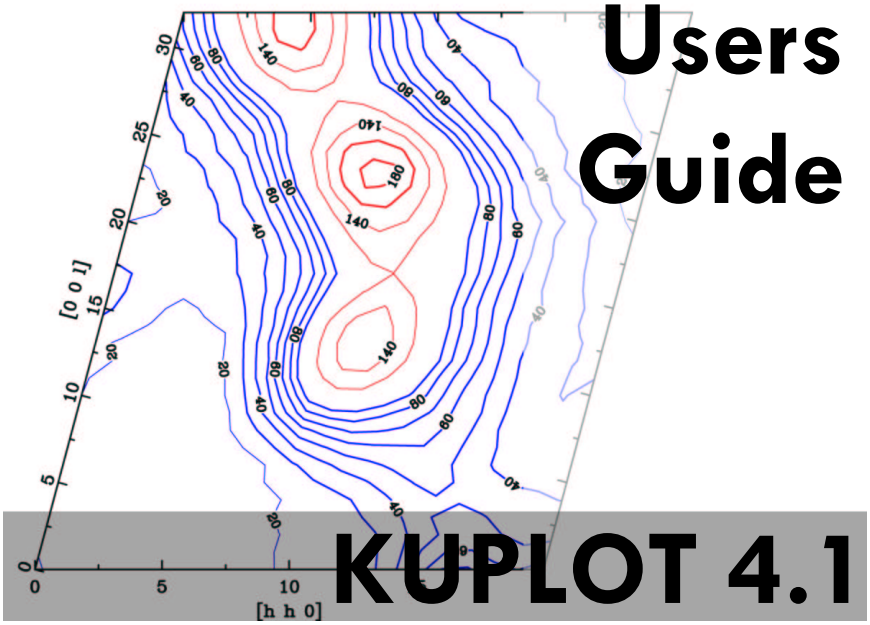
Users Guide



KUPLOT 4.1

written by

Thomas Proffen
Email: tproffen@lanl.gov

Reinhard Neder
Email: reinhard.neder@mail.uni-wuerzburg.de

Document created: February 27, 2003

Preface

Disclaimer

The *KUPLOT* software described in this guide is provided without warranty of any kind. No liability is taken for any loss or damages, direct or indirect, that may result through the use of *KUPLOT*. No warranty is made with respect to this manual, or the program and functions therein. There are no warranties that the programs are free of error, or that they are consistent with any standard, or that they will meet the requirement for a particular application. The programs and the manual have been thoroughly checked. Nevertheless, it can not be guaranteed that the manual is correct and up to date in every detail. This manual and the *KUPLOT* program may be changed without notice.

KUPLOT is intended as a public domain program. It may be used free of charge. Any commercial use is, however, not allowed without permission of the authors.

Acknowledgments

The routines for the line editing were taken from the program GNUPLOT. *KUPLOT* uses now the *PGPLOT* library written by T.J. Pearson, California Institute of Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is KUPLOT ?

KUPLOT is an interactive plotting program controlled by a command language. *KUPLOT* is part of the diffuse scattering simulation package *DISCUS*, however it can be used totally independent of the *DISCUS* software.

KUPLOT is now using the *PGPLOT* library for the actual plotting and supports all graphic devices supported by *PGPLOT* such as X-window terminal or POSTSCRIPT output. As mentioned before, the program is controlled by a command language which includes a FORTRAN style interpreter which allows the use of variables, loops and conditional i statements. This results in a high degree of flexibility and allows the creation of quite complex graphics. It also allows *KUPLOT* to be used to process large numbers of data files and produce the desired plots automatically.

KUPLOT can process simple 1d data files. The program supports normal line graphs, marker, error bars as well as spline interpolation between the data points. Line color, marker color, line type, line width and various other parameters can be adjusted. *KUPLOT* allows one to plot 3D data sets using contour lines, colored bitmaps or both. The colormap for the bitmap can be freely changed using the FORTRAN interpreter (see section 6.1). The program also allows one to define different contour line sets for one data set, e.g. finer spaced lines for diffuse scattering and larger space lines in a different color for the Bragg peaks. A page can be divided into different plot areas (see chapter 4). Each frame can contain graphs or the contents of a text file. Frames can have different background colors. Each frame has its own parameter set like e.g. title, axis labels, fonts.

Loaded data sets can be manipulated using the FORTRAN style interpreter or a variety of build-in functions. An integrated FIT sublevel, which allows to fit the following functions to a given 1D data set: polynomial, n Gaussians and n Lorentzians. Furthermore 2D data sets can be fitted using a set of 2D Gaussians. Additionally *KUPLOT* allows fitting a user defined function to a loaded data set.

1.2 What is new ?

Now you just have installed the new version of *KUPLOT* and the question is what are the new features ?

Version 4.1

This version brings a lot of bug fixes and a self-extracting installer for the Windows platform. Several new commands were added: `conv` will calculate the convolution of two loaded data sets. The command `deriv` will calculate the derivative and `spline` will interpolate a given data set onto the grid given by another data set. The command `mouse` has now optional parameters allowing one to use the mouse pointer in macros without bringing up the button interface. Coordinates are returned in the 'res[]' variables. *KUPLOT* can now read *GSAS* data files and supports logarithmic axes.

Version 4.0

The main difference is that the program uses the *PGPLOT* rather than its own plotting routines. This change allowed the integration of some 'mouse' driven functions that can be accessed using the new command `mouse`. Additionally the program now allows one to use special commands controlling the appearance of text, e.g. sub- and superscript or using the Å character (see chapter 2).

Other new features include the import scans from SPEC (X-ray data collection software by CSC, Chicago) files (see 2.1) and new data analysis and manipulation commands (see 7). A new feature of the command language is the interaction with files using the commands 'fget' and 'fput' (see 6). The command itself can now be preceeded by spaces thus allowing one to indent e.g. loops or if blocks.

For a complete list of changes refer to the file '*CHANGES.LOG*' which can be found in the *kuplot/prog* directory of the distribution.

1.3 Getting started

After the program *KUPLOT* is installed properly and the environment variables are set, the program can be started by typing: 'kuplot' at the operating systems prompt. Information about how to compile and install the program can be found in the file *INSTALL* on the top level of the distribution directory. All array sizes that might need to be adjusted to your needs are defined in the file '*kuplot.inc*'. Check the comments within this file for an explanation of the various variables.

The program uses a command language to interact with the user. The command 'exit' terminates the program and returns control to the shell. All commands of *KUPLOT* consist of a command verb, optionally followed by one or more parameters. All parameters must be separated from one another by a comma ','. There is no predefined need for any specific sequence of commands. *KUPLOT* is case sensitive, all commands and alphabetic parameters MUST be typed in lower case letters. If *KUPLOT* has been compiled using the '*-DREADLINE*' option (see installation files) basic line editing and recall of commands is possible. For more information refer to the reference manual or check the online help using ('help command input').

Symbol	Description
"text"	Text given in double quotes is to be understood as typed.
<text>	Text given in angled brackets, is to be replaced by an appropriate value, if the corresponding line is used in *KUPLOT*. It could, for example be the actual name of a file, or a numerical value.
'text'	Text in single quotes exclusively refers to *KUPLOT* commands.
[text]	Text in square brackets describes an optional parameter or command. If omitted, a default value is used, else the complete text given in the square brackets is to be typed.
{text | text}	Text given in curly brackets is a list of alternative parameters. A vertical line separates two alternative, mutually exclusive parameters.

Table 1.1: Used symbols

Names of input or output files are to be typed, as they will be expected by the shell. If necessary include a path to the file. Only the first four letters of any command verb are significant, all commands may be abbreviated to the shortest unique possibility. At least a single space is needed between the command verb and the first parameter. No comma is to precede the first parameter. A line can be marked as comment by inserting a '#' as first character in the line.

The symbols used throughout this manual to describe commands, command parameters, or explicit text used by the program *KUPLOT* are listed in Table 1.1.

1.4 More help

There are several sources of information, first *KUPLOT* has a build in online help, which can be accessed by entering the command 'help' or if help for a particular command <cmd> is wanted by 'help' <cmd>. The online help is also available as printed version in the file 'kupl_cmd.ps' in the directory 'doc' of the distribution. This manual describes background and principle functions of *KUPLOT* and should give some insight in the ways to use this program. Additionally there is an **interactive tutorial** guiding through the different functions of the program. To start the tutorial, go to the directory 'tutorial' of the distribution, start *KUPLOT* and enter 'tutorial' to begin the tour.

KUPLOT is distributed as part of the diffuse scattering simulation software *DISCUS*. However, *KUPLOT* can be used as general plotting program separate from the *DISCUS* program package. To find out about recent updates of *KUPLOT* or to get further information visit the *KUPLOT* WWW homepage at the following sites:

http://www.totalscattering.org/discus/kuplot.html
http://www.uni-wuerzburg.de/mineralogie/crystal/discus/kuplot.html

Chapter 2

Plotting 1D data

In this chapter we will learn how to read, plot and print simple 1D data sets, e.g. xy-data. Chapter 3 explains the plotting of 2D data sets. However, many of the commands to modify a plot discussed in this chapter also apply to the plotting of 2D data.

The simplest data file is a text file with x- and y-coordinates (x_i, y_i) for each point in a separate line, as in the example data file '*test.xy*' listed below:

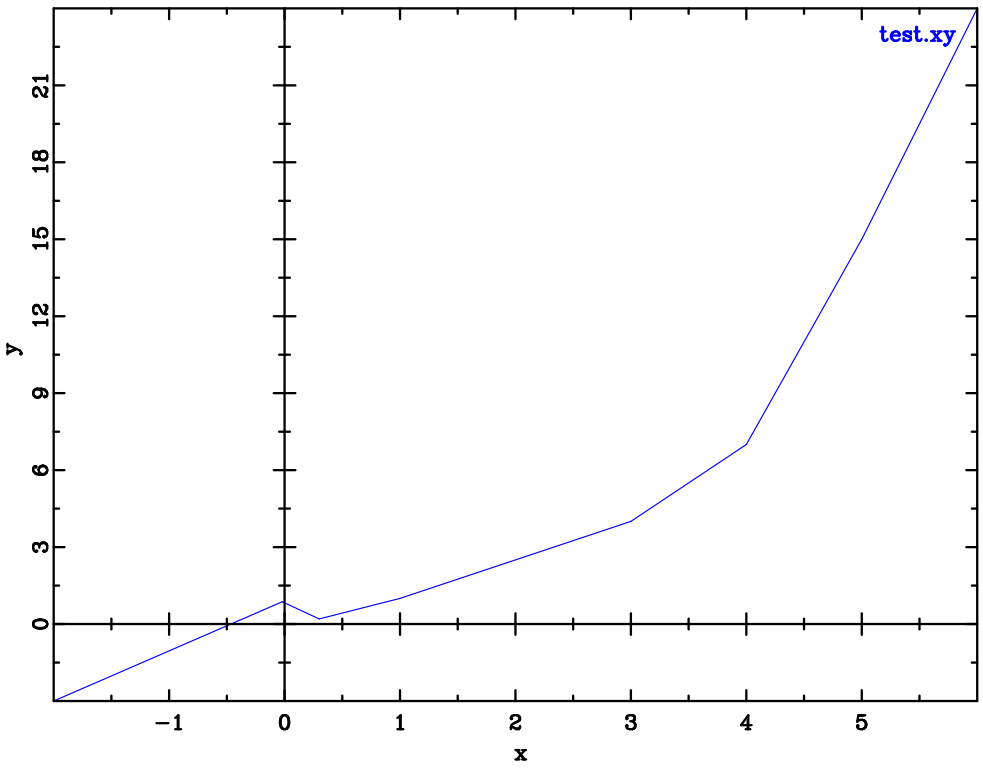


Figure 2.1: The first *KUPLOT* plot

```
-2.00 -3.00
-0.02  0.87
  :     :
```

In order to plot the file '*test.xy*', we only need to enter the following two commands at the *KUPLOT* input prompt:

```
load xy,test.xy
plot
```

The 'load' command reads the specified data file, which is in our case of the type 'xy'. The command 'plot' finally displays the plot on the screen. The resulting plot for this example can be seen in Figure 2.1. The following sections of this chapter describe the supported file formats and how to change, print and save the resulting plots.

2.1 File formats

KUPLOT reads data sets using the 'load' command. The simplest file format is a text file containing values for x and y on a separate line for each data point as in the previous example. To read more than one data set just repeat the 'load' command. The maximum number of data sets that can be read and the maximum total number of data points are defined in the file '*kuplot.inc*' and might be adjusted before *KUPLOT* is compiled. The current limits can be displayed using the command 'show config'. The 'rese' command clears the currently loaded data sets and the next file is read as data set one again.

The various file formats for 1D data supported by *KUPLOT* are summarized in Table 2.1. The file format is specified as first parameter for the 'load' command followed by the name of the file to be read. Note that the 'xy' and 'yx' formats may contain standard deviations $\sigma_{x,y}$ for each data point which may be used to plot error bars using the 'etyp' command in *KUPLOT*.

After a data set is read, *KUPLOT* displays the number of points read and the x- and y-limits. In our previous example, the screen output after entering the 'load xy,test.xy' looks like that:

```
Reading 2 columns ...
Data set no.:   1

  Filename : test.xy                          (     8 points )
  Range  x :   -2.000     to    6.000
  Range  y :   -3.000     to    24.00
```

Since our example file contains no σ values, *KUPLOT* can only read two columns. The file is associated with data set 1. This data set number is subsequently used to alter the appearance of the plot for a given data set (see next section).

2.2 Making a nicer plot

In the example shown in Figure 2.1 we have simply used the defaults of *KUPLOT* and plotted the data set directly. The program *KUPLOT* offers a variety of commands to alter the appearance of the plot itself and

Format	Description
cr	Special crystal structure format exported by *DISCUS*. Each line contains x,y,z, marker type, colour and marker size for the atom at the given coordinates. Note, the first two columns are uses for plotting, the z value is ignored (see *DISCUS* plot sublevel).
ma	Only the x value of each data line is read, y is set to zero. This allows the plotting of tickmarks (e.g. at Bragg peak positions in powder diffraction data) at the x-axis.
sc/st	This command allows one to read scans from a SPEC file. For details refer to the online help.
gs	This command allows one to read GSAS files. For details refer to the online help.
xx	Each line contains only one value that is taken as y value and the point number is stored as x value.
xy	Each line of the data file contains x,y and optionally standard deviations σ_x and σ_y.
yx	Each line of the data file contains y,x and optionally standard deviations σ_y and σ_x.
special	Various other file formats are available to read specific information from data files created by the diffractometer control software for the instruments MAN I and MAN II at the FRM I. For details refer to the online help for the command 'load'.

Table 2.1: Supported file formats for 1D data

the representation of each loaded data set. We are using the same data set '*test.xy*' as in the previous example (with added σ values) and create the plot shown in Figure 2.2. The macro file used to create this plot (Fig. 2.2) is listed below. Note, that the line numbers are added for easy reference within this manual and are not part of the actual macro file.

```
 1  load xy,test.xy
 2  #
 3  achx x-axis [units]
 4  achy y-axis [units]
 5  tit1 \fs KUPLOT demonstration
 6  tit2 Example 2
 7  #
 8  grid on
 9  fnam off
10  #
11  skal -2.5,7.5,-4.5,29.5
12  mark 2,5
13  #
14  lcol 1,6
15  lwid 1,0.5
16  mtyp 1,3
```

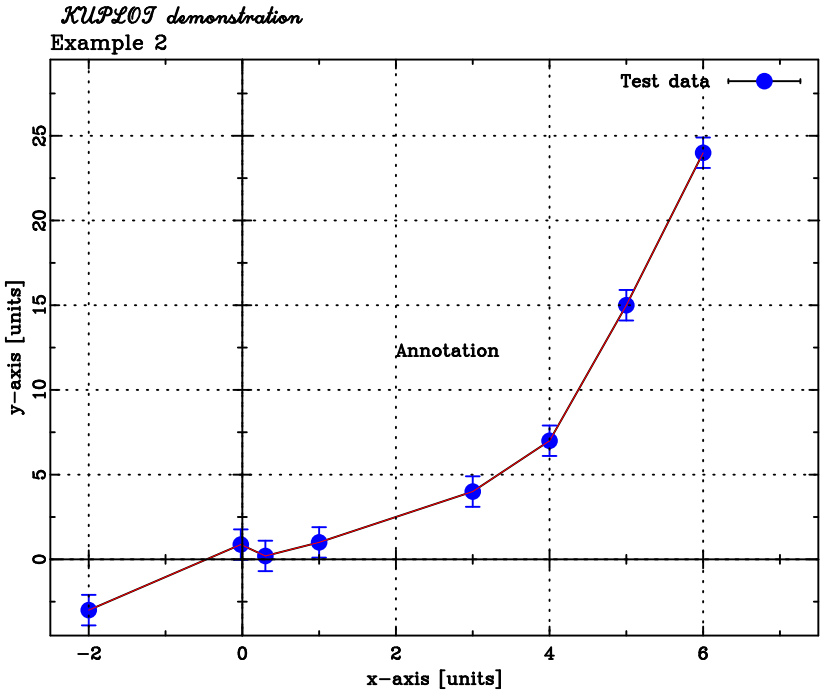


Figure 2.2: Customized *KUPLOT* plot

```
17  mcol 1,3
18  msiz 1,0.4
19  etyp 1,2
20  #
21  sleg 1,"Test data"
22  sann 1,"Annotation",2,12
23  plot
```

In line 1 of this macro we read the data set from file '*test.xy*' as in the previous example. The axes labels and the text for the first and second title line are set in lines 3–6. The grid of dashed lines at positions of the major tick marks is enabled in line 8 and the plotting of the filenames corresponding to the data sets in the upper left corner of the plot is switched off (line 9). Next we define the extend of the plot (line 11) to be from -2.5 to 7.5 in the x-direction and from -4.5 to 29.5 in the y-direction. The tick mark interval is set to 2.0 and 5.0 (line 12). All these settings affected the complete plot, whereas the following commands act on data set 1, which is given as the first parameter to all commands in lines 14–19. First the line color is set to black (line 14). The first six colors are coded as the 6 default pen colors on a HP plotter, i.e. red, green, blue, purple, yellow and black. Those default colours can be redefined using the command 'color'. In the following lines, the line width, marker type, marker color and marker size are defined (lines 16–18). The different marker types supported by *KUPLOT* are shown in Figure 2.3. In addition to the markers shown, *KUPLOT* can access all markers defined in the *PGPLOT* library. The marker number is simply 100 plus

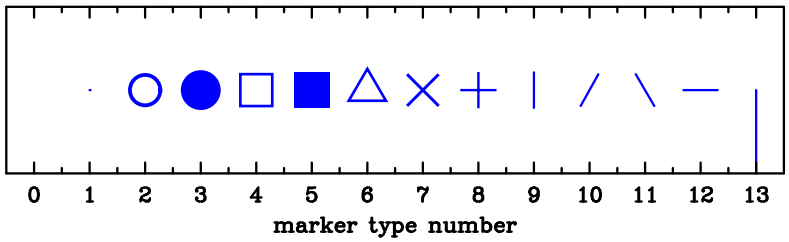


Figure 2.3: *KUPLOT* marker types

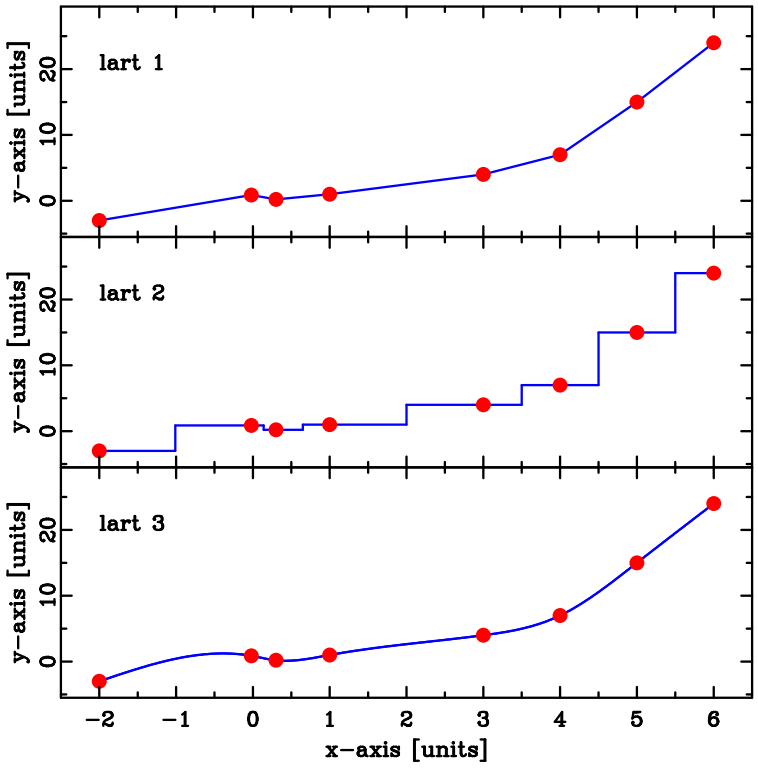


Figure 2.4: *KUPLOT* plotting styles

the *PGPLOT* marker code. Refer to the *PGPLOT* documentation for a list of those markers. Error bars in y-direction based on the σ_y values read from the input file '*test.xy*' are enabled in line 19. The caption "Test data" is defined in line 21 and the annotation "Annotation" at the coordinates (2.0,12.0) is specified by the 'sann' command in line 22. Note, the value 1 in line 30 stands for the first annotation and not for data set one. Finally the plot is displayed on the screen (line 23). In this example, the data points (x_i, y_i) were simply connected by a straight line. *KUPLOT* offers two alternative plotting styles: a histogram style and cubic spline interpolation. The style is selected by the 'lart' command followed by the data set number and the desired style. Style '1' is the default and connects the data points by a straight line as in the previous

Command	Function
\u	start superscript, or end subscript
\d	start sub-, or end superscript
\b	backspace (draw next character on top of current one)
\fn	switch to normal font (1)
\fr	switch to roman font (2)
\fi	switch to italic font (3)
\fs	switch to script font (4)
\\	backslash
\x	multiplication sign $\times$
\.	centered dot $\cdot$
\A	Å

Table 2.2: Text control characters

example ('lart 1,1'). This style is shown in Figure 2.4 in the bottom view graph. The histogram style '2' is shown in the middle view graph of the same Figure. Here each data point (x_i, y_i) sits in the center of a histogram step. The command to change the plotting style of data set 1 to histogram would be 'lart 1,2'. The third mode ('lart 1,3' for data set 1) is cubic spline interpolation. The interpolated spline goes through all data points (x_i, y_i) and has a continuous first derivative. The spline interpolation of the data set '*test.xy*' can be seen as top view graph in Figure 2.4. The default number of interpolated points for a plot is 500. However, the value, determined by the variable *MAXSP* might be altered in the file '*kuplot.inc*' before *KUPLOT* is compiled.

2.3 Fonts and special characters

The program *KUPLOT* offers the user various ways to alter the appearance of the text parts of the plot. All these functions are accessed by the command 'font'. If no parameters are given, the current settings will be displayed on the screen, as shown below:

```
Font settings for frame no.:   1
   Overall font scaling factor  :   2.50

   Font  where ?             size   fontname           font-id   color
   ------------------------------------------------------------------------
     1   Main title line       16.  Roman                   2       6
     2   Subtitle line         14.  Roman                   2       6
     3   Axis labels           12.  Roman                   2       6
     4   Numbers at axis       12.  Roman                   2       6
     5   Text in text frame    12.  Standard                1       6
     6   Filename & caption    12.  Roman                   2       6
```

As can be seen from the output, there are six different types of fonts, the main and second title line, the

alpha	\ga	α	\gA	A	beta	\gb	β	\gB	B
gamma	\gg	γ	\gG	Γ	delta	\gd	δ	\gD	Δ
epsilon	\ge	ϵ	\gE	E	zeta	\gz	ζ	\gZ	Z
theta	\gh	θ	\gH	Θ	iota	\gi	ι	\gI	I
kappa	\gk	κ	\gK	K	lambda	\gl	λ	\gL	Λ
mu	\gm	μ	\gM	M	nu	\gn	ν	\gN	N
xi	\gc	ξ	\gC	Ξ	omicron	\go	o	\gO	O
pi	\gp	π	\gP	Π	rho	\gr	ρ	\gR	P
sigma	\gs	σ	\gS	Σ	tau	\gt	τ	\gT	T
upsilon	\gu	υ	\gU	Υ	phi	\gf	ϕ	\gF	Φ
chi	\gx	χ	\gX	X	psi	\gq	ψ	\gQ	Ψ
omega	\gw	ω	\gO	Ω					

Table 2.3: Access to Greek characters

Figure 2.5: *KUPLOT* fonts

axis labels, the axis numbering, text in a text frame (see section 4) and finally the font used for annotations and captions. Each of those types is associated with a font size given in points, a font style given as name (e.g. Roman) and number (e.g. 2) and finally a color in our example pen number 6 or black. The four font styles available in *KUPLOT* are shown in Figure 2.5. *KUPLOT* allows the user to apply an overall scale factor to the font size or change font sizes individually. As an example let us change the color of the first title line to red. This would be done using the command:

```
font col,1,1
```

The first parameter tells *KUPLOT* what we want to change, here the color. Next we have the number of the font type, here 1 for the first title line. Finally we specify the desired color, here pen 1 for red. One benefit of using the *PGPLOT* library is that the user can access special characters and commands within all text lines, e.g. titles, axis labels and text in text frames. The supported control commands are listed in Table 2.2 and a list of Greek characters is given in Table 2.3. For example to create an axis label $Å^{-2}$ the text input would read *\A\u-2\d*. Note that sub- and superscript always have to be given in pairs. These examples can only give a brief introduction in the different commands of *KUPLOT* allowing the user to alter the appearance of the plot. For details on these commands and their parameters refer to the online help, the command reference or the interactive tutorial.

2.4 Drawing bonds

The program *KUPLOT* allows the user to connect points in the plot that are separated by a given distance. The distance is specified relative to the current x-axis. To calculate the distances, the aspect ratio and the angle between the axes is used. These values can be defined by the user via the commands 'aver' and 'angl' as we will discuss in somewhat more detail in section 3.2. Consequently you will get the desired connection only for the correct aspect ratio and angle between the axes. This feature can be used to include bonds when plotting a structure file exported by *DISCUS*. An example for the compound PHTP (Perhydrotriphenylen) is shown in Figure 2.6. The macro file used to create this picture is listed below:

```
 1  rese
 2  load cr,phtp.cr
 3  #
 4  skal 0.0,28.8,0.0,28.8
 5  mark 14.4,14.4
 6  #
 7  aver 1.0
 8  angl 120.0
 9  #
10  msiz 1,0.35
11  grid on
12  fnam off
13  #
14  achx a-axis (\A)
15  achy b-axis (\A)
16  #
17  bond 1,1.4,0.2,1,1,1.5
18  #
19  plot
```

First we reset the program and read the data (line 2). In line 4 the plot area is defined followed by the setting for the tick mark interval (line 5). In our case the lattice parameter is $a = b = 14.4$Å. The command 'aver' defines the ratio between units on the y- and x-axis. Since both are in Å, the desired ratio is one. The angle between the axes is set to 120° (line 8). Next we set the marker size (line 10), turn the plotting of the grid on (line 11) and disable the plotting of the filenames (line 12). The axes labels are set in lines 14 and 15. So far nothing new, now comes the command 'bond' that defines the bonds between carbon atoms we like to plot. The first parameter '1' is the number of the bond definition. *KUPLOT* allows one to define multiple distances. The second and third parameter set the distance to 1.4Å $\pm 20\%$. The last optional parameters in line 17 set the bond color to red (pen 1), select a solid line (type 1) and set the line width to 1.5. All that remains is to plot the result (line 19). One should be aware that *KUPLOT* calculates the distances between *all* data points and compares the to the selected distance interval which can be rather slow for large data file. To disable a bond definition simply reenter the command with distance zero, e.g. 'bond 1,0.0'.

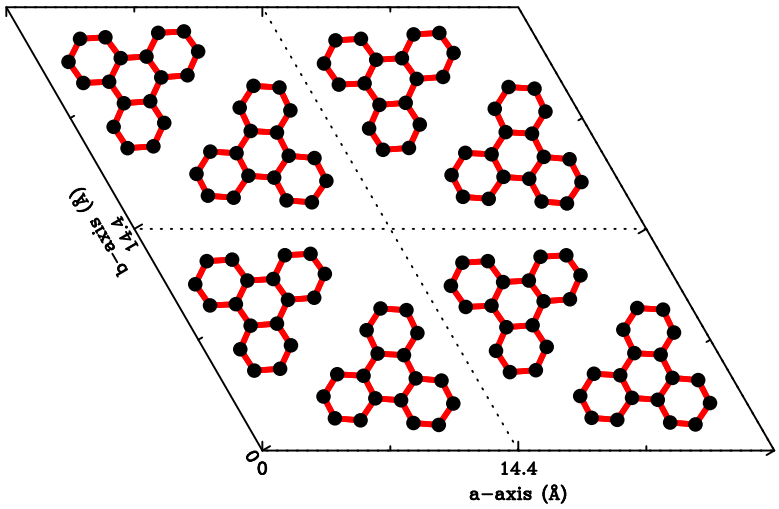


Figure 2.6: Example for plotting bonds

2.5 Saving data sets

The program *KUPLOT* allows the user to save loaded data sets to a file. This might be necessary after a data set was altered or in cases where only part of the data shall be saved. To save a data set, use the command 'ksav' followed by the number of the data set to be saved. After entering the 'ksav' sublevel, you need to specify the output filename and the file format for the output file. In this section we will only discuss the saving of 1D data sets and the only available file format is 'xy' (see previous example). The following commands will save the loaded data set 1 (e.g. '*test.xy*' from the example above):

```
1  ksav 1
2  outfile export.xy
3  form xy,0.0,5.0
4  show
5  run
```

In line 1 we enter the 'ksav' sublevel. The parameter specifies the data set we want to save, here data set 1. Next the output filename is set to '*export.xy*' (line 2). The file format is set to 'xy' and the optional further parameters give x-limits (line 3), i.e. only data points with x-values between 0.0 and 5.0 will be saved. Note, that the default range are defined by the current plot window limits. Thus if the plot window set by the command 'skal' is smaller than the extend of the actual data set, only point that are visible in the current plot are saved if no limits are given with the 'form' command. The command 'show' (line 4) lists the current limits and finally the data are written to the specified file via the 'run' command (line 5). The command 'run' also exits the 'ksav' sublevel. To exit without actually writing a file use the 'exit' command.

The 'ksav' command has many additional options to save 2D data such as format conversion and the export of cross sections. These functions are discussed in chapter 3.4.

2.6 Printing and exporting the plot

Viewing the plot on the screen is one thing, but we certainly need to print plot at some stage or save is for e.g. import in a text processing program. *KUPLOT* supports two different output formats that can be saved or used for printing. The defaults are POSTSCRIPT and GIF associated with the parameters 'ps' and 'pic' respectively. However, they can be set to any device supported by *PGPLOT* by altering the variables *dev_name* in the file '*blk_appl.f*' of the *KUPLOT* distribution. The variable *dev_prn* in the same file sets the default print command. To print the current plot to the default printer, use 'prin ps'. To reach other printers, the corresponding print command can be specified as a second parameter to the 'prin' command as in the example below:

```
prin ps,"lp -d myprinter -h "
```

Here the printer 'myprinter' is used. Check local documentation or ask your system administrator how to access the desired printer. In order to save the graphics file rather than printing it, use the 'save' command with the first parameter being 'ps' or 'pic' for POSTSCRIPT or GIF output respectively. The second parameter is the name of the output file, e.g. 'save ps,plot.ps'.

Chapter 3

Plotting 2D data

In the previous chapter we have learned about using *KUPLOT* to plot 1D data sets. Now we will extend the usage of *KUPLOT* to 2D data sets, i.e. xyz-data. The coordinates x and y are within the drawing plane and the z values are represented by contour lines at given z_c values, by the bitmap color or both. Changing the general appearance of the plot such as defining title lines, labels or tick mark intervals was described in section 2.2 of this manual and works exactly the same way when plotting 2D data sets.

3.1 File formats

As described in section 2.1 for 1D data sets, 2D data sets are read using the command 'load'. Multiple data sets might be read by repeating the command 'load' and 1D and 2D data sets might be imported in any combination. The command 'rese' clears the currently loaded data sets and the next file is read as set number one again.

A summary of the supported 2D file formats is given in Table 3.1. All file formats are "normal" text files that can be viewed and edited with every text editor. Since these files are normally larger in size compared to a binary version, archived files might be compressed with standard UNIX tools like 'compress' or 'gzip'. The standard file format for 2D data is the so-called NIPL format which was developed at the Insitut für Mineralogie und Kristallographie in München. The NIPL format ('ni') is defined as follows:

```
1      nx ny
2      xmin xmax ymin ymax
3ff    z z z z ...
```

The first line contains the number of data points in x- and in y-direction. The next line gives the x- and y-limits of the plot area. All following lines contain the z-values row by row starting in the lower left corner. The z-values are real numbers and not restricted in any way. However, the value -9999.0 is reserved for excluded regions (see below).

The optional third parameter listed in Table 3.1 is the name of a file containing excluded regions, i.e. areas of the data file that should be excluded from the plot. An example is shown in Figure 3.1. The plot

Format	add. Parameters	Description
de	$\Delta x, \Delta y$	Reads xy-file and creates a 2D data set with the number of points that are inside a specified grid box $\Delta x, \Delta y$ as z-values.
ni	[exclude]	Reads NIPL file format (see text). The optional parameter is a file containing excluded regions.
pg		Reads PGM file format (see text).
zz	$\Delta x, \Delta y$	Reads xyz-file and bins the data to the given grid size $\Delta x, \Delta y$

Table 3.1: Supported file formats for 2D data

on the bottom was created by reading the NIPL file '*test.nipl*' using the command 'load ni, test.nipl' and thus reading all data. The data shown here are actually neutron diffuse scattering from calcium stabilized zirconia collected by R.B. Neder at the neutron source in Garching, Germany. Sometimes one wants to exclude certain regions within the data from the actual plot, in our example the strong Bragg peaks. This can be done by creating a file with coordinates of rectangles within the plotting area that should be excluded. Such a file can be created using a text editor. In our example, the excluded region file '*test.excl*' looks like this:

```
10
0.87  1.13  0.81  1.17
2.85  3.15  0.81  1.26
1.85  2.34  3.70  4.28
2.82  3.18  2.79  3.28
0.87  1.18  2.81  3.23
0.00  0.24  3.68  4.28
0.00  0.12  1.73  2.24
0.00  0.58  0.00  0.79
1.78  2.22  0.00  0.35
1.90  2.17  1.81  2.13
```

The first line contains the number of rectangle coordinates in the file followed by 'xmin,xmax,ymin,ymax' for each excluded region. Rereading the file '*test.nipl*' using the command 'load ni, test.nipl, test.excl' results in the right view graph of Figure 3.1. Note that *KUPLOT* will replace all z-values within an excluded region by -9999.0 and those values will be ignored by most *KUPLOT* functions. If a file read with excluded regions is saved later on, the data within those excluded regions are lost.

The second more general 2D file format is the PGM (portable graymap) format which is defined as part of the PNM format Jef Poskanzer. Various programs are capable of reading PNM files and the '*pnmplus-package*' is a freely available collection of tools and conversion programs from PNM to virtually any other graphics format. *KUPLOT* only supports ASCII PGM files which are defined as follows:

```
1       P2
2       nx ny
3       255
```

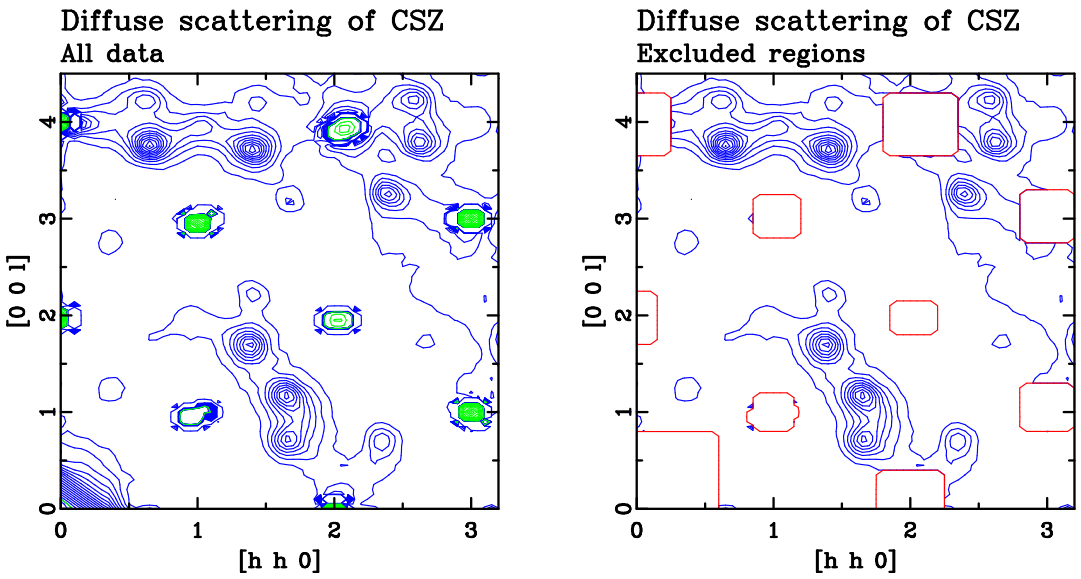


Figure 3.1: Usage of excluded regions for 2D files

```
4ff    z z z z z
```

The code 'P2' in the first line identifies the file as ASCII PGM format. The dimensions nx and ny of the data set are given in the next line. The PGM format is an integer bitmap and the number in line 3 gives the maximum depth (here 255 or 8 bit). Although the PGM format is generally not restricted to 8 bit resolution, it was found that many programs are not capable of reading PNM files with a larger resolution. Comments can be included in the file starting with a '#' as first character. The actual data are integer values written row by row starting (in contrast to NIPL files) from the top left corner. Besides being limited to integer values ranging from $0 \rightarrow 255$, the PGM file format does not contain any information about the x and y limits and *KUPLOT* sets x- and y-values to the pixel number. However, the x- and y-values might be manipulated (see chapter 7) and the file can be saved in NIPL format (see section 3.4), thus allowing the conversion of PGM files to NIPL files and *vice versa*.

The PGM and NIPL files are restricted to data points being on a equidistant rectangular grid. The file format 'zz' can be used to read any data file containing the data points (x_i, y_i, z_i) on a separate line. This is practically the extension of the 'xy' file format. However, *KUPLOT* can internally only work with z-values lying on a rectangular equidistant grid. Thus the read data points are binned to a grid size defined by Δx and Δy given as additional parameters to the 'load' command. Each resulting grid point contains the average of all data points within the file that fall within the area of the specific grid point. If the specified grid size is too small, the resulting plot will contain empty grid points. The following command reads the XYZ file '*test.xyz*' and bins it to a grid size of $\Delta x = \Delta y = 0.05$:

```
load zz,test.xyz,0.05,0.05
```

This file format can also be used to rebin data sets to a broader grid by saving the NIPL (or PGM) file as XYZ file and rereading the file with the new desired grid size.

The file format 'de' (for density) works similar to the previously discussed 'zz' format but rather than averaging the z-values in each grid point, the resulting z-value is the number of points (x_i, y_i) falling within the corresponding grid volume which is defined as before by the additional parameters Δx and Δy. The program reads the first two real numbers in each line as x- and y-value. For example *DISCUS* can export the atom positions of all atoms without the translational part, i.e. all atoms are in one unit cell. The resulting file can be read using the 'cr' format resulting in a marker for each atom. Alternatively the same file (since the first two numbers are x and y) can be read using the 'de' format which will result in a density plot which can be displayed either using contour lines or a bitmap.

3.2 Customizing the plot

The command 'hart' selects the usage of contour lines, bitmaps or both and is discussed in the next section. Here we want to concentrate on commands used to change contour lines. The base level, the interval and the number of contour line levels are defined using the command 'hlin':

```
hlin 1,100,50,10
hlin 2,10,10,9,%
```

KUPLOT supports multiple sets of contour lines, e.g. one set at low levels to display diffuse scattering and one set at higher levels with a different spacing for the stronger Bragg peaks. Each set can be plotted in a different color. The two example commands above illustrate the 'hlin' command. The first command sets the values for contour line set 1. The contours start at a value of 100 and increase in steps of 50. A total of 10 contour levels is drawn, e.g. the highest level corresponds to 600. The second 'hlin' commands sets values for the second contour line set, but the additional parameter '%' indicates, that the numbers are taken as percentage of $\Delta z = zmax - zmin$ of all loaded data sets rather than absolute z-values. In our example the base level is 10% of Δz, the contour lines are stepped in 10% intervals and 9 levels are plotted. Thus the highest level corresponds to 100%. The command 'hpak' determines how many contour line sets are actually plotted. The default is the usage of all sets defined by the 'hlin' command. Color and line type for the individual contour line sets can be changed using the commands 'hcol' and 'htyp'.

Sometimes a specific aspect ratio of the x- and y-axis or a specific angle between the two axis is required to obtain a non distorted picture of the data. *KUPLOT* allows the user to specify an aspect ration using the 'aver' command. As default *KUPLOT* determines the aspect ratio is such a way, that the resulting plot is as large as possible. This default can be restored by entering the command 'aver' without further parameters. Alternatively, the desired ratio can be given as parameter to the 'aver' command. In Figure 3.2 we show an example of a contour plot illustrating some of the features discussed above. The commands used to create the contour lines shown are listed below:

```
1  aver 0.707
```

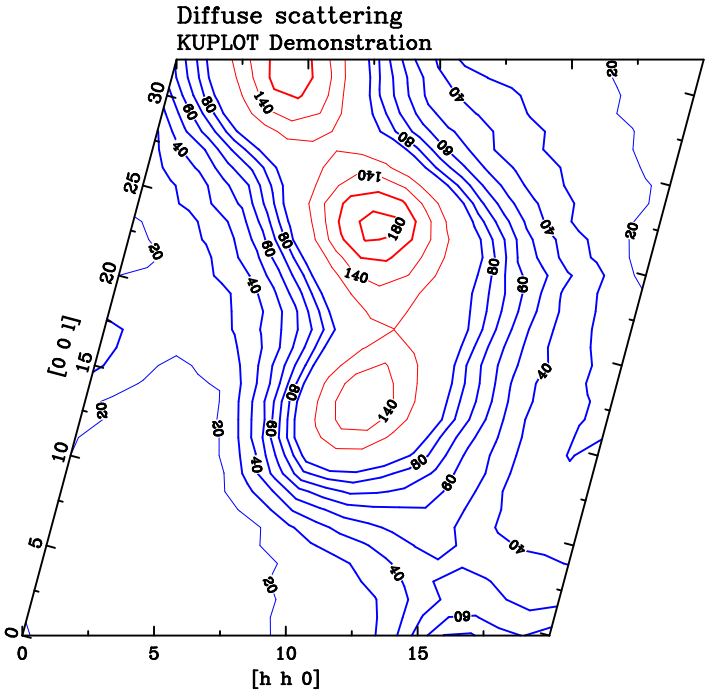


Figure 3.2: Customizing contour plots

```
2  angl 75.0
3  #
4  hlin 1, 10,10, 9
5  hlin 2,120,20,10
6  hcol 1,1,3
7  hcol 1,2,1
8  hlab 1,2
```

In line 1 we set the aspect ratio of the x- and y-axis to $1/\sqrt{2} \approx 0.707$ and the angle between the axes to a value of 75° (line 2). In lines 4–5 we define two contour line packages, the first one giving contour lines at values of $z_c = 10, 20, ..., 100$ the second one at values $z_c = 120, 140, ..., 320$. In line 6 the color for the first contour package for data set one is set to pen 3 (blue). Next the color for the second package for the same data set is set to pen 1 (red). Finally the labelling of contour lines for data set one is enabled. The second parameter in line 8 specifies that every second contour line starting from the base line is labelled. As always, check the online help for more detailed information on the commands used.

3.3 Using bitmaps

The command controlling the appearance of a 2D plot is 'hart'. Its first parameter determines the data set, the second parameter the type of plot. To use only contour lines set the second parameter to '1', for bitmap

display use ’2’ and to have both a bitmap and contour lines set the second parameter of the ’hart’ command to ’3’. In Figure 3.3 we can see the same plot using just a bitmap on the bottom and using additional contour lines on the top. The wedge showing the z range of the bitmap colors is activated by setting a label for the z-axis using the command ’achz’.

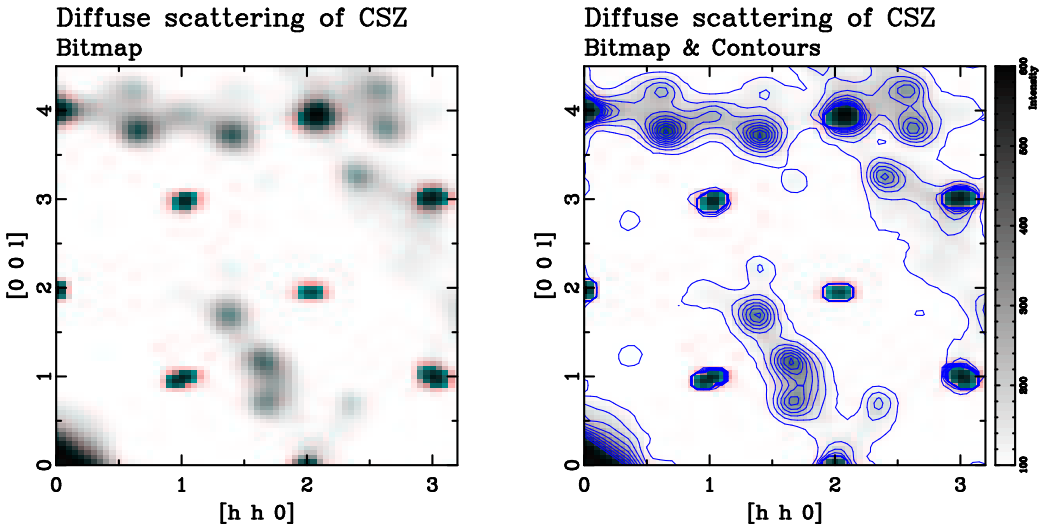


Figure 3.3: Example for bitmap plots

The bitmaps used by *KUPLOT* have 240 color entries, the remaining 16 colors are reserved for other functions. The z-range to be converted to those 240 colors is determined by the minimal and maximal contour level for the first set defined by the command ’hlin’. Subsequently the bitmap color range and the contour levels are the same for the first set of contour levels. In some cases one might want to create a plot where the bitmap and the contour lines have different ranges. This can be done like in the following example:

```
hlin 1,0,50,1,%
hlin 2,0,5,20,%
htyp 1,0
htyp 2,1
```

The levels set by the first ’hlin’ command determine the z-range used to create the bitmap, i.e. 0% to 50% of the z-range. We do this in a single step since only the maxima are used to determine the bitmap. The second ’hlin’ command sets the contour lines we actually want to plot on top of the bitmap. In order to suppress the first set of contour lines we set the line type for set 1 to ’0’, i.e. no line using the first ’htyp’ command. The line type for the second contour set is set to ’1’, i.e. solid line, as done in the last line of the example above.

KUPLOT has three default color maps used to display bitmaps. The map is selected by the command ’cmap’. For a description of the different color maps refer to the online help. The selected map for the

examples displayed in Figure 3.3 is 'gray'. The command 'cmap' is also used to save or read a color map from a file. Additionally the current color map can be altered using the variable *cmap*. For details about the usage of variables see section 6.1 of this users guide.

3.4 Saving data

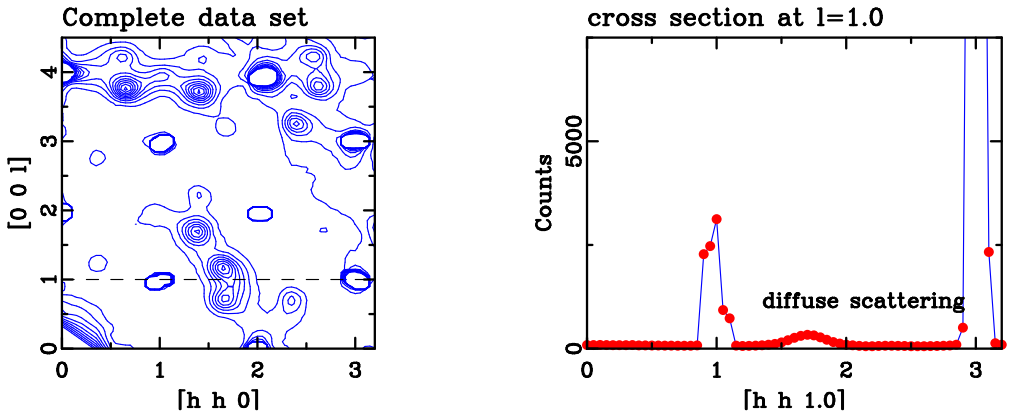


Figure 3.4: Extracting data from 2D data sets

In contrast to 1D data where we could mainly save a given data set in the 'xy' format, there are several options to extract and save data from a 2D data set. As described in section 2.5, the first step is to enter the save sublevel using the command 'ksav' followed by the number of the data set to be saved. As before the output filename is set by the command 'outfile' and the saving process is started via 'run'. However, there are now many more possible settings for the 'form' command.

Format	Parameters	Description
ni	[x1,x2,y1,y2]	Nipl file, given area
pg	[x1,x2,y1,y2]	PGM file (ASCII), given area
gn	[x1,x2,y1,y2]	XYZ file (gnuplot), given area
sx	y-value	Cross section ‖ x at y-value
sy	x-value	Cross section ‖ y at x-value
mx	i	Cross section ‖ x through maximum #i
my	i	Cross section ‖ y through maximum #i
sk	ik	Cross section along xy of data set ik
sl	x1,y1,x2,y2,n	Cross section from x1,y1 to x2,y2 with n points

Table 3.2: Save options for 2D data sets

The first three of those formats listed in Table 3.2 save the data set or a subsection as 2D data set either

in NIPL, PGM or XYZ format. As before the area that is saved is determined by the current size of the plot window determined by the command 'skal'. Alternatively the x-limits $x1$ and $x2$ and y-limits $y1$ and $y2$ can be specified as additional parameters to the 'form' command. Note that when using PGM as output format, the z-values are converted to integer and should range from $0 \rightarrow 255$. The command 'thresh' allows to control how the z-values are converted to this range. The other formats in Table 3.2 are used to extract a cross section from the 2D data set. The created output file is a normal 'xy' file. The parameters 'sx' and 'sy' extract a cross section parallel to the x- or y-axis at the given y- or x-value (see example below). Rather than specifying these y- or x-values, the corresponding coordinates of maximum i determined by the command 'smax' (see section 7) can be used ('mx' or 'my'). The command above will extract the z-values at the corresponding grid points. The last two options can extract data from any value of x and y by interpolation. The format 'sk' will use the coordinates x and y from data set (1D) number 'ik' to determine the z-values to be extracted whereas 'sl' will extract 'n' points along a straight line defined by the points $(x1, y1)$ and $(x2, y2)$.

A simple example how to extract a cross section is show in Figure 3.4. The original data set is shown on the left. The cross section parallel x (or [hh0]) at y (or l) equals 1.0 is marked by a dashed line. The resulting 1D plot of the cross section is shown on the right panel of the figure. The corresponding commands to create the data file shown at the top is listed below:

```
1  ksav 1
2  outfile test.cut
3  form sx,1.0
4  run
```

In line 1 we enter the save sublevel. In our example the data set to be used is data set number one. The output filename is set to '*test.cut*' (line 2) and the format is set to 'sx', i.e. cross section parallel to x at $y = 1.0$ (line 3). Finally the data file is written after the command 'run' is entered (line 4).

Chapter 4

Using frames

In this chapter we will introduce the usage of frames that enable *KUPLOT* to display more than one view graph in a single plot. A detailed description of all frame related commands is available as part of the online help which can be accessed by entering 'help frames' at the input prompt.

4.1 Introduction

Frames enable *KUPLOT* to display multiple view graphs within a single plot. The layout is defined by the user. A simple example with two graphs being plotted on a single page is shown in Figure 4.1. Let us have a look at the macro file that was used to create the figure and learn step by step how to use this feature of *KUPLOT*. Note that the line numbers shown in the listing below are used for easy reference in this manual and are not part of the actual macro file.

```
 1  load xy,s287.xy
 2  load xy,s288.xy
 3  #
 4  tit1 Experiment A
 5  achx \gw (\uo\d)
 6  achy Counts
 7  mark 0.1,100
 8  buff 0.08
 9  fnam off
10  fram on
11  #
12  nfra 2
13  #
14  afra 1
15  kfra 1,1
16  tit2 Scan 287
17  #
18  afra 2
```

```
19  kfra 2,2
20  tit2 Scan 288
21  #
22  plot
```

The macro starts with the reading of two data sets (lines 1–2). Next title, axes labels and marker intervals are set (lines 4–7) as in previous examples. The command 'buff' (line 8) alters the space around the view graph reserved for axis numbering and titles. The value is the fraction of the total width or height of the plot. In our case we reserve 8% of the page on all four sides of the plot as buffer space. In line 9 the plotting of the filename is disabled and in the following line the plotting of a border around each frame is enabled. So far we have used no frame related commands and a plot at this stage would show both data sets in a single view graph. In line 12 we then specify that we want to use two frames, i.e. have two view graphs on our plot. At this stage, all settings from the current plot (fram 1 in case we have used frames before) are copied as defaults to all other frames. Entering command 'plot' now would result in two equivalent view graphs side by side on the plot. Thus all setting that are common to all frames should be made *before* the command 'nfra' is used. Now we need to customize the two frames. The command 'afra' determines for which frame the following commands are used. First we alter settings for frame 1 by entering 'afra 1' (line 14). Next we specify that this frame should only contain the data from data set one (line 15) and we enter an individual second title line (line 16). The same procedure is repeated for frame 2 (lines 18–22) which should include

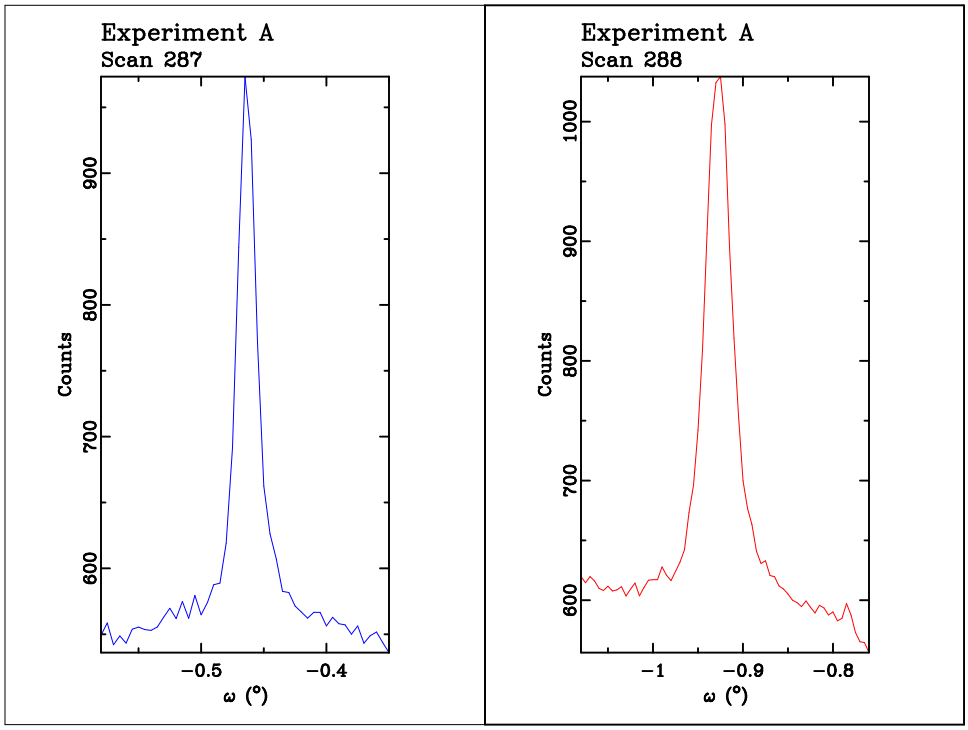


Figure 4.1: Simple example using frames

data set 2 and a different subtitle line. The command 'plot' (line 22) will result in a picture similar to the one in Figure 4.1.

Command	Description
afra	Sets the active frame for user input
bfra	Sets background color for specified frame
cfra	Copies frame parameters
fram	Defines if a border is plotted around each frame
kfra	Defines contents of frames (data sets or text)
nfra	Sets number of frames (default = 1)
sfra	Defines position and size of a frame

Table 4.1: *KUPLOT* commands related to frames

The usage of frames is rather simple and most *KUPLOT* settings are individual settings for the different frames. All frame related commands are summarized in table 4.1. The following two sections contain more complex examples using frames. Additionally the interactive tutorial (see section 1.4) gives a further inside into the usage of frames.

4.2 Example 1: Using two different y-axes

In this example, we will use frames to produce a plot that uses two different y-axes to display two different data sets. One y-axis labelling will be on the left hand side of the view graph, the other one on the right hand side. The resulting plot is shown in figure 4.2. Here we display the scaling factor f (right axis) and the background parameter b (left axis) of a Reverse-Monte-Carlo refinement as function of the cycle number as an example plot.

We will discuss the corresponding macro file listed below step by step. Again the line numbers are not actually part of the macro. Lines 1–6 are nothing special, we read the two data files and set common features like title lines. Next we set the number of frames to two (line 8). Since we want to plot the two data sets on top of each other using just a different y-axes, we have to overwrite the default location of the frames using the 'sfra' command (lines 10–11). The coordinates of the plot area range from 0.0 to 1.0 in x- and y-direction. In our example both frames cover the complete plot area.

```
1  load xy,back.xy
2  load xy,scal.xy
3  #
4  fnam off
5  tit1 RMC test simulations
6  tit2 Frame demonstration
7  #
8  nfra 2
9  #
```

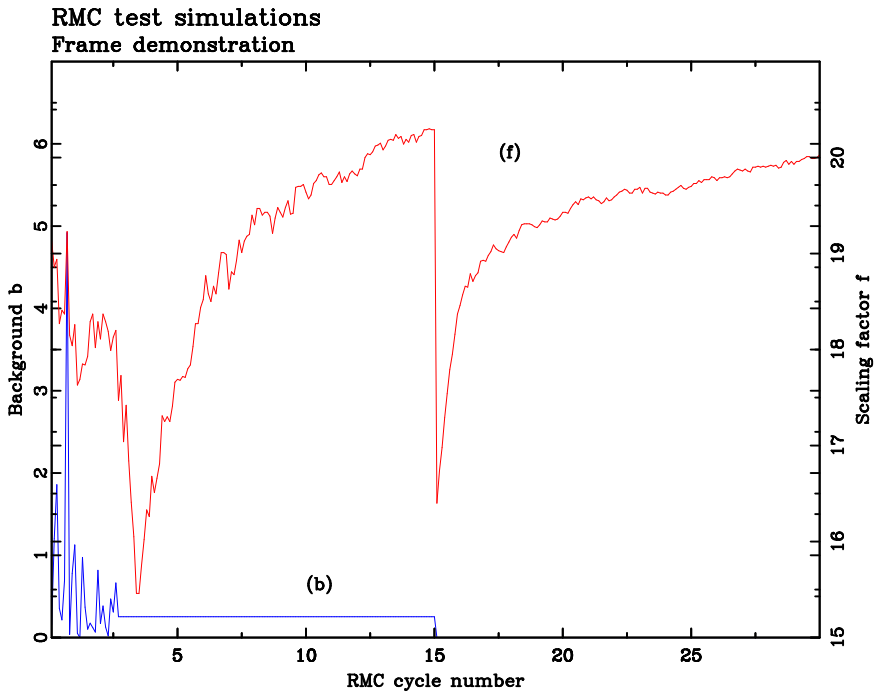


Figure 4.2: Using different y-axis with frames

```
10  sfra 1,0.0,0.0,1.0,1.0
11  sfra 2,0.0,0.0,1.0,1.0
12  #
```

The next step is to input the settings for both frames. First we set the input focus to frame 1 (line 13) and specify data set 1 to be used in that frame (line 14). The 'fset' command in line 15 determines the layout of the view graph and setting number '3' is the default plot layout, i.e. box, labels and tick marks in x- and y-direction and lines at $x = 0, y = 0$. Finally we set the size of the plotting window (line 16), the tick mark intervals (line 17) and the labels for the axes (lines 18–19).

```
13  afra 1
14  kfra 1,1
15  fset 3
16  skal 0.1,30.0,0.0,7.0
17  mark 5.0,1.0
18  achx RMC cycle number
19  achy Background b
20  #
```

The same as before has to be done for the second frame. Lines 21–22 set the focus to the second frame and select data set 2 to be displayed. The negative value of the parameter of the 'fset' command (line 23) indicates that the y-axis label and numbers are to be plotted on the right hand side rather than on the left

hand side which is the default. Again the extend of the plotting window and the tick marks are set (lines 24–25). Note that we need to specify the same plotting range ($0.1 \rightarrow 30.0$) in x-direction and a different one in y-direction compared to the settings of frame 1. The x-axis label is switched off (line 26) because we have already the label from the other frame. The y-axis needs a new label (line 27). Finally we define two annotations (line 28–29). Note that the given coordinates are with respect to the scaling of frame 2.

```
21  afra 2
22  kfra 2,2
23  fset -3
24  skal 0.1,30.0,15.0,21.0
25  mark 5.0,1.0
26  achx
27  achy Scaling factor f
28  sann 1,"(b)",10.0,15.5
29  sann 2,"(f)",17.5,20.0
30  #
31  plot
```

Obviously the same plot could have been achieved by scaling one of the data sets to the y-range of the other, but the information about the original range of y-values of one of the data sets would have been lost in the plot in contrast to the example using frames given here.

4.3 Example 2: Connected view graphs

In this section we will learn how we can create plots where several viewgraphes share one or more sides. This is done using the command 'buff' we have used in the first part of this chapter. An example plot is shown in Figure 4.3.

The macro used to create the plot is displayed below. As usual we start resetting *KUPLOT* (line 1) and we select portrait orientation for this plot (line 2). Next the data sets are loaded (lines 4–7). This is followed by general settings (lines 8–12) since all our frames will have the save y-axis label, plot window and tick mark interval.

```
 1  rese
 2  orient port
 3  #
 4  load xy,tot.calc
 5  load xy,dif_ga.calc
 6  load xy,par_ga.calc
 7  #
 8  fnam off
 9  fset 2
10  achy G(r) [\A\u-2\d]
11  skal xmin[1],xmax[1],1.2*ymin[1],1.2*ymax[1]
12  mark 2.0,10.0
```

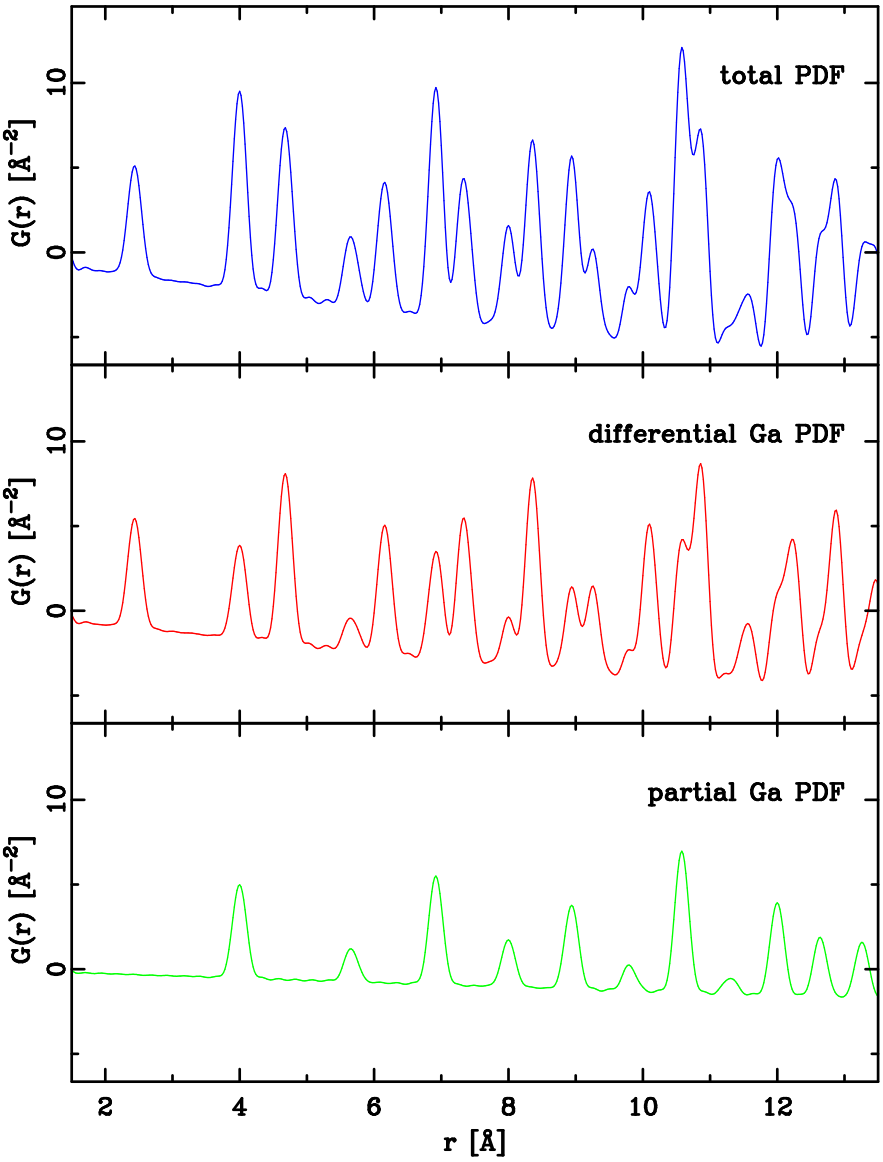


Figure 4.3: Example for connected view graphs

Next we define the frames, three on top of each other with no gap in between them. Since the middle frame will have no buffer space on the bottom and top we need to make it smaller in order for the plot area to have the same size. Here we use variables for the task. The value of the buffer size for title and axes labels is stored in 'r[1]' (line 14), here 0.1. The remaining plot space for each panel is calculated and stored in 'r[2]' (line 15). It is simply the full size (1.0) minus two times the buffer size for the title of the top panel and the axis label and numbering of the bottom panel. Next we select three frames (line 17) and set the

frame corners using the variables we just defined. The x-range for all panels is 0.0 to 1.0. The y ranges are determined by simply adding the heights of the panels and the buffering space together (lines 18–20). Simply calculate the numbers and it becomes clear.

```
13  #
14  r[1] = 0.1
15  r[2] = (1.0-2.0*r[1])/3.0
16  #
17  nfra 3
18  sfra 1,0.0,2.0*r[2]+r[1],1.0,3.0*r[2]+2.0*r[1]
19  sfra 2,0.0,    r[2]+r[1],1.0,2.0*r[2]+    r[1]
20  sfra 3,0.0,0.0          ,1.0,    r[2]+    r[1]
```

In the next section the setting for the individual frames are entered. Most of the commands were already discussed in the previous chapters. The command 'buff' can either be used with a single parameter like in the example in section 4.1 in which case the buffer space around the view graph is the same in all directions. Alternatively one can supply four parameters for the required free space on the left and right side and the bottom and top of the view graph, respectively. In our example this first frame located on the top will have no buffer space at the bottom (line 23). Apparently we have to turn the numbering for the x-axis off (line 25). Now we repeat the settings for the other two frames. For the middle frame we have no buffer space on the top an bottom (line 29) and for the bottom frame we have only buffer space at the bottom (line 34). Also we need to set a x-axis label for the bottom panel (line 38).

```
21  #
22  afra 1
23  buff 0.1,0.1,0.0,r[1]
24  kfra 1,1
25  achx OFF
26  sann 1,"total PDF",13.0,10.0,right
27  #
28  afra 2
29  buff 0.1,0.1,0.0,0.0
30  kfra 2,2
31  achx OFF
32  sann 1,"differential Ga PDF",13.0,10.0,right
33  #
34  afra 3
35  buff 0.1,0.1,r[1],0.0
36  kfra 3,3
37  sann 1,"partial Ga PDF",13.0,10.0,right
38  achx r [\A]
```

Frames are a flexible tool and can be placed anywhere on the plot area and may also overlap. A even slightly more complex example using frames is discussed in the next section.

4.4 Example 3: Advanced frame usage

The final example for the usage of frames is more complex and the resulting plot is shown in Figure 4.4. *KUPLOT* is capable of creating quite complex plots and e.g. allows one to create complete transparencies for a talk.

The macro used to create the plot is displayed below. As usual the data sets are loaded first (lines 1–2) followed by general settings (line 4). Next we select 4 frames (line 6) and define the desired frame layout (lines 8–11) and individual background colors for each frame (lines 13–15). Note that the default background color is white. The colors are given as RGB (red, green, blue) values ranging from 0.0 to 1.0.

```
 1  load xy,test.cut
 2  load ni,test.nipl
 3  #
 4  fnam off
 5  #
 6  nfra 4
 7  #
 8  sfra 1,0.0,0.0,0.5,0.9
 9  sfra 2,0.5,0.4,1.0,0.9
10  sfra 3,0.5,0.0,1.0,0.4
11  sfra 4,0.0,0.9,1.0,1.0
12  #
13  bfra 2,0.9,0.9,0.9
14  bfra 3,0.9,0.9,0.9
15  bfra 4,0.7,0.7,0.7
```

The next part of the macro file contains the settings for frame 1. After setting the focus to this frame (line 19), data set 1 is selected for this frame (line 20). The following commands specify various settings and were already explained in previous examples. The 'font' command in line 32 increases the font size of all fonts used for this frame by 10%.

```
16  #
17  # Frame 1 with cross section
18  #
19  afra 1
20  kfra 1,1
21  buff 0.4
22  lcol 1,6
23  mtyp 1,3
24  mcol 1,1
25  msiz 1,0.2
26  skal 0.0,3.2,0.0,5000.0
27  mark 1,1000
28  achx [h h 1.0]
29  achy Counts
30  tit1 Cross section
```

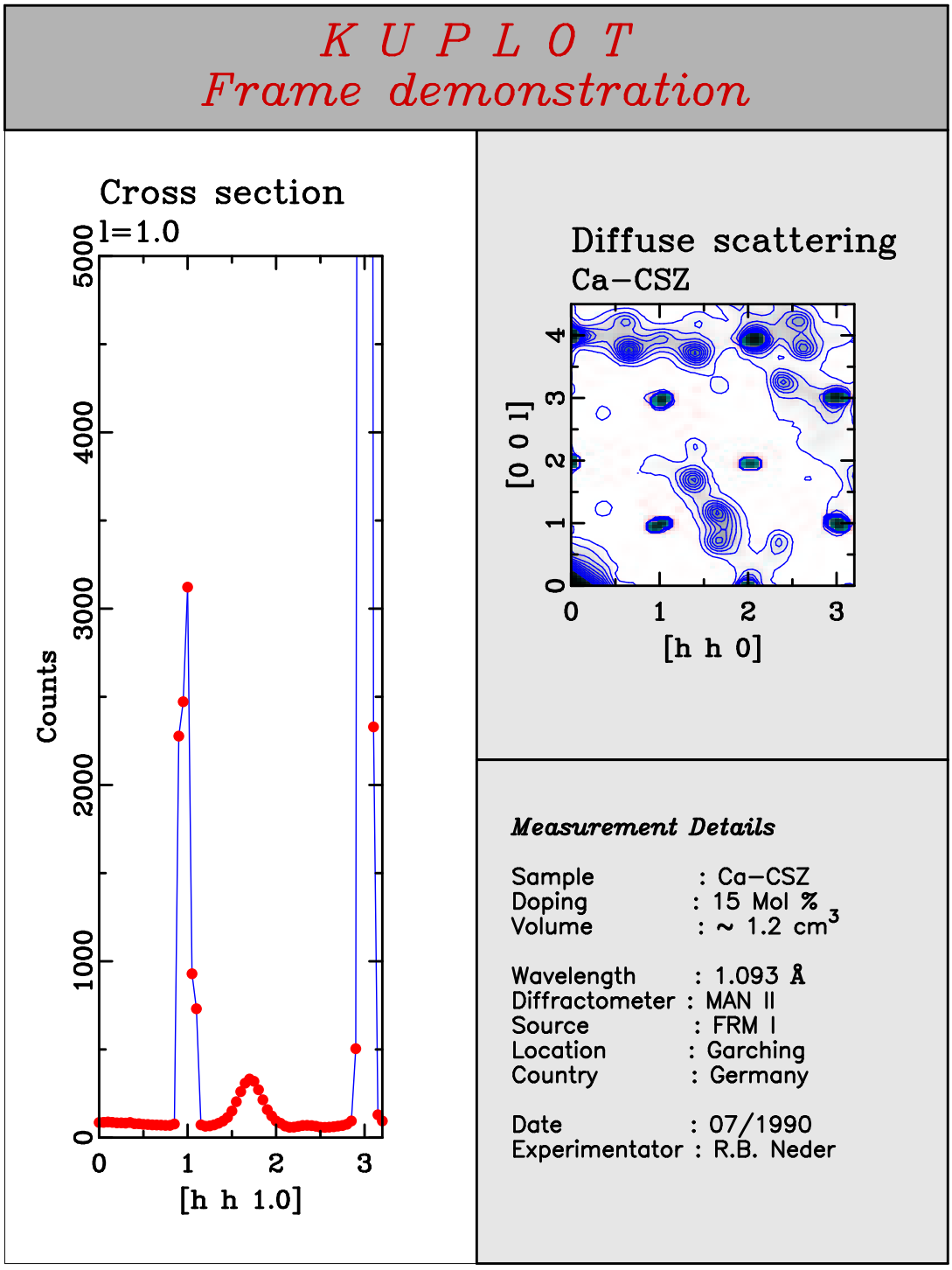


Figure 4.4: Advanced frame example plot

```
31  tit2 l=1.0
32  font size,1.1
```

Next we have the settings for frame 2 containing the 2D data set number 2, the diffuse neutron scattering of Ca-CSZ we used as example before. Again the various commands were explained in earlier examples.

```
33  #
34  # Frame 2 with data plot
35  #
36  afra 2
37  kfra 2,2
38  glat 2,3
39  hart 2,3
40  hcol 2,1,3
41  hlin 1,100,50,12
42  mark 1,1,0,0
43  aver 0.707
44  achx [h h 0]
45  achy [0 0 l]
46  tit1 Diffuse scattering
47  tit2 Ca-CSZ
48  font size,1.1
```

The next frame contains text rather than data. The text for this frame is read from the file '*ref.txt*' which contains exactly the text you can see at the bottom right corner of the plot in Figure 4.4. In lines 52–56 the justification of the text, the font type and font size are specified. The text file may contain the same special characters and control sequences discussed in section 2.3.

```
49  #
50  # Frame 3 with text
51  #
52  afra 3
53  kfra 3,ref.txt
54  font just,left
55  font typ,5,1
56  font siz,5,12
```

Finally we specify another text file containing the title line of the plot. We select the text to be centered (line 62) and set the font to 'italics' (font 3), set the color to dark red (pen 7) and increase the size of the font to 24 points.

```
57  #
58  # Frame 4 with title text
59  #
60  afra 4
61  kfra 4,tit.txt
```

```
62  font just,center
63  font typ,5,3
64  font col,5,7
65  font siz,5,24
```

Currently one has not much control about the layout in a frame containing text but that might change in future version of the program *KUPLOT*.

Chapter 5

Using the mouse

To enter the mouse mode, simply use the command ’mouse’. The plot window will now contain a number off buttons in addition to the view graph as can be seen in Figure 5.1.

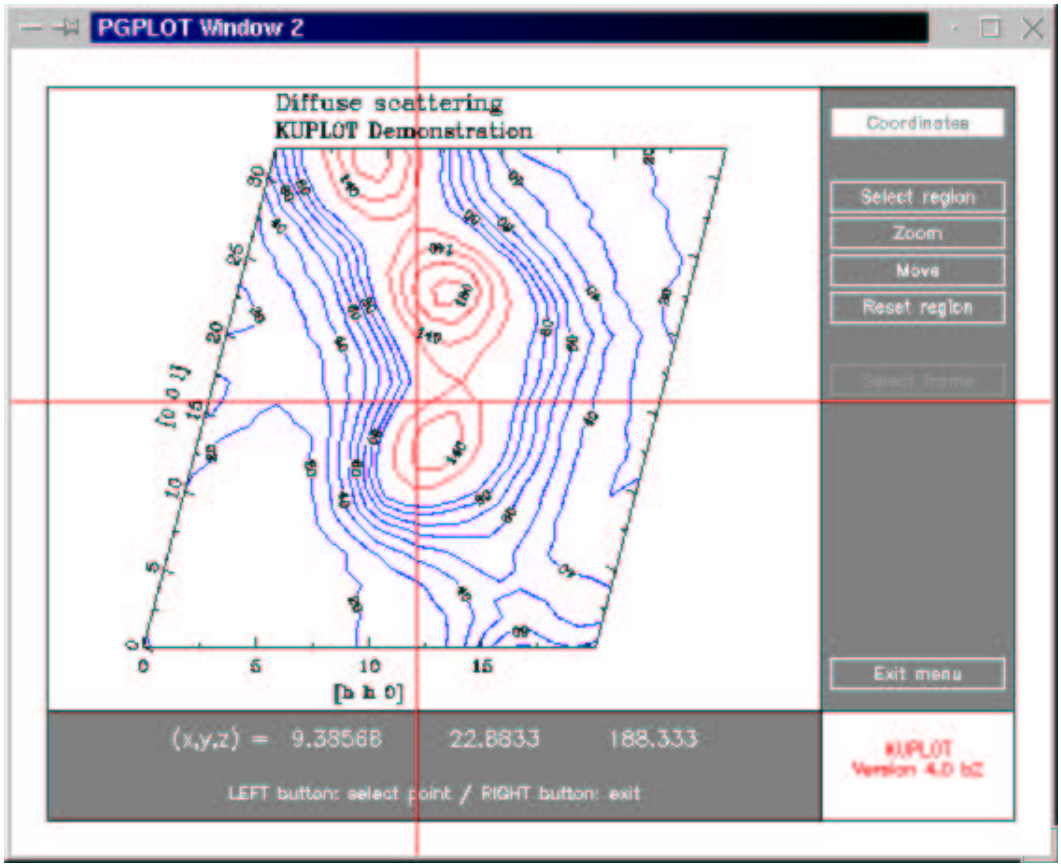


Figure 5.1: *KUPLOT* in ’mouse’ mode

The button 'coordinates' allows the user to display the coordinates of the mouse pointer by clicking the left mouse button. The values are displayed below the plot. In case of 2D files, the z-value of the closest data point is also shown. The right mouse button terminates the coordinate display function. The next group of buttons allows the user to use the mouse to modify the potting area. 'Select region' does exactly that by clicking on the lower left corner of the desired area first. A moving frame will appear and the user can select the other corner of the new region. The buttons 'zoom' and 'move' perform these functions using all three mouse buttons. A short help text explaining which button is doing what appears below the plotting area once a function is selected. 'Reset region' allows the user to quickly display the complete range of data again. In cases where the current plot contains frames (see chapter 4), the active frame is marked by a light red background color. All functions will affect the active frame only. The focus can be moved to another frame via the 'select frame' button, which is only active when there is more than a single frame. Finally to get back to the command prompt of *KUPLOT* use the 'exit menu' button. Note that in cases where commands are written to a macro file via the command 'learn', all region functions will write the corresponding 'skal' command in the macro file. One should also be careful when using the 'mouse' command in macro files, since *KUPLOT* will not execute any further commands in until the 'exit menu' button is hit.

Alternatively, the 'mouse' command can have additional parameters which specify what will be selected using the mouse pointer. In this case no buttons will show up and the coordinates will be returned in the 'res[]' variables. For details on valid parameters simply type 'help mouse' at the *KUPLOT* command prompt.

Chapter 6

FORTRAN style interpreter

The program includes a FORTRAN style interpreter that allows the user to program complex modifications. The interpreter provides variables, linked to data sets and free variables, loops, logical construction, basic arithmetic and built in functions. Commands related to the FORTRAN interpreter are '=', 'break', 'do', 'else', 'elseif', 'enddo', 'endif', 'eval', 'if'.

6.1 Variables

All variables are denoted by a name, immediately followed by a left square bracket [, one or more indices and a right square bracket].

Example : i[1], r[0], i[i[1]], x[1,2]

The left square brackets **must** immediately follow the name of the variable. Blanks within the square brackets are not significant. The index can be any integer expression, especially any of the integer variables again. Two free variables are provided, an integer, i[n], and a real, r[n] variable. Each of these variables is actually an array of dimension n given in squared brackets. The allowed range for n is 0 to MAXPARAM, which is defined at compilation time by the parameter MAXPARAM in the file '*param.inc*', see appendix B for further help.

Beside these variables i[n] and r[n] for general use a large variety of variables is linked to information of the currently loaded data sets in *KUPLOT*. Some of these variables can not be modified, others like the individual values of data sets can be altered, thus allowing to modify the loaded data. Table 6.1 shows a summary of data related variables. The values of these variables marked with * are read-only and can not be altered. Note that the variable 'n[1]' can be overwritten, however this results in data sets being lost. To allocate space for a new data set use the command 'allocate', just increasing the value of 'n[1]' will *not* work. One possible application for example is to remove just the last data set rather than using 'rese' and loading all other data sets again. The following line would just remove the last data set:

```
n[1] = n[1]-1
```

Variable	Description
n[1]	Number of loaded data sets (**now writable**)
n[2]*	Maximum allowed number of data sets
n[3]*	Maximum allowed number of frames
n[4]*	Maximum allowed number of annotations
n[5]*	Maximum allowed number of bonds
nx[<ik>]*	Number of points in x-direction for data set <ik>
ny[<ik>]*	Number of points in y-direction for data set <ik>
ni[<ik>]*	Type of data set <ik>. (0 for 1D and 1 for 2D data)
np[<ik>]*	Number of points of 1D data set <ik>
xmin[<ik>]*	Minimum x-value of data set <ik>
xmax[<ik>]*	Maximum x-value of data set <ik>
ymin[<ik>]*	Minimum y-value of data set <ik>
ymax[<ik>]*	Maximum y-value of data set <ik>
zmin[<ik>]*	Minimum z-value of data set <ik>
zmax[<ik>]*	Maximum z-value of data set <ik>
pwin[i]*	Current plotting dimensions (1:xmin,2:xmax,3:ymin,4:ymax)
x[<ik>,<ip>]	X-value of data point <ip> of data set <ik>
y[<ik>,<ip>]	Y-value of data point <ip> of data set <ik>
dx[<ik>,<ip>]	Value of σ_x for data point <ip> of data set <ik>
dy[<ik>,<ip>]	Value of σ_y for data point <ip> of data set <ik>
z[<ik>,<ix>,<iy>]	Z-value of point <ix>,<iy> of data set <ik>
axis[1,i]	Angle of labels for axis i (1=x,2=y)
axis[2,i]	Length of major ticks for axis i (1=x,2=y)
axis[3,i]	Length of minor ticks for axis i (1=x,2=y)
axis[4,i]	Subdivisions between ticks, axis i (1=x,2=y)
axis[5,i]	Distance numbers - axis i (1=x,2=y)
axis[6,i]	Distance label - axis i (1=x,2=y)
cmap[<ic>,3]	Color map entry in R,G,B (0..1) for color <ic>
cmax[1]*	Number of bitmap colors available
p[<i>]	Fit variable number <i> (see section 8)

Table 6.1: *KUPLOT* specific variables

Another variable is res[i] which contains the results of a particular *KUPLOT* command. The variable res[0] contains the number of elements returned in res[i]. Check the online help for the different commands to see what results might be returned in this variable.

6.2 Arithmetic expressions

KUPLOT allows the use of arithmetic expressions using the same notation as in FORTRAN. Valid operators are '+', '-', '*', '/' and '**'. Expressions can be grouped by round brackets (and). The usual hierarchy for the operators holds. Values of expressions can be assigned to any modifiable variable. If you know FORTRAN (or another programming language) you will have no problems with these examples.

```
i[0] = 1
r[3] = 3.1415
r[i[1]] = 2.0*(i[5]-5.0/6.5)
```

6.3 Logical expressions

Logical expressions are formed similar to FORTRAN. They may contain numerical comparisons using the syntax: *<arithmetic expression><operator><arithmetic expression>*. The allowed operators within *KUPLOT* are *.lt., .le., .gt., .ge., .eq.* and *.ne.* for operations less than, less equal, greater than, greater equal, equal and not equal, respectively.

Logical expressions can be combined by the logical operators *.not.*, *.and.* and *or*. The following example shows an expression that is true for values of i[1] within the interval of 3 and 11, false otherwise.

```
i[1].ge.3 .and. i[1].le.11
```

Logical operations may be nested and grouped using round brackets (and). For more examples see section 6.6.

6.4 Intrinsic functions

Several intrinsic functions are defined. Each function is referenced, as in FORTRAN, by its name **immediately** followed by a pair of parentheses (and) that include the list of arguments. Trigonometric and arithmetic functions are listed in Table 6.2. Table 6.3 contains various random number generating functions.

6.5 Loops

Loops can be programmed in *KUPLOT* using the 'do' command. Three different types of loops are implemented. The first type executes a predefined number of times. The syntax for this type of loop is

do <variable> = <start>,<end> [,<increment>]
. . . commands to be executed . . .
enddo

Type	Name	Description
real	sin(r) cos(r) tan(r)	Sine, cosine and tangent of <r> in radian
real	sind(r) cosd(r) tand(r)	Sine, cosine and tangent of <r> in degrees
real	asin(r) acos(r) atan(r)	Arc sin, cosine, tangent of <r>, result in radian
real	asind(r) acosd(r) atand(r)	Arc sin, cosine, tangent of <r>, result in degrees
real	sqrt(r)	Square root of <r>
real	exp(r)	Exponential of <r>, base e
real	ln(r)	Logarithm of <r>
real	sinh(r) cosh(r) tanh(r)	Hyperbolic sine, cosine and tangent of <r>
real	abs(r)	Absolute value of <r>
integer	mod(r1, r2)	Modulo <r1> of <r2>
integer	int(r)	Convert <r> to integer
integer	nint(r)	Convert <r> to nearest integer
real	frac(r)	Returns fractional part of <r>

Table 6.2: Trigonometric and arithmetic functions

Loops may contain constants or arithmetic expressions for <start>, <end>, and <increment>. <increment> defaults to 1. The internal type of the variables is real. The loop counter is evaluated from *(<end> - <start>)/<increment> + 1*. If this is negative, the loop is not executed at all. The parameters for the counter variable, start end and increment variables are evaluated only at the beginning of the do - loop and stored in internal variables. It is possible to change the values of <variable>, <start> and/or <end> within the loop without any effect on the performance of the loop. This practice is not encouraged, could, however, be an unexpected source of errors.

The second type of loop is executed while <logical expression> is true. Thus it might not be executed at all. The syntax for this type of loop is

Type	Name	Description
real	ran(r)	Uniformly distributed pseudo random number between 0.0 and 1.0. Argument <r> is a dummy
real	gran(r1, typ)	Gaussian distributed random number with mean 0 and a width given by <r1>. If <typ> is "s" <r1> is taken as sigma, if <typ> is "f" <r1> is taken as FWHM.
real	gbox(r1, r2, r3)	Returns pseudo random number with distribution given by a box centered at 0 with a width of <r2> and two half Gaussian distributions with individual sigmas of <r1> and <r3> to the left and right, respectively.

Table 6.3: Random number functions

do while <logical expression>
. . . commands to be executed . . .
enddo

The last type of loop is executed until <logical expression> is true. This loop, however, is always executed once and has the following syntax

do
. . . commands to be executed . . .
enddo until <logical expression>

In the body of commands any valid *KUPLOT* command can be used. This includes calls to the sublevels, further do loops or macros, even if these macros contain do loops themselves. The maximum level of nesting is limited by the parameter *MAXLEV* in the file '*doloop.inc*'. If necessary adjust this parameter to allow for deeper nesting. All commands from the first 'do' command to the corresponding 'enddo' are read and stored in an internal array. This array can take at most *MAXCOM* (defined in file '*doloop.inc*' as well) commands at every level of nesting. If lengthy macro files are included in the do loop, this parameter might have to be adjusted.

If a do loop (or an if block) needs to be terminated, the 'break' command will perform this function. The parameter on the 'break' command line gives the number of nested levels of 'do' and 'if' blocks to be terminated. The interpreter will continue execution with the first command following the corresponding 'enddo' or 'endif' command. An example is given below, note, that the line numbers are only given for better orientation and are no actual part of the listed commands.

```
1  do i[2]=1,5
2    do i[1]=1,5
3      if ((i[1]+i[2]) .eq 6) then
4        break 2
5      endif
6    enddo
7  enddo
```

In this example, the execution of the inner do-loop will stop as soon as the sum of the two increment variables i[1] and i[2] is equal to 6. The program continues with the 'enddo' line of the outer do - loop. Notice that two levels need to be interrupted, the if block and the innermost do loop. If the parameter had been equal to one, only the if block would have been interrupted, while the innermost do loop would have continued without break.

6.6 Conditional statements

Commands can be executed conditionally by using the 'if' command. Analogous to FORTRAN, the if-control structure takes the following form:

if (<logical expression>) then
... commands to be executed ...
elseif (<logical expression>) then
... commands to be executed ...
else
... commands to be executed ...
endif

The logical expressions are explained in section 6.3. Enclosed within an if block any valid *KUPLOT* command can be used. This includes calls to the sublevels further if blocks, do loops or macros, even if these macros contain if blocks or do loops themselves. The 'elseif' and 'else' section is optional. The maximum level of nesting is limited by the parameter *MAXLEV* in the file '*doloop.inc*'. If necessary adjust this parameter to allow for deeper nesting. All commands from the first 'if' command to the corresponding 'endif' are read and stored in an internal array. This array can take at most *'MAXCOM'* (defined in file '*doloop.inc*' as well) commands at every level of nesting. If lengthy macro files are included in the do loop, this parameter might have to be adjusted.

If an if block (or a do loop) needs to be terminated, the 'break' command will perform this function. The parameter on the 'break' command line gives the number of nested levels of 'do' and 'if' blocks to be terminated. The interpreter will continue execution with the first command following the corresponding 'enddo' or 'endif' command. See the example in section 6.5 for further explanations.

```
1  load xy,test.dat
2  #
3  do i[1]=1,np[1]
4    if(dy[1,i[1]].gt.0.0) then
5      y[1,i[1]]=(1.0+0.1*(0.5-ran(0)))*y[1,i[1]]
6    endif
7  enddo
```

The example listed above illustrates the use of loops and conditional statements within *KUPLOT*. Again, the line numbers are given for easy reference and not part of the actual *KUPLOT* input. First a 1D data set is loaded (line 1). In line 3 starts a do-loop over all data points of data set 1. The variable np[1] contains this information (see Table 6.1 in section 6.1). In this example a $\pm 5\%$ random noise shall be added to all data points with $\sigma_y > 0$. The conditional statement in line 4 is true for those points. Since the function 'ran' produces a uniformly distributed pseudo random number in the range 0.0 to 1.0, the factor the y-value is multiplied with (line 5) ranges from 0.95 to 1.05.

6.7 Filenames

Usually, file names are understood as typed, including capital letters. Unix operating systems distinguish between upper and lower case typing ! However, sometimes it is required to be able to alter a file name

e.g. within a loop. Thus, *KUPLOT* allows the user to construct file names by writing additional (integer) numerical input into the filename. The syntax for this is:

"string%dstring",<integer expression>

The file format MUST be enclosed in quotation marks. The position of each integer must be characterized by a '%d'. The sequence of strings and '%d's can be mixed at will. The corresponding integer expressions must follow after the closing quotation mark. If the command line requires further parameter (like 'addfile' for example) they must be given after the format-parameters. The interpretation of the '%d's follows the C syntax. Up to 10 numbers can be written into a filename. All of the following examples will result in the file name '*a1.1*':

```
i[5]=1
load xy,a1.1
load xy,"a%d.%d",1,1
load xy,"a%d.%d",4-3,i[5]
```

The second example shows how filenames are changes within a loop. Here the files '*data1.calc*' to '*data 11.calc*' will be loaded.

```
do i[1]=1,11
..
load xy,"data%d.calc",i[1]
..
enddo
```

6.8 Macros

Any list of valid *KUPLOT* commands can be written to an ASCII file and executed indirectly by the command '@<filename>'. The commands must start in the left most column of the file and are otherwise executed as typed. Macro files can be written by any editor on your system or be generated by the 'learn' command. 'learn' starts to remember all the commands that follow and saves them into the file given on the 'learn' command. The learn sequence is terminated by the 'lend' command. The default extension of the macro file is '*.mac*'. Macro files can be nested and even reference themselves directly or indirectly. This referencing of macro files is, however, just a nesting of the corresponding text of each macro, not a call to a function. All variable retain their values. If an error occurs while executing a macro, *KUPLOT* immediately stops execution of all macros and returns to the interactive prompt. If the macro switched to a sublevel, and the error occurred inside of this sublevel, *KUPLOT* will remain within this sublevel the interactive prompt corresponding to this sublevel is returned.

On the command line of the macro command '@', optional parameters can be supplied. Within the macro these have to be referenced as '$1', '$2' etc. Upon execution of the macro the formal parameters '$n' are replaced by the character string of the actual values from the command line. As any other command

parameters, these parameters must be separated by comma. If a formal parameter is referenced inside a macro without a corresponding parameter on the command line, an error message is given. An example is given below:

```
# Adds two numbers supplied as command line parameters.
# The value is stored in variable defined by parameter three
#
$3 = $1 + $2
```

If this macro is called with the following line, @add 1,2,i[4], the result is stored in variable i[4] which now has the integer value 3.

If the program *KUPLOT* is started with command line parameters, e.g. `kuplot 1.mac 2.mac`, the program will execute the given macros in the specified order, in our example first `1.mac` then `2.mac`. If a macro is not found in the current working directory, a system macro director is searched. This system macro directory is located at `path_to_binary/sysmac/kuplot/`. Commonly used macro files might be installed in this directory.

6.9 Working with files

The command language offers the user several commands to write variables to a file or read values from a file. First a file needs to be opened using the command 'fopen'. In the standard configuration, the program can open five files at the same time. The first parameter of all file input/output related commands is the unit number which can range from 1 to 5. The commands 'fget' and 'fput' are used to read and write data, respectively. The following example illustrates the usage of these commands:

```
 1  fopen 1,sin.dat
 2  fput 1,'Cool sinus function'
 3  #
 4  do i[1]=1,50
 5    r[1]=i[1]*0.1
 6    r[2]=sin(r[1])
 7    fput 1,r[1],r[2]
 8  enddo
 9  #
10  fclose 1
```

In line 1 we open the file '*sin.dat*' and write a title (line 2). If the file already exists it will be overwritten. Note that the text must be given in *single* quotes. Text and variables may be mixed in a single line. Next we have a loop calculatin $y = \sin(x)$ and writing the resulting x and y values to the open file (line 7). Finally the file is closed (line 10). To read values from a file use simply the command 'fget' and the read numbers will be stored in the specified variables. In contrast to writing to a file, mixing of text and number is not allowed when reading data. However, complete lines will be skipped when the command 'fget' is entered without any parameters.

Chapter 7

Manipulating and analyzing data

In this chapter we will discuss *KUPLOT* functions that allow the analysis and manipulation of data. The first two sections deal with data manipulation, first using build-in functions and later using variables. The last section of this chapter summarises data analysis functions of *KUPLOT*.

7.1 Simple data calculations

A common task is the numerical manipulation of a data set, e.g. to add some constant to a data set or inverse all y-values. The command 'ccal' offers a variety of manipulation functions for a specific data set. The command 'kcal' on the other had allows simple arithmetic operations between two data sets. The commands and valid operations are listed in Table 7.1.

Command	Operation	Description
ccal	add	Performs $x_i = x_i + a$
	exp	Performs $x_i = \exp(x_i)$
	inv	Performs $x_i = \frac{1}{x_i}$
	log	Performs $x_i = \ln(x_i)$
	mul	Performs $x_i = f \cdot x_i$
	sqr	Performs $x_i = \sqrt{x_i}$
	squ	Performs $x_i = x_i^2$
kcal	add	Performs $x_i''' = x_i'' + x_i'$
	sub	Performs $x_i''' = x_i'' - x_i'$
	mul	Performs $x_i''' = x_i'' \cdot x_i'$
	div	Performs $x_i''' = x_i''/x_i'$

Table 7.1: Data manipulation functions

Note that x_i in Table 7.1 stands for x-, y-, z- and σ_x- or σ_y-values depending on the given parameters.

The following simple command will multiply all y-values of data set one with the factor 1.75:

```
ccal mul,wy,1,1.75
```

The parameter 'mul' indicates that a multiplication is to be performed using the y-values which are selected by the next parameter ('wy'). Finally data set one and the desired factor of 1.75 are specified. For x- and z-values use 'wx' and 'wz', the standard deviations σ_x and σ_y are selected using the parameters 'dx' and 'dy'.

Another feature of *KUPLOT* is the capability to create a data set from an arithmetic expression rather than reading it from a file. This is done using the command 'func'. The following two commands demonstrate usage of 'func' to create a 1D data set ($y = \sin(x)$) and a 2D data set ($z = \sin(x) \cdot \cos(y)$).

```
func sin(r[0]),0.0,6.3,0.1
func sin(r[0])*cos(r[1]),0,6,0.1,0,6,0.1
```

Note that the variable 'r[0]' is used for the x-argument and r[1] is used as y-argument. Thus values previously stored in these two variables are destroyed by the 'func' command. The following commands are the range and the grid size in the two directions. In our first example, the desired x-range is $0.0 \rightarrow 6.3$ with a grid size of $\Delta x = 0.1$. This results the creation of a data set with 64 points. The second 'func' command shown above creates a 2D data set ranging from 0.0 to 6.1 in x- and y-direction with a grid size of $\Delta x = \Delta y = 0.1$ given a size of 61x61 data points. Alternatively, space for a new data set can be initialize using the command 'alloc' and the data values are then calculated using the FORTRAN style interpreter of *KUPLOT* (see section 7.3. However, the creation of large data sets this way from an arithmetic expression might be relatively slow.

7.2 Data manipulation

7.2.1 Data smoothing

KUPLOT has two different smoothing functions that can be used for 1D as well as 2D data. The first type of smoothing is a simple sliding average. The corresponding command is 'glat'. The name is a reminder that the first *KUPLOT* version was in German (smoothing in German is 'glätten'). The smoothing is performed by a sliding average of n neighboring points. The value of n is given as parameter of the command 'glat'. An example for the smoothing operation is given in Figure 7.1. The top view graph shows the raw data showing quite noisy contour lines. The picture below shows the same data after the data set was smoothed with a value of $n = 7$ using the command 'glat 1,7' assuming the values are stored as data set one. It is apparent from Figure 7.1 that this type of averaging broadens the peaks and does not preserve the peak heights. An alternative way to smooth data implemented in *KUPLOT* is the Savitzky-Golay algorithm which is in principle a weighted sliding average. The weight is given by a polynomial of user definable order (the default is 2). The command for the later type of smoothing is 'smooth' with parameters identical to 'glat'. However, the minimum number of n is five. The bottom view graph in Figure 7.1 was smoothed with $n = 7$

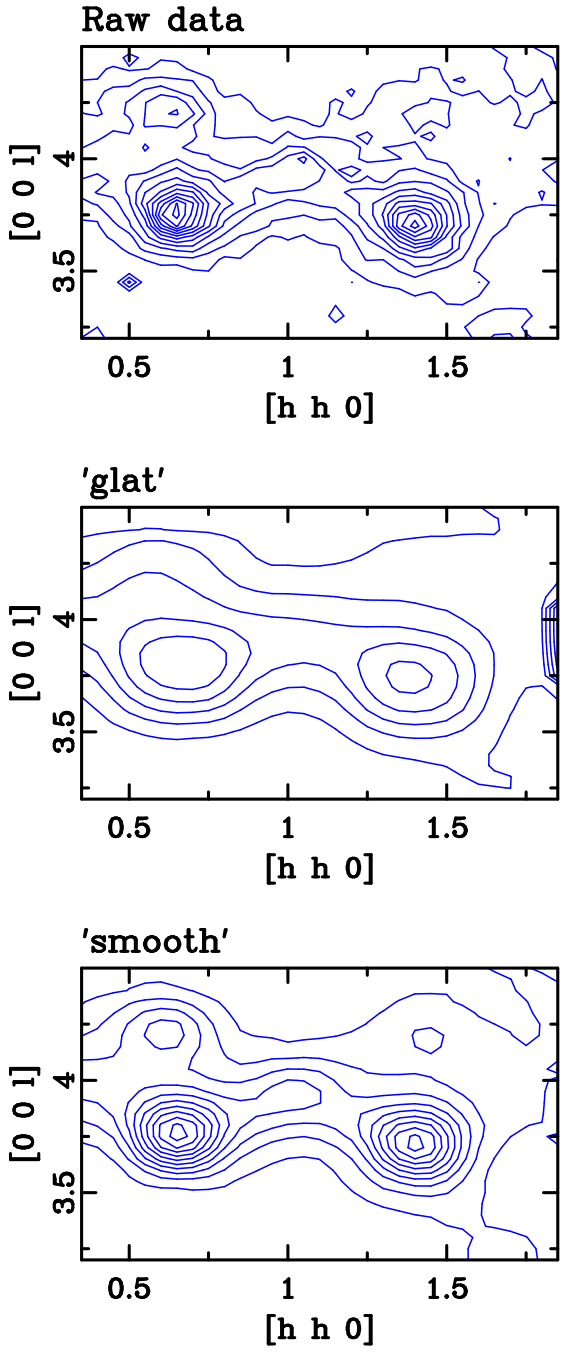


Figure 7.1: Demonstration of data smoothing

using the command 'smooth'. It can be clearly seen that the widths and heights of the peaks are much better preserved. For a detailed discussion about the smoothing algorithm and its limits see e.g. Numerical Recipes by Press, Flannery, Teukolsky & Vetterling, Cambridge University Press, 1989.

As a default, 2D data sets are per smoothed in x and y direction. The user can restrict the smoothing to only one direction by adding the optional parameter "x" or "y" to the 'glat' or 'smooth' command. Again the reader might refer to the online help for more detailed information on the smoothing commands.

7.2.2 More data manipulation

In this section we will briefly mention other commands that allow the manipulation of data. A data set can be simply sorted increasing in x values via the command 'sort'. Another function is 'rebin' which transform the data to a user defined grid spacing in x. Note that this command is limited to 1D data sets. To rebin 2D data save them as xyz file and re-load them with 'load zz,..' and the desired grid spacing.

Sometimes two data sets differ by a scaling factor of an offset. The command 'match' allows to find the scaling and/or offset that gives the best agreement between two data sets. The scaling and bacground is directly applied. Note that this function only works for data sets, that have identical points in x. The command 'merge' allows one to combine data sets. Points with common x-values are averaged.

7.3 Data manipulation using variables

A very flexible way to manipulate data is the use of variables. Details about the FORTRAN style interpreter and the usage of variables were discussed in chapter 6. Two simple examples are given here for illustration.

In the first example we assume having a data set containing measured intensities as function of some angle ω. The following *KUPLOT* commands will create values for σ_y according to the relation $\sigma_y(i) = \sqrt{y(i)}$.

```
1  do i[1]=1,np[1]
2    dy[1,i[1]]=sqrt(y[1,i[1]])
3  enddo
```

The first line starts a DO loop over all data points of data set one, assuming that is where our data are stored. The variable i[1] is the loop control variable and np[1] contains the number of data points of data set one. For a complete list of variables refer to section 6.1 of this users guide. Next the value of σ_y for data point i[1] is computed (line 2). The first parameter in 'dy[1,i[1]]' specifies data set one. Finally (line 3) the loop is closed. The corresponding plot with the error bars of the newly created values of σ_y is shown in Figure 7.2.

The second example shows how to create a new data set using variables. We assume that we have two data sets loaded, both having the same length and identical x-values. We want to create a new data set with the same x-values and y-values that are the average of the y-values of the two load data sets, thus $y_i''' = \frac{1}{2}(y_i'' + y_i')$. Here y''' stands for the new data set whereas y'' and y' represent the values of the two loaded data sets. The *KUPLOT* commands for our task are listed below:

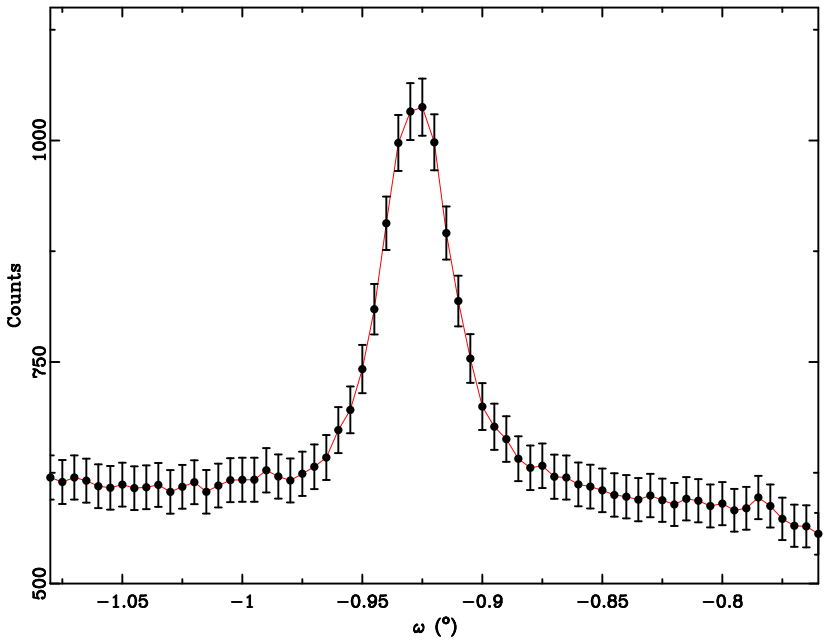


Figure 7.2: Plot of data manipulation example

```
1  alloc aver.xy,np[1]
2  #
3  do i[1]=1,np[1]
4    x[3,i[1]]=x[1,i[1]]
5    y[3,i[1]]=0.5*(y[1,i[1]]+y[2,i[1]])
6  enddo
```

First we have to allocate space for the new data set. This is normally done by the command 'load' or 'func', but the command 'alloc' allows the user to create an empty data set, just what we need here. The name '*aver.xy*' given as parameter to the 'alloc' command in line 1 is the internal name for the data set, e.g. showing up in the top left corner of the plot or when using the 'show' command. This is an arbitrary name and **no** file of that name needs to exist. The second parameter specifies the size of the new data set, in out case the same size as data sets one (variable np[1]) or two. Next we have a loop as in the previous example over all data points (line 3). In lines 4–5 the x- and y-values of the new data set number 3 are computed. Since we assumed that both original data sets have the same x-values, we just use one of them as x-values for the new set. Finally the loop is closed (line 6).

Although the usage of variables allows one to perform the same functions as the 'ccal' and 'kcal' commands, the usage of variables is much slower especially for large data sets.

7.4 Data analysis

This section describes the data analysis functions of *KUPLOT*. Apparently variables can also be used to calculate averages and analyse data sets. The contents of a variable or result of an expression can be displayed with the command 'eval'. Another wide area of data analysis is to fit a theory function to a data set. The least square fitting functions of *KUPLOT* are discussed in the next chapter. As example for this chapter we use a subsection of the diffuse scattering data displayed in previous examples. The data are shown in Figure 7.3. The circles mark positions of maxima found in the data set by the command 'smax'.

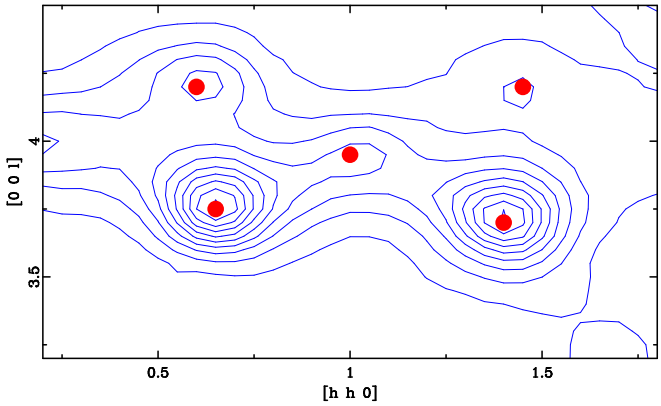


Figure 7.3: Marking of maxima within a plot

A maximum determined by 'smax' is defined as a point where all n neighbouring points have smaller y- or z-values compared to the reference point. The value of n is the second parameter of the 'smax' command, the first is the data set number. The maxima marked in Figure 7.3 were determined with 'smax 1,3' assuming we are dealing with data set one. The command 'ptyp' allows one to select a symbol analog to 'mtyp' to mark the positions of the determined maxima. Furthermore the positions of the found maxima are displayed on the screen. The output for our example is shown below:

```
Found maxima data set   1 (ifen =   3) :

   No.        pos. x       pos. y         value
   ------------------------------------------------
     1          .600        4.200           272.778
     2          .650        3.750           567.111
     3         1.000        3.950           273.889
     4         1.400        3.700           509.667
     5         1.450        4.200           156.778
```

You might verify these coordinates as the marked positions in figure 7.3. Other functions allow the user to determine the integral or mean values of a given area of the data set. Next we will determine the integral of the left diffuse peak in figure 7.3. This is done using the command

```
inte 1, .423750, .903750, 3.4716, 3.99168
```

The first parameter specifies the data set number followed by the area to be integrated given as $x_{min}, x_{max}, y_{min}$ and y_{max}. If those last 4 parameters are omitted, the complete current plotting window is used. The screen output of this command is:

```
Integration result for data set   1 :
   x-range    :    .4238     to    .9038
   y-range    :    3.472     to    3.992
   Integral   :    57.06     +-    .3650      (    100 pkt)
```

The command 'mean' with similar parameters can be used to calculate mean values and standard deviations in the given region. The region above was determined using the 'mouse' functions (see 5).

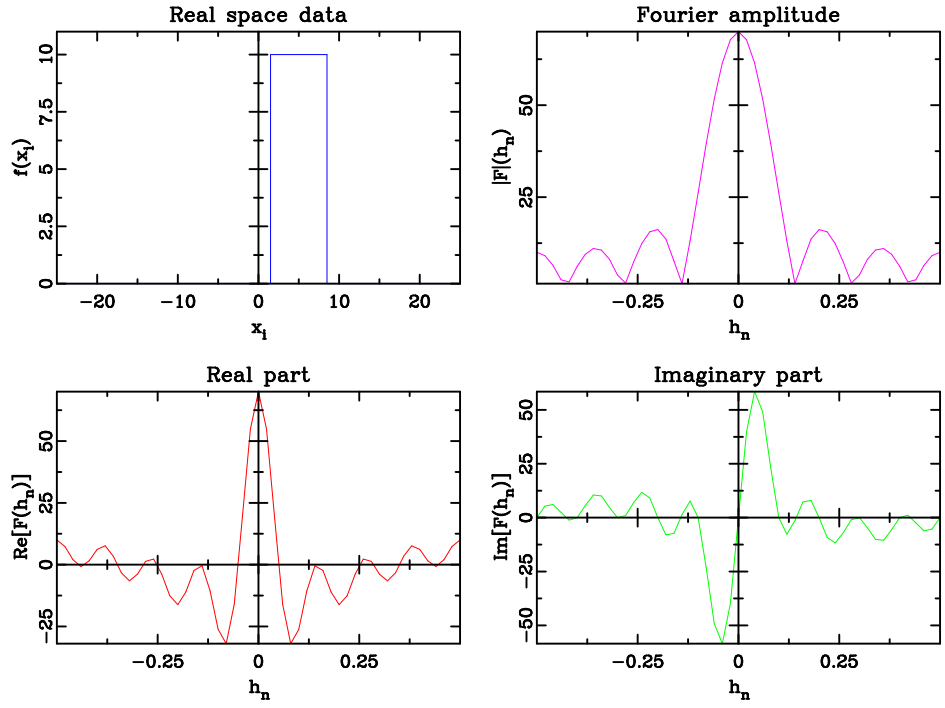


Figure 7.4: Fourier transform of box function

7.5 Fourier transform

KUPLOT can calculate a discrete Fourier transform of a 1D or 2D data set. Let us start with a little mathematic first. Details can again be found in the Numerical Recipes. The function we want to Fourier transform is $f(x_i)$ with $i = 1, ..N$ with a constant grid size of Δx. This gives us the following range in Fourier space:

$$h_n = \frac{n}{N\Delta x} \quad \text{with} \quad n = -\frac{N}{2}..\frac{N}{2} \tag{7.1}$$

The Fourier transform is calculate as

$$F(h_n) = \Delta x \sum_{i=1}^{N} f(x_i) \left[\cos(2\pi h_n x_i) + i \sin(2\pi h_n x_i)\right]. \tag{7.2}$$

The Fourier transform is a complex quantity and *KUPLOT* allows one to select the real and imaginary part and/or amplitude and phase angle to be stored as new data sets.

Let us consider a simple example, a box function as displayed in the upper left panel of Figure 7.4. Our data set has $N = 50$ points and a grid size of $\Delta x = 1.0$. Using equation 7.1 we find the grid size in Fourier space to be $1/50 = 0.02$ and a calculated range of $-1/2\Delta x = -0.5$ to 0.5. In our example the command for the calculation of the Fourier transform is:

```
four ria
```

The parameter 'ria' specifies that we want to keep the **r**eal and **i**maginary part as well as the **a**mplitude of the Fourier transform. All three parts and the box function are shown in Figure 7.4.

Chapter 8

Least square fitting

This chapter describes the least-square (LS) fitting segment of the program *KUPLOT*. The user can choose between polynom, Gaussian and Lorentzian for 1D data or a Gaussian for 2D data. Furthermore this version of *KUPLOT* supports user defined theory functions thus allowing for virtually any fit function desired. A description of the LS method itself is beyond the scope of this manual and the reader might refer to various text books on statistics or numerical mathematics. For a list of all fit related commands of *KUPLOT* check the online help or the command reference manual. Furthermore, the interactive tutorial contains a fit example.

8.1 Fitting 1d data sets

Starting point for a LS refinement is a loaded data set (x_i, y_i), the selection of a suitable theory function $y(x_i, p)$ to describe the data and a set of starting values for the fit parameters p. The program *KUPLOT* offers the following three predefined theory functions and additionally allows the use of a user defined function (see section 8.3).

$$y(x_i,p) = \sum_{n=0}^{N} p_n x_i^n \tag{8.1}$$

$$y(x_i,p) = p_1 + p_2 x_i + \sum_{n=1}^{N} p_{1,n} \cdot \exp\left\{\frac{-(x_i - p_{2,n})^2}{\sigma_n^2}\right\} \tag{8.2}$$

$$y(x_i,p) = p_1 + p_2 x_i + \sum_{n=1}^{N} p_{1,n} \cdot \left\{\frac{\sigma_n^2}{(\sigma_n^2 + 4(x_i - p_{1,n}))^2}\right\} \tag{8.3}$$

$$with \quad \sigma_n = \begin{cases} p_{3,n} \cdot p_{4,n} & \text{for } x_i < p_{2,n} \\ \frac{p_{3,n}}{p_{4,n}} & \text{else} \end{cases}$$

Let us have a closer look at these theory functions. The first one shown in equation 8.1 is a simple polynom of the order of N which can be defined by the user. The fit parameters p_n are the corresponding

coefficients. Note that *KUPLOT* actually numbers the fit parameters starting with one, i.e. parameter one corresponds to the term x^0.

The next function (equation 8.2) is a sum of N Gaussians and a linear background defined by the first two parameters p_1 and p_2. The number of Gaussians to be used is defined by the user. Each Gaussian is represented by a set of four fit parameters: $p_{1,n}$ is its peak height and $p_{2,n}$ is its peak position. The half width of the Gaussian (as well as the Lorentzian) is defined by a half width parameter $p_{3,n}$ and an asymmetry parameter $p_{4,n}$. Obviously the symmetric case is given by $p_{4,n} = 1.0$. The Lorentzian theory function (equation 8.3) is defined by a similar set of parameters. Currently both function types can not be used in combination except as user defined theory function as described in section 8.3.

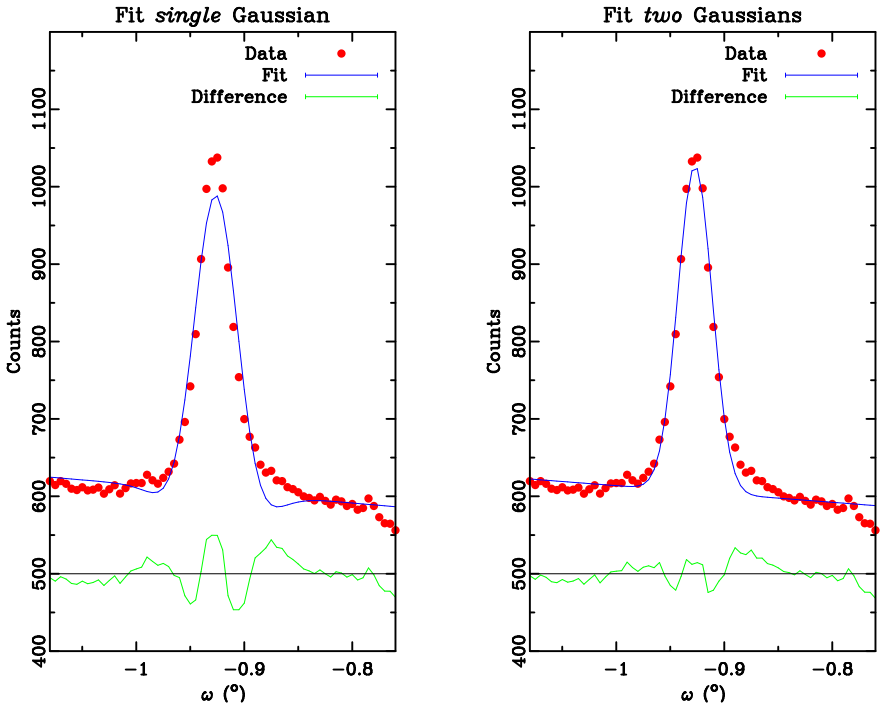


Figure 8.1: Fit results for 1D example

Now we will see the fitting segment of *KUPLOT* in action in the following example. The data set is the profile we have used in examples before in Figure 4.1. Assuming the corresponding data are loaded as set one, the next step is to enter the LS sublevel of the program *KUPLOT*. This is done by the command 'fit' followed by the data set number that should be used as observed data (line 1). The macro is show below and as before the line number are not part of the macro file itself.

```
1  fit 1
2  #
3  mfen 7
4  func gaus,1
5  par
```

```
6  #
7  run
8  save
```

The command in line 3 sets the 'window' size used to determine maxima within the plot which are used as defaults for the starting values. The procedure is equivalent to the command 'smax' explained in section 7.4. For our example the setting to 7 points will determine the main maximum correctly. Next (line 4) the theory function is specified to be one Gaussian. Note that *KUPLOT* determines the default starting values for the fitting parameters p at this stage and the command 'mfen' has to be entered before to have an effect. Furthermore all current values of the fit parameters, e.g. from a previous run, are replaced by the new default values. The command 'par' (line 5) lists the current parameter settings and after checking them on the screen we are ready to go. The fit is started via the command 'run' in line 7 of our example command file. The fit progress is displayed on the screen until the fits converges to a minimum or the maximum number of cycles is reached. This maximum cycle number can be altered using the 'cyc' command in the fitting sublevel. In our example it took 9 iterations to reach the minimum and the resulting parameters are listed in table 8.1 on the left. Finally the results are saved (line 8) as two data files containing the calculated and difference data set and a text file summarizing the results.

Parameter	**One Gaussian**	**Two Gaussians**	
		Gaussian 1	Gaussian 2
Background p_1	506.(16)	501.(10)	
Background p_2	-107.(18)	-106.(11)	
Peak height	421.(9)	315.(10)	134.(10)
Peak position	-0.9270(4)	-0.9277(3)	-0.9250*
Half width	0.0374(9)	0.0278(9)	0.0640*
R-value	2.1%	1.2%	

Table 8.1: Results of example 1D fit

The resulting plot of the observed and calculated data as well as the difference between them is shown in Figure 8.1 again on the left. Although there a reasonable agreement is achieved, there are clearly visible differences especially at the tails of the Gaussian. Note that we have fitted a symmetric Gaussian which is the default, i.e. the parameter $p_{4,n}$ is set to 1.0 and not refined. To refine this parameter use the command 'par 6,1'. The first number is the number of the fit parameter (since we have two background parameters it is 4+2=6) and the next number can be 1 for refining or 0 for keeping the corresponding parameter fixed. An optional third parameter allows one to alter the value of the fit parameter.

To improve the fit we will try to fit two Gaussians located at the same position to the data set. This will allow to model the sides of the peak in a better way. The corresponding macro is shown below. Lines 1–4 are the same as before except that we select two Gaussians rather than one. Again we assume the profile is loaded as data set one.

```
 1  fit 1
 2  #
 3  mfen 7
 4  func gaus,2
 5  #
 6  par 7,1,0.0
 7  par 8,1,p[4]
 8  par 9,1,p[5]
 9  par10,0,p[6]
10  par
11  #
12  run
13  save
```

Since *KUPLOT* will only find one maximum a warning will be displayed and we have to set the starting values for our second Gaussian manually (lines 6–9). We set the peak height to 0.0 and take the other parameters from the settings for the first Gaussian using the variables p[i]. Just to be sure we list the current settings again (line 10) and the fit is started in line 12 by the 'run' command. Finally the results are saved as before (line 13). The resulting parameters are listed in Table 8.1. A measure for the quality of a fit is the R-vale which is defined as:

$$R = \sqrt{\frac{\sum_{i=1}^{N} w_i(y_i - y(x_i,p))^2}{\sum_{i=1}^{N} w_i y_i^2}} \tag{8.4}$$

Here the sum goes over all N data points (x_i, y_i) and $y(x_i, p)$ is the theory function. *KUPLOT* offers a variety of weights w_i to be selected to the fit. The default we have used in our example is $w_i = \frac{1}{y_i}$. Check the online help for the command 'wic' to obtain more information about supported weighting schemes. Inspection of the results for this second fit shown in Table 8.1 and Figure 8.1 clearly show that the second fit describes the data much better. However the parameters marked with * in Table 8.1 could not be refined and remained at their starting values. Possible steps to avoid that problem are to refine the parameters one by one or change the setting of 'URF' which controlled the 'speed' of the fit. A small value, e.g. 0.1, will converge quickly to the minimum but in case of a more complex problem, the fit might fail. A larger value of 'URF' on the other hand will change the parameters in each iteration by a smaller amount and the fit might work. However care has to be taken to not end up in a local minimum. The value of 'URF' (which stands for something very German) is altered via the command 'urf'. The authors recommend playing around with the various settings of the fit sublevel to get familiar with the LS fitting process.

8.2 Fitting 2d data sets

Actually there is no principle difference between fitting a 2D or a 1D data set. The only theory function of 2D data currently available in *KUPLOT* is a set of N 2D Gaussians as defined in equation 8.5. Each Gaussian is

defined by eight parameters, peak height $p_{1,n}$, peak position $p_{2,n}$ and $p_{3,n}$ in x- and y-direction, half widths $p_{4,n}, p_{5,n}$ and asymmetry parameters $p_{7,n}, p_{8,n}$ for each direction and a new parameter $p_{6,n}$ that defines the angle between the main axes of the Gaussian and the coordinate system. One additional parameter describes a flat overall background.

$$
\begin{aligned}
z(x_i, y_i, p) &= p_1 + \sum_{n=1}^{N} p_{1,n} \cdot \exp\left\{\frac{r_x^2}{\sigma_n^2}\right\} \cdot \exp\left\{\frac{r_y^2}{\tau_n^2}\right\} \\
\text{with } r_x &= \cos(p_{6,n})(x_i - p_{2,n}) + \sin(y_i - p_{3,n}) \\
r_y &= -\sin(p_{6,n})(x_i - p_{2,n}) + \cos(y_i - p_{3,n}) \\
\sigma_n &= \begin{cases} p_{4,n} \cdot p_{7,n} & \text{for } x_i < r_x \\ \frac{p_{4,n}}{p_{7,n}} & \text{else} \end{cases} \\
\tau_n &= \begin{cases} p_{5,n} \cdot p_{8,n} & \text{for } y_i < r_y \\ \frac{p_{5,n}}{p_{8,n}} & \text{else} \end{cases}
\end{aligned}
\tag{8.5}
$$

This equation certainly looks more complex than the 1D equivalent and we will illustrate the fitting of three 2D Gaussians to a section of the diffuse scattering data used in the 2D examples before. The data and the fit result are shown in Figure 8.2. The observed data are shown in the top view graph. The maxima used to determine the starting values for the fit are marked by black circles. One of the major mistakes when using the fit routine of *KUPLOT* are wrong starting values. The calculated data are shown in the middle view graph using identical contour line settings as for the data. The bottom view graph finally shows the difference between observed and calculated data. The solid black line is the zero level, red lines mark negative values and blue lines stand for positive contour levels. Note that the contour line interval used for the difference plot is only $\frac{1}{5}$ of the interval used for the observed and calculated data in order to make the differences more visible.

The macro file used for this example file is shown below. Again we assume the observed data are already loaded as data set one.

```
 1  fit 1
 2  #
 3  mfen 3
 4  func gau2,3
 5  par
 6  #
 7  run
 8  #
 9  par  7,1
10  par 23,1
11  #
12  run
13  save
14  #
15  exit
```

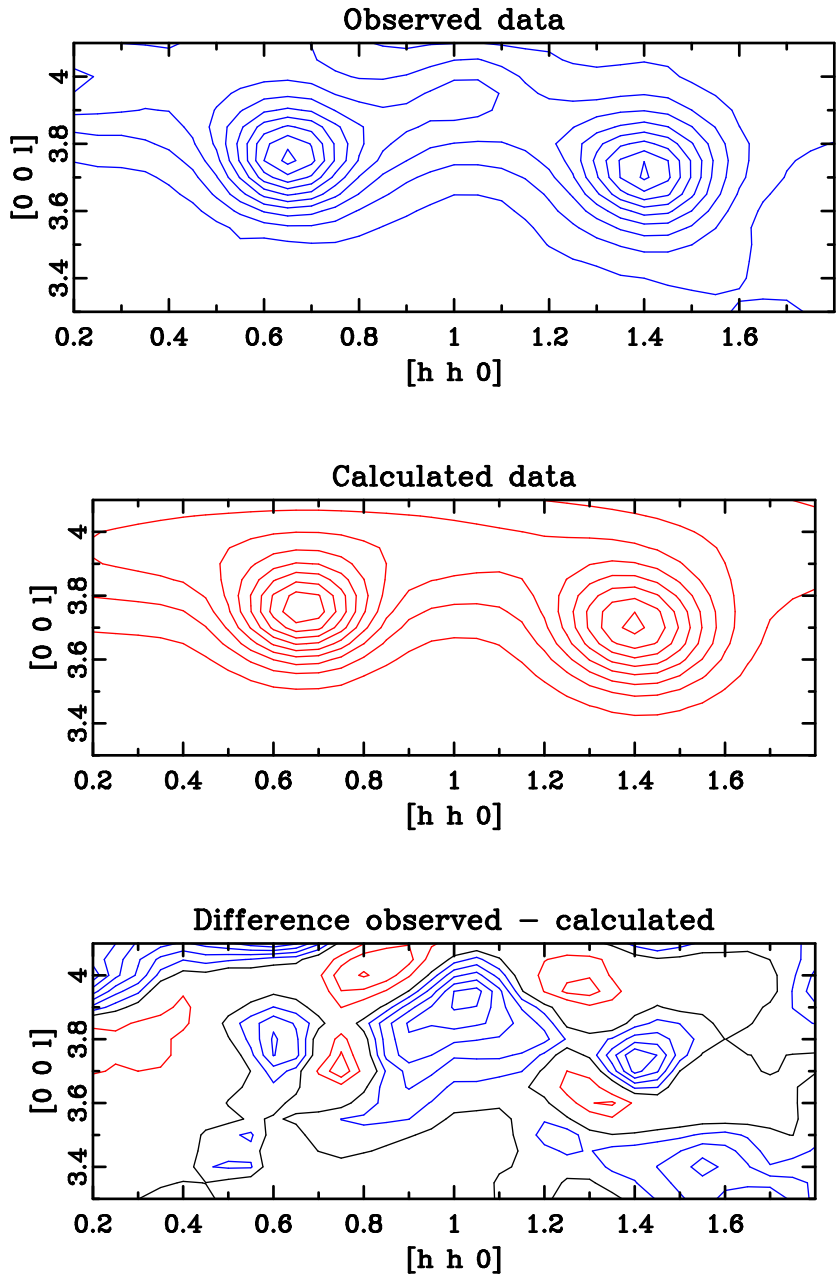


Figure 8.2: Fit results for 2D example

First we enter the fit sublevel (line 1) to fit the theory function to data set one. Before we select three 2D Gaussian theory functions in line 4 we set the window size for determination for maxima used as starting values to 3 data points. This will result in the three maxima marked in Figure 8.2. Line 5 displays the current parameter settings for checking. Next (line 7) the fit is started. As default the parameter $p_{6,n}$ determining the angle between the principle axes of the Gaussian and the coordinate axes and the asymmetry parameters $p_{7,n}$ and $p_{8,n}$ will be kept fixed. After the first fit, the orientation of the Gaussian (parameter $p_{6,n}$) for Gaussians one and three (the two strong peaks) are refined as well (lines 9–12). Finally the data sets containing the calculated data and the difference as well as the resulting parameters are saved. When fitting interactively the command 'plot' within the fit sublevel will plot the current fit on the screen.

Finally we want to have a look at the produced text file containing the fit results (file extension *.erg*) for this example fit. The line numbers are given for reference and as for the macro files not part of the actual output file. The first part contains general information about the fit such as a title (lines 2–3), the used theory function (line 4), the data file used (line 5), the number of data points and parameters (lines 6–7). Note that this is the total number of parameters including those not actually refined. Lines 8–10 show the current URF (see 8.1), the maximum number of iterations, the setting for maxima determinations and the used weighting scheme, in our case $w_i = \frac{1}{z_i}$. The next block of data in the output file show details of the fitting process itself. Line 14 lists the value for $\sum(z_i - z(x_i, y_i, p))^2$ for the last and the previous iteration step. The difference between both is shown in the next line 15. A big difference normally indicates a 'crashed' fit. The value of URF changes during the refinement and the final value is shown in line 16. If this value is not close to 1.0 at the end of the fit something has gone wrong. The next line shows the resulting R value and the expected R value. The ration of these values is also known as the goodness-of-fit. Finally all correlations between fitted parameters that are greater than 0.8 are listed. We are lucky because our fit contains no highly correlated variables.

```
 1   General fit parameter settings :
 2     Title            : KUPLOT 2D fit demonstration
 3                        January 1998
 4     Fit function     : Gaussian (2D)
 5     Data file name   : test.fit
 6     # of data pts.   :    561
 7     # of parameters  :     25
 8     Urf              :     .1000
 9     Max. cycle       :     30
10     MFEN for maxima  :      3
11     Weighting scheme : w(i) = 1.0/i
12
13   Information about the fit :
14     Sum n-1    :  1038.98                Sum n   :  1040.29
15     Difference :  1.31653
16     Urf final  :  1.00047
17     R4 value   :  .106378                R exp   :  .077327
18
19   Correlations larger than 0.8 :
```

```
20     ** none **
21
```

All remaining information of the output file are results of the parameters used in the theory function. In line 22 the number and type of functions fitted are shown. Next we have the result for the overall background p_1 and its standard deviation (line 24). The 'pinc' in the last column indicates whether the parameter was refined (pinc = 1) or kept fixed to its starting value (pinc = 0). The values for each parameter and the corresponding 'pinc' value can be altered via the command 'par'.

```
22   Fitted  3 Gaussian(s) :
23
24     p( 1) : backgr. 1 :  69.2818     +-  .998061    pinc : 1.
25
```

The rest of the output file contains three identical blocks with the resulting parameters of the three Gaussians fitted to the observed data. We show here just the results for Gaussian 1 (left maximum).

```
26   Gaussian :   1
27
28     p( 2) : peak      :  407.332     +-  11.3879    pinc : 1.
29     p( 3) : position x:  .665616     +-  .002372    pinc : 1.
30     p( 4) : position y:  3.74364     +-  .002798    pinc : 1.
31     p( 5) : fwhm a    :  .237793     +-  .005545    pinc : 1.
32     p( 6) : fwhm b    :  .235389     +-  .005938    pinc : 1.
33     p( 7) : angle a,x : -98.8801     +-  85.1452    pinc : 1.
34     p( 8) : asym. a   :  1.00000     +-  .000000    pinc : 0.
35     p( 9) : asym. b   :  1.00000     +-  .000000    pinc : 0.
36             integral  :  25.8344     +-  1.97653
```

Note that the asymmetry parameters were not refined, thus kept at the default setting for a symmetric function. The orientation for this Gaussian (line 7) has a very large error which can be understood from the resulting half width parameters (lines 31–32) which show a nearly isotropic result. Thus the definition of an orientation is quite arbitrary. This result is displayed here for demonstration purposes, normally the fit needs to be repeated with a fixed orientation parameter for this Gaussian. The third Gaussian however (right maxima) has a significant orientation parameter of $p_{6,n} = 33(8)^\circ$. Finally (line 36) the integral for each Gaussian with its standard deviation is given.

8.3 User defined fit functions

The final section of this chapter will demonstrate the use of an arbitrary fit function. The fit process is not different from the example described above, just the theory function is entered as *KUPLOT* type of expression. Assuming your theory function should be

$$y(x_i, p) = p_1 \cdot \exp(-p_2 x_i) \cdot \sin(x_i + p_3), \tag{8.6}$$

the corresponding command to define this theory function in the fit sublevel of *KUPLOT* would look like this

```
func fx,3,p[1]*exp(-p[2]*r[0])*sin(r[0]+p[3])
```

The parameter 'fx' stands for a user defined function. The next parameter is the number of fit parameters to be used for that function. As last parameter, the function itself is given. The variable r[0] stands for x_i similar to the command 'func' (see section 7.1). The fit parameters are represented by variables p[i]. Since the nature of the fit function is unknown, there is no calculation of default starting values. They have to be supplied by the user via the command 'par'. Note that *KUPLOT* is calculating the expression once after the 'func' command is given to perform a syntax check. If no proper starting values for p[i] have been defined this calculation might fail, e.g. by a division by zero. After the function is successfully defined, the fit process proceeds as described in the previous sections. The definition of a 2D theory function works exactly the same way, use r[1] for the values of y_i. The dimension of the user defined theory function (1D or 2D) is given by the data set used for the fit.

The derivatives needed for the least square refinement are calculated numerically in contrast to the usage of analytical derivatives used for the build in theory functions. In extreme cases the calculation of these derivatives and subsequently the fit itself might fail. In those cases it might be necessary to renormalize the observed values or to use a specialized fit package after all.

Appendix A

List of commands

In this appendix, a list of all current *KUPLOT* commands is given. Note that commands marked with an asterix '*' branch to a sublevel of *KUPLOT* where further commands not included in these lists are used. Detailed descriptions of **all** commands can be found in the command reference manual and the online help of the program.

A.1 Alphabetical list of commands

Command	**Description**
#	Comment, the rest of the line will be ignored
@	Execution of a macrofile
=	Assigns the value of an expression to a variable
achx/y/z	Changes label for x-, y- or z-axis
alloc	Allocates space for new data set without reading a file
angl	Sets the angle between x- and y-axis
aver	Sets the aspect ratio between the y- and the x-axis
break	Interrupts a loop or conditional statement
buff	Sets buffer space around plot for labels etc.
ccal	Mathematical manipulation of a data set
cmap	Selects colourmap for bitmap plots
continue	Continues *KUPLOT* after 'stop' command
do	Start of a do loop
echo	Echos a string
else	Default block in an if construction
elseif	Alternative block in an if construction
enddo	End of a do loop
endif	End of an if construction

etyp		Selects the type of error bars to be plotted
eval		Evaluates an expression for interactive display
exit		Ends the *KUPLOT* program
fclose		Closes i/o file (**new**)
fit	*	Enters least square fit sublevel
fget		Gets data from file (**new**)
fnam		Toggles the plotting of the filenames within the plot
font		Sets font, font size and font colour
fopen		Opens file for 'fget' or 'fput' (**new**)
fset		Sets type of frame and axis labeling
frames	*	Check this help entry for summary on the usage of frames
fput		Writes user variables to file (**new**)
func		Create data set from given functional expression
glat		Smoothing of data sets (German: glätten)
grid		Toggles plotting of a grid at major tick mark positions
hart		Defines plotting mode for 2D data (contours, bitmap)
hcol		Sets colour of contour lines
help		Gives on line help
hlab		Enables contour line labelling (**new**)
hlin		Sets contour line starting level and intervals
hpak		Sets number of contour line definitions
htyp		Sets line type for contour lines
if		Begin of an if construction
inte		Calculates integral of specified data set
ksav	*	Sublevel to save loaded data sets
kcal		Mathematical operations between data sets
lart		Sets line plotting style (line, steps, spline)
lcol		Sets line colour
learn		Starts a learn sequence
lend		Ends a learn sequence
load		Reads data set from file
ltyp		Sets line type
lwid		Sets line width
mark		Sets interval and starting value for axis labeling
match		Matches two data sets via scale and offset (**new**)
mcol		Sets marker colour
mean		Calculates average and standard deviation of specified data set
merge		Merges several data sets (**new**)
mouse		Enters 'mouse' input mode (**new**)

mtyp	Sets marker type
msiz	Sets marker size
orient	Sets orientation of the plot (landscape, portrait)
plot	Display plot on the screen
prin	Print current plot
ptyp	Sets type for peak markers
rdef	Read *KUPLOT* settings from file
rebin	Rebins data to a new grid (**new**)
rese	Reset *KUPLOT*
sann	Define annotations for plot
save	Save current plot
sdef	Write current *KUPLOT* settings to file
seed	Initialize the random number generator
set	Set various parameters
show	Shows various *KUPLOT* settings
skal	Defines plot area
sleg	Define optional caption for given data set
smax	Determine maxima of specified data set
smooth	Smoothes data (weighted smoothing) (**new**)
sort	Sort specified data set with respect to its x-values
stop	Stops *KUPLOT* macro, resume with 'cont'
system	Executes a shell command
tit1/2	Sets first/second title line
wait	Wait for user input

A.2 Functional list of commands

Program control

Command	**Description**
@	Execution of a macrofile
=	Assigns the value of an expression to a variable
break	Interrupts a loop or conditional statement
continue	Continues *KUPLOT* after 'stop' command
do	Start of a do loop
else	Default block in an if construction
elseif	Alternative block in an if construction
enddo	End of a do loop

endif	End of an if construction
exit	Ends the *KUPLOT* program
if	Begin of an if construction
learn	Starts a learn sequence
lend	Ends a learn sequence
rdef	Read *KUPLOT* settings from file
rese	Reset *KUPLOT*
sdef	Write current *KUPLOT* settings to file
seed	Initialize the random number generator
set	Set various parameters
stop	Stops *KUPLOT* macro, resume with 'cont'
system	Executes a shell command
wait	Wait for user input

Changing plot

Command		**Description**
achx/y/z		Changes label for x-, y- or z-axis
angl		Sets the angle between x- and y-axis
aver		Sets the aspect ratio between the y- and the x-axis
buff		Sets buffer space around plot for labels etc.
fnam		Toggles the plotting of the filenames within the plot
font		Sets font, font size and font colour
fset		Sets type of frame and axis labeling
frames	*	Check this help entry for summary on the usage of frames
grid		Toggles plotting of a grid at major tick mark positions
mark		Sets interval and starting value for axis labeling
mass		Sets plot size with respect to its the world coordinates
orient		Sets orientation of the plot (landscape, portrait)
sann		Define annotations for plot
skal		Defines plot area
sleg		Define optional caption for given data set
tit1/2		Sets first/second title line

Changing appearance 1D data

Command	**Description**
etyp	Selects the type of error bars to be plotted

lart	Sets line plotting style (line, steps, spline)
lcol	Sets line colour
ltyp	Sets line type
lwid	Sets line width
mcol	Sets marker colour
mtyp	Sets marker type
msiz	Sets marker size
ptyp	Sets type for peak markers

Changing appearance 2D data

Command	**Description**
cmap	Selects colourmap for bitmap plots
hart	Defines plotting mode for 2D data (contours, bitmap)
hcol	Sets colour of contour lines
hlab	Enables contour line labelling (**new**)
hlin	Sets contour line starting level and intervals
hpak	Sets number of contour line definitions
htyp	Sets line type for contour lines
lart	Sets line plotting style (line, steps, spline)
lwid	Sets line width

Data analysis and manipulation

Command		**Description**
ccal		Mathematical manipulation of a data set
eval		Evaluates an expression for interactive display
fit	*	Enters least square fit sublevel
func		Create data set from given functional expression
glat		Smoothing of data sets (German: glätten)
inte		Calculates integral of specified data set
kcal		Mathematical operations between data sets
match		Matches two data sets via scale and offset (**new**)
mean		Calculates average and standard deviation of specified data set
merge		Merges several data sets (**new**)
mouse		Enters ’mouse’ input mode (**new**)
smax		Determine maxima of specified data set
sort		Sort specified data set with respect to its x-values

rebin		Rebins data to a new grid (**new**)
smooth		Smoothes data (weighted smoothing) (**new**)

Input and Output

Command		**Description**
alloc		Allocates space for new data set without reading a file
fclose		Closes i/o file (**new**)
fget		Gets data from file (**new**)
fopen		Opens file for 'fget' or 'fput' (**new**)
fput		Writes user variables to file (**new**)
ksav	*	Sublevel to save loaded data sets
plot		Display plot on the screen
prin		Print current plot
save		Save current plot to a file
show		Shows various *KUPLOT* settings

Appendix B

Installation

In this appendix we will briefly describe the installation process of the program *KUPLOT*. *KUPLOT* is part of the *Diffuse* package which includes the programs *PDFFIT* and *DISCUS* as well as *KUPLOT*. Refer to the section corresponding to your operating system for installation information.

Windows

The Windows version of the *Diffuse* is distributed as a self-extracting installer. Simply download the file `Diffuse-4.1-win32.exe` (or the most current version) and run it by double clicking on the downloaded file. This will start the installation process and the dialog shown in Figure B.1 will appear. Simply follow the instructions on the screen and that is all, you are ready to use and of the programs that are part of the *Diffuse* package.

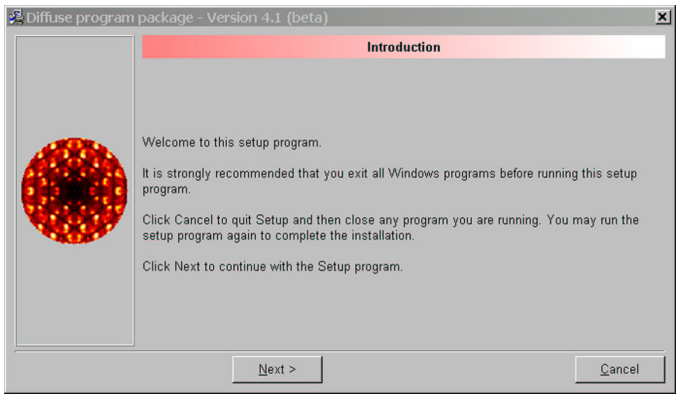


Figure B.1: Windows installer for *Diffuse*.

Unix / Linux

First one needs to obtain the complete 'diffuse' archive named `Diffuse-4.1.tar.gz` or the most recent version. Next you need to unpack the archive using the commands

```
gzip -d Diffuse-4.1.tar.gz
tar -xvof Diffuse-4.1.tar
```

This will create a directory 'diffuse' containing the distribution. Within this directory there are separate directories `discus`, `kuplot` and `discus` containing the three different programs as well as a directory `lib_f77` which contains command language related routines common to all three programs. In each program directory, you will find the following directory structure:

```
./prog     : Contains the source code
./doc      : Contains POSTSCRIPT version of the documentation
./tutorial : Contains the tutorial for the program
```

After proceeding to the `kuplot/prog` directory you need to choose a `Makefile` that suits your needs and customize the `Makefile` as well as one configuration file '`config.inc`'. Next copy the UNIX style Makefile using the command

```
cp Makefile.unix Makefile
```

This version of *KUPLOT* requires the graphics library *PGPLOT 5.2* to be installed on your system. The library can be obtained free of charge from the World Wide Web at *http://astro.caltech.edu/~tjp/pgplot/*. As default the Makefile uses the variable PGPLOT_DIR to locate the library. Make sure the variable is set properly or adjust the location directly in the Makefile. Next the program is compiled and linked by executing the command `make` followed by `make install` if all went well. Make sure an appropriate path for the binaries to be installed is set in the Makefile. After that you can perform a `make clean` to remove the binary and the object files from the source directory. If you want to install the program manually you have to put the files '`kuplot`' and '`kuplot.hlp`' in the same directory.

Before you actually can use the online help of the program, an environment variable `KUPLOT` has to be set to the path where the program is installed in. This can be done e.g. in the `.login` or `.cshrc` file using the command `setenv KUPLOT /path/to/kuplot` for the `csh` of `set KUPLOT=/path/to/kuplot; export KUPLOT` is you are using the bourne shell. If this path is also included in your search path you can start the program simply by entering `kuplot`.

The progam *KUPLOT* can also be installed on a DEC VMS machine. Tools like `gzip` and `tar` are freely available from the internet. The file `Makefile.vax` is a compilation script that is part of the distribution.

Please register

Please register yourself as a user of one or more programs of the *Diffuse* program package. In the top directory of the distribution you will find the file `REGISTER`. Please fill in the corresponding information and send the file via email to *proffen@pa.msu.edu*. Thank you for registering, a feedback is always a strong

motivation to proceed with the program development and making it available to the public. By registering you also enable us to inform you of program updates, workshops and other related topics.