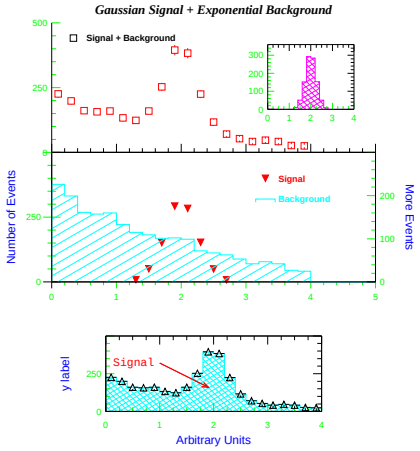
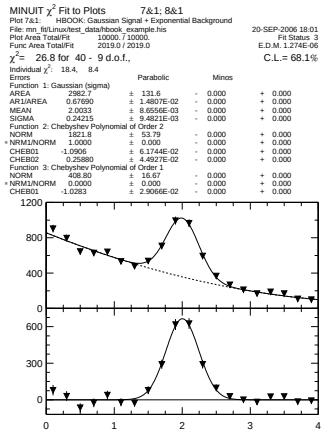




Mn_Fit

A Tutorial

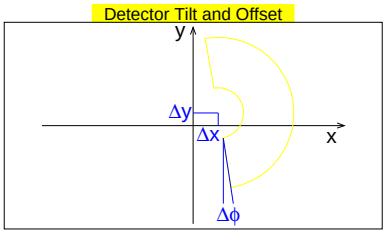
Version 1.0

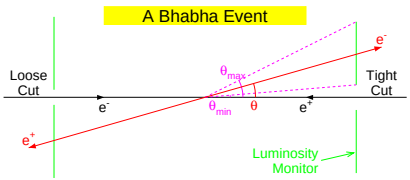
September 20, 2006

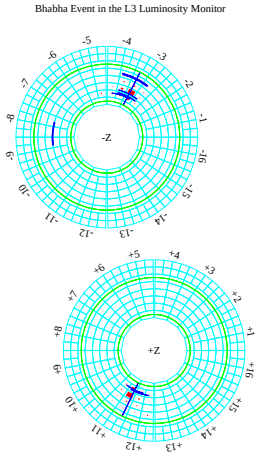
Ian C. Brock

Physikalisches Institut der Universität Bonn
Nußallee 12, D-53115 Bonn

EMail: Ian.Brock@cern.ch or brock@physik.uni-bonn.de
Homepage: http://www-zeus.physik.uni-bonn.de/~brock/mn_fit.html







Contents

List of Examples

Plotting

Quick Fitting

Functions

Accessing Information

Fitting and MINUIT

Manipulating Histograms

Manipulating Ntuples

Sizes

List of Figures

1 Introduction

Mn_Fit is an interactive fitting and plotting package, that uses MINUIT [1] to fit histograms or data read in from a file and optionally displays the fit results on a screen. Hardcopies of pictures can easily be made with a quality suitable for publication. Mn_Fit can also be used purely as a plotting package without using the fitting.

The usual problem with such packages is overcoming a threshold level of knowledge so that one can use the program. This guide is supposed to ease that process. I will first give a series of brief examples of how to do simple things with Mn_Fit. For a full description of all the commands see the Mn_Fit manual and/or the online help, which is available with the `HELP` command once you have started Mn_Fit.

The reference manual gives more details on how to start Mn_Fit. If it has been installed in a standard system directory on a Unix machine you should just be able to give the command `mn_fit` to start it. If, in addition, you have the `DISPLAY` variable set properly, you can just type `<CR>` when you are prompted for the output device.

The generic Mn_Fit commmands to do something are given in a double box, while the explicit examples are given in an oval box. Assuming that Mn_Fit has been installed properly you should be able to give the command `exec filename`, where the `filename` is in the framed box, to run any of the examples. It does not make sense in a guide like this to give all the arguments of some commands (such as the `set` commands). In these cases I will indicate that there are further arguments with .... Optional arguments are given in square brackets `[optional]`.

The examples given below can all be found in the directory `$MN_FIT_SYS/help/tutorial`. The files always start with the `set default` command to set all options back to their default values. With a normal Mn_Fit installation the HBOOK example histograms are stored in the file `$MN_FIT/help/hbook_example.his`. This is probably a link to `../test_data/hbook_example.his`. If you have installed you own version of `mn_tutorial.tar.gz` then `hbook_example.his` will be in the `tutorial` directory. The examples have been tested with Mn_Fit Version 5.06. They should run with this and any later version. Compatibility with versions less than 5.00 is not guaranteed, as the Mn_Fit directory structure changed with this version.

This guide is written assuming that you are working on a computer under the Unix operating system. See Appendix C for how to adapt the commands for other systems.

2 Generalities

All Mn_Fit commands are in English and can be abbreviated to the point of ambiguity (*e.g.* `plot 1` or `pl 1` are equivalent, but `p 1` is ambiguous and a list of the possible commands will be given). Commands are not case sensitive (although Unix filenames are). Within Mn_Fit you can just type in the command and it will prompt you for any arguments that are needed.

You can either type Mn_Fit commands on the command line or put a series of commands in a file and then execute that file. For the examples in this tutorial you could type in all the commands yourself. An easier way is to just run the relevant file (`exec filename`) which contains the commands. By default the commands that are executed are echoed on the screen so you can see what is in the file.

In this guide I use the word "histogram" in a very loose sense to include both histograms and series of data points (*e.g.* a table of (x, y) values and optionally their errors). As far as possible the program should sort out what sort of data is being used and act accordingly. Similarly, although there are a series of commands for Ntuples (`ntuple project, plot, scan` etc.), they

also work equally well on histograms. A "plot" refers to the drawing of one or more "histograms" on the screen.

Within Mn_Fit the default comment character is `!` and a `-` at the end of a line indicates that the next line is a continuation line. If you are more used to Unix comment characters you can change the defaults using the `set character` commands.

If you want to keep the current value of a number use the `=` sign. You can also (mostly) perform a simple arithmetic operation on the current value with the syntax `=+n, =-n, =*n, =/n` where `n` is a number.

2.1 Histogram Identifiers

Identifiers are numbers used to identify a histogram, Ntuple or series of data points. Instead of the usual single identifier for each plot Mn_Fit has two identifiers associated with each "histogram". This permits significantly more flexibility in numbering your histograms and enables you to group together histograms which are associated with each other. To select a "histogram" with a particular secondary identifier when you are asked for a histogram number use the syntax `id&idb` (*e.g.* `10&1` to get histogram 10 which has secondary identifier 1). If the secondary identifier is omitted, the default will be used. The default can be changed with the `SET IDB` command. Any plots read in after this command have the new secondary identifier. Whenever a histogram identifier is requested you can use the syntax `id&idb`, even though this is usually abbreviated to `id` in this guide.

For some commands, *e.g.* `INDEX, FETCH, PLOT`, you can specify a range of plots which the command applies to. The range is delimited by a `:`. You can specify a range of primary or secondary identifiers or both. In general if you give as primary identifier 0, this means all histograms you have read in or created; *e.g.* `FETCH filename 0` will fetch all the histograms in file `filename`.

As in most cases when you are asked to input a number, it is also possible to use a register, parameter etc. to specify a histogram identifier or secondary identifier (see Section 6.2). More examples on how to use secondary identifiers will be given as we go along.

3 Plotting Data

In order to plot a histogram, you can either fetch it from a file where it has been stored by another program (*e.g.* HBOOK [2] or ROOT [3]) or typed in via a text editor, or you can create a histogram within Mn_Fit. The most common procedure is to book and fill your histograms with HBOOK or ROOT and then to store them in a file using a subroutine such as HRPUT or HROUT or equivalent ROOT output routines. Such histograms are stored in ZEBRA [4] format in a direct access file (usually called an RZ file) or a ROOT file.

3.1 HBOOK Histograms

If your histograms are stored in an RZ file give the following commands to plot a histogram:

```
fetch filename id
plot id
```

`filename`	is the name of the file with the histogram
`id`	is the histogram identifier.

If you have ROOT histograms change the `fetch` command to `root_open filename` followed by `root_fetch id`.

For example:

Example P1: `tutorial/plot1`

```
!
! Mn_Fit tutorial plotting example 1
! Plotting an HBOOK histogram
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 1
plot 1
```

will make a plot of a generated Gaussian distribution.

3.2 Data Files

Another possibility is to type in the histogram entries using a text editor (or create the same thing with a program). This has the advantage that you can just look at the data, but is not as compact or flexible a format as an HBOOK RZ file or a ROOT file. In this case the commands are:

```
dat_fetch filename
plot id
```

`filename`	is the name of the file with the histogram
`id`	is the histogram identifier.

Note that with this command (the command `READ DATA` is equivalent), you read in all the histograms in a file, whereas you can select which histograms you want to fetch with the `FETCH` command.

For example you could create the file `plot2.mnd`:

```
id 1
title A test histogram
order x y dy
data
1 2 0.3
2 5 0.4
3 8 0.6
4 10 0.5
5 7 0.2
7 1 0.2
end
```

The `order` card says which column is for which variable, *i.e.* column 1 has the x values, column 2 the y values and column 3 the error on the y values. Then give the commands:

Example P2: `tutorial/plot2`

```
!
! Mn_Fit tutorial plotting example 2
! Plotting data stored in a text file
!
set default
dat_fetch $MN_FIT/help/tutorial/plot2.mnd
plot 1
```

If you do this you see that the scale is certainly not optimal. You can improve this by either adding a `limit` card, *e.g.*:

```
limit 0 8
```

before the `data` card, or giving the commands:

Example P3: `tutorial/plot3`

```
!
! Mn_Fit tutorial plotting example 3
! Setting limits
!
set default
dat_fetch $MN_FIT/help/tutorial/plot2.mnd
set x limit 0 8
plot 1
!
! Resets the limits back to default
!
set x limit 0 0
fetch $MN_FIT/help/tutorial/hbook_example.his 7
plot 7
set x limit 0 4
set y limit 0 450
plot 7
```

To be more flexible you can also omit the `order` card and give the command `set order x y dy` before the `dat_fetch` command. This is most useful if you want to read in certain columns from a table.

3.3 Booking and Filling inside Mn_Fit

You can also book and fill histograms inside Mn_Fit. As options you can specify whether the histogram should be binned or not and whether the errors should also be stored:

```
book/bin/err id 'title' ndim nbinx xlo xhi ...
fill id x [weight]
book/unbin/[no]err id 'title' ndim maxpnt
fill id x y [dx dy]
```

`id`	is the histogram identifier,
`title`	is the histogram title,
`ndim`	is the number of dimensions of the histogram or variables in the case of an unbinned histogram,
`nbinx`	is the number of bins,
`xlo`	is the lower boundary of the histogram,
`xhi`	is the upper boundary of the histogram,
`maxpnt`	is the maximum number of points for which space should be reserved.

The commands `book` and `histogram book` are synonymous, as are `fill` and `histogram fill`.

For example if you would book and fill the data from the previous example inside Mn_Fit you would give the following commands:

Example P4: `tutorial/plot4`

```
!
! Mn_Fit tutorial plotting example 4
! Booking and filling histograms inside Mn_Fit
!
set default
book/unbin/err 1 'Data points filled in Mn_Fit' 1 10
fill 1 1 2 0.0 0.3
fill 1 2 5 0.0 0.4
fill 1 3 8 0.0 0.6
fill 1 4 10 0 0.5
fill 1 5 7 0.0 0.2
fill 1 7 1 0.0 0.2
! Set the limits for the data
partition 1 0 8
set y mode log
set y lim 0.5 20
plot 1
```

Note that the x errors must be given explicitly in this case. As I don't know when you have finished filling with such data you should use the `partition` command to set the boundaries of the plot. `partition` can also be used to select part of a histogram. The `set y mode log` command plots the y axis using a log scale. You have to set the plot limits after selecting the log mode, to ensure that you do not have illegal plot boundaries for a logarithm.

3.4 Modifying Plots

There is usually something in the style of the plots that is not to your taste. Almost all aspects of the plots can be modified using the `set` command. Some typical commands that are used

include:

```
set default
set symbol number
set colour symbol name|number
set x tick ntick nbtick ...
set x label 'text' =
set y mode log ...
```

`number`	is the symbol number you want,
`name`	is the name or number of the colour ,
`ntick`	is the number of ticks on the x-axis,
`nbtick`	is every how many ticks a big tick is drawn with a number on the scale

Default symbols are defined for the different styles of input data:

```
Symbol  1 for 1-d histograms
       -1 for a series of points without errors
      -32 for a series of points with errors
       12 for 2-d histograms
        1 for scatter plots
```

The list of all possible symbols is given in Appendix A.

The `set x|y|z mode` command changes the way the scale is shown. Choices are `real, integer, log, date, time`.

The current setting of any option can be seen with the `show` command. For most `set` commands you can just replace the command with `show`. If this does not work then use `help show` to see the other possibilities, e.g. `show sizes` to see the settings of many text sizes etc.

For example, if you want to have red filled symbols with the y error bars shown, a number on the y scale every 4 ticks and labels on the x and y axes, give the following commands:

Example P5: `tutorial/plot5`

```
!
! Mn_Fit tutorial plotting example 5
! Changing the plot
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 2
set colour symbol red
! Filled squares with y errors shown
set symbol -32
! 10 ticks with a big tick every 2 and the first scale offset by 1
set x tick 10 2 1
set y tick 7 4
set x label 'Size (cm)' =
set y label 'Number of Entries' =
plot 2
show label
show tick
```

The `set default` command sets all options back to their default values. The `=` after the label commands is for the position of the labels. When asked for a position, size, etc. an `=` sign means take the current value. If you do not give it you will be prompted for their values. You can then hit `<CR>` to keep the current default.

3.5 Overlays

On many occasions you will want to display more than one histogram in the same picture. The usual commands for this are:

```
overlay id symbol
overlay/diff id symbol
```

`id`	is the histogram identifier,
`symbol`	is the symbol number,
`/diff`	gives the 2nd histogram a different y scale.

The symbol number is a required argument (you will be prompted with a suggested value). You can also give the symbol number with the `plot` command, but in this case it is optional. Options for Mn_Fit commmands are usually given with the VMS syntax of a /.

For example:

Example P6: `tutorial/plot6`

```
!
! Mn_Fit tutorial plotting example 6
! Overlays
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 5
fet 6:7
plot 5
over 6 2
over/diff 7

redraw
```

Note that if you want to `fetch` more histograms from the last file that you opened you do not need to give the filename. You can fetch a range of histograms by giving the form `id1:id2`. The blank line after `over/diff` means take the default symbol number. The `redraw` commmand remakes the last plot and in this case automatically removes the ticks for the first histograms from the right axis because of the `over/diff` command.

If you want to show error bars on overlayed plots that have the same binning, you can use the `xshift` command to move the points a bit. An example of this is given in Section /ref-sec:oper:arith, example 9.1.

3.6 Two-Dimensional Histograms

You can plot two-dimensional histograms in many ways. The most common commands are:

```
plot id
2dim lego id theta phi
2dim surface id theta phi
2dim contour id
```

`theta`	is the polar viewing angle,
`phi`	is the azimuthal viewing angle.

The following example shows some of the possibilities:

Example P7: `tutorial/plot7`

```
!
! Mn_Fit tutorial plotting example 7
! 2-dimensional histograms
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 10
set
  default
! Windows 2x2 with 1.5cm spacing
  wind 2 2 1.5 1.5
endset
!
set x zero off     !Turn off the lines at x=0 and y=0
set y zero off
plot 10                 !Plot with area proportional to number of entries
plot 10 1               !Plot with random dots within a bin
set z limit 0 10        !Sets the vertical axis limits for the next plots
2dim lego 10 30 30      !Lego plot
2dim surf/c1 10 45 45   !Rebinned version as a colour surface plot
```

Note that you can either give the whole of the `set` command on one line or just give the command `set` and then a series of commands on the following lines.

Mn_Fit also supports true scatter plots, where the x, y coordinates of each of the data points are stored. These can be stored as HBOOK Ntuples in a disk file, as a text file using the `ntuple` card, or booked and filled within Mn_Fit.

4 Fitting - Quick and Dirty

I almost hesitate to put this section so near the beginning of the tutorial, as my experience is that "quick and dirty" fitting should generally be avoided! My philosophy is that as Mn_Fit always uses MINUIT for the function minimization (except if you are spline fitting), when you start to fit you should get immediate access to the full power of MINUIT, plus the extra commands that I have built in for Mn_Fit. However, I recognise that sometimes one just wants to quickly fit a Gaussian to see what the mean/width is without having to type in too many commands. For this purpose, for a very limited number of functions, you can fit and view the result with a single command.

```
fit/function/type id
set fit commands ...
```

`function`	is the name of the function to fit to,
`type`	is the type of the fit.

Allowed functions are `Gaussian, flat, line` and the allowed fit types are `chi, likelihood`. You can specify which commands are executed when the fit is run - the default is `minimize` and `display`.

4.1 Fitting a Gaussian

The simplest case is the fitting of a Gaussian to a histogram. For example you can make a likelihood fit to the generated Gaussian that was shown in the first plotting example:

Example Q1: `tutorial/quickfit1`

```
!
! Mn_Fit tutorial quickfit example 1
! Fitting a histogram with a Gaussian
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 1
plot 1
fit/gaus/like 1
exit
```

Note that the default commands leave you inside MINUIT so that you can then play around more if you want to.

The `display` command shows you the histogram you are fitting with the result of the fit overlayed. You get a list of all the fit parameters and their errors. In addition you can see the χ^2 and/or likelihood for the fit and its confidence level, the area of the histogram and the function(s) and the fit status. All this information is very useful in deciding whether your fit worked properly and whether it is a good fit.

4.2 Gaussian + Background

Example Q2: `tutorial/quickfit2`

```
!
! Mn_Fit tutorial quickfit example 2
! Fitting a histogram with a Gaussian and a straight line
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 7
part 7 0 4
fit/gaus/line 7
0
exit
```

In this example we fit using 2 functions. The histogram only has entries between 0 and 4, so we first `partition` it so that only that part is fit. Other methods of selecting the fit region will be given in the main fitting section. The type of fit was not given on the command line, so Mn_Fit issues a prompt for the fit type. The quality of the fit here is not particularly good, as the generated background was an exponential rather than a straight line.

5 Functions

Mn_Fit has a large number of functions already built-in. A series of commands exist to include these functions in fits and also to plot them. In order to do something with a function you first have to `add` it to the list of functions in use.

```
function add name
function add comis file.f nfun
function info
help function list
function delete
function use
function plot|overlay|histogram
function edit
function compile
set function ...
```

`name`	is the name of the function to fit to,
`file.f`	is the name of a FORTRAN file which contains a user defined function,
`nfun`	is the function number within the COMIS function.

The command `help function list` gives a list of the available functions. The `function edit` and `function compile` commands are for COMIS functions.

5.1 Plotting Simple Functions

Example F1: `tutorial/function1`

```
!
! Mn_Fit tutorial functions example 1
! Adds and plots a Gaussian and a Chebyshev polynomial
!
set default
function delete 0
fun add gauss sigma
1000
0
1
! Plot the first function (Gaussian) with secondary id 1 and symbol -1
function plot 1 &1 1
-5 5
function add chebyshev 2
100
1
0.3
! Overlay the 2nd function (Chebyshev) with secondary id 2 and symbol -2
function overlay 2 &2 2
function info
```

In this example the `function delete 0` removes any existing functions. First a Gaussian (with the width expressed in terms of standard deviation) is added and is plotted from -5 to 5. Then a 2nd order Chebyshev polynomial is overlayed. In commands such as `function plot` where a list of function numbers can be given the `&` or a `<CR>` indicates the end of the list. The same comment applies to the `ntuple project|plot` command. Whenever you plot a function in Mn_Fit it is stored as a series of data points in a "histogram" so that you can then apply any normal histogram operations to it. If the plot is part of a `display` command then the function is stored with the same primary identifier as the histogram you are fitting. If not, it is stored with primary id 98765, as indicated in the header at the top of the picture.

5.2 COMIS Functions

If the function that you want to fit with is not already defined you can write your own using COMIS [5]. COMIS stands for "Compilation and Interpretation System", and enables you to write FORTRAN functions and run them inside Mn_Fit, without recompiling and linking the whole program. From within a COMIS function you can also call any of the Mn_Fit predefined functions. Use the `help function list` command to find the function number to call. If you give the command `function add file.f` and `file.f` does not yet exist, a skeleton function will be created.

Example F2: `tutorial/function2`

```
!
! Mn_Fit tutorial functions example 2
! COMIS function consisting of 2 Gaussians with a variable separation
!
set default
function delete 0
function add comis $MN_FIT/help/tutorial/function2.f 0 n
5
Two Gaussians
AREA
MEAN
SIGMA
AR2/AR1
DMEAN
100
3
0.8
0.5
3
!
function plot 0 &1 1 0 10
```

Within one FORTRAN function you can define several functions for plotting and fitting. In this example this feature is not used and the functon is not given to the editor, hence the syntax `fun add comis tutorial/function2.f 0 n`. If you leave off the `0 n`, function 0 will be added and you will be asked if you want to edit the file.

The FORTRAN function is:

Example F2: tutorial/function2.f

```
      FUNCTION XMNCMI(XX,YY,NP,DPAR,NUSER,WFERR)
C
C     This is a user defined function consisting of 2 Gaussians,
C     both with the same  width, where the ratio of the areas
C     and the separation are the extra free parameters
C
C     XX     is first variable
C     YY     is the second variable
C     NP     is number of parameters
C     DPAR   are the parameters in DOUBLE PRECISION
C     NUSER  is the user function number
C     WFERR  is the error on the function (0 in most cases)
C
      IMPLICIT NONE
C
      REAL          XMINNM,XMAXNM,XBINNM,YMINNM,YMAXNM,YBINNM
      COMMON/MNUSR/ XMINNM,XMAXNM,XBINNM,YMINNM,YMAXNM,YBINNM
      LOGICAL QDEBUG
      INTEGER NDEBUG
      COMMON/MNDBG/ QDEBUG,NDEBUG
C     Mn_Fit registers - Do not overwrite this common block
      REAL REGIS
      COMMON/MNREGI/REGIS(0:400)
C
      DOUBLE PRECISION XMNCMI,XMNCLC
      EXTERNAL XMNCLC
      REAL XX,YY
      INTEGER NP,NUSER
      DOUBLE PRECISION DPAR(20),WFERR,WXX,WYY,WMN1,WMN2,WPAR(3)
      INTEGER NCALL,II
C
      DATA NCALL/0/
C
      NCALL = NCALL + 1
      WXX = DBLE(XX)
      WYY = DBLE(YY)
C
C     First Gaussian (function 6) - copy the parameters to local storage
      WPAR(1) = DPAR(1)
      WPAR(2) = DPAR(2)
      WPAR(3) = DPAR(3)
      WMN1 = XMNCLC(6,WPAR,3,0,XX,YY)
C
C     Second Gaussian - calculate the parameters
      WPAR(1) = DPAR(1) * DPAR(4)
      WPAR(2) = DPAR(2) + DPAR(5)
      WPAR(3) = DPAR(3)
      WMN2 = XMNCLC(6,WPAR,3,0,XX,YY)
C
      XMNCMI = WMN1 + WMN2
C
      IF (NCALL.LE.10 .OR. MOD(NCALL,100).EQ.0) THEN
          WRITE(6,*) 'NCALL',NCALL,', DPAR',(DPAR(II),II=1,NP)
     +     ,WMN1,WMN2,XMNCMI
      ENDIF
C
      END
```

This example adds a COMIS function which consists of 2 Gaussians. The extra parameters are the ratio of the 2 areas and the separation between the means.

6 Accessing Information

6.1 Histograms

There are a number of commands that allow you to see which histograms exist in a file, which you have already fetched, what the properties of the histograms are etc.

```
index [id1[:id2]]
directory [id1[:id2]]
cdir name
ldir
zdir
dump id
histogram dump id
```

`id1`	is the first histogram id,
`id2`	is the second histogram id,
`name`	is the name of the HBOOK directory,

The commmands `index` and `directory` are synonymous and list the histograms that you have fetched into memory. The command `cdir, ldir` and `zdir` enable you to look at the HBOOK memory and disk file contents. The 2 dump commands are extremely useful for looking at how histograms are defined and exactly what the contents of them are. You can also use them to look at individual events in Ntuples. The commands `dump`, `histogram dump` and `ntuple dump` are synonymous, except inside MINUIT, where `dump` gives a dump of the fit calculation for each data point.

Example F1: `tutorial/info1`

```
!
! Mn_Fit tutorial information example 1
! Index and dumping the contents of a histogram or Ntuple
!
set default
delete 0
! Fetch some histograms
fetch $MN_FIT/help/tutorial/hbook_example.his 0
ldir
set idb 100
fetch 1:10
! Examples of index and directory
index
dir 1:5
dir 0&100
! Dump some histograms
histogram dump 1 y
! Set default secondary id back to 0
set idb 0
histogram dump 31 5 8
```

In this example first all histograms in memory are deleted and the histograms in the example file are fetched. Then the secondary identifier is modified and histograms $1 \rightarrow 10$ are fetched (Normally you would fetch the histograms from another file). A few examples of how to use the `index|directory` command are shown as well as the output from the `ldir` command. Finally a `dump` is made of histogram 1 as well as events $5 \rightarrow 8$ of Ntuple 31. Note that histogram `1&100` is dumped as that is the current default secondary identifier.

6.2 Numbers, Registers, Parameters, etc.

One of the strengths of Mn_Fit is the ability to easily access all histogram and Ntuple contents, function parameters, registers etc. at all places in the program. In almost all cases these values can be used whenever Mn_Fit requires a number.

```
deposit var = expression
examine var
show register number
set plot id default
```

`var`	is a register, parameter, histogram bin,
`expression`	is an arithmetic expression,
`id`	is the histogram identifier,
`number`	is a register number or a range of numbers (m:n).

The following lists the possibilities for giving a number:

- A number. *e.g.* 4, -5.2, 3.2E+02. It is not necessary to give the decimal point for real numbers.

- A register

  Rn or IRn where n can be from 0 to 400.

  Note that you are allowed to fill registers 0-99 with whatever you wish using the `DEPOSIT` command. All register contents are stored as real numbers. The integer format `IR` is used when you want to convert the contents to an integer to put them in a text string.

  The contents of registers 100 and above are given in Appendix A. These registers are filled with the command `set plot id default`. Registers > 300 contain extra variable names that you have defined. Up to 200 variables are allowed.

- A function parameter, its parabolic error, its positive or negative error, or its limits. The syntax to use is:

  Pn(m) for the parameter

  ERRn(m) for the parabolic error on the parameter

  ERNn(m) for the negative MINOS error on the parameter

  ERPn(m) for the positive MINOS error on the parameter

  LOLIMn(m) for the lower limit on the parameter value

  HILIMn(m) for the upper limit on the parameter value

  where `n` is the function number and `m` is the parameter number. If you are fitting you can also use just the MINUIT parameter number *e.g.* `Pn`.

- The centre of a bin, the bin width, the contents or the error on the contents (including asymmetric errors). The syntax to use is:

  Xn(m) for the bin centre or the x value of a point

  DXn(m) for half the bin width or the error on the x value

  Yn(m) for the bin contents or the y value of a point

  DYn(m) for the error on the y value

  DNXn(m) for half the bin width or the negative error on the x value

  DNYn(m) for the negative error on the y value

  DPXn(m) for half the bin width or the positive error on the x value

  DPYn(m) for the positive error on the y value

  where n is the plot number (including the optional secondary identifier if needed) and m is the bin number. For 2-dimensional histograms give both bin numbers. *e.g.* `Xn(l,m)`. However note that it is not possible to use the Y value of a bin, as Y **ALWAYS** means the bin contents for histograms. You can calculate the y value of a particular bin or the y bin width using registers 134, 135 and 136. For plots with asymmetric errors `DX` and `DY` are interpreted as the negative errors if you want to get the value and as both errors if you are using the `DEPOSIT` command.

- The value of an Ntuple variable. The syntax to use is:

Xn(m,nvar) or

Xn(m,tvar) where n is the plot number, m is the event number, nvar is the variable number and tvar is the variable name.

For cuts and expressions for projections you can just give the variable name.

You can also define new variable names using the syntax `deposit name = value`. The variable names must be alphanumeric expressions of maximum length 8 characters and can also include `$` and `_`. Variable names which match one of the names below (*e.g.* `R, ERR`), or a FORTRAN intrinsic function are not allowed nor is the name `ALL`. The names are converted to upper case. The new variables can be deleted using the `remove` command.

```
e.g. dep pi = 3.14159
     dep r2 = sin(pi/2)
```

These extra variables are stored in registers > 300. Note that numbers are always floating, with the exception of `IR`.

As mentioned at the beginning of this guide, in general if you want to keep the current value of a number use the `=` sign. You can also add, subtract, multiply or divide the current value by using the syntax `=+1.0, =*R2`, etc.

6.3 Text Formatting for Plots

You should normally just type in the text you want to see. The HIGZ routine IGTEXT is used to interpret the various escape characters that can be used in order to get superscripts, subscripts etc. There are a number of changes from the standard IGTEXT input in the way you give a text string to make it easier and quicker to get what you want. Upper and lower case characters should be entered as you want them to be, including Greek and special characters. To switch modes precede the string you want in a different mode by the relevant escape character given below. You will stay in the new mode until you give the termination character. The exception is `@` which is only valid for the next character The following escape characters can be used: IGTEXT:

```
[  go to Greek
]  end of Greek
"  go to special symbols
#  end of special symbols
^  go to superscript
?  go to subscript
~  go to Zapf Dingbats font (HIGZ Postscript only)
!  end of superscript or subscript or Zapf Dingbats
&  backspace one character
$  termination character (do not use in your strings)
@  to get any of the escape characters (if they are in the
   IGTEXT character set)
```

The following characters can be entered directly:

```
a,b...z     lowercase alphabetic
A,B...Z     uppercase alphabetic
0,1...9     the digits
,           comma
.           period
;           semicolon
```

```
:           colon
+           plus sign
-           minus sign
*           star
/           slash
=           equals
_           underscore
|           vertical bar
'           quote or apostrophe
< >         less than, greater than
( )         left/right parentheses
{ }         curly brackets
```

The following sequences will be printed as a single character:

```
=<          less than or equal to
>=          greater than or equal to
->          right arrow
<-          left arrow
+/-         plus or minus
```

German letters with umlauts as well as the sharp s (also called sz) can be obtained with the syntax `\"a` etc. The sharp s is `\"S`.

As the < and > signs can be typed in directly, you cannot use them for switching between lower and upper case. This only affects getting small versions of digits. You can force a switch by preceding the symbol by an `@`.

For more details on what special characters, Greek letters, German special characters etc. exist see Appendix A.

For example
`@<1234@>` will get you $_{1234}$,
`'[p]^+![p]^-! Invariant Mass (GeV/c^2!)@!'` will get you $\pi^+\pi^-$ Invariant Mass (GeV/c^2)!
and
`Die Bl\"atter sind gro\"S'` will get you Die Blätter sind groß when printed on a Postscript printer.

If you are required to give a title, comment, key, or axis label the text should be enclosed in single quotes, if you want to also put other parameters on the same line. If you omit the quotes the whole of the rest of the line will be used. This enables you to put the parameters associated with the text (*e.g.* comment position) on the same command line:

```
comment id new 'This is a comment positioned at 10,10' 10.0 10.0
```

Text which is always a single word (parameter names, filenames etc.) should not be included in quotes. If you want to pass quoted text as a parameter to a macro, enclose the whole string in double quotes:

```
exec test "'Text to pass'"
```

If you want to pass or give a null string (for example set the axis label back to a blank), give 2 single quotes:

```
Give axis label: ''
```

A `<CR>` always keeps the current text.

As well as giving normal text, it is also possible to include the values of parameters or registers etc. in a text string. The format to use is:

```
Text {parameter[,format]} MORE TEXT
```

i.e. a section enclosed in braces signifies that this part should be translated into text. `Parameter` can be a number, register, parameter, bin contents etc. `Format` is a FORTRAN format statement including parentheses. If you omit the format statement the default format is `(1PG12.5)` and `(I8)` for real and integer numbers respectively. Integers are integer registers (syntax `IR`) and any user variable name that starts with `I`. `NINT` is used to convert real numbers to integers.

This form of text string applies to comments, keys, axis labels, titles etc. To include a normal { in a text string, precede it by a `@`.

Superscripts and subscripts are terminated by a `!`. `!` is also used for command line history in Unix version, therefore you have to quote `!` with a backslash to get an exclamation mark if you give the command interactively. In addition if `!` is defined as the comment character you must also enclose the expression in single quotes:

```
set x label '(GeV/c^2\!)'
```

If you put the command in a file, then do not include the backslash, but you must still include the quotes:

```
set x label '(GeV/c^2!)'
```

6.4 Adding Text to Plots

Text and explanations of symbols can be added to plots with the `comment` and `key` commands.

```
comment id new 'text' ...
key id new symbol 'text' ...
```

`id`	is the histogram id thast the comment or key is associated with,
`text`	is the text you want to write.

The contents of registers, parameters etc. can also be put into text strings and either written to the screen or a file, or added to plots.

Example I2: `tutorial/info2`

```
!
! Mn_Fit tutorial information example 2
! Adding a comment and key to a plot
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 1
plot 1 1 345
! Store the information on this histogram in registers 100 -> 200
set plot 1 default
! Draw the key 1cm from the top left corner of the histgoram
deposit r1 = r201 + 1
deposit r2 = r204 - 1
key 1 new 3345 'Histogram Symbol' r1 r2 0.4 = l cm
! Put a comment on the value of the mean just above the mean position
dep r1 = r134
dep r2 = r127
comment 1 new 'Mean = {r134,(f6.2)}' r1 r2 0.4 = c plot
```

In this example the key is drawn using screen coordinates (*i.e.* cm), while the comment is positioned using plot coordinates. Keys are always left adjusted regardless of what you give for the position option. The command `set plot id default` loads the registers 100 → 300 with the information on the histogram. The values and format of the registers are specified in the text string using the `{var,(format)}` syntax. In Mn_Fit the minimum and maximum number of entries also include the errors on the data points.

6.5 Making Tables

Suppose you make a series of fits and want to output the result in a table. This example shows how to project an Ntuple into a series of histograms with different cuts and then how the resultant histograms are fit and the results put into another histogram as well as being written to a table.

Example I3: `tutorial/info3`

```
!
! Mn_Fit tutorial information example 3
! Making a series of fits and outputing the results in a table
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 31
!
function delete 0
cut delete 0
cut new flat > r1
cut new flat <= r2
cut use -1
! Set the dump filename and write out the header
set dump file $TMPDIR/info3.fil
set dump lpt
message 'Fit Results'
message ''
set dump tty
book/unbin/err 1&11 'Fit Results' 1 4
do i=1,4
  dep r1 = (@i-1)
  dep r2 = r1 + 1
  ntuple plot 31 gaus1 &@i 20 -5 5
  fit/like/gauss 31&@i
  dep r3 = 0.5*(r1+r2)
  fill 1&11 r3 p1(2) 0.5 err1(2)
! Write the result to a file
  set dump lpt
  message 'Fit @i: Area = {p1(1),(1pg11.4)} +/- {err1(1),(1pg11.4)}'
  message '        Mean = {p1(2),(1pg11.4)} +/- {err1(2),(1pg11.4)}'
  set dump tty
  exit
enddo
!
plot 1&11
fit/chi/flat 1&11
exit
set dump close
shell cat $TMPDIR/info3.fil
```

In this example two cuts are set using registers as the cut variables. The command `cut use -1` means `AND` all the defined cuts. Events are selected in intervals of 1 unit in the variable `flat` and the Ntuple is projeted onto the `Gaus1` axis. Note that the values of the registers are accessed at the time of the `ntuple plot` command. More details on Ntuple manipulation can be found in Section 10. The results of the fit are stored in `p` and `err`, where the first number is the function number and the second is the variable number – in this case function 1 and variable

2 (the `MEAN`). Once the projections have been made and a Gaussian fit done, the means for each interval are fit to a flat line. As can be seen from the `DISPLAY` all the values are consistent with 1, as is the fit. Do loop variables must be single letters and are referenced insde the do loop using the syntax `@i`. `message` normally writes to the screen, but the command `set dump lpt` sends the text to the dump file (by default `mn_dump.dat`).

7 Fitting and MINUIT

Not suprisingly as Mn_Fit was written to do fitting there are many commands that are related to fitting. In the first part I will give examples of some of the more common ones. An introduction to many of the possibilities is given in the Mn_Fit manual in the section "Fitting with Mn_Fit". I will start with the command syntax that is used if you are fitting with a single histogram. Examples of fitting with multiple histograms will be given later. In this section I will only dicuss fitting with MINUIT. Spline fitting and smoothing will come in a later section.

7.1 Simple Fits

```
fit/type id ...
display
minimize
include xlo xhi
exclude ylo yhi
```

`id`	is the histogram id that you want to fit,
`type`	is the fit type – chi,likelihood,slikelihood,
`xlo`	is the lower limit for an exclusion or inclusion,
`xhi`	is the upper limit for an exclusion or inclusion.

As a first example I show how to fit a histogram with a Gaussian signal and a polynomial background. Chebyshev polynomials are usually the most stable and converge fastest.

Example M1: `tutorial/fit1`

```
!
! Mn_Fit tutorial fitting example 1
! Fitting with a Gaussian and a Polynomial Background
!
set default
delete 0
! Fetch some histograms
fetch $MN_FIT/help/tutorial/hbook_example.his 7
! Add the functions - Gaussian and 2nd order Chebyshev
fun del 0
fun add gauss s
1000
1.5
1
fun add cheb 2
100
0
0
! Likelihood fit
fit/like 7
! Exclude the region which has no data
excl 4 10
minimize
display
exit
```

Although a Gaussian is by far the most common function to fit with, it does not have particularly nice properties. You will often find that you have to set the start value for the mean fairly close to where you think it should be, otherwise it will not find the correct peak. The excluded part of the function is shown as a dotted line in the plot. This implies that even if you exclude regions you should make sure that any fit functions you write will not blow up in these regions.

8 Miscellaneous

This section collects together a number of miscellaneous useful commands.

8.1 Hardcopies

There are 2 ways of making hardcopies. Either you can make a file (usually Postscript) of the last picture on the screen (`hardcopy` command), or you can `capture` the hardcopy device and then execute the normal plotting commands. In Mn_Fit it is not possible to write simultaneously to the screen and a hardcopy device.

```
hardcopy device
capture device
close
```

`device`	is the device name.

Although there are many possible output devices - see HELP Screen_Devices and HELP Hardcopy_Devices, by far the most common ones are: `Postscript` for a Postscript output (Portrait) which can be printed immediately, and `EPost` for encapsulated Postscript output for later inclusion in documents.

A useful device is `none`, which allows Mn_Fit to go through the motions of plotting without actually drawing anything. This enables you not to make plots of repetitive fitting procedures, even if the `display` command is in the command list.

8.2 Windows or Zones

On may occasions you want to show several histograms on the same page. In Mn_Fit this is known as dividing the page into "windows". In PAW they are called "zones".

```
window nwindx nwindy sepx sepy
no_window
set autotrim on|off
set autosize on|off
```

`nwindx`	is the number of windows horizontally,
`nwindy`	is the number of windows vertically,
`sepx`	is the separation between the windows horizontally in cm,
`sepy`	is the separation between the windows vertically.

When windowing by default the sizes of the title, ticks, scale and labels will be reduced (if they are set to their default values). Use the command `set autosize off` command to turn off this feature. If you specify 0 separation between the windows then the scale between the windows will automatically be suppressed. In these cases the last number on the scale usually overlaps with the first one on the next plot. By default the last number is therefore not shown. Use the command `set autotrim off` to turn off this feature.

8.3 Opening and Closing Histogram Files

If you just want to open an HBOOK RZ file you can use the commnd `hb_open` or simply `open`. You can close any open files (useful if you have a job that is going to write to the file) using the command `hclose`. Note that for historical reasons the command `close` closes any open hardcopy files.

8.4 The Mouse

The mouse can be used to position comments and keys and to `draw` things like lines, arrows, boxes and polygons. When asked to give the position move the cursor to the correct position and click on the left mouse button. If you are changing an item and are happy with its current position, just put the cursor anywhere on the picture and click on the right mouse button. Use the command `set mouse on|off` to turn on or off use of the mouse.

9 Manipulating Histograms

9.1 Arithmetic Operations

A whole series of commands are available for manipulating histograms:

`add`	`id1 id2 id3 scale1 [scale2]`
`subtract`	`id1 id2 id3 scale1 [scale2]`
`multiply`	`id1 id2 id3 scale1 [scale2]`
`divide`	`id1 id2 id3 scale1 [scale2]`
`efficiency`	`id1 id2 id3`
`average`	`id1 id2 id3`

`id1`	is the identifier of the 1st histogram,
`id2`	is the identifier of the 2nd histogram,
`id3`	is the identifier of the output histogram,
`scale1`	is a scale factor to apply to the 1st histogram,
`scale2`	is a scale factor to apply to the 2nd histogram

Note that the efficiency command calculates the errors of the new histogram assuming a binomial distribution. This in turn assumes that the errors on each of the bins are Poisson, with the mean given by the bin contents.

You can also move or scale the data for the x, y and z axes. This can be useful if you are trying to overlay plots with points at the same x values and overlapping errors.

`normalize`	`id1 id2 norm`
`scale`	`id1 id2 scale`
`xshift`	`id1 id2 shift`
`xscale`	`id1 id2 scale`

`id1`	is the identifier of the 1st histogram,
`id2`	is the identifier of the 2nd histogram,
`norm`	is the total area of the new histogram,
`scale`	is a scale factor to mutitply the contents by.

There are similar commands `yshift, zshift, yscale, zscale` and the commmand `scale` is the same as `yscale` for one-dimensional histograms and the same as `zscale` for two-dimensional histograms.

Example O1: `tutorial/oper1`

```
!
! Mn_Fit tutorial histogram manipulation example 1
! Shifting the data so that it can be overlayed
!
set default
dat_fetch $MN_FIT/help/tutorial/oper1.mnd
xshift 1 1&1 -0.05
xshift 2 2&1 +0.05
!
set title user 'Shifted and Not Shifted Histograms'
wind 1 2 0 0
plot 1 -31
over 2  32
!
set x lim -0.5 4.5
plot 1&1 -31
over 2&1  32
```

In this example two sets of data points have the same x values. The top plot shows what would (and often does) happen if you simply overlay them. The second plot shows how to use `xshift` so that both results can be seen clearly. Note the use of 0 window separation automatically suppresses the bottom scale on the upper plot. The command works equally well on true histograms.

9.2 Slices and Bands

There are not yet any built in commands for making slices and bands. You have to use a combination of `cut` and `project` inside a do loop. The file `tutorial/info3` shows you how this can be done, as well as providing a nice illustration of how to use Mn_Fit registers.

10 Manipulating Ntuples

There are a series of commands for manipulating Ntuples, a selection of which are given here:

```
ntuple project id var1 [var2] &idb ...
ntuple plot    id var1 [var2] &idb ...
ntuple dump id
cut new ...
cut use ...
```

`id`	is the identifier of the Ntuple,
`var1`	is the name or number of the variable to project onto

A example of projecting Ntuples was already given in Section 6.5. As a simple starting point using the same Ntuple:

Example N1: `tutorial/ntuple1`

```
!
! Mn_Fit tutorial ntuple example 1
! Projecting a row-wise Ntuple
!
set default
fetch $MN_FIT/help/tutorial/hbook_example.his 31
!
cut delete 0
! 1-dimensional histogram
ntuple plot 31 gaus1 &1 -1
! 2-dimensional scatter plot
ntuple plot 31 gaus1 gaus2 &2 0 -5 5 0 10
! 2-dimensional histogram
window 1 2 0 0
ntuple project 31 flat (flat+gaus1) &3 8 0 4 10 0 10
plot 31&3
! Profile plot
ntuple sprof 31 flat (flat+gaus1) &4 8 0 4 0 20
plot 31&4
```

In this example first a projection is made onto a 1-dimensional histogram and then onto a 2-dimensional scatter plot. For the scatter plot the limits for each axis are given in this example. In the 3rd picture a 2-dimensional histogram and a profile plot are made. For the profile plot the errors are the spread on the data points. As you can see profile plots are often a more compact way of conveying the same information, but you lose information about the shape of the distribution and the tails.

Projections of Ntuples (and histograms) always get the same primary identifier as the orginal Ntuple and the user should give the secondary identifier. Expressions should be enclosed in parentheses. This is not strictly necessary as Mn_Fit looks for arithmetic operations in the projected variable, but it helps to avoid problems.

Note that for row-wise Ntuples you do not have to give the limits for the projection, as they are stored with the Ntuple. This is not the case for column-wise Ntuples, so that for the moment you have to give the limits for the projection by hand. This will be fixed in the near future at the cost of looping over the Ntuple twice. It will therefore always be quicker to give the binning if you know what you want.

11 Cleaning up Data

You have your data and want to play around with it, fit it to various distributions etc. However there a some points that clearly do not belong to the distribution, or you booked your histogram badly and it would take a week to rerun over the data or ... Here are a few examples of how to get round such problems.

11.1 Bad Profile Plot Entries

Profile plots are often used as a convenient way of plotting 2 variables. For example the difference in the intercept of a particle with a detector and the measured signal position vs. the particle momentum. A common problem is that at some point you have insufficient statistics and you get some entries with no error or an error that is obviously too small. If you try to then fit the data it is not possible to calculate a sensible χ^2 and the fit fails.

This example is not yet written!

12 Sizes

In this section I will attempt to collect together most of the commands that change the sizes of things in plots.

```
set x|y size   val
set x|y psize  val
set x|y margin val
set ssize  val
set tsize  val
set idsize val
set usize  val
set dsize  val
```

`val`	is the size in cm,

The meaning of the commands is:

size is the size of the histogram in the picture

psize is the overall size of the picture

margin is the size of the left—bottom margins.

The example shows many of the sizes discussed in this section. Because it is fairly long it is split into 2 macros (which is good practice anyway!).

Example S1: `tutorial/size1`

```
!
! Mn_Fit tutorial sizes example 1
! Picture and histogram sizes and margins
!
exec $MN_FIT/help/tutorial/size1_setup
plot 1
!
! Overall picture size
draw arrow 1 red 2 cm -3
0.0 1.0
h_xpsize 1.0
comment 1 new 'x psize' 9.0 1.2 r3 = c cm
draw arrow 1 red 2 cm -3
1.0 0.0
1.0 h_ypsize
comment 1 new 'y psize' 0.8 12.5 r3 90 c cm
!
! Margins
draw arrow 1 red 2 cm -3
0.0 5.0
r201 5.0
dep r1 = 0.5*(0.0+r201)
comment 1 new 'x margin' r1 5.2 r3 = c cm
draw arrow 1 red 2 cm -3
5.0 0.0
5.0 r203
dep r2 = 0.5*(0.0+r203)
comment 1 new 'y margin' 4.8 r2 r3 90 c cm
!
! Plot size
draw arrow 1 red 2 cm -3
r201 5.0
r202 5.0
dep r1 = 0.5*(r201+r202)
comment 1 new 'x size' r1 5.2 r3 = c cm
draw arrow 1 red 2 cm -3
5.0 r203
5.0 r204
dep r2 = 0.5*(r203+r204)
comment 1 new 'y size' 4.8 r2 r3 90 c cm
```

Example S2: `tutorial/size1_setup`

```
!
! Mn_Fit tutorial sizes example 1
! Called by size1 to setup the sizes
!
set default
! Fetch some histograms
fetch $MN_FIT/help/tutorial/hbook_example.his 1
!
! Put the plot sizes and the x label offset in variables
dep r3 = 0.3
dep h_xpsize = 18
dep h_ypsize = 25
dep h_xsize  = 12
dep h_ysize  = 18
dep h_xmarg  =  3
dep h_ymarg  =  4
dep l_xoff   = 0.5 * h_xsize
!
! Set the sizes
set
  x psize h_xpsize
  y psize h_ypsize
  x size  h_xsize
  y size  h_ysize
  x margin h_xmarg
  y margin h_ymarg
  font -1004
  box on
  idsize 0.25
  x label 'set x label'  l_xoff -1.5 0.3 = r
  y label 'set y label' -1.0 +2.0 0.3
endset
```

A Symbols, Fonts and Hatching

The complete set of Mn_Fit symbols is shown in Fig. 1 and the standard Mn_Fit registers are given in Table 1.

The same fonts are available in Mn_Fit as in PAW. For some examples see Fig. 4. Note that the font and precision are specified with the form `spfff`, where `s` is the sign of the font, `p` is the precision and `fff` is the font, e.g. `-1013` to get font -13 with precision 1. The fonts available depend on the version of the graphics package used. In Fig. 4 HIGZ Postscript fonts are shown.

The complete character set for the device independent HIGZ font (font 2000) is shown in Fig. 2, while that for HIGZ Postscript font -13 is shown in Fig. 3. Note that you only see the Zapfdingbats fonts when you look at or print the Postscript file. Use the `HARDCOPY` or `CAPTURE` command to make such a file.

HIGZ Postscript fonts can be obtained using precision 0,1, or 2. They are identical when you make a hardcopy. The differences are how they appear on the screen. If you use Postscript fonts and precision 2 you get the hardware default font on the screen. If you use precision 1 then the standard IGTEXT font (font 2000) is used on the screen and the Postscript font is used when printing. If you use precision 0, you get the Postscript fonts on the screen, but not superscripts, subscripts, Greek characters etc. In addition the alignment of the text on the screen is often not right for precision 0 and 2.

The default font everywhere is -1004 i.e. IGTEXT on the screen and Helvetica Postscript font when printing. If you want the standard IGTEXT font you should give the command `set font 2000`.

Device independent hatchings are available in all HIGZ interfaces and some examples are shown in Fig. 5.

Hatch -3 is often used, because it gives a nice shaded overlay. However note that this is really a pattern (because you cannot see through it) and if you use it in an overlay it obscures the ticks. The solution is to do a `plot/noclear` on the 1st id again to get the scale redrawn. The solid fill patterns (100) and examples of hatch -3 are shown in Fig. 6.

The `KEY` command can be used to provide a legend of the different symbols and shading used in a plot. Some examples are given in Fig. 7.

A picture showing the standard Mn_Fit sizes and the appropriate commands is shown in Fig. 8.

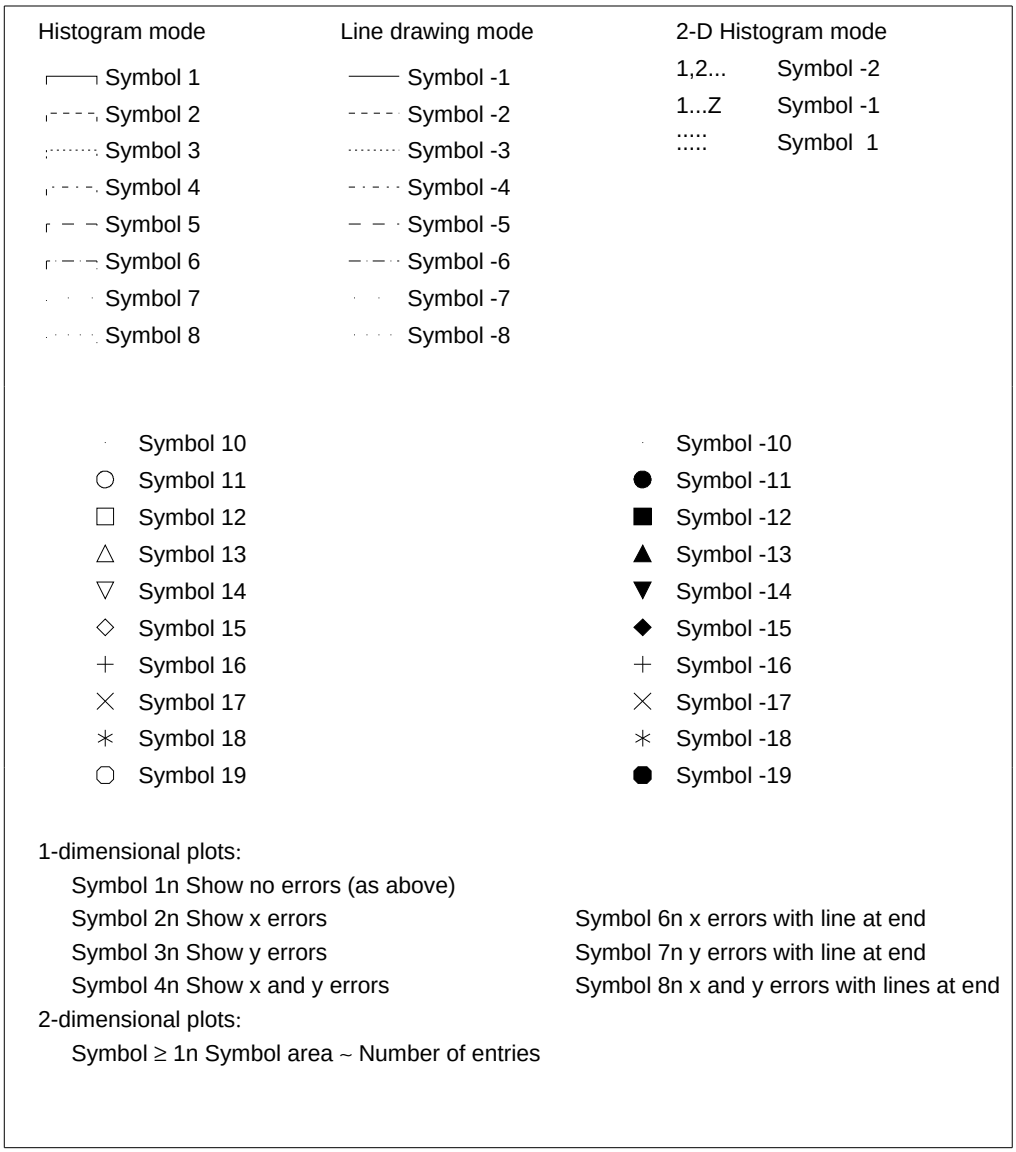
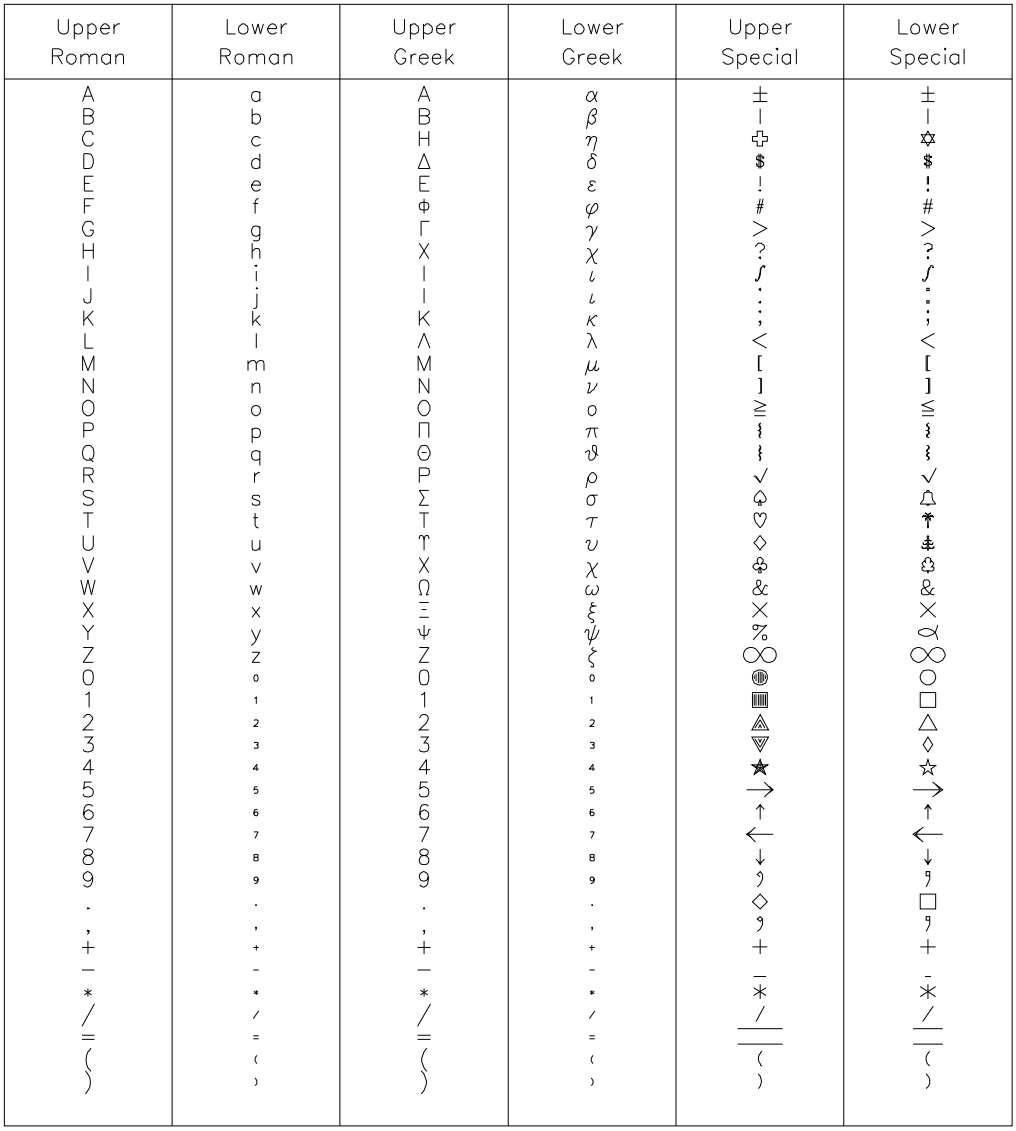


Figure 1: Mn_Fit Symbols

101	The result of a `SUM` or `INTEGRATE` command
102	The number of points summed over in `SUM`
111	The chi**2 or likelihood from the fit
112	The confidence level of a fit
113	The fit status (3 means converged properly)
114	The EDM (estimated distance to minimum) of the fit
Registers 121 to 200 and 231 to 257 are filled if you give the command `SET PLOT id [&idb] DEFAULT`	
121	The plot identifier
122	The secondary plot identifier
123	The number of entries (histograms) or points
124	The dimension of the plot (positive for histograms, negative for Ntuples and a series of points).
125	The area under the plot (i.e. sum of weights)
126	The minimum number of entries (weight)
127	The maximum number of entries (weight)
128	The creation date of the histogram (yymmdd)
129	The creation time of the histogram (hhmm)
131	The number of bins on x-axis (0 for Ntuples and points)
132	The lower limit of the x-axis
133	The upper limit of the x-axis
134	The mean value for the x-axis
135	The sigma for the x-axis
136	The number of bins on y-axis (0 for Ntuples and points)
137	The lower limit of the y-axis
138	The upper limit of the y-axis
139	The mean value for the y-axis
140	The sigma for the y-axis
	etc. up to 14th dimension of an Ntuple.
Registers 201-204 contain the positions of the corners of the current plot in cm:	
201	x position left
202	x position right
203	y position bottom
204	y position top
Registers 205-210 contain the limits used for the drawing of each of the axes in plot coordinates:	
205	x minimum
206	x maximum
207	y minimum
208	y maximum
209	z minimum
210	z maximum
Registers 231-257 contain total contents, as well as underflows and overflows of the default histogram:	
231	Underflows x-axis
232	Contents x-axis
233	Overflows x-axis
For 2 dimensional histograms 9 registers are filled (contents in register 235), while for 3-dimensional histograms 27 registers are filled (contents in register 244).	
Registers > 300 contain extra variable names that you have defined. Up to 200 variables are allowed.	

Table 1: The list of standard Mn_Fit registers.

Font 2000, i.e. IGTEXT



[	Switch to Greek	]	Switch back from Greek
"	Switch to special	#	Switch back from special
?	Switch to subscript	!	Switch back from subscript
^	Switch to superscript	!	Switch back from superscript
&	Backspace one character	@	Print an escape character

Figure 2: HIGZ Portable Software Characters and the escape characters to switch modes.

Font -1004, i.e. Postscript - Helvetica

Upper Roman	Lower Roman	Upper Greek	Lower Greek	Upper Special	Lower Special	Upper Zapf	Lower Zapf
A	a	A	α	±	≈	✡	❁
B	b	B	β	|	≡	✢	✲
C	c	H	η	∃	⊥	✣	✵
D	d	Δ	δ	∀	∂	✤	✺
E	e	E	ε	!	*f*	✥	✹
F	f	Φ	ϕ	#	⊂	✦	✸
G	g	Γ	γ	>	⊃	✧	✷
H	h	X	χ	?	⊃	★	✺
I	i	I	ι	∫	⊇	☆	✹
J	j	I	ι	:	∉	✪	✸
K	k	K	κ	;	⊂	✯	✷
L	l	Λ	λ	<	⊆	✰	●
M	m	M	μ	[	∈	✰	❍
N	n	N	ν	]	∉	✭	■
O	o	O	ο	≥	∇	✭	❑
P	p	Π	π	{	∧	✮	❒
Q	q	Θ	θ	}	∨	✶	❑
R	r	P	ρ	√	⇔	✮	❒
S	s	Σ	σ	♠	⇐	✵	▲
T	t	T	τ	♥	⇑	✳	▼
U	u	Y	υ	♦	⇒	❁	◆
V	v	X	χ	♣	⇓	✶	❖
W	w	Ω	ω	≤	&	✷	◗
X	x	Ξ	ξ	×	ϖ	✸	❙
Y	y	Ψ	ψ	∝	~	✹	❙
Z	z	Z	ζ	∞	✖	✺	▮
0	0	0	0	⊗	⊗	❍	❍
1	1	1	1	⊕	⊕	❑	❑
2	2	2	2	∅	∅	❒	❒
3	3	3	3	◊	◊	✓	✓
4	4	4	4	•	•	✔	✔
5	5	5	5	→	→	✕	✕
6	6	6	6	↑	↑	✖	✖
7	7	7	8	←	←	✗	✗
8	8	8	8	↓	↓	✘	✘
9	9	9	9	↔	↔	✙	✙
.	.	.	.	.	.	✚	✚
,	,	,	,	,	,	✛	✛
+	+	+	+	+	+	✜	✜
-	-	−	−	∠	∠	✝	✝
*	*	*	*	〈	〈	✞	✞
/	/	/	/	÷	÷	✟	✟
=	=	=	=	≠	≠	†	†
(	(	(	(	≡	≡	✠	✠
)	)	)	)	″	″	✡	✡

[	Switch to Greek	]	Switch back from Greek
"	Switch to special	#	Switch back from special
?	Switch to subscript	!	Switch back from subscript
^	Switch to superscript	!	Switch back from superscript
~	Switch to Zapf Dingbats	#	Switch back from Zapf Dingbats
&	Backspace one character	@	Print an escape character

Figure 3: Postscript Version of HIGZ Portable Software Characters and the escape characters to switch modes.

HIGZ Postscript Fonts

Font/Prec	
- 1/0	*abcdefghijklmno ABCDEFGHIJKLMNO 123*
- 2/0	**abcdefghijklmno ABCDEFGHIJKLMNO 123**
- 3/0	***abcdefghijklmno ABCDEFGHIJKLMNO 123***
- 4/0	abcdefghijklmno ABCDEFGHIJKLMNO 123
- 5/0	*abcdefghijklmno ABCDEFGHIJKLMNO 123*
- 6/0	**abcdefghijklmno ABCDEFGHIJKLMNO 123**
- 7/0	***abcdefghijklmno ABCDEFGHIJKLMNO 123***
- 8/0	abcdefghijklmno ABCDEFGHIJKLMNO 123
- 9/0	*abcdefghijklmno ABCDEFGHIJKLMNO 123*
-10/0	**abcdefghijklmno ABCDEFGHIJKLMNO 123**
-11/0	***abcdefghijklmno ABCDEFGHIJKLMNO 123***
-12/0	αβηδεφγχιικλμνο ΑΒΗΔΕΦΓΧΙΙΚΛΜΝΟ 123
-13/0	abcdefghijklmno ABCDEFGHIJKLMNO 123
-14/0	❁❂❃❄❅❆❇❈❉❊❋●❍■❏ ✡✢✣✤✥✦✧★✩✪✫✬✭✮✯ ✑✒✓
-15/0	*abcdefghijklmno ABCDEFGHIJKLMNO 123*
-16/0	**abcdefghijklmno ABCDEFGHIJKLMNO 123**
-17/0	***abcdefghijklmno ABCDEFGHIJKLMNO 123***
-18/0	abcdefghijklmno ABCDEFGHIJKLMNO 123
-19/0	*abcdefghijklmno ABCDEFGHIJKLMNO 123*
-20/0	**abcdefghijklmno ABCDEFGHIJKLMNO 123**
-21/0	***abcdefghijklmno ABCDEFGHIJKLMNO 123***
-22/0	αβχδεφγηιϕκλμνο ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟ 123
-23/0	abcdefghijklmno ABCDEFGHIJKLMNO 123
-24/0	❁❂❃❄❅❆❇❈❉❊❋●❍■❏ ✡✢✣✤✥✦✧★✩✪✫✬✭✮✯ ✑✒✓

Figure 4: Examples of HIGZ Postscript Fonts

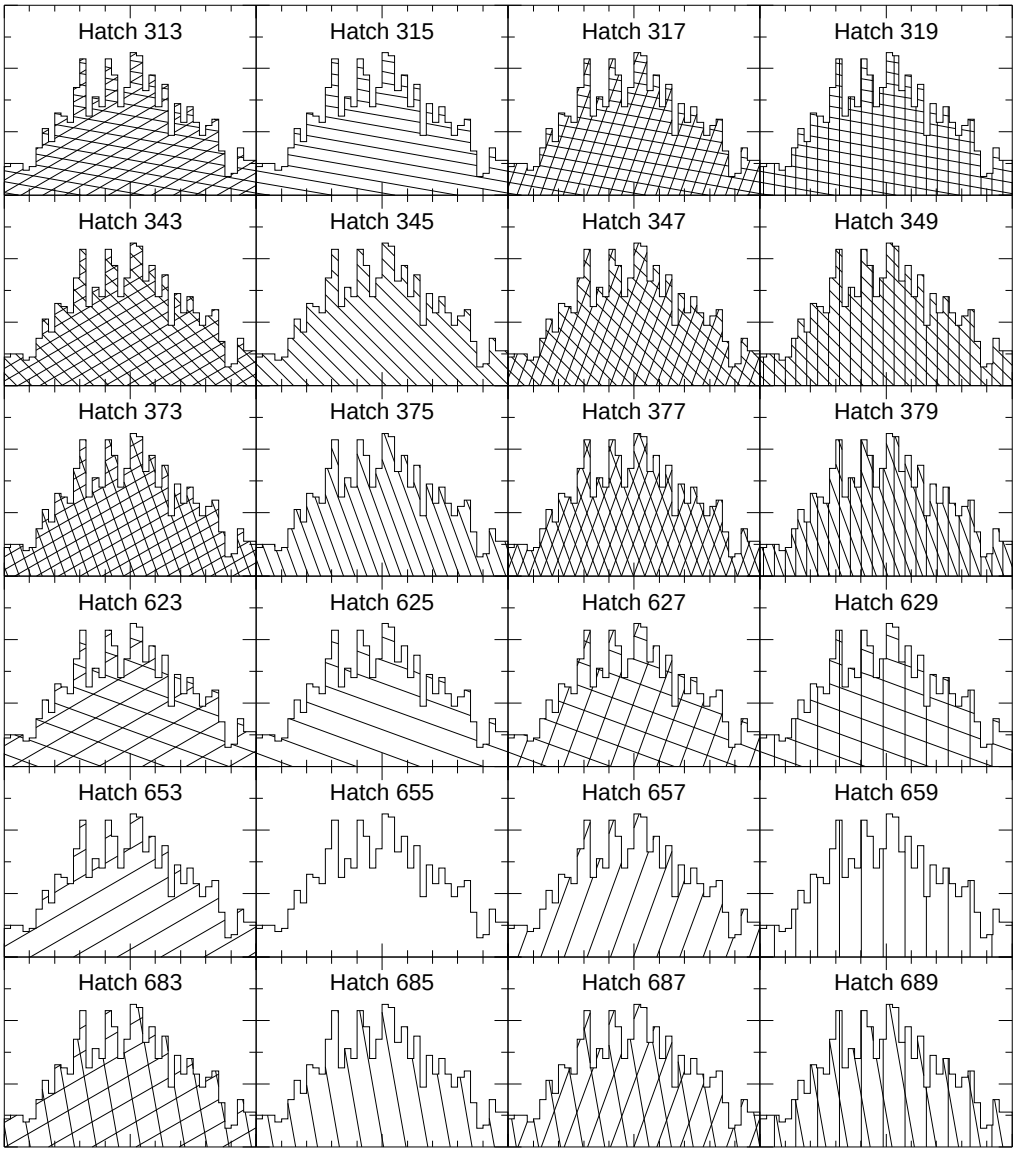


Figure 5: Examples of HIGZ Hatching

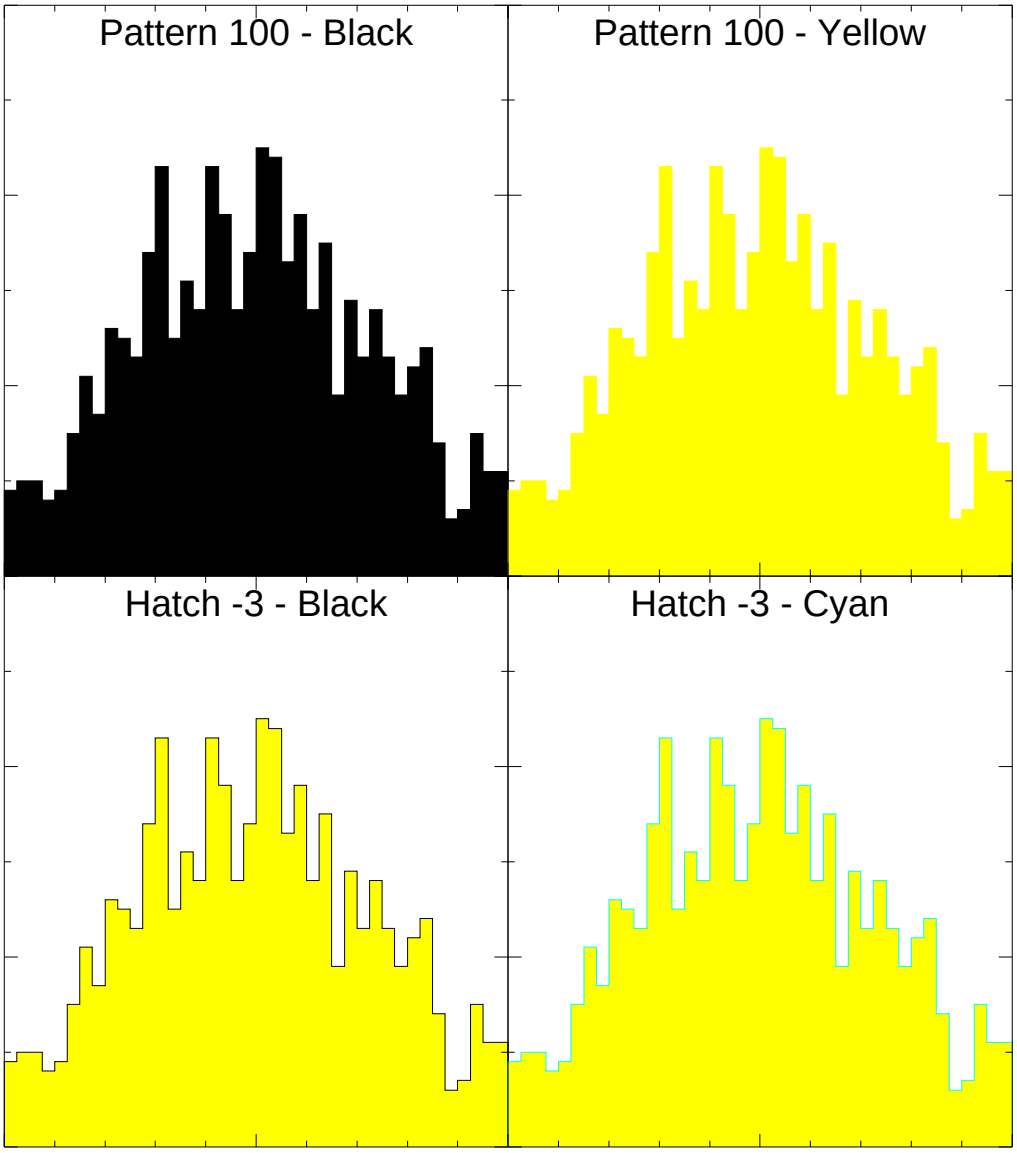


Figure 6: Examples of Patterns

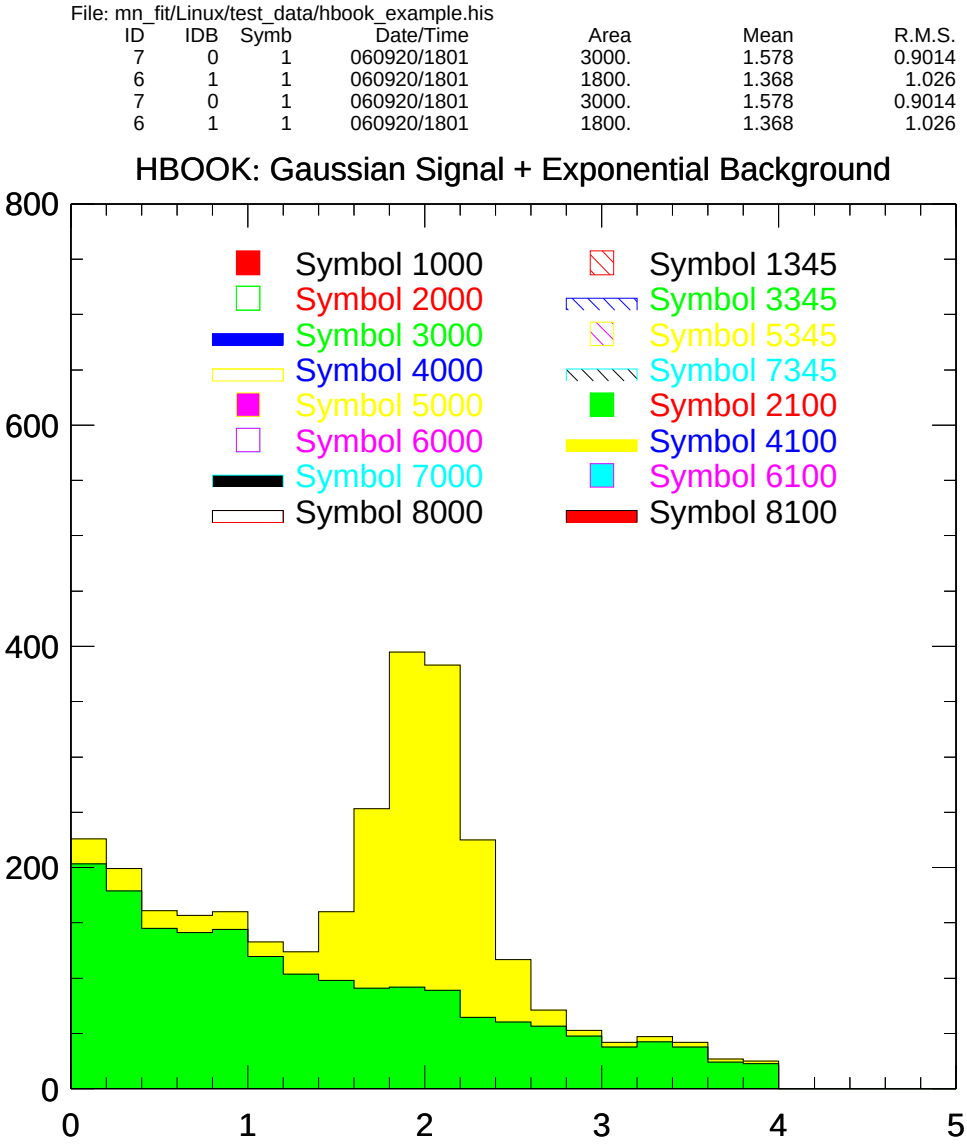


Figure 7: Examples of Keys

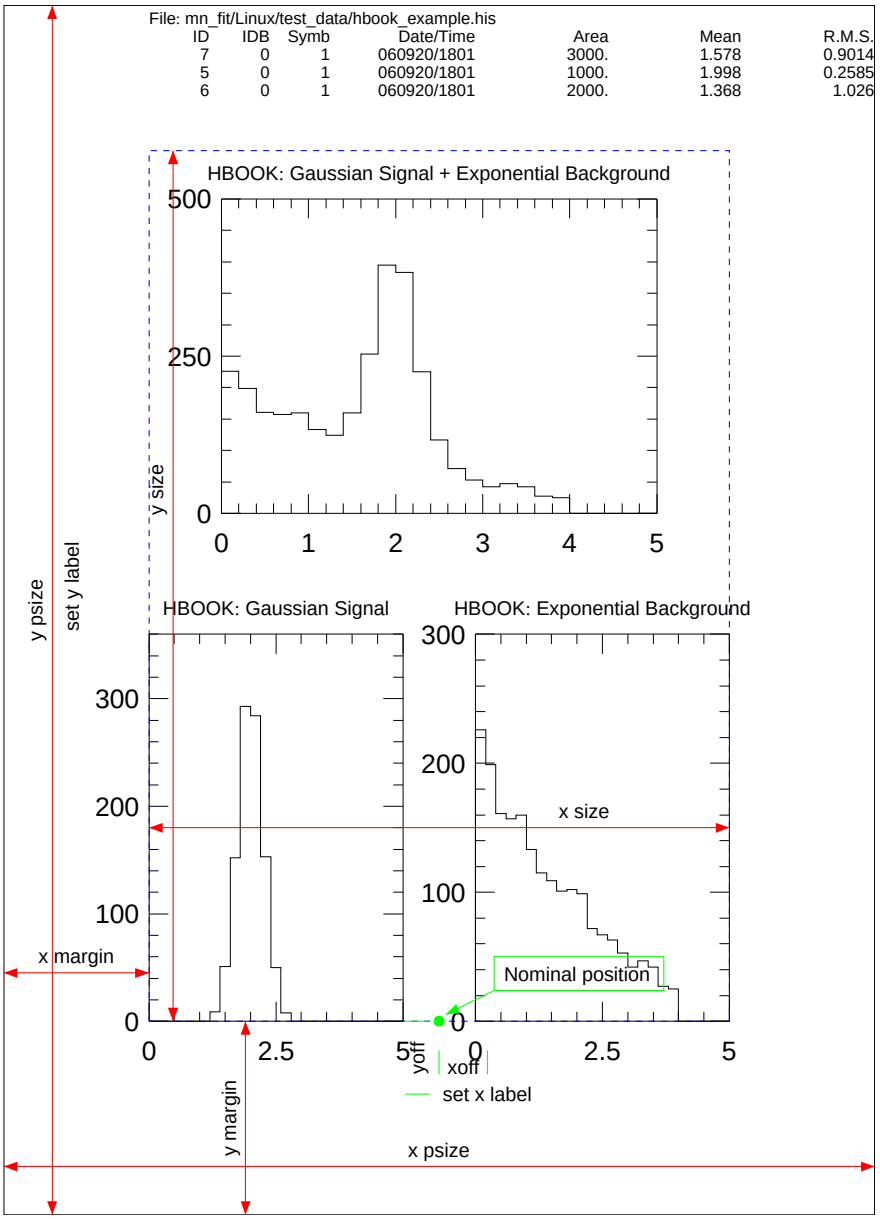


Figure 8: Picture sizes and the commands to set them

B Font Problems?

If your output stops after Fig. 3 then you have to edit the `mn_tutorial.tex` file and replace `font00` with `font14`.

C Mn_Fit on Other Machines

Mn_Fit currently runs under the Unix and VMS operating systems. This guide has been written as if you are on a machine using the Unix operating system. Use the following substitutions if you are using another operating system.

C.1 VMS

Instead of environment variables logicals are used on VMS machines. In all commands where environment variables are used (*e.g.* `$MN_FIT`), use the following replacements:

```
$MN_FIT/help/        mn_fit_help:
$MN_FIT/test/        mn_fit_test:
```

References

[1] CERN Program Library. MINUIT – Function Minimization and Error Analysis. D506.

[2] CERN Program Library. HBOOK - Reference Manual. Y250.

[3] R. Brun et al. ROOT – An Object-Oriented Data Analysis Framework. http://root.cern.ch.

[4] CERN Program Library. ZEBRA – Overview of the ZEBRA System. Q100/Q101.

[5] CERN Program Library. COMIS – Compilation and Interpretation System. L210.

Index