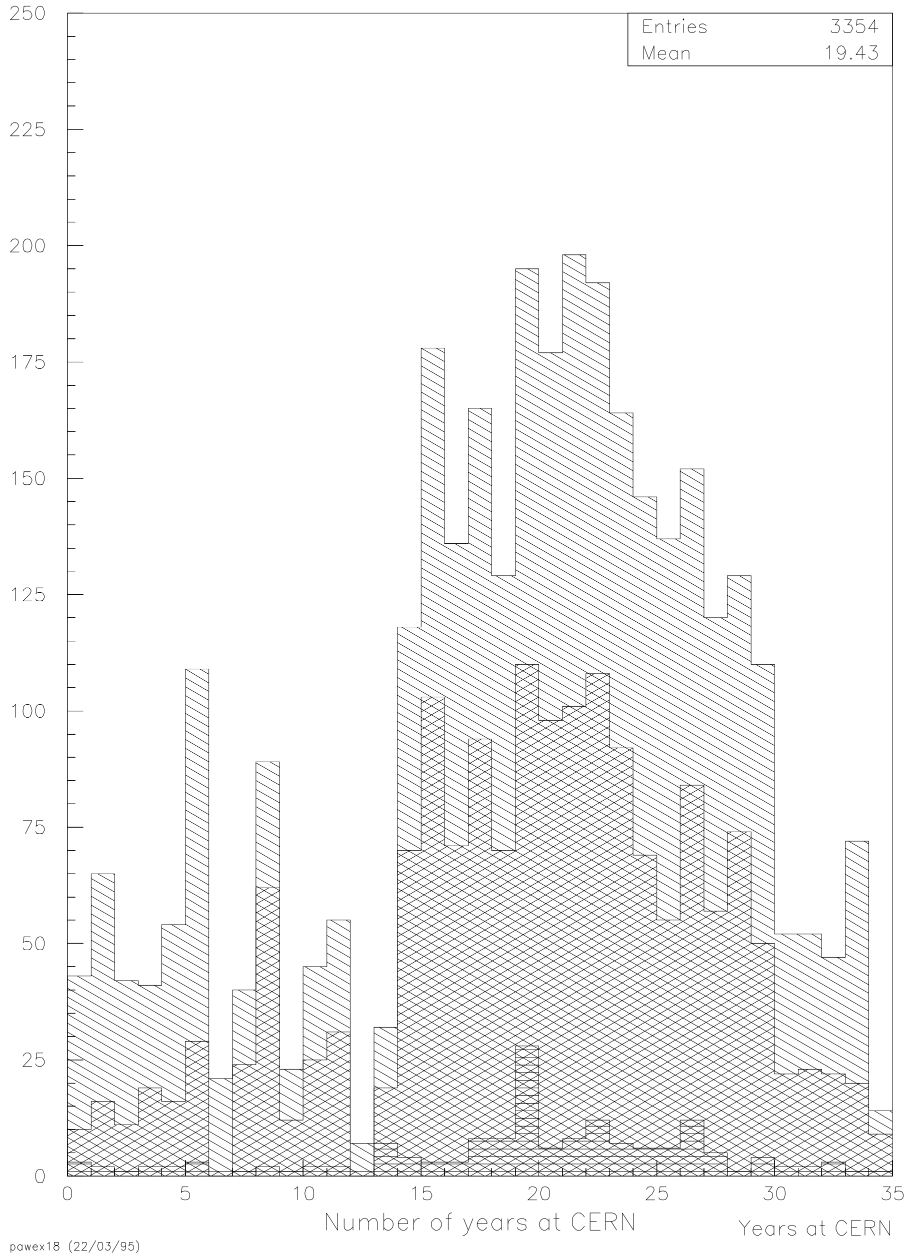


Figure 34: Exec pawex18.kumac

NT/SCAN output for a Row Wise Ntuple

```
+-------+-------------+-------------+-------------+-------------+-------------+
| ENTRY |   AGE       |   SERVICE   |   CHILDREN  |   GRADE     |   NATION    |
+-------+-------------+-------------+-------------+-------------+-------------+
|    48 |  56.000     |  34.000     |  .00000E+00|  7.0000     |  7.0000     |
|   194 |  62.000     |  27.000     |  .00000E+00|  7.0000     |  7.0000     |
|   213 |  56.000     |  26.000     |  .00000E+00|  6.0000     |  7.0000     |
|   214 |  45.000     |  26.000     |  .00000E+00|  6.0000     |  7.0000     |
|   216 |  56.000     |  19.000     |  .00000E+00|  5.0000     |  7.0000     |
|   266 |  63.000     |  26.000     |  .00000E+00|  13.000     |  7.0000     |
|   267 |  59.000     |  32.000     |  .00000E+00|  13.000     |  7.0000     |
|   273 |  55.000     |  26.000     |  1.0000     |  12.000     |  7.0000     |
|   275 |  53.000     |  26.000     |  1.0000     |  11.000     |  7.0000     |
|   279 |  51.000     |  30.000     |  .00000E+00|  6.0000     |  7.0000     |
|   315 |  56.000     |  25.000     |  .00000E+00|  8.0000     |  7.0000     |
|   318 |  64.000     |  26.000     |  .00000E+00|  6.0000     |  7.0000     |
|   320 |  49.000     |  26.000     |  .00000E+00|  6.0000     |  7.0000     |
|   327 |  59.000     |  19.000     |  .00000E+00|  5.0000     |  7.0000     |
|   328 |  51.000     |  25.000     |  .00000E+00|  5.0000     |  7.0000     |
+-------+-------------+-------------+-------------+-------------+-------------+
More...? ( <CR>/N/G ) n
 ==>     15 events have been scanned
```

NT/SCAN output for a Column Wise Ntuple

```
+-------+-------------+-------------+-------------+-------------+-------------+
| ENTRY |   AGE       |   SERVICE   |   CHILDREN  |   GRADE     |   NATION    |
+-------+-------------+-------------+-------------+-------------+-------------+
|    48 | 56          | 34          | 0           | 7           | FR          |
|   194 | 62          | 27          | 0           | 7           | FR          |
|   213 | 56          | 26          | 0           | 6           | FR          |
|   214 | 45          | 26          | 0           | 6           | FR          |
|   216 | 56          | 19          | 0           | 5           | FR          |
|   266 | 63          | 26          | 0           | 13          | FR          |
|   267 | 59          | 32          | 0           | 13          | FR          |
|   273 | 55          | 26          | 1           | 12          | FR          |
|   275 | 53          | 26          | 1           | 11          | FR          |
|   279 | 51          | 30          | 0           | 6           | FR          |
|   315 | 56          | 25          | 0           | 8           | FR          |
|   318 | 64          | 26          | 0           | 6           | FR          |
|   320 | 49          | 26          | 0           | 6           | FR          |
|   327 | 59          | 19          | 0           | 5           | FR          |
|   328 | 51          | 25          | 0           | 5           | FR          |
+-------+-------------+-------------+-------------+-------------+-------------+
More...? ( <CR>/N/G ) n
 ==>     15 events have been scanned
```

A graphical scan of the ntuples

```
    hi/file 2 rwn_aptuple.hbook
    nt/print 10
❶   ZONE 2 3
❹   SET HTYP 245
❷❹  NT/SCAN 10 OPTION=SA CATEGORY AGE:STEP COST
```

❶ Each “spider” is diplayed according to the current zone setting.

❷ When the option S (Spider plot) is specified, each event is presented in a graphical form (R versus PHI plot) to give a multi dimensional view of the event. Each variable is represented on a separate axis with a scale ranging from the minimum to the maximum value of the variable. A line joins all the current points on every axis where each point corresponds to the current value of the variable. The option “A” allows to display the “Average” spider.

❸ The list of variable to be scanned may contain a list of the original variables, expressions of the original variables or/and ranges of variables. A range can be given in the following form:

```
:          means all variables (default).
var1:var2  means from variable var1 to variable var2 included.
var1:      means from variable var1 to the last.
:var2      means from variable 1 to variable var2
```

❹ The Spider plot is drawn according to the histogram attributes (HCOL, HTYP etc ...)

Option “Spider” in NTUPLE/SCAN

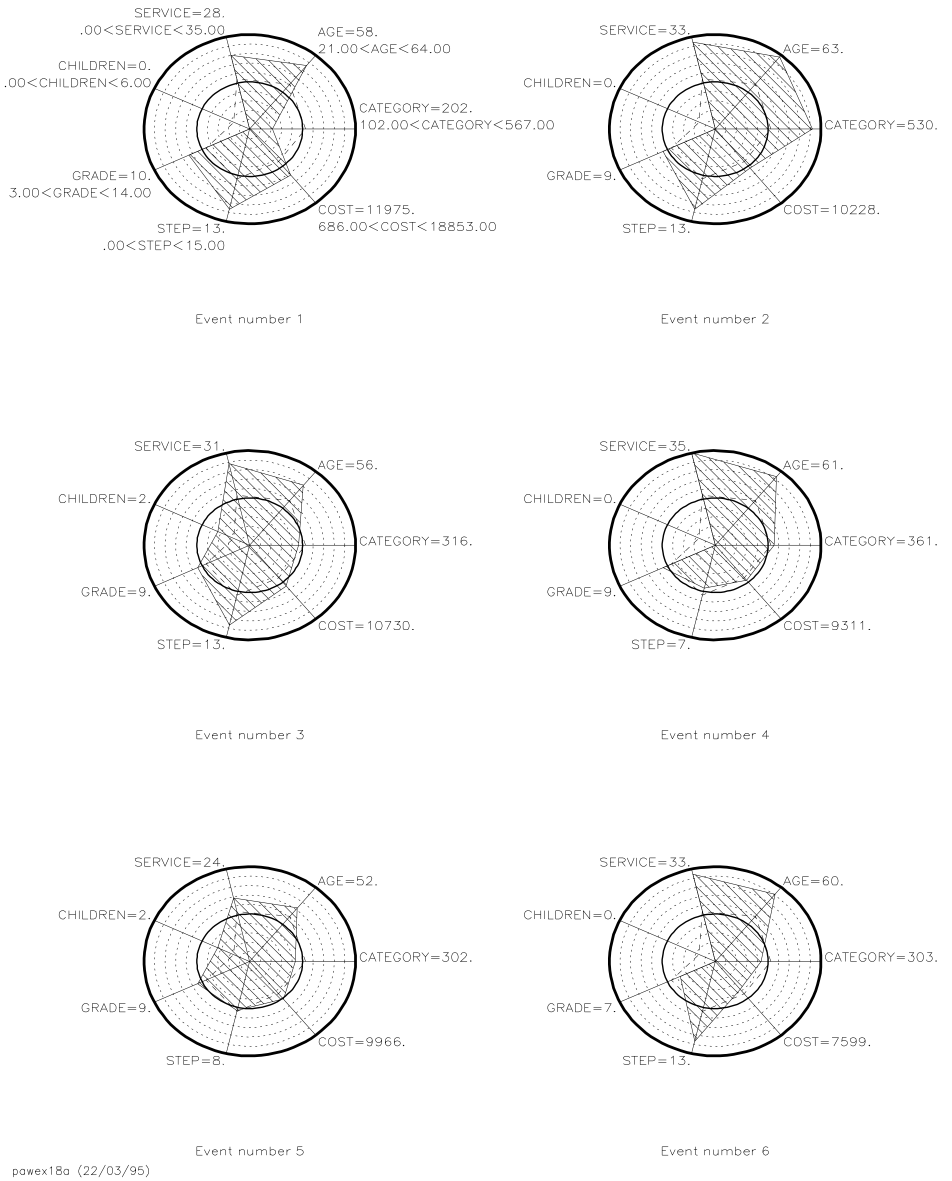


Figure 35: Exec pawex18a.kumac

Use of Ntuple masks and loops

```
    hi/file 2 rwn_aptuple.hbook
    1dhisto 20 'Distribution by grade' 12 3 15
    max 20 700
    ntuple/plot 10.grade IDH=20
❶   NT/MASK STMASK n 3500
❷   NT/LOOP 10 STEP=15>>STMASK(1)
    nt/loop 10 grade>4.and.step=13>>stmask(2)
    nt/loop 10 _
    (grade=13.and.step=10).or.(grade=14.and.step=7)>>stmask(3)
    NT/PLOT 10.GRADE _
    STMASK(1).OR.STMASK(2).OR.STMASK(3)>>STMASK(4) OPTION=S IDH=20
❸   NT/MASK STMASK P
❹   NT/MASK STMASK C
    set CHHE 0.35
❺   EXEC LEGEND 245 9.3 10.3 610 640 'All Staff'
❺   EXEC LEGEND 244 9.3 10.3 560 590 'Staff at end of grade'
```

❶ `NT/MASK` (p 90) perform operations with masks. A mask is a direct-access file with the name `MNAME.MASK` (here `STMASK.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.

❷ The symbol "≫" in `NT/LOOP` (p 84) and `NT/PLOT` allows to fill the mask according to the selection function.

❸ This command allows to print the definition of the mask.

❹ The option "`C`" in `NT/MASK` close the mask.

❺ A general macro to draw a legend (see next page).

❻ Try `NT/PLOT 10.GRADE STMASK(4)`: It produce the same result as the last `NT/PLOT` of the macro.

❼ Compare the execution time (with `TIMING`) of the two following commands:

```
NTUPLE/PLOT 10.GRADE (GRADE=13.AND.STEP=10).OR.(GRADE=14.AND.STEP=7)
NTUPLE/PLOT 10.GRADE STMASK(3)
```

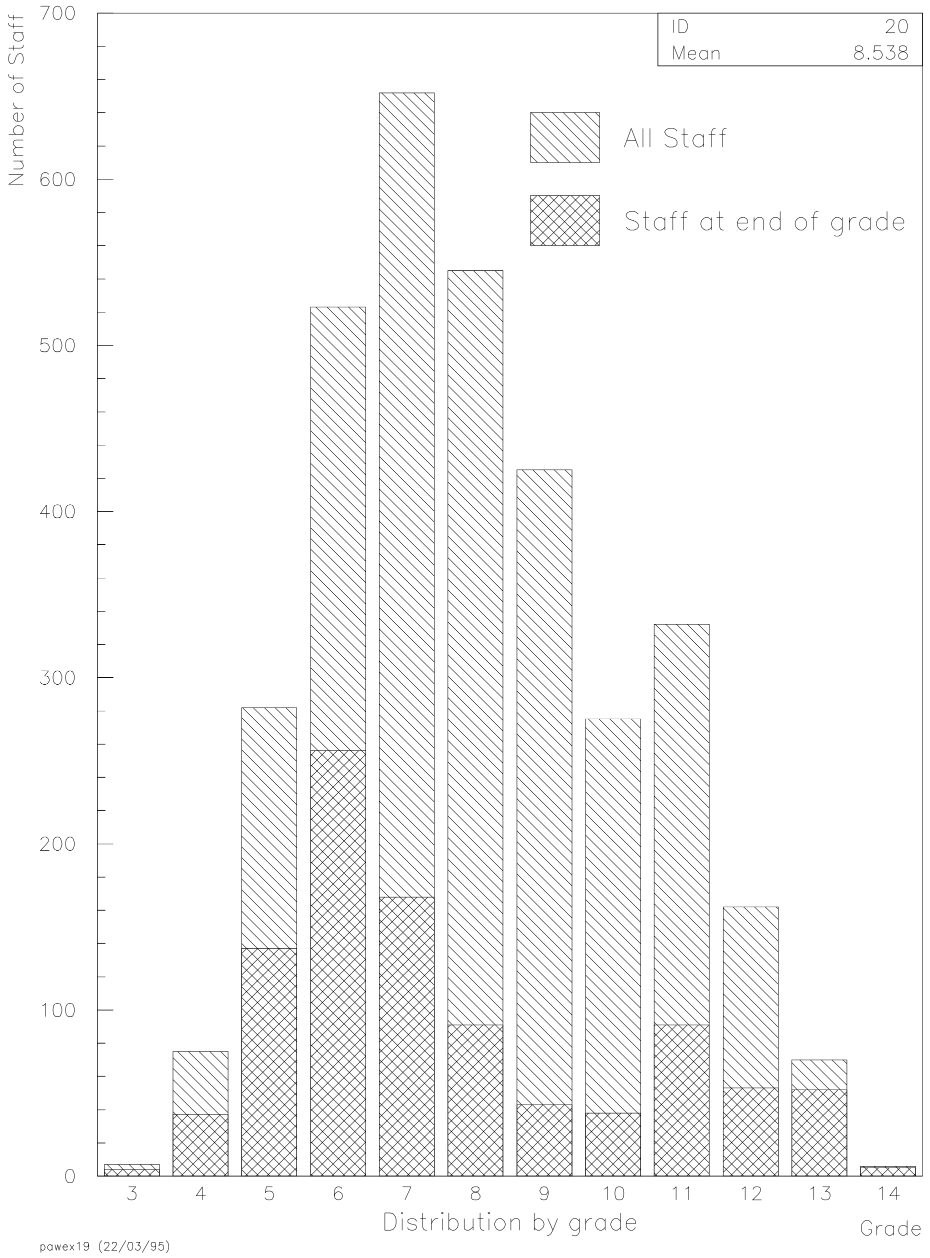



Figure 36: Exec pawex19.kumac

Use of Ntuple masks and loops

Output of the command NT/MASK STMASK P

```
=====> Current active selections in mask STMASK

Bit  Nevents    Selection
  1      41     STEP=15

  2     877     GRADE>4.AND.STEP=13

  3      57     (GRADE=13.AND.STEP=10).OR.(GRADE=14.AND.STEP=7)

  4     975     STMASK(1).OR.STMASK(2).OR.STMASK(3)
```

A general macro to draw a legend

```
     Macro Legend
❶    TYPE = [1]    | Type of hatches
❶    X1   = [2]    | X bottom left corner of the box.
❶    X2   = [3]    | X top right corner of the box.
❶    Y1   = [4]    | Y bottom left corner of the box.
❶    Y2   = [5]    | Y top right corner of the box.
❶    TEXT = [6]    | Text to be printed
     Set FAIS 3
     Set FASI [TYPE]
     Set BORD 1
     Box [X1] [X2] [Y1] [Y2]
     Set TXAL 03
❷    XT = [X2]+$GRAFINFO('?CHHE')
❷    YT = ([Y2]+[Y1])/2
❷    Itx [XT] [YT] [TEXT]
     Return
```

❶ Input parameters 1 to 6 are copied into locale variables with meaningful names.

❷ The text is positionned according to the text size and the box position.

The use of Ntuple Cuts

```
   hi/file 2 rwn_aptuple.hbook
❶  CUT $1 MOD(FLAG,2).EQ.0
❶  CUT $2 MOD(FLAG,4)>1
   1d 20 'Male/female and resident/non-resident Staff' 13 1 14
❷  OPT BAR
❷  SET BARW 0.4
❷  SET BARO 0.1
   max 20 600
❸  LABELS 1 13 AG DD DG EF EP FI LEP PE PS SPS ST TH TIS
   set ndvx 13.15
   set ndvy -506
   ntuple/plot 10.division IDH=20
   set htyp 244
   ntuple/plot 10.division $2        option=S idh=20
   set baro 0.5
   set htyp 145
   ntuple/plot 10.division $1        option=S idh=20
   set htyp 154
   ntuple/plot 10.division $1.and.$2 option=S idh=20
   atitle 'Division' 'Number of staff'
```

❶ NTUPLE/CUTS (p 89) allows to manipulate cuts. A cut identifier has the format $nn.

It is possible to store the cuts in a file with the option "W" and read them afterwards with the option "R". When a cut is defined it can be used in commands like NT/PLOT, NT/PROJ etc ...

It also possible to define "graphical cuts" with the option "G". These cuts are specified interactively with the mouse.

Graphical cuts are only operational for plots of the original Ntuple variables, not for expressions of these variables.

Note also:

❷ The "BAR" option and the attributes "BARW" and "BARO" allow to draw bar charts. OPTION BAR is also active on LEGO plots.

❸ LABELS (p 116) used with SET NDVX or SET NDVY allows to produce alphanumeric labeling on histograms with a numerical labelling.

❹ Histograms with alphanumeric binning are now available in **HBOOK**. A set of routines is available to manage such histograms. In **PAW**, the command SORT (p 66) allows to reorder the labels. The next example demonstrates the possibilities offer by the SORT command.

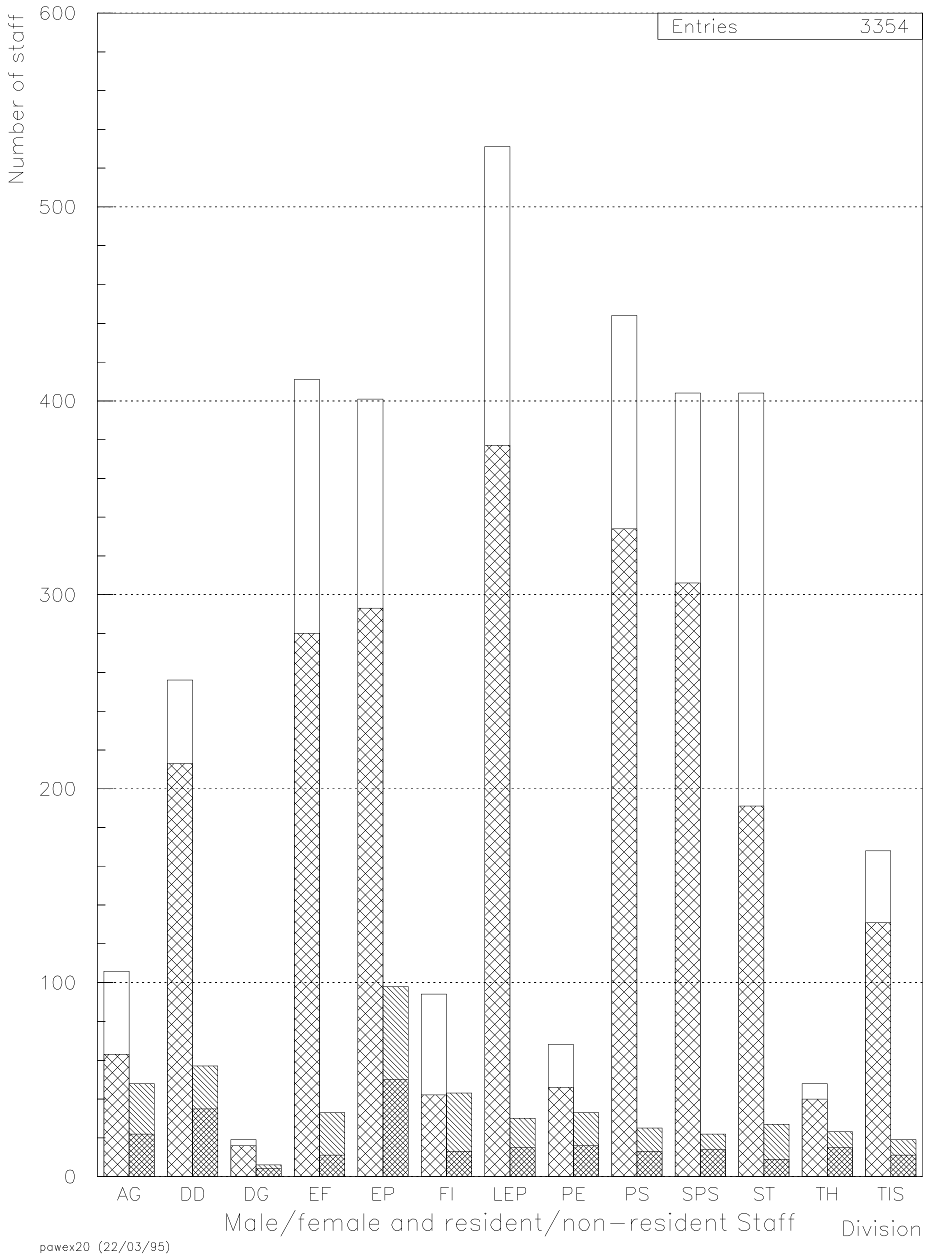


Figure 37: Exec pawex20.kumac

The command SORT usage

```
    hi/file 2 cwn_aptuple.hbook
    zone 1 3
❶   NTUPLE/PLOT 11.DIVISION
❷   SORT 1000000 XD
❸   H/PLOT 1000000
❷   SORT 1000000 XV
❸   H/PLOT 1000000
```

❶ Create the histogram 1000000. This is an histogram with character labelling.

❷ Reorder the labels according to the option.

❸ Plot the histogram 1000000 with reordered labels.

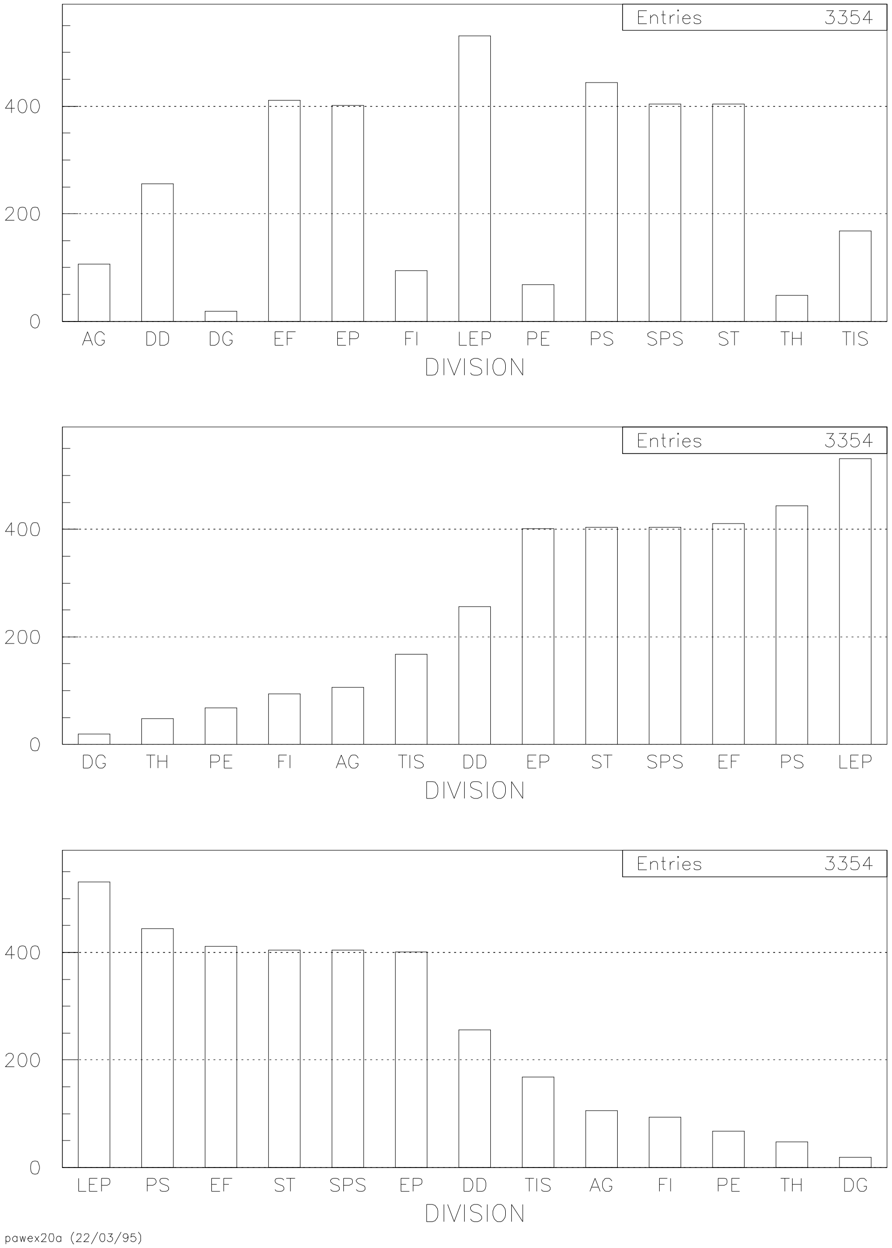


Figure 38: Exec pawex20a.kumac

Ntuple and 2D histograms: profile histograms

How to create a profile histogram from a Ntuple

```
   hi/file 2 rwn_aptuple.hbook
   zone 1 2
   set MTYP 3
❶  NT/PLOT //LUN2/10.age%grade
❷  NT/PLOT //LUN2/10.age%grade option=prof
```

❶ The symbol "%" is used to produce multiple dimensional distributions with ntuples. The maximum number of dimension is 10. The command `NT/PLOT` produce a bi-dimensional distribution represented as a scatter plot with the current marker type.

❷ When the option `PROF` is used, a profile histogram is produce. A profile histogram, is a 1D histogram which gives for each value of `X` the mean value of `Y` and its RMS (for more details see the **HBOOK** manual: routine `HBPROF`).

Ntuple and 2D histograms: profile histograms

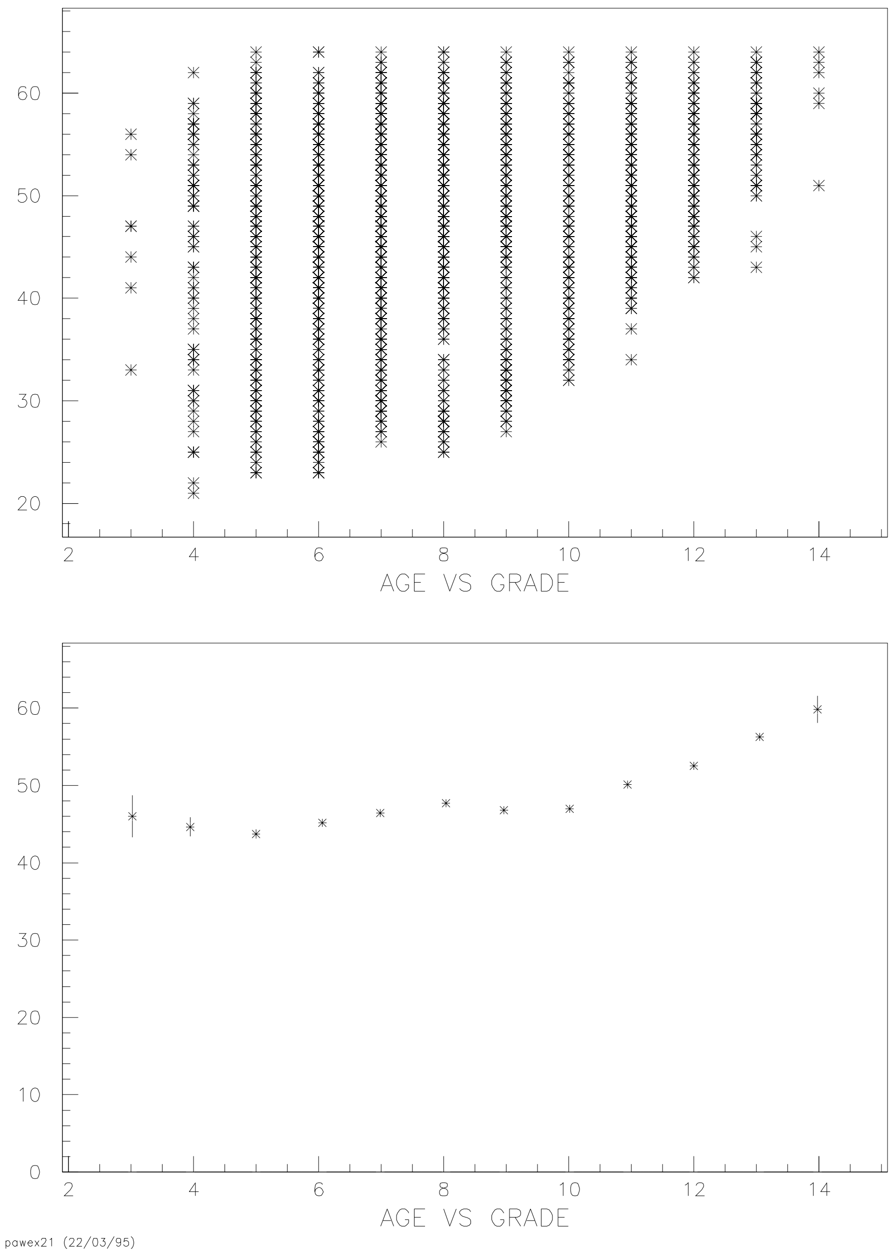


Figure 39: Exec pawex21.kumac

2D Ntuple distributions and 2D histograms projections

```
   hi/file 2 rwn_aptuple.hbook
   clr
   2d 20 ' ' 12 3 15 16 0 16 0.
❶  NT/PROJECT 20 //lun2/10.STEP%GRADE
   lego 20 20 40
❷  PROX 20
❸  H/PRO 20
❹  H/PLOT 20.prox
```

❶ NT/PROJ (p 85) allows to fill an histogram with data read in a Ntuple without plotting the result.

❷ Create the projection onto the x axis. The commands PROX (p 59) , PROY (p 60) , SLIX (p 60) , SLIY (p 60) , BANX (p 60) and BANY (p 60) allows to define projections.

❸ Fill the projection (p 52) .

❹ Plot the projection.

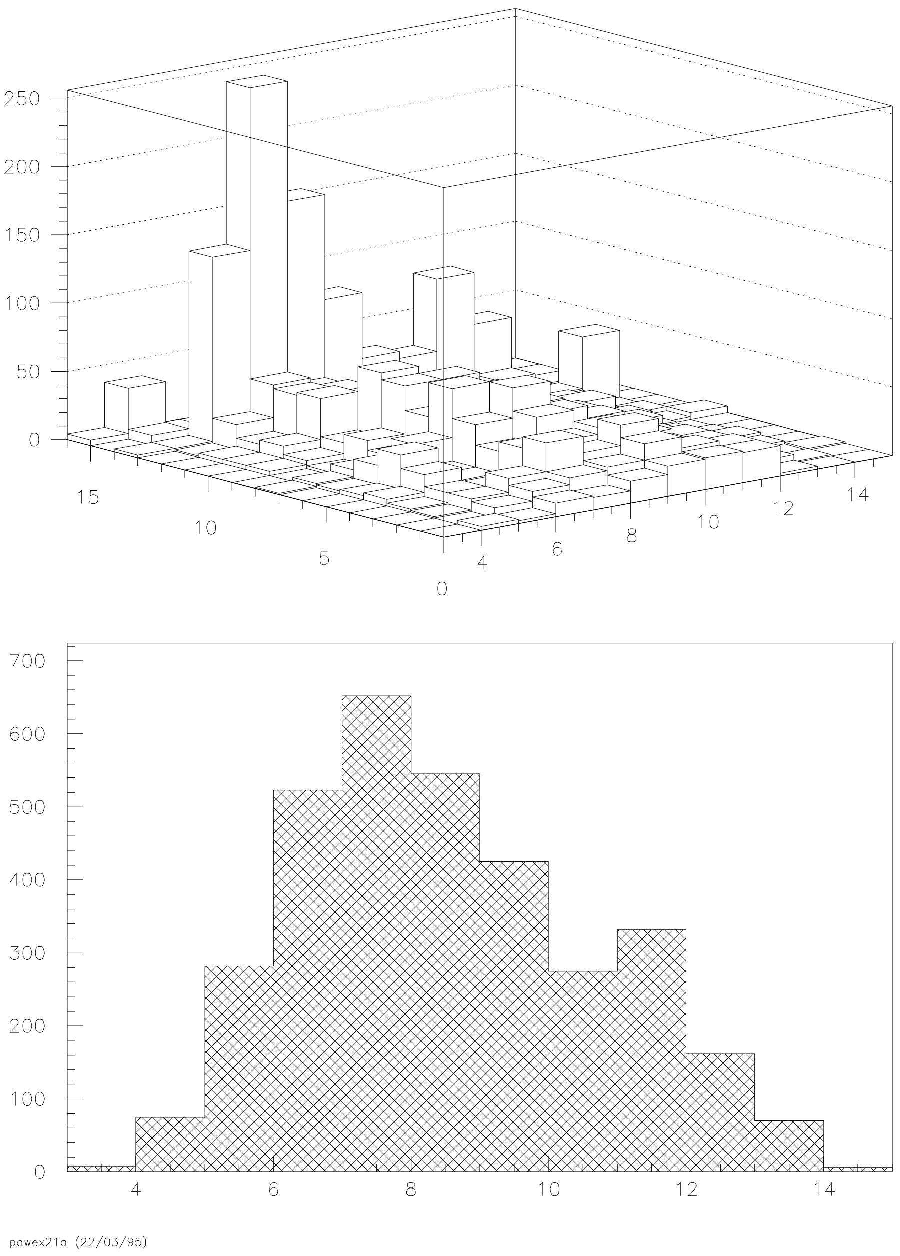


Figure 40: Exec pawex21a.kumac

Copy a Ntuple variable into a Vector

```
         hi/file 2 rwn_ptuple.hbook
❶        UWFUNC 10 copy.f E
❷        NT/LOOP 10.age copy.f
         zone 1 2
         vect/draw x
         vect/plot x
```

The routine copy.f

```
      REAL FUNCTION COPY(XDUMMY)
      REAL
     +CATEGORY,DIVISION,FLAG    ,AGE     ,SERVICE ,CHILDREN,
     +GRADE   ,STEP    ,NATION  ,HRWEEK  ,COST
      COMMON/PAWIDN/IDNEVT,VIDN1,VIDN2,VIDN3,VIDN(10),
     +CATEGORY,DIVISION,FLAG    ,AGE     ,SERVICE ,CHILDREN,
     +GRADE   ,STEP    ,NATION  ,HRWEEK  ,COST
*
❸     VECTOR X(3354)
      X(IDNEVT)=VIDN1
      END
```

❶ This command (p 91) allows to define the skeleton of the FORTRAN routine used by NTUPLE/LOOP (p 84) .

❷ For each event, NTUPLE/LOOP calls copy.f.

❸ The declaration VECTOR 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.

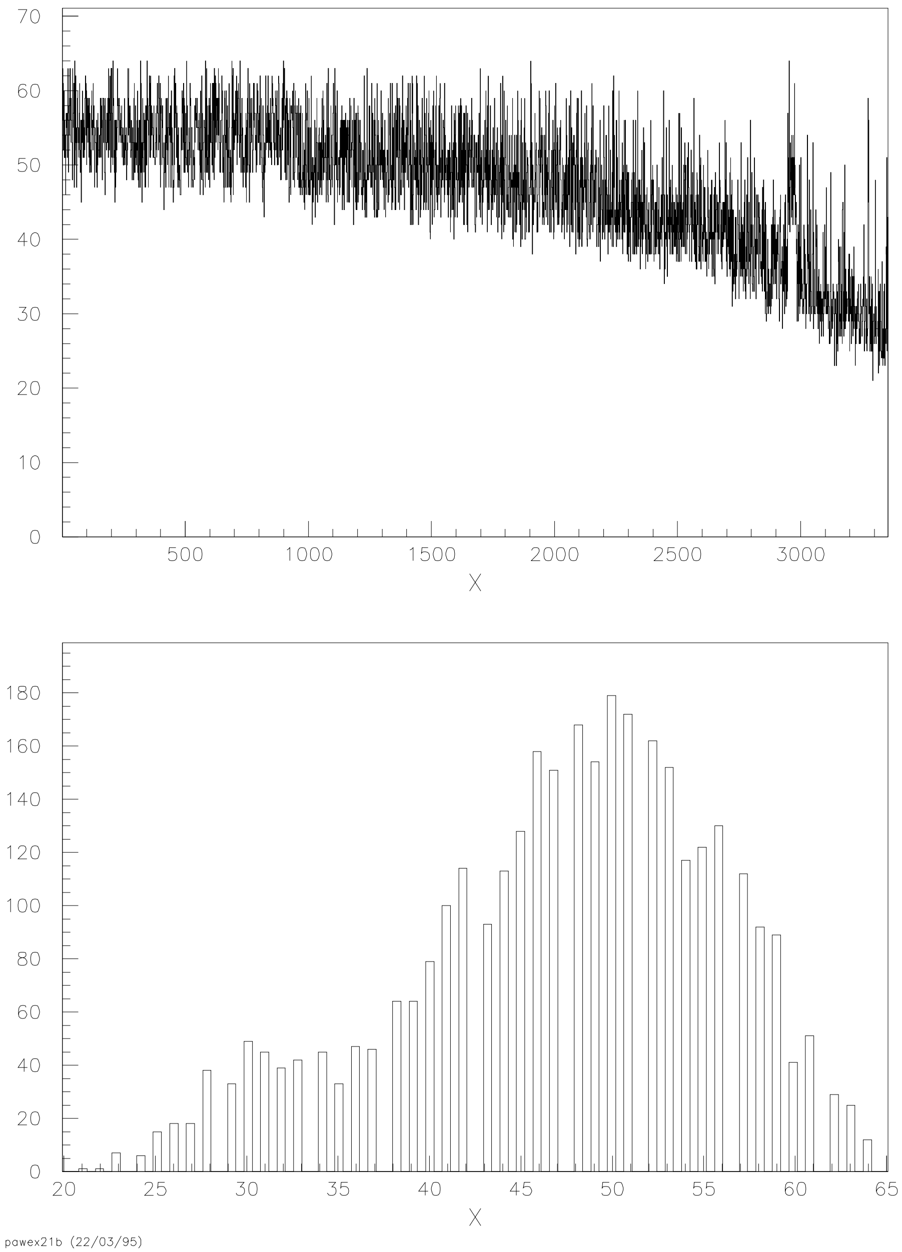


Figure 41: Exec pawex21b.kumac

This example create a new "CERN population" Ntuple with all the people of 50 years old counted twice.

Usage of NTUPLE/MERGE and NTUPLE/DUPLICATE

```
    opt stat
    hi/file 1 rwn_aptuple.hbook
❶   HI/FILE 2 RWN_APTUPLE_MERGE.HBOOK ! N
❷   NTUPLE/DUPLICATE //LUN1/10 12 title='CERN Population +50'
    hrout 12
❸   NTUPLE/MERGE //LUN1/10 //LUN2/12
    hrout 12
❸   NTUPLE/MERGE //LUN1/10 //LUN2/12 AGE=50
    hrout 12
    nt/plot //lun2/12.age
```

❶ Create a new hbook file.

❷ The command `NTUPLE/DUPLICATE` (p 82) create an empty Ntuple (here 12) with the same structure as a reference one (here //lun1/10).

❸ The command `NTUPLE/MERGE` (p 85) allows to concatanate Ntuple. Note that some selection criteria can be apply on the first ntuple identifier. This command acts only one Row Wise Ntuples.

Duplicate and merge Ntuples

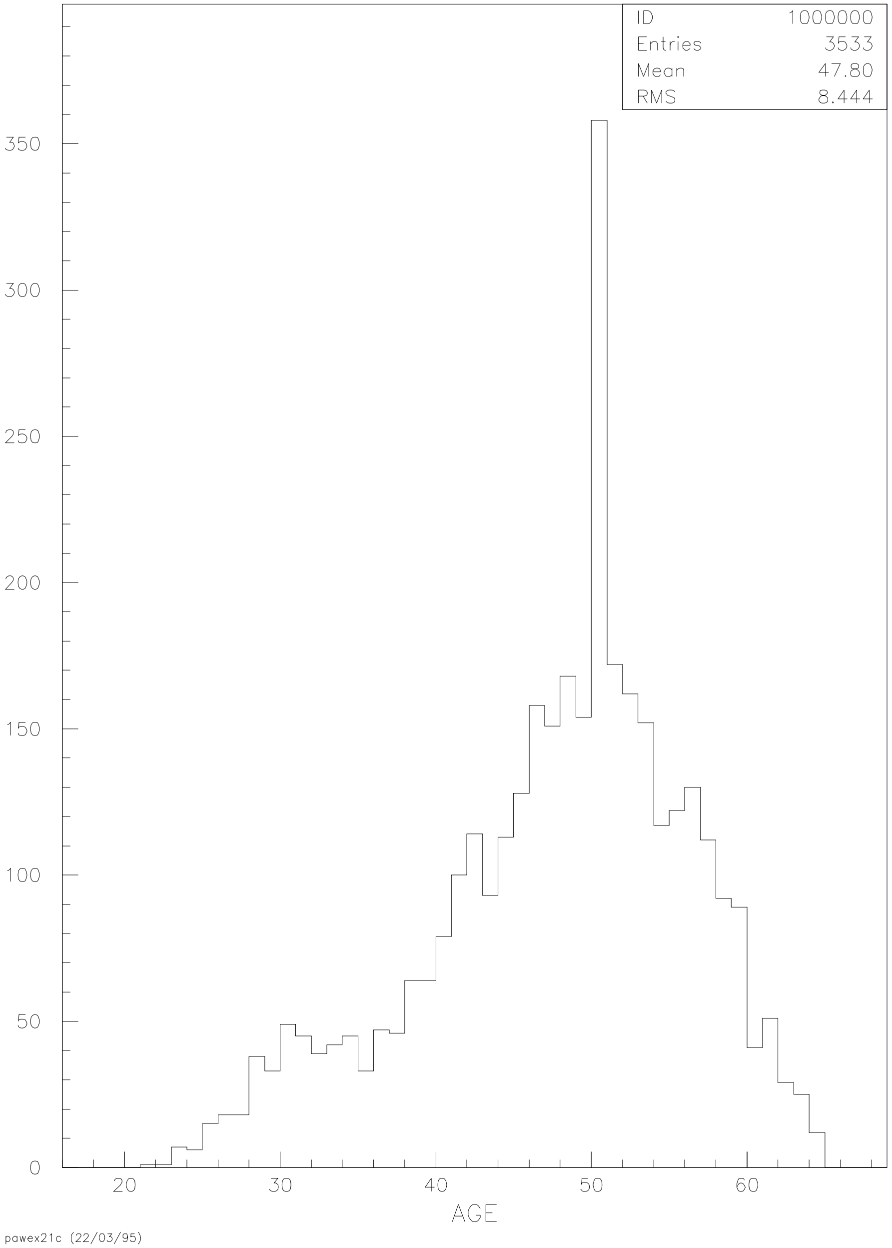


Figure 42: Exec pawex21c.kumac

Usage of NTUPLE/HMERGE

```
     opt stat
❶    HMERGE CWN_APTUPLE_MERGE.HBOOK CWN_APTUPLE.HBOOK CWN_APTUPLE.HBOOK
     HI/FILE 1 CWN_APTUPLE_MERGE.HBOOK
❷    NT/PLOT 11.AGE
```

❶ The command HMERGE (p 82) merge HBOOK files containing histograms and/or ntuples. Ntuples are merged and histograms with the same identifier are added.

❷ As we can see in the statistics box, the new Ntuple has now the double of entries.

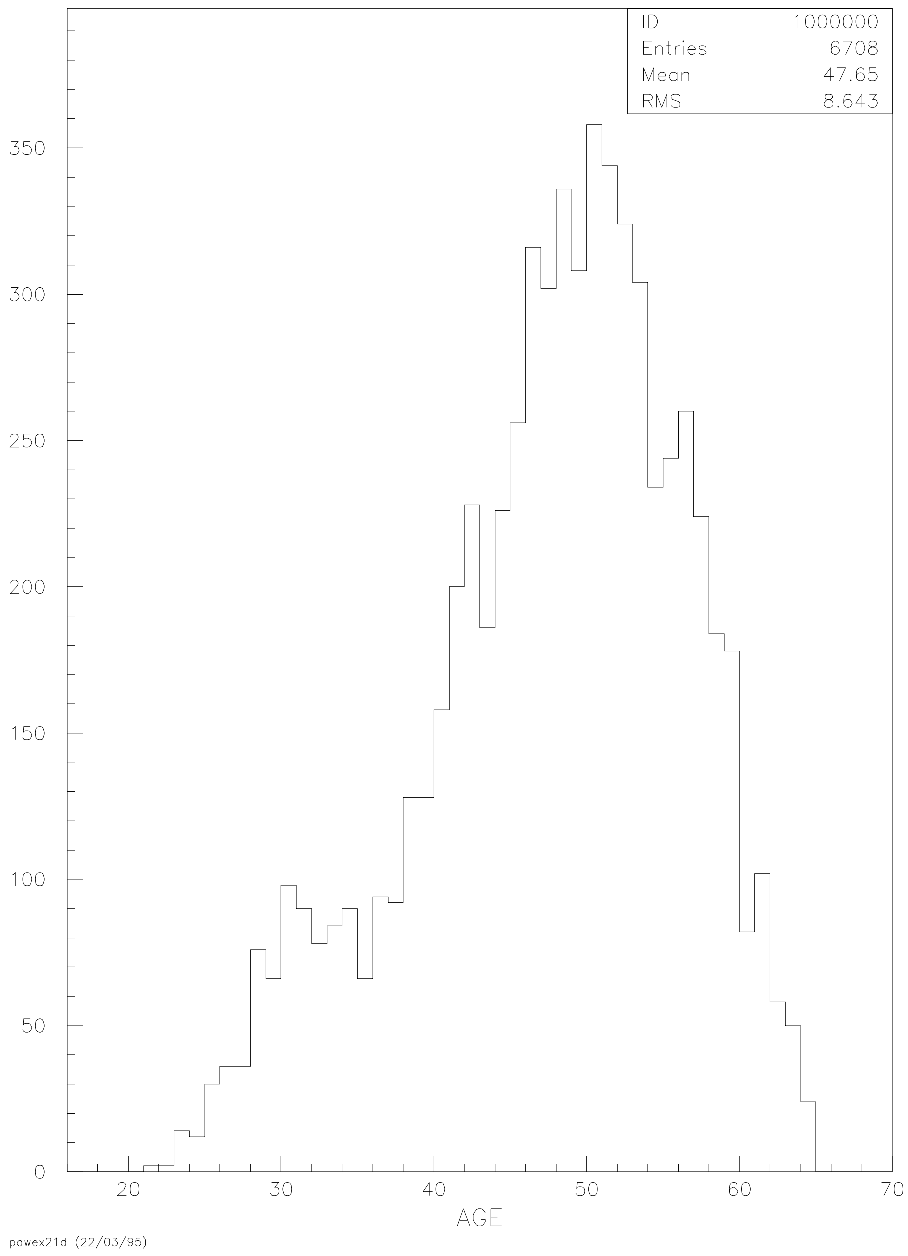


Figure 43: Exec pawex21d.kumac

This example simulate a CERN population of 335400 people.

A 10MB ntuple chain

```
   opt stat
❻  SET SMGU 0.2 ; SET SMGR 0.5 ; SET CSIZ .35
❶  CHAIN MB05   cwn_aptuple.hbook cwn_aptuple.hbook cwn_aptuple.hbook  _
                cwn_aptuple.hbook cwn_aptuple.hbook
❶  CHAIN MB1    MB05 MB05
❶  CHAIN MB10   MB1 MB1 MB1 MB1 MB1 MB1 MB1 MB1 MB1 MB1
❷  CHAIN
❸  CHAIN MB1>
❹  CD //MB10
   Nt/plot 11.age
❺  CHAIN -MB10 ; CHAIN -MB1 ; CHAIN -MB05
```

❶ Create the chain.

❷ List all the chains.

❸ Give the tree of the chain MB1.

❹ Set the current chain (MB10).

❺ Delete the chains MB10, MB05 and MB1.

❻ Changes the statistics size and position.

List of the chains and tree of MB1.

```
MB05      MB1       MB10

MB1
   MB05
      cwn_aptuple.hbook   cwn_aptuple.hbook   cwn_aptuple.hbook
      cwn_aptuple.hbook   cwn_aptuple.hbook
   MB05
      cwn_aptuple.hbook   cwn_aptuple.hbook   cwn_aptuple.hbook
      cwn_aptuple.hbook   cwn_aptuple.hbook
```

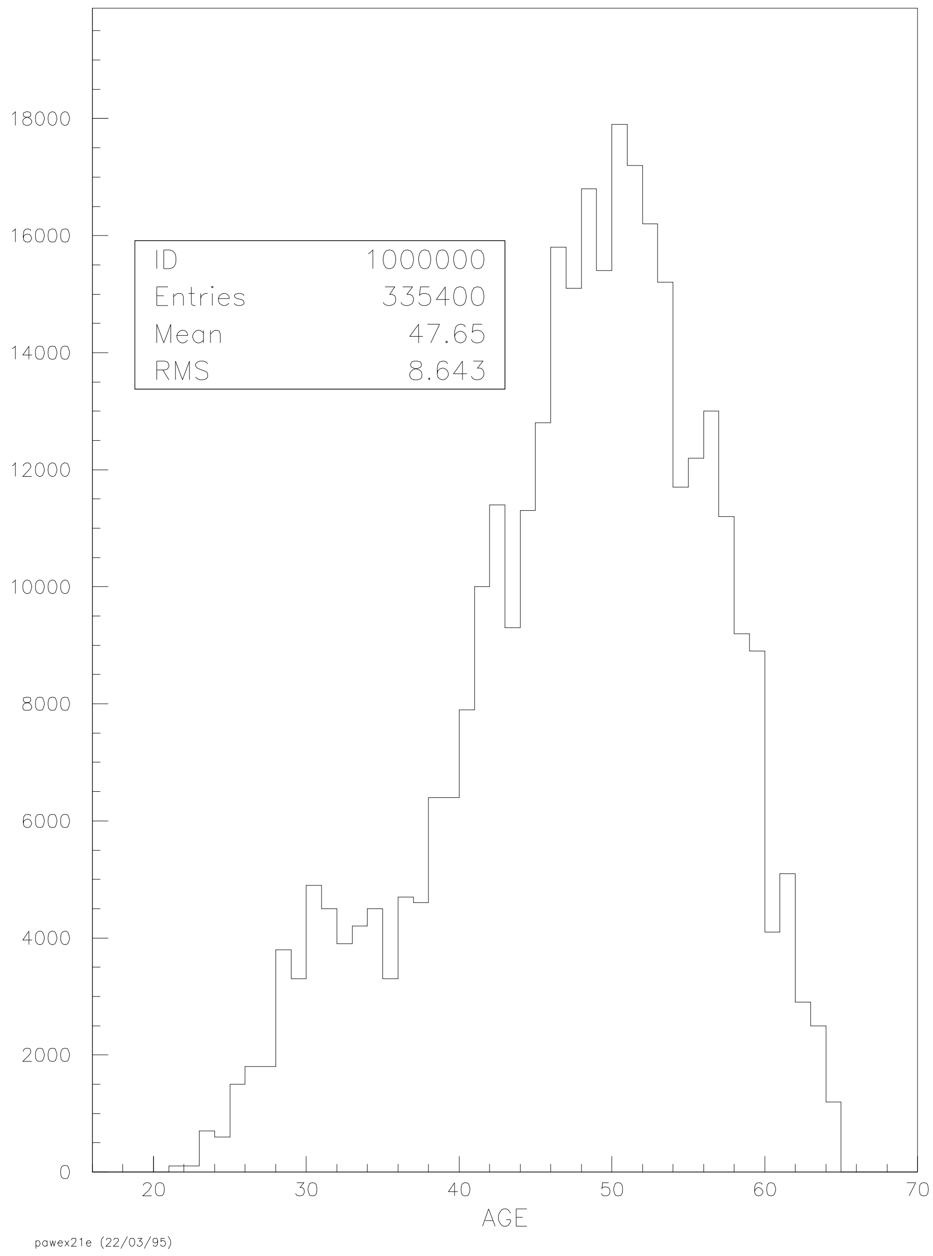



Figure 44: Exec pawex21e.kumac

In this example, a RWN ntuple is duplicated and filled with a subset of the original one. The subset is: “the people with at least 3 childrens”.

A sub-Ntuple with children ¿= 3

```
  opt stat
  hi/file 1 rwn_aptuple.hbook
❶ HI/FILE 2 RWN_APTUPLE_DUP.HBOOK ! N
❷ NT/DUPL //LUN1/10 100
  *
  application comis quit
        real function dup(dum)
❹       INCLUDE ?
❺       IF (CHILDREN.GE.3) CALL HFN(100,CATEGORY)
        dup=1.
        end
  Quit
  *
❸ NT/LOOP //LUN1/10 dup
  hrout 100
  nt/plot //lun1/10.children
  set hcol 1105
  nt/plot //lun2/100.children option=s
```

❶ Create a new **HBOOK** file to store the “subset-Ntuple”

❷ Duplicate the structure of the original Ntuple. The subset will be stored in this duplicated ntuple.

❸ Loop over the original Ntuple with the function “dup”.

❹ The special **COMIS** statement avoid to do `UWFUNC` (p 91) .

❺ When an event is selected, `HFN` is called to fill the “subset-Ntuple”

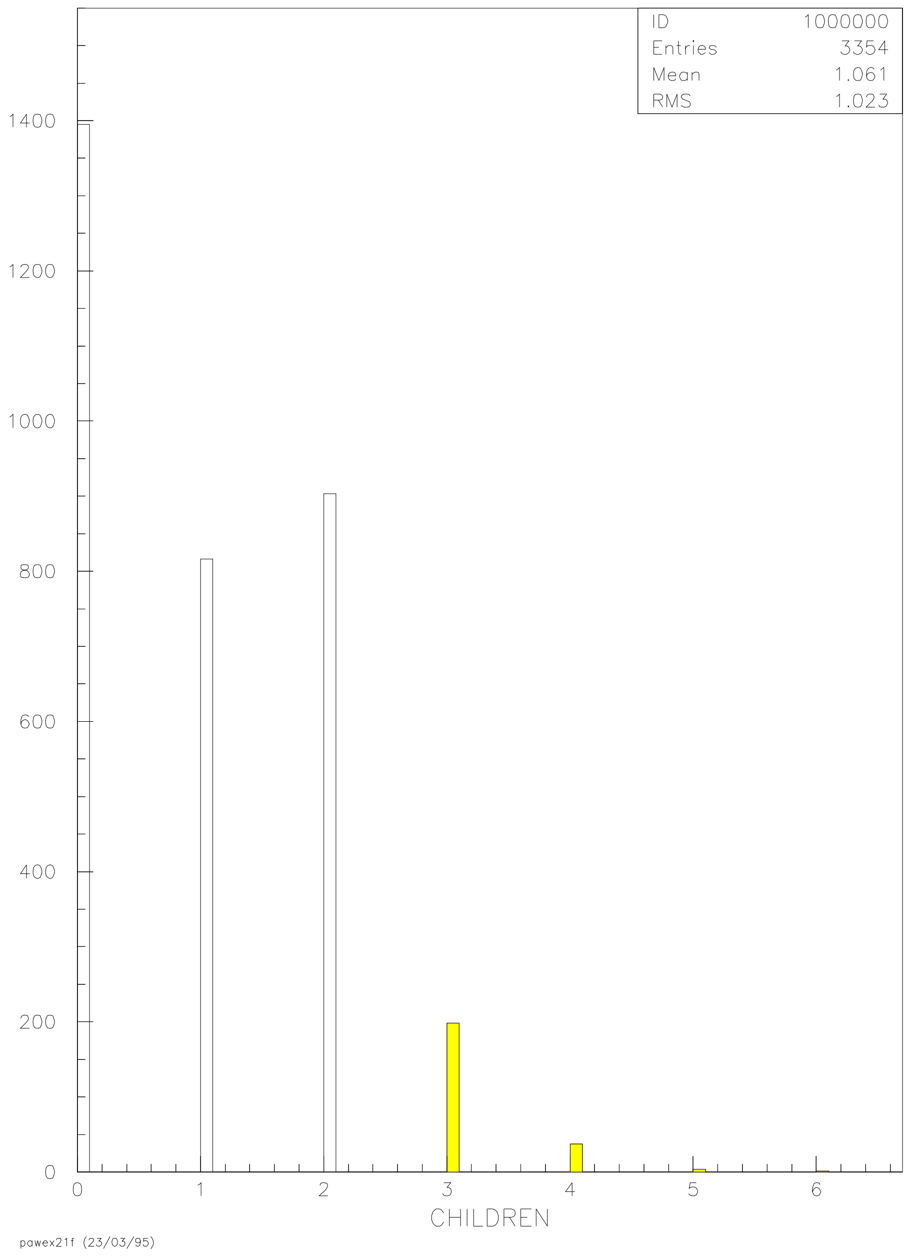


Figure 45: Exec pawex21f.kumac

CW-Ntuple duplicated with selection (1)

In this example the “subset-Ntuple” is filled with a direct call to a **COMIS** function, without the command `NT/LOOP` (p 84) .

A sub-Ntuple with nation = ’FR’

```
     opt stat
     hi/file 1 cwn_aptuple.hbook
❶    UWFUNC //LUN1/11 CWN.INC
     hi/file 2 cwn_aptuple_dup.hbook ! N
     nt/dupl //lun1/11 110
     *
     application comis quit
           subroutine ntdup(id1,id2)
           include 'cwn.inc'
❸          CALL HNOENT(ID1,NOENT)
           do ievent=1,noent
❹             CALL HGNT(ID1,IEVENT,IERR)
              if (ierr.ne.0) goto 20
              if (nation.eq.'FR') then
❺                CALL HFNT(ID2)
              endif
           enddo
       20  continue
     *
           end
     quit
     *
❷     CALL NTDUP(11,110)
❷     HROUT 110
     *
     nt/plot //lun1/11.children
     set hcol 1105
     nt/plot //lun2/110.children option=s
     set hwid 8
     nt/plot //lun1/11.children nation='FR' option=s
```

❶ Generate the include file coresponding to the original ntuple. Note that `UWFUNC` (p 91) generate an include file if the extension of the file name is `.inc`.

❷ Call the routine which duplicate the Ntuple, and save the “subset-Ntuple” on disk.

❸ Get the number of entries in the original ntuple in order to loop over all the events.

❹ Get the event number IEVENT.

❺ Fill the “subset-Ntuple” with the selected events.

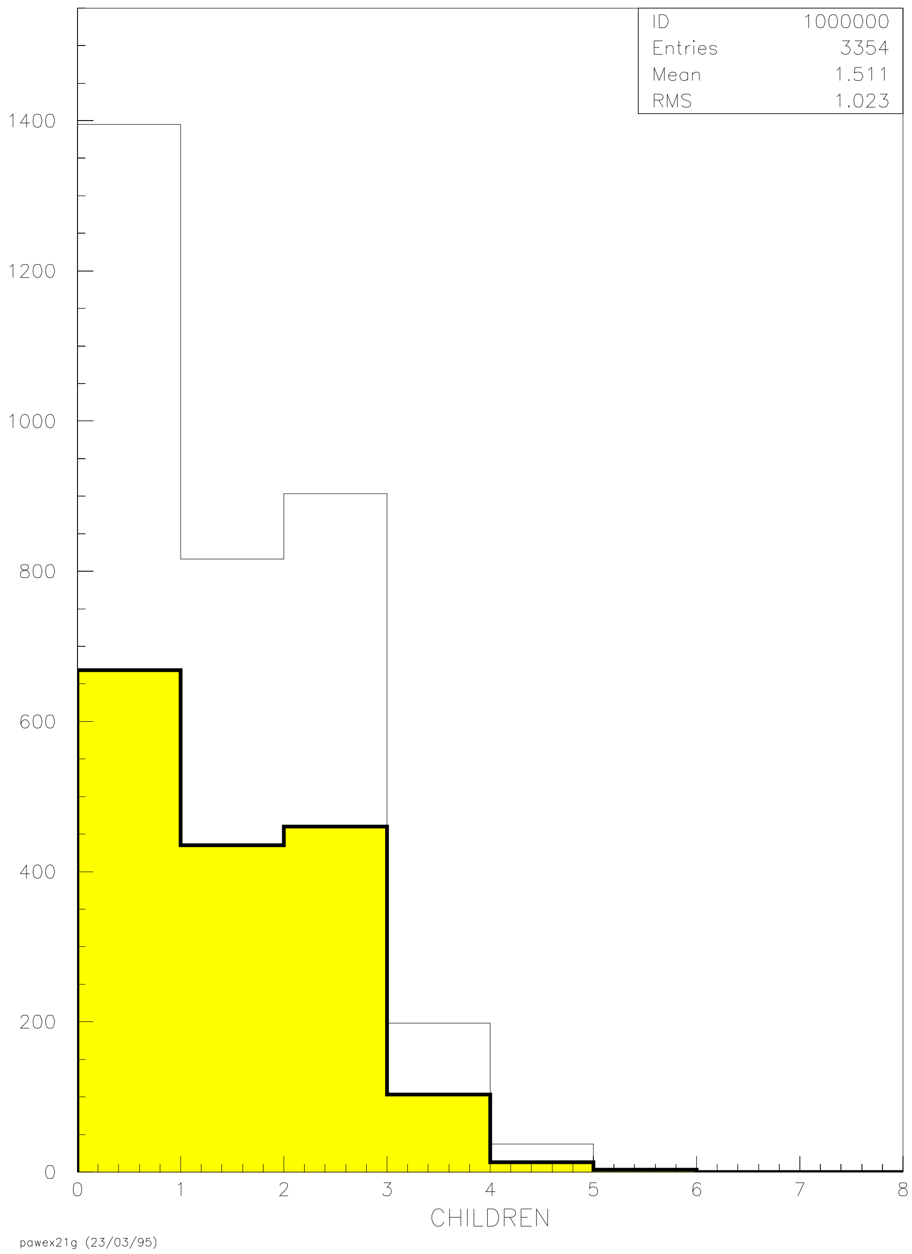


Figure 46: Exec pawex21g.kumac

CW-Ntuple duplicated with selection (2)

This example is similar to the previous one, except that more operations are done is the **COMIS** routine.

A sub-Ntuple with nation = 'IT'

```
     opt stat
     hi/file 1 cwn_aptuple.hbook
     uwfunc //lun1/11 cwn.inc
     hi/file 2 cwn_aptuple_dup.hbook ! N
     *
     application comis quit
           subroutine ntdup(Id1,Id2)
           include 'cwn.inc'
❶          CALL HNTDUP(ID1,ID2,-1,' ','A')
           call hnoent(Id1,Noent)
           do ievent=1,noent
              call hgnt(id1,ievent,ierr)
              if (ierr.ne.0) goto 20
              if (Nation.eq.'IT') then
                 call hfnt(id2)
              endif
           enddo
❶       20 CALL HROUT(ID2,ICYCLE,' ')
     *
           end
     quit
     *
     call ntdup(11,110)
     *
     nt/plot //lun1/11.children
     set hcol 1105
     nt/plot //lun2/110.children option=s
     set hwid 8
     nt/plot //lun1/11.children nation='IT' option=s
❷    CLOSE 0
```

❶ The ntuple duplication and its ouput on disk can be done inside the **COMIS** routine itself.

❷ Note that if 0 is given the command `CLOSE` (p 143) all the opened files are closed.

CW-Ntuple duplicated with selection (2)

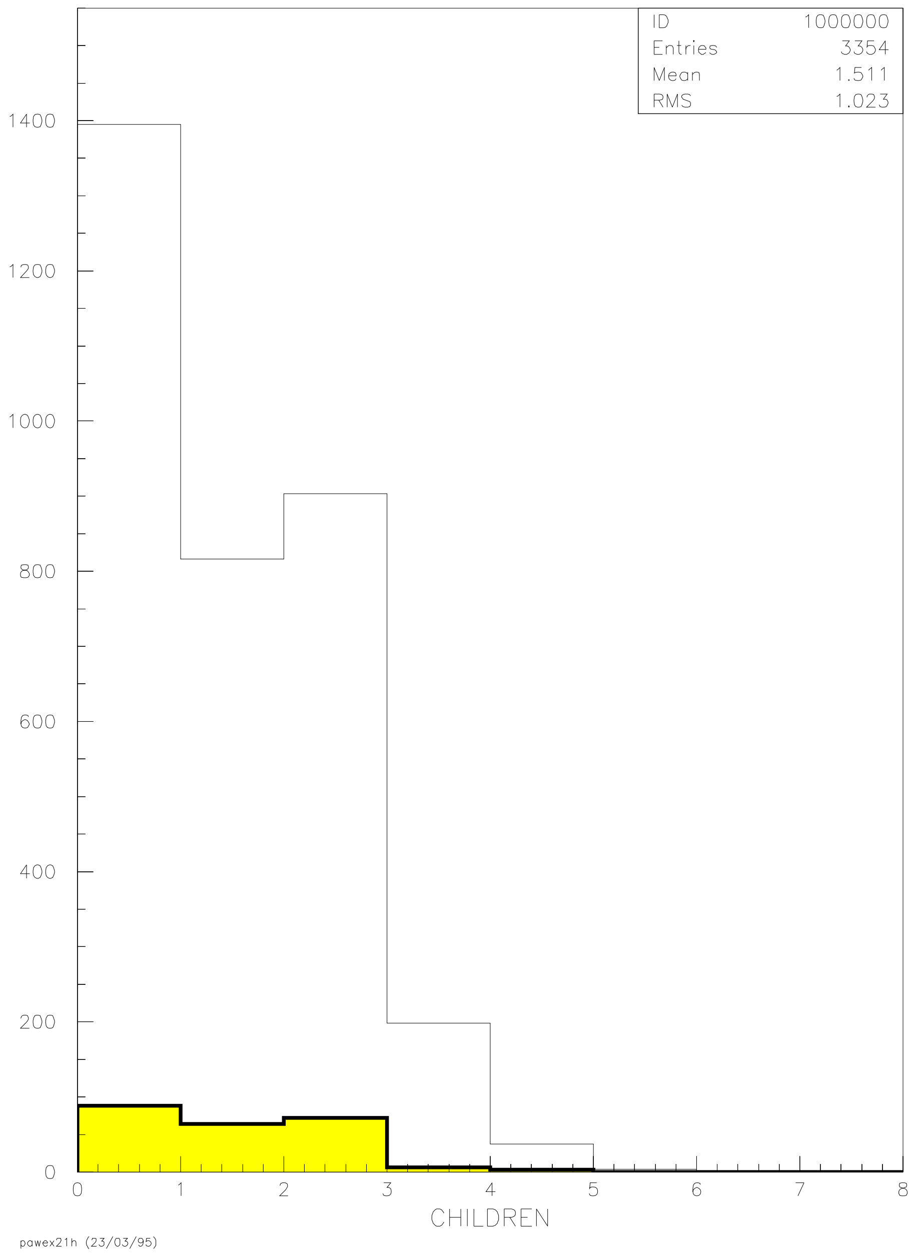


Figure 47: Exec pawex21h.kumac

Examples of the SIGMA processor (1)

```
      zone 2 2
❷     APPLICATION SIGMA
         X=ARRAY(200,0#2*PI)
         sinus=sin(x)
         sinx=sin(x)/x
❷     EXIT
      gra 200 x sinus
      set dmod 2
      gra 200 x sinx l
      set dmod 0
❶     SIGMA x=array(300,0#8)
      sigma g=cosh(x)+sin(1/(.1+x*x))
      gra 300 x g
      sigma x=array(300,0#3)
❸     GRAPH 300 x $SIGMA(cosh(x)+sin(1/(.1+X*X)))
      sigma x=array(300,0#1)
❹     GRAPH 300 x $RSIGMA(cosh(x)+sin(1/(.1+X*X)))
```

This example (and the next one) shows how to use the array manipulation package **SIGMA**. There are four ways to give directives to **SIGMA**. The complete information on **SIGMA** is given in the chapter *SIGMA* of the **PAW** manual.

❶ Precede the statement by the prefix `SIGMA`.

❷ 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**.

❸ 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).

❹ The **PAW** system function `$RSIGMA`. This function has be to used in **COMIS** calls expecting a `REAL` argument, e.g.

```
CALL file.f($RSIGMA(sqrt(x(1)))
```

Otherwise the value may be passed as an `INTEGER` if the **SIGMA** result turns out to be a whole number.

Note also:

The system function `$FORMAT(number,format)` to format a number according to a Fortran-like `FORMAT` string, e.g. `$FORMAT([x],F9.3)`. Supports F,E,G,I, and Z (hexadecimal). The complete list of the system functions available is given on next page.

Examples of the SIGMA processor (1)

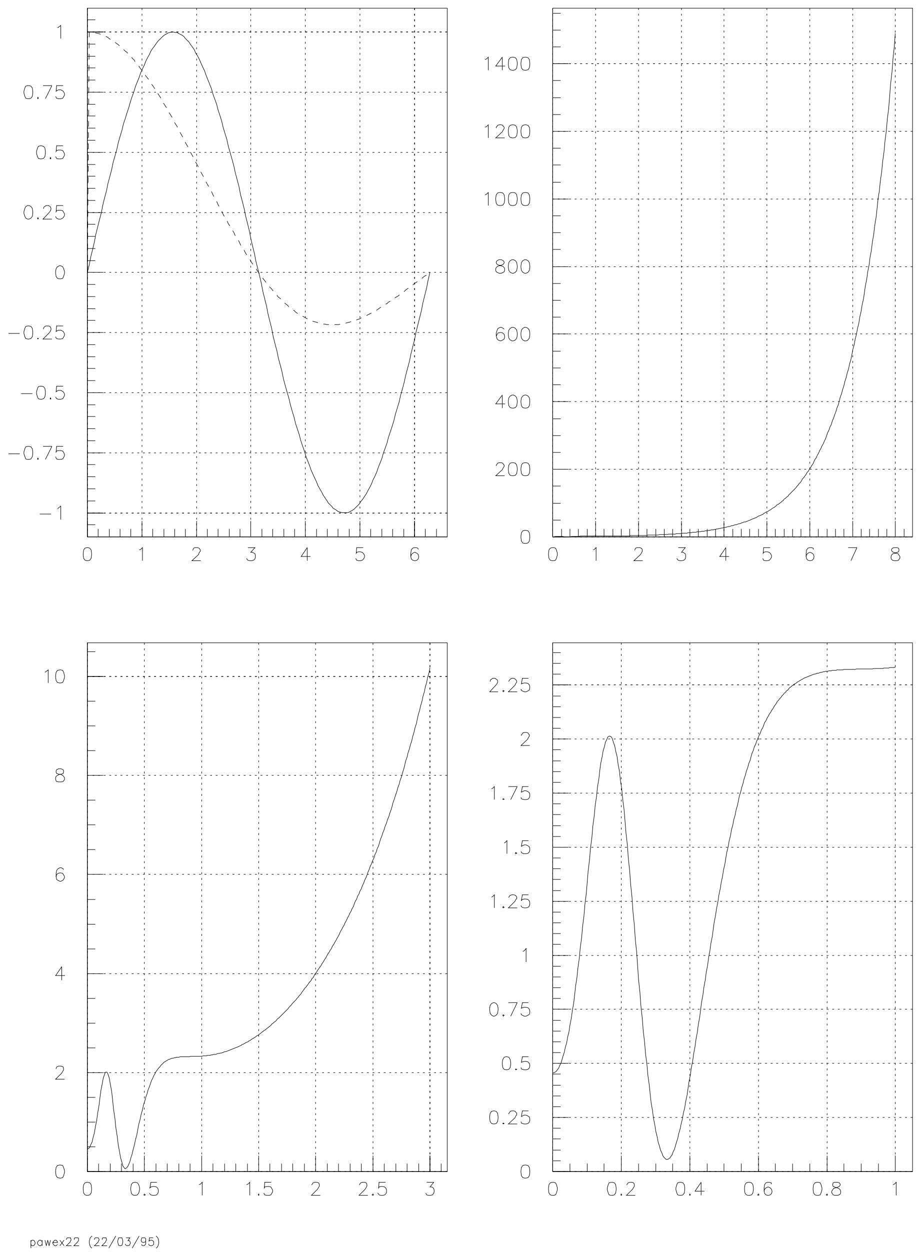


Figure 48: Exec pawex22.kumac

Examples of the SIGMA processor (1)

The function is literally replaced, at run-time, by its current value. The following functions are available:

$DATE	Current date in format DD/MM/YY
$TIME	Current time in format HH.MM.SS
$CPTIME	CP time elapsed since last call (in sec)
$RTIME	Real time elapsed since last call (in sec)
$VDIM(VNAME,IDIM)	Physical length of vector VNAME on dimension IDIM (1..3)
$VLEN(VNAME,IDIM)	As above, but for the logical length (i.e. stripping trailing zeroes)
$NUMVEC	Current number of vectors
$VEXIST(VNAME)	Index of vector VNAME (1..$NUMVEC or 0 if VNAME does not exist)
$SUBSTRING(STRING,IX,NCH)	STRING(IX:IX+NCH-1)
$UPPER(STRING)	STRING changed to upper case
$LOWER(STRING)	STRING changed to lower case
$LEN(STRING)	Length of STRING
$INDEX(STR1,STR2)	Position of first occurence of STR2 in STR1
$WORDS(STRING,SEP)	Number of words separated by SEP
$WORD(STRING,K,N,SEP)	Extract N words starting at word K
$QUOTE(STRING)	Add quotes around STRING
$UNQUOTE(STRING)	Remove quotes around STRING
$EVAL(Expression)	Result of the Expression computed by KUIP
$SIGMA(Expression)	Result of the Expression computed by SIGMA
$RSIGMA(Expression)	As above but a decimal point is added to integer results
$FORMAT(number,format)	Format a number according to Fortran. e.g. $FORMAT(1.5,F5.2) ==> ' 1.50' $FORMAT(123,I5.5) ==> '00123'
$ARGS	Command line at program invocation
$KEYNUM	Address of latest clicked key in style GP
$KEYVAL	Value of latest clicked key in style GP
$LAST	Latest command line executed
$ANUM	Number of aliases
$ANAM(I)	Name of I-th alias
$AVAL(I)	Value of I-th alias
$STYLE	Current style as defined by SET/STYLE
$OS	Operating system name, e.g. UNIX or VMS
$MACHINE	Hardware or Unix brand, e.g. VAX or HPUX
$PID	Process ID
$IQUEST(I)	Value of IQUEST(I) status vector
$ENV(var)	Value of environment variable
$FEXIST(file)	1 if file exists or 0 otherwise
$SHELL(cmd,N)	N'th line of shell command output (Unix only)
$SHELL(cmd,sep)	Shell output with newlines replaced by sep
$SHELL(cmd)	Same as $SHELL(cmd,' ')
$CALL('fun(args)')	Call a Fortran REAL FUNCTION

Examples of the SIGMA processor (1)

$ICALL('ifun(args)')	Call an INTEGER FUNCTION
$LCALL('lfun(args)')	Call a LOGICAL FUNCTION and return 0 or 1
$DCALL('dfun(args)')	Call a DOUBLE PRECISION FUNCTION
$HCDIR()	Current Hbook working directory
$HEXIST(id)	1 if histogram ID exists or 0 otherwise
$HINFO(id,'ENTRIES')	Number of entries
$HINFO(id,'MEAN')	Mean value
$HINFO(id,'RMS')	Standard deviation
$HINFO(id,'EVENTS')	Number of equivalent events
$HINFO(id,'OVERFLOW')	Content of overflow channel
$HINFO(id,'UNDERFLOW')	Content of underflow channel
$HINFO(id,'MIN')	Minimum bin content
$HINFO(id,'MAX')	Maximum bin content
$HINFO(id,'SUM')	Total histogram content
$HINFO(id,'NSLIX')	Number of X slices
$HINFO(id,'NSLIY')	Number of Y slices
$HINFO(id,'NBANX')	Number of X bandes
$HINFO(id,'NBANY')	Number of Y bandes
$HINFO(id,'NPROX')	Projection X (0 or 1)
$HINFO(id,'NPROY')	Projection Y (0 or 1)
$HINFO(id,'XBINS')	Number of bins in X direction
$HINFO(id,'XMIN')	Lower histogram limit in X direction
$HINFO(id,'XMAX')	Upper histogram limit in X direction
$HINFO(id,'YBINS')	Number of bins in Y direction
$HINFO(id,'YMIN')	Lower histogram limit in Y direction
$HINFO(id,'YMAX')	Upper histogram limit in Y direction
$HTITLE(id)	Histogram title
$GRAFINFO('XZONES')	Number of zones in X direction
$GRAFINFO('YZONES')	Number of zones in Y direction
$GRAFINFO('NT')	Current Normalization Transformation number
$GRAFINFO('WNXMIN')	Lower X limit of window in current NT
$GRAFINFO('WNXMAX')	Upper X limit of window in current NT
$GRAFINFO('WNYMIN')	Lower Y limit of window in current NT
$GRAFINFO('WNYMAX')	Upper Y limit of window in current NT
$GRAFINFO('VPXMIN')	Lower X limit of viewport in current NT
$GRAFINFO('VPXMAX')	Upper X limit of viewport in current NT
$GRAFINFO('VPYMIN')	Lower Y limit of viewport in current NT
$GRAFINFO('VPYMAX')	Upper Y limit of viewport in current NT
$GRAFINFO('TXALIH')	Horizontal text alignment
$GRAFINFO('TXALIV')	Vertical text alignment
$GRAFINFO('TXFONT')	Text font
$GRAFINFO('TXPREC')	Text precision
$GRAFINFO('?attr')	HPLOT/HIGZ attribute (HELP SET for valid names)
$RGBINFO(icol,'R')	Weight of Red in color table
$RGBINFO(icol,'G')	Weight of Green in color table
$RGBINFO(icol,'B')	Weight of Blue in color table
$CUT(n)	Cut expression $n
$CUTEXPAND(string)	Replace $n in the (quoted) string by $CUT(n)

Examples of the SIGMA processor (2)

```
      zone 2 2
❶❸    SIGMA X=ARRAY(200,0#5)
❷     SIGMA A=8
      sigma b=.01
❸     SIGMA Y=EXP(-X)*SIN(A*X)+B*X*X
      gra 200 x y
      sigma x=array(200,0#2*pi)
      sigma s=sin(x)
      sigma s2=s/2
      sigma c=cos(x)
      sigma c2=c/2
      sigma s4=s/4
      sigma c4=c/4
      gra 200 s c
      gra 200 s2 c l
      gra 200 s4 c l
      gra 200 s c2 l
      gra 200 s2 c2 l
      gra 200 s4 c2 l
      gra 200 s c4 l
      gra 200 s2 c4 l
      gra 200 s4 c4 l
      sigma a=array(100,0#59.77)
      sigma nc=nco(a)
      sigma y=cos(a)*a
      sigma x=sin(a)*a
      gra nc x y
      sigma a=a*2.55555
      sigma y=cos(a)*a
      sigma x=sin(a)*a
      gra nc x y
```

❶ The command V=ARRAY(L,x1#x2) allows to create a vector V with the length L and initialize it in the range x1,x2.

❷ All the objects managed by **SIGMA** are vectors . In this example A is vector of length 1.

❸ The resulting vectors (if they don't exist) are created automatically by **SIGMA** (here Y).

Examples of the SIGMA processor (2)

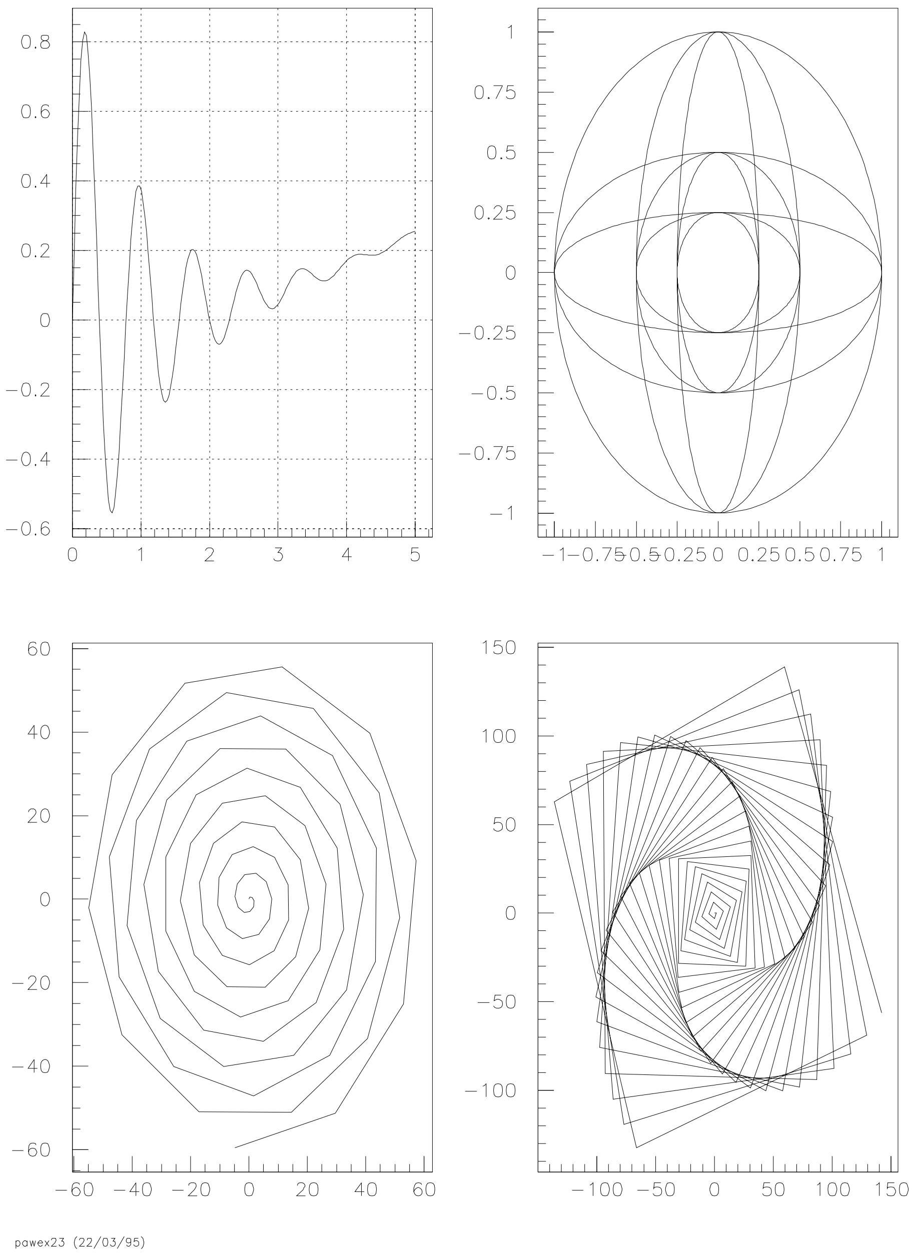


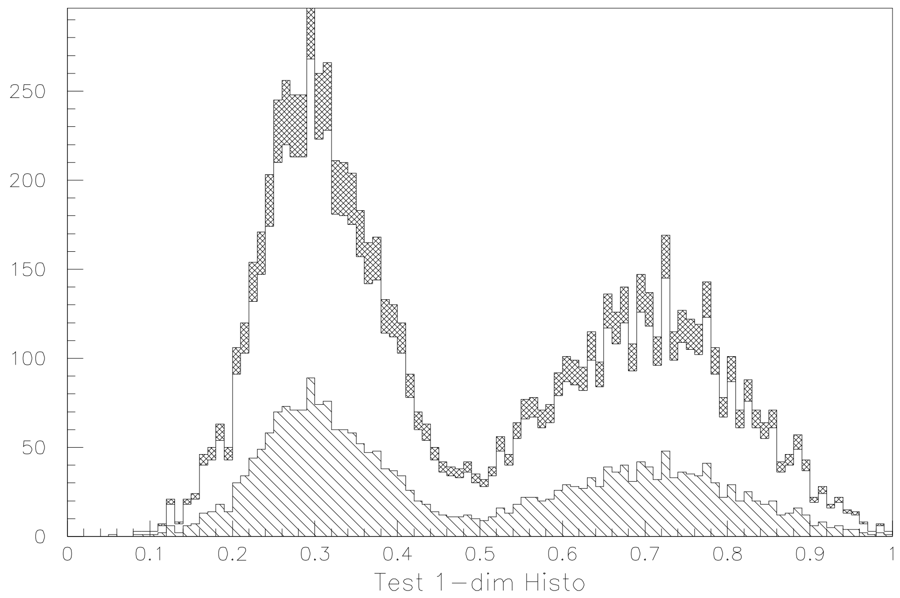
Figure 49: Exec pawex23.kumac

Graphical operations on histograms (Keep and Add)

```
      histogram/file 45 pawhists.hbook
      zone 1 2
      set htyp 245
❶     H/PL 120 K
      set htyp 254
❶     H/PL 110
      set htyp
❷     H/PL 110 +
      set htyp 144
      hi/pl 130 +
```

❶ The option "K" in the command HIST/PLOT keep the histogram in memory at the graphics level to allows updating. If no zone is defined, the option "K" is not necessary.

❷ If an histogram is kept in memory (automatically or via option "K") it is possible to add the content of an other histogram with option "+".



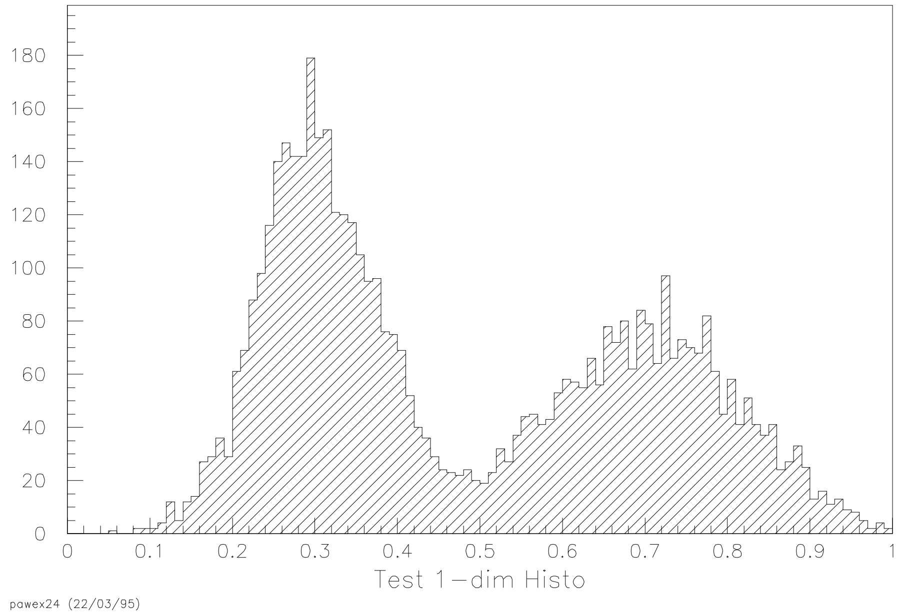


Figure 50: Exec pawex24.kumac

Keep and Update

```
        MACRO DICE 1=50
        set hcol 1001
        set ndvx -11.05
❻       OPT STAT
❶       CALL DICE.F([1])
        hi/fit 3 g
```

FORTRAN routine dice

```
        subroutine dice(n)
        ifirst=1
❷         CALL HBOOK1(3,'Playing with two dice',11,2.,13.,0.)
          do 3 j=1,n
            ix1=6.*rndm(.01234)+1
            ix2=6.*rndm(.56789)+1
❸           CALL HFILL(3,FLOAT(IX1+IX2),0.,1.)
            if (ifirst.eq.1) then
❹              CALL HPLOT(3,'BK',' ',0)
               ifirst=0
            else
❺              CALL HPLOT(3,'BU',' ',0)
            endif
          enddo
        end
```

❶ This macro call a **COMIS** routine only to be faster. The **COMIS** routine can be replaced by a macro, in particular the options "K" and "U" are also available in command HIST/PLOT (try HELP H/PL).

❷ The histogram is also booked in the FORTRAN program. The corresponding **PAW** command is 1DHISTO.

❸ Two random numbers between 1 and 6 are generated and the histogram is filled with the sum of this numbers to simulate dice playing.

❹ The first time the histogram is plotted the option "K" is used to keep in memory a copy of the histogram in order to update it later.

❺ With the "U" option, **PAW** looks at the current kept histogram contents and update the plot with the new contribution without redrawing everything. This mechanism is used in data acquisition.

❻ The statistics are also updated.

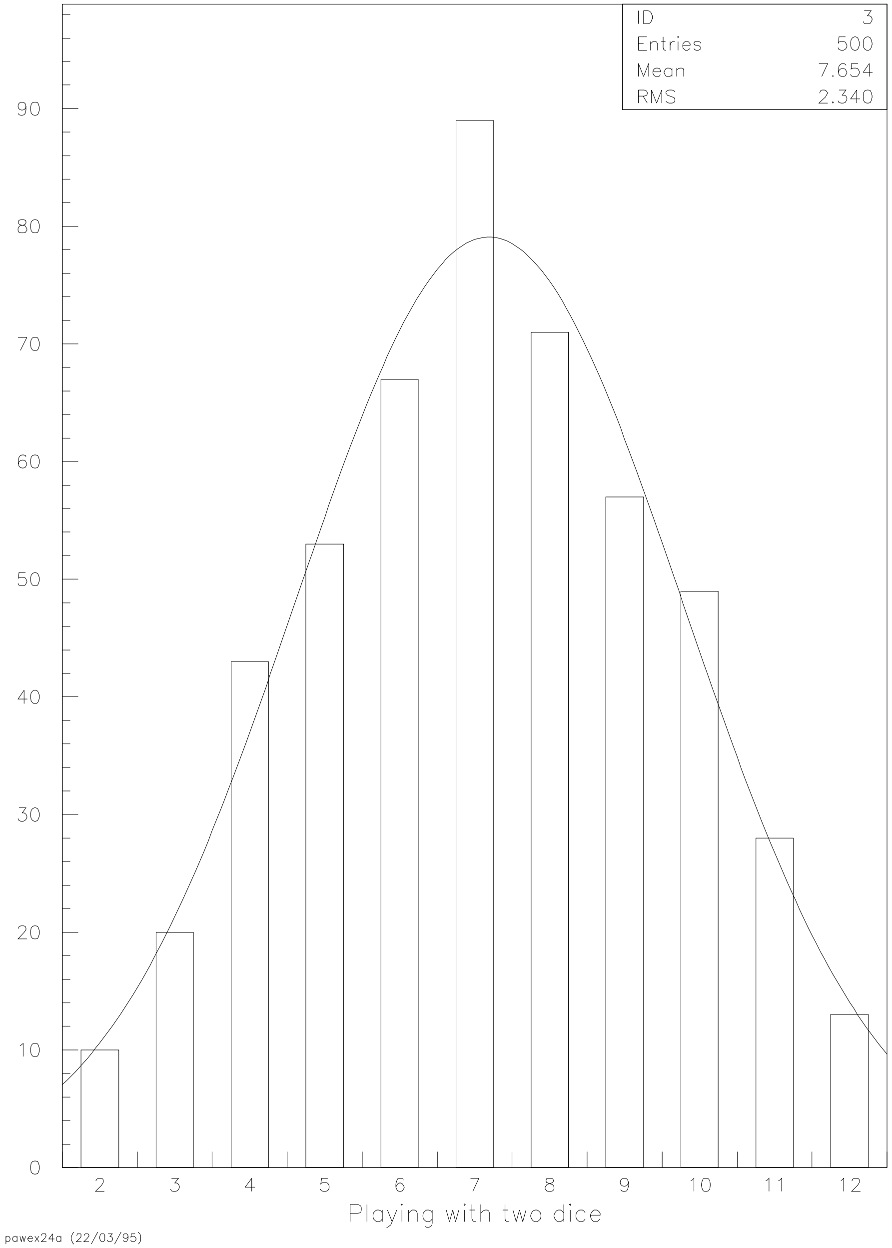


Figure 51: Exec pawex24a.kumac

Updating plots in real time (2)

Keep and Update

```
    MACRO UPDATE 1=50
❶   APPLICATION COMIS QUIT
       subroutine fill
       do 10 i=1,1000
          call rannor(x,y)
          call hf1(1,x,1.)
          call hf1(2,y,1.)
          call hf1(3,(x-y)/(x+y),1.)
 10    continue
       end
❶   QUIT
    cd //pawc
    h/del 0
    1dhisto 1 x 100 -4 4
    1dhisto 2 y 100 -4 4
    1dhisto 3 (x-y)/(x+y) 100 -6 6
    zone 2 2
❶   CALL FILL
    Set 2buf 1
    h/pl 1 k
    h/pl 2 k
    zon 1 2 2 s
    h/pl 3 k
❷    do i=1,[1]
       opt NSTA
❶      CALL FILL
       h/pl 1 u
       h/pl 2 u
       opt STAT
       h/pl 3 u
       call igterm
    enddo
```

❶ The COMIS routines can be declare via the an `APPLICATION`. When a such routine is called, the extension `.f` should not be specified.

❷ On Unix systems, you can activate the command below instead of the loop

```
IDLE 1 'opt nsta;call fill; h/pl 1 u ; h/pl 2 u ;op stat; h/pl 3 u ;call igterm'
```

The command `IDLE`, execute a command if program is idle. The command string is executed if there was no keyboard activity during `SEC` seconds.

```
 * KUIP/IDLE SEC [ STRING ]
```


Updating plots in real time (2)

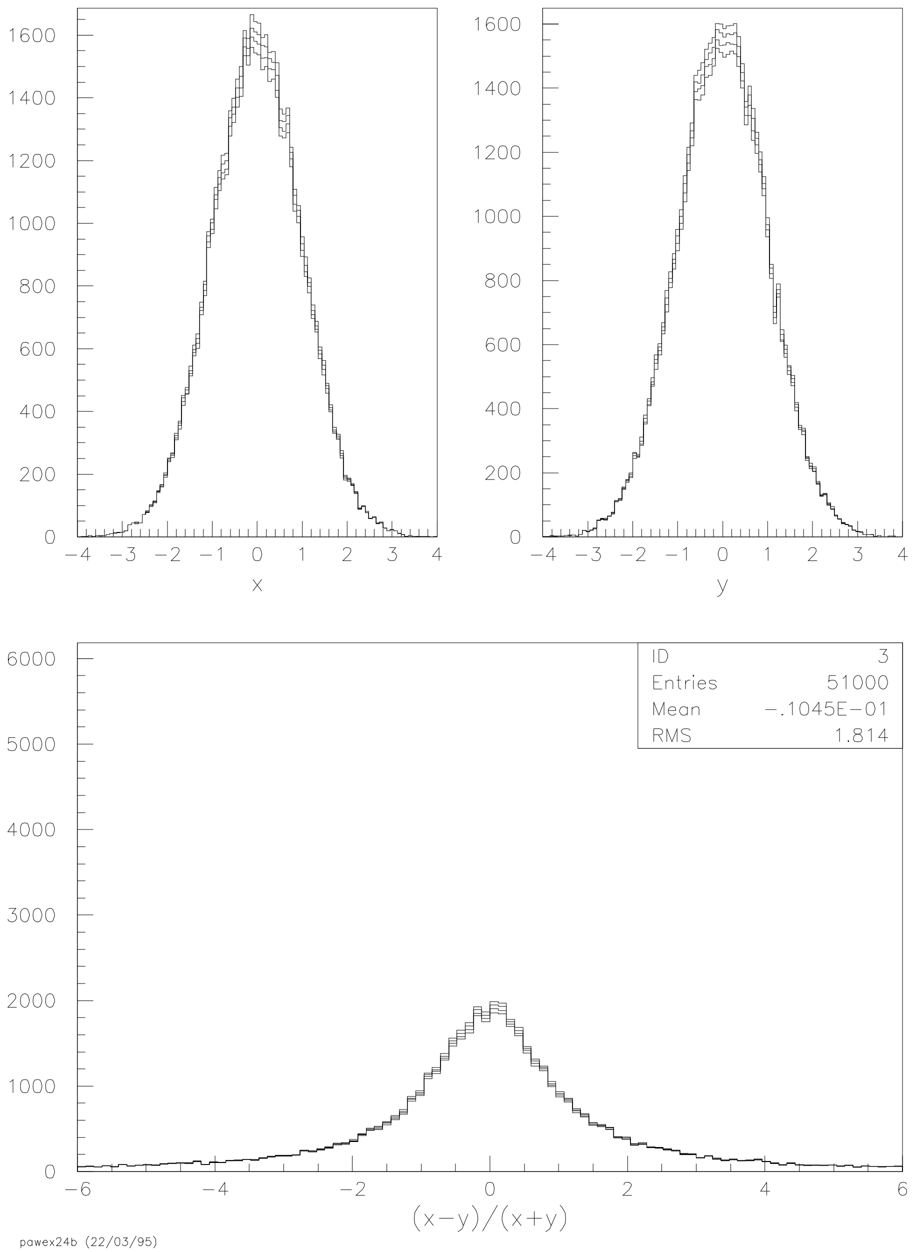


Figure 52: Exec pawex24b.kumac

Merge pictures onto one plot

```
    histogram/file 1 pawhists.hbook
❺   SWITCH Z
❶   PIC/CR MERGE2
    set htyp 354
    hi/pl 110
    set htyp 345
    hi/pl 110(31:40) s
❶   PIC/CR MERGE1
    set htyp 354
    hi/pl 110(31:40)
❷   IZPICT MERGE2 C
    switch g
❸   PI/MERGE MERGE1 .5 .5 .3 D
❹   PI/DEL *
```

This example shows some application of the **HIGZ** pictures.

❶ `PI/CREATE` (p 127) allows to create a new graphic picture in memory. After this call, all the graphic generated

❷ `IZPICT` (p 129) is the generic function to perform all kind of actions on the **HIGZ** pictures. Here the picture `MERGE2` is set as the current picture.

❸ `PI/MERGE` (p 128) allows to merge a picture into the current picture.

❹ `PI/DEL` (p 127) allows to delete a picture from memory. To delete a picture from a file the command `SCRATCH` (p 127) should be used.

❺ The command `SWITCH` set the graphics switch to control plotting output to terminal (G) and/or picture in memory (Z).

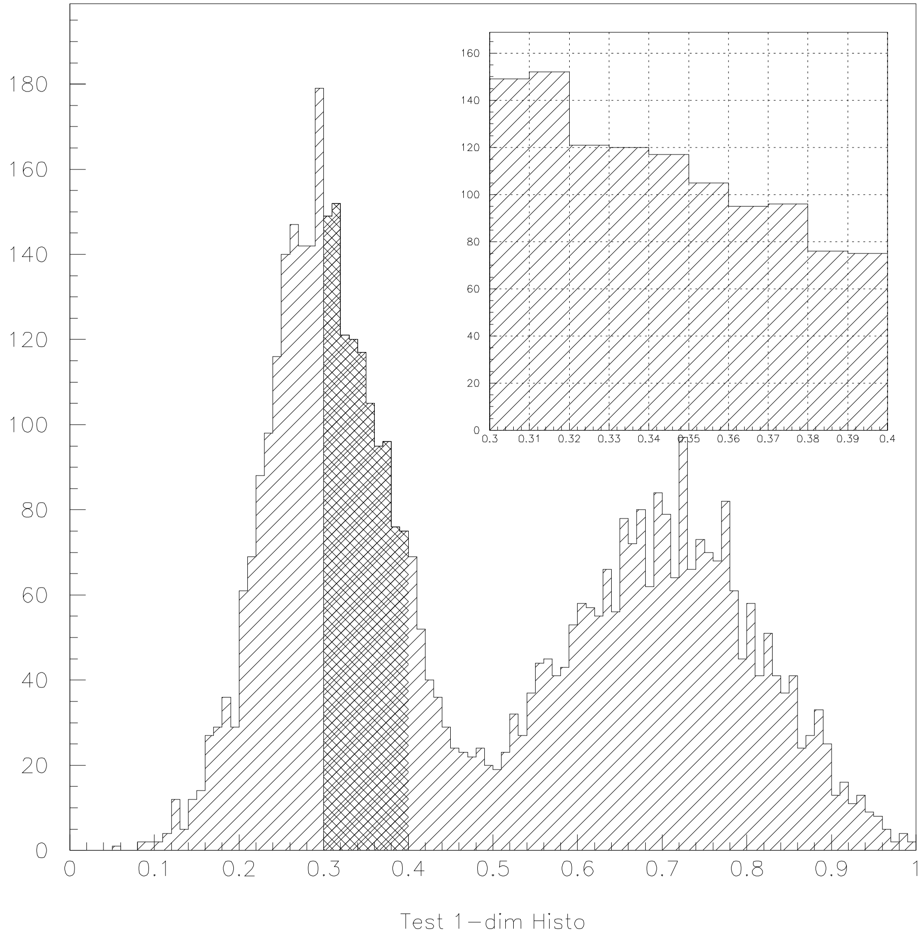


Figure 53: Exec pawex25.kumac

This macro can be used to print the tutorial examples

```
MACRO PRINTEX 1=1
FOR/FILE 44 pawex[1].ps
METAFILE 44 -111
EXEC PAWEX[1]
CLOSE 44
SHELL  local print command pawex[1].ps
```

The PostScript workstation types have the following format:

`-[Format][Nx][Ny][Type]`

Where:

Format is an integer between 0 and 99 which defines the format of the paper. For example if `Format`=3 the paper is in the standard A3 format. `Format`=4 and `Format`=0 are the same and define an A4 page. The A0 format is selected by `Format`=99.

Nx, Ny specify respectively the number of zones on the x and y axis. `Nx` and `Ny` are integers between 1 and 9.

Type can be equal to:

- **1** Portrait mode with a small margin at the bottom of the page.
- **2** Landscape mode with a small margin at the bottom of the page.
- **4** Portrait mode with a large margin at the bottom of the page.
- **5** Landscape mode with a large margin at the bottom of the page. The large margin is useful for some PostScript printers (very often for the colour printers) they need more space to grip the paper for mechanical reasons. Note that some PostScript colour printers can also use the so called "special A4" format permitting the full usage of the A4 area; in this case larger margins are not necessary and `Type`=1 or 2 can be used.
- **3** Encapsulated PostScript. This `Type` permits the generation of files which can be included in other documents, for example in LaTeX files. Note that with this `Type`, `Nx` and `Ny` must always be equal to 1, and `Format` has no meaning. The size of the picture must be specified by the user via the command `SIZE`. Therefore the workstation type for Encapsulated PostScript is -113. For example if the name of an Encapsulated PostScript file is `example.eps`, the inclusion of this file into a LaTeX file will be possible via (in the LaTeX file):

```
\begin{figure}
\epsffile{example.eps}
\caption{Example of Encapsulated PostScript in LaTeX}
\label{EXAMPLE}
\end{figure}
```


How to print all the tutorial examples on one page

```
MACRO PRINTALL
FOR/FILE 44 all.ps
METAFILE 44 -471
DO I=1,26
   EXEC PAWEX[I]
ENDDO
CLOSE 44
SHELL  local print command all.ps
```

Note also:

The command `PICTURE/PRINT` allows to print the current picture in memory onto a PostScript file, and if required send it to the default PostScript printer.

PAW++ panels creation

```
     MACRO Panel
*
❶    ICON debug   debug.px
❶    ICON df      df.px
❶    ICON laser   laser.px
❶    ICON shell   shell.px
❷    PANEL 0 R
❸    PANEL  1.01 'Exec pawex32#debug'     debug
     Panel  2.01 'Shell df'     df
     Panel  3.01 'pi/print'     laser
     Panel  4.01 '-shell'       shell
❹    PANEL 0 D '  ' '100x390+0+0'
     Return
*
     MACRO DEBUG
     panel 0
     panel 1.01 'Trace ON'
     panel 1.02 'Trace OFF'
     panel 0 D '  ' '170x100+112+0'
     Return
```

This example shows how to create and dispaly panels with PAW++.

❶ Define an icon. The icon description is describe in a bitmap file.

❷ Reset the current panel in memory.

❸ Define one item in the current panel.

❹ Display the current panel.

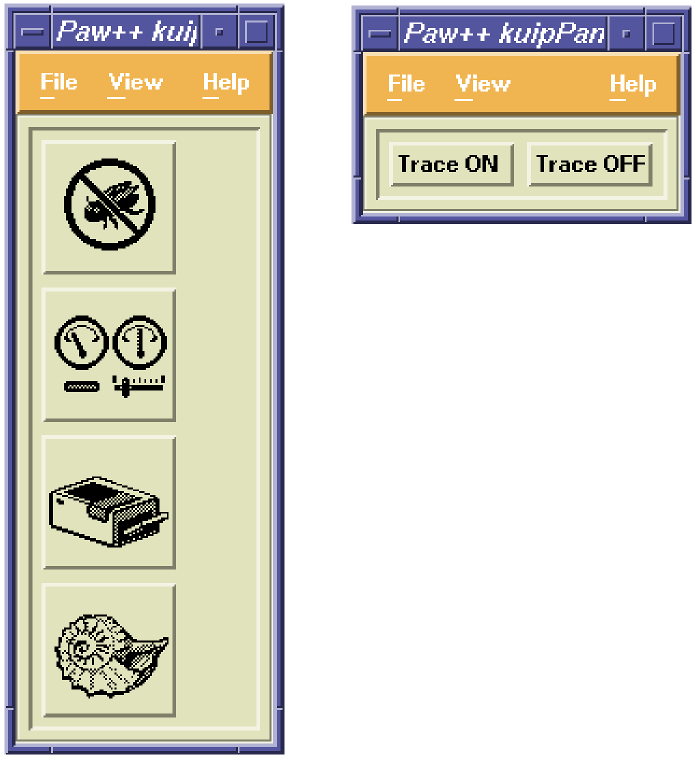


Figure 54: Exec pawex32.kumac