Tv
User–Manual

Andreas Fitzler[1]
Institute for Nuclear Physics
University of Cologne

July 13, 2000

[1]e–mail: fitz@ikp.uni-koeln.de

Contents

List of Figures

List of Tables

Preface

Introduction

Tv is a spectra– and matrix–analysis program which runs under the operating system UNIX and has been successfully tested under SUNOS 5.5.1, Ultrix 4.3A, HP-UX 9.01 and Linux. This manual documents the use and simple customization of the program. Its intention is to teach the beginner all basic functions of the program and as a reference manual for the advanced user. This documentation describes the program as it and not the physical background. Only now and then fundamental vocabulary from the analyzing of nuclear physics data is explained and a little theory of the form of spectra is given.

   This manual is available both as postscript file and as hypertext. The URL of the hypertext–form is:

   http://ikp.uni–koeln.de/˜fitz/Tv_user-manual/Tv_user-manual.html.
You can get a postscript version of the manual from this site, too.

Organisation of the book

Chapters 1 to 3 are recommended to be read by beginners at starting their work with Tv. Chapters 4 and 5 are intended as introduction to the beginner and as reference guide to the more sophisticated user respectively. Chapter 6 is a reference guide to all commands of Tv in alphabetical order. A more detailed description of the contents of the chapters is given in the following list:

Chapter 1 Describes the typographic conventions used to emphasize certain parts of the text, as there are for example commands and filenames.

Chapter 2 Gives an introduction to the main topics of Tv with examples to each item. The basic commands are explained in their most simple form and you will be able to use the program to do simple evaluation after reading this chapter.

Chapter 3 Describes the basic usage of Tv starting from commandline arguments, windows and input modes via the usage of the mouse and the command– and menu–concept to the defining of hotkeys, writing of command files and sending commands from outer space.

Chapter 4 Shows how to load one or more spectra and perform all necessary operations to analyze the data, as there are choosing of the best window type, calibration, recalibration, normalization, arithmetic operations, integration and fit as well as peaksearch.

Chapter 5 Explains all necessary operations to work with matrices, cuts, projections, spectra and buffers.

Chapter 6 Shows a detailed list of all Tv–commands. This chapter is addressed
to advanced users who want to know all the mysteries and secrets. It lists
all commands in alphabetical order and gives a short description for each of
them.

Copying

Before copying the program Tv you must obtain an appropriate license from the
authors. To get a license, fill in the attached **License Agreement for Tv** (see
appendix E p. 107), sign it, and send the original and one copy to the authors at the
address found in the agreement. No fee is charged for licensing Tv. All registered
users will be informed about new releases of Tv.

Chapter 1

Syntax in this manual

1.1   Introduction

To increase the legibility of this manual, various font styles are used. In this chapter the meaning of these typographic conventions is explained.

1.2   Commands

Tv commands are printed in a **boldfaced sans serife font** and formatted in the following manner:

    menu> **command** **<argument1>** **[<argument2>]** **<argument3 . . . >**

    menu> **command** **{<argument4>** **|** **<argument5>}**

menu> is the prompt indicating that Tv is accepting input in the current menu. The command can be entered complete or in short form as explained later (see section 3.7 p. 20). Arguments enclosed in brackets [ ] are optional. Three dots after the argument name indicate that a list of arguments of this type may follow. From arguments enclosed in braces { } exactly one must be used. The vertical line ('|') in the example above means that you can specify argument4 or argument5 (exclusive 'or').

1.3   Command–Shortcuts — Hotkeys

Frequently used commands can be executed with one or more keystrokes in cursor mode (see section 3.4.2 p. 18). These hotkeys are shown in the manual in a box, like:

    $\boxed{\texttt{keycode}}$

For example, the hotkey which displays the full spectrum is:

    $\boxed{\texttt{f}}$

Since there is only a limited number of characters on your keyboard, combinations of keys have been introduced to increase the number of abbreviable commands. You must press each character in such a combination. For example to plot a spectrum in xfig–format you have to press the keys $\boxed{\text{P}}$ and $\boxed{\text{x}}$ one after the other which is written in the manual like:

    $\boxed{\text{P}}\,\boxed{\text{x}}$

The default hotkeys are printed in appendix A on page 83.

1.4   Program output

Terminal–output from Tv is printed in `typewriter font` like in the following way:

```
spectrum ./co56b68.127'8k.le4:1 read to #0
```

1.5   Filenames

Filenames are printed in *italics* like:

site.tvrc

Chapter 2

Introduction

Contents

2.1 Introduction

This chapter is intended for the novice user to get an overview of the main topics of Tv and to describe its basic operations. It is started from the commandline with:

> **tv&**

but since Tv writes its output so standard out, you better enter:

> **tv > /dev/null 2>&1 &**

Two windows appear on your screen which are a **text window (named tv)** for echoing commands and the displaying of results and status information and a **graphic window (named tv–root)** for displaying the spectra you are working with. Both window types are described in detail in section 3.3 on page 16 and displayed in figure 3.2 on page 18 and figure 3.1 on page 16.

2.2 Input modes

Tv supports two major modes for entering commands, called **edit–mode** and **cursor–mode**.

By default Tv starts in edit–mode where input is focussed to the text window which means that all keys you type are printed in the text window and will be evaluated after pressing $\boxed{\text{CR}}$. As long as the cursor is placed over the text window you are in edit–mode.

In cursor–mode mode entered keys are used as hotkeys (see section 3.4.2 p. 18) when the cursor is placed over the graphic window (the cursorform is pirate ☠ then).

To switch between edit– and cursor–mode place the cursor over the graphic window and press $\boxed{\text{Esc}}$. Most operations in this introduction are performed in cursor–mode.

2.3   Spectrum

2.3.1   Loading a single spectrum

In order to examine a single γ–spectrum switch to the cursor mode and press $\boxed{\text{g}}$ to **load a spectrum**. You will be queried the name of the spectrum file. Enter its name and press $\boxed{\text{CR}}$. The spectrum will be displayed in the tv–root window (see figure 2.1).
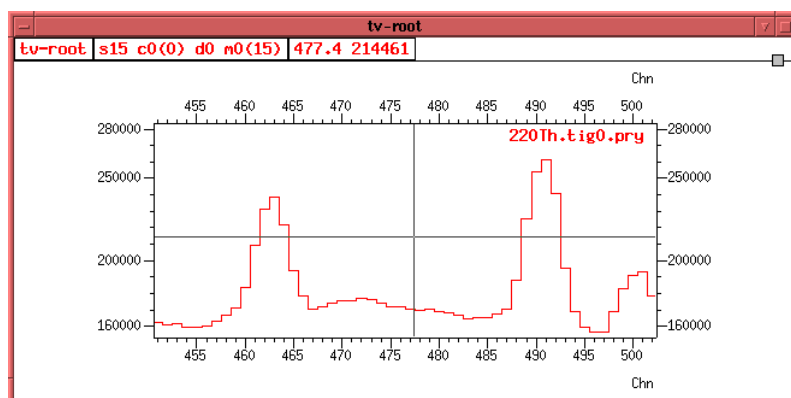
Figure 2.1: Graphic–display window with a ^{56}Co spectrum loaded.

Tv uses the program library **mfile for spectrum I/O**. Spectra written in mfile format are preceded by a header with information about the spectrum format. If you want to read spectra not written in mfile format, the mfile routines try to guess the format. If this fails you have to explicitly specify the spectrum format, for example to read a spectrum with 8192 channels that was written in low endian format with 4 bytes you enter:

    tv> s co56b68.127'8k.le4:1

In the example above **s** stands for   **spectrum**. See section 3.7 on page 20 for a description of abbreviating commands. A detailed desription of spectra formats is given in appendix C.2 on page 101.

2.3.2   Moving through the spectrum

You can determine the x– and y–position of the cursor from the status–window in the graphic–window. To **zoom** a part of the spectrum set two markers by pressing $\boxed{\text{Spacebar}}$ for each over the desired position and press $\boxed{\text{e}}$ to expand. You also can use keys $\boxed{1}$ to $\boxed{9}$ to zoom in and $\boxed{0}$ or $\boxed{-}$ ($\boxed{1}$ to $\boxed{9}$) to zoom out with the cursor defining the center. To **move the viewport** to the left (or right) press $\boxed{<}$ (or $\boxed{>}$) which automatically adjusts the y–scale according to the peaks displayed. To move the viewport without adjusting the y–scale use $\boxed{,}$ (or $\boxed{.}$). Alternatively you can press $\boxed{\text{i}}$ to **jump to a certain position** you will be queried. This position is the energy for calibrated spectra which is equal to the channels for uncalibrated spectra. The full spectrum is displayed by pressing $\boxed{\text{f}}$.

The **mouse** can be used to zoom the spectra and move through them, too as is explained in section 3.5 on page 19.

2.3.3 Fitting a spectrum

The easiest way to determine (or **fit**; see 4.12 p. 38) the position, width and volume of a peak is to use the **quickfit** feature of Tv. Place the cursor over the peak to be fitted and press Q . The result of this operation is shown in figure 2.2 on page 9 and the **short status** output is printed below:

```
fit of spectrum #0 './co56b68.127'
spectrum #0:  ./co56b68.127 [Chn]
#0 pos 1501.0690(66) wdt 2.883(14) vol 63636(324)
```

The last line of the result output shows fitted position, width and volume of the peak. In parentheses the errors of fitted values are given.

The quickfit guesses position and fit–region of the peak which leads to trouble if you have to fit merged peaks. They are usually fitted as one peak by the quickfit. Furthermore it does not consider background–regions which leads to a misestimation of the subtracted background.

As a consequence of the last paragraph you have to define **fit– and background– regions** as well as **peak markers** to obtain best fit results. You have to define the fit–region, i.e. the region of the spectrum containing the peak(s) to be fitted. Press r to define the left border of the fit–region and press r again for the right border of the fit–region. Now mark all peaks to be fitted inside the fit–region by pressing p over each peak. Then you can **perform the fit** by pressing F .

Optionally you can further improve the fit by defining an arbitrary number of **background–regions** using the hotkey b once for left border and once for right border of each background–region. Press B to fit spectral background inside the background–regions and F to fit peak(s). The background fitted to the background–regions will be used for the fit inside the fit–region. Note, though Tv supports only one fit–region, any number of background–regions can be defined for the fit. Figure 2.2 shows a spectrum with fit.
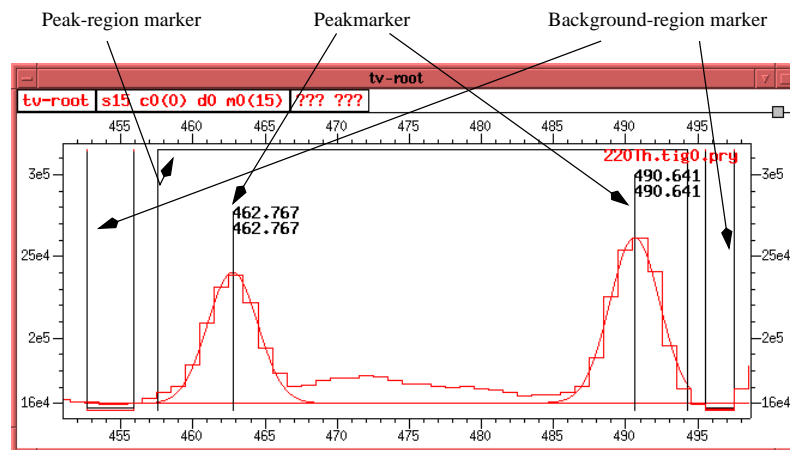
Figure 2.2: Graphic–display window with a ^{220}Th projection spectrum loaded and a fit attached. You can see the fit–region surrounding the peaks which have peak markers on them, as well as two background–regions with the fitted background–function. Moreover the fit–function is displayed.

To remove either a quickfit or a normal fit use the hotkey - F . This will remove the region markers from the graphic window and delete the fit. You have to do this before you can fit another peak.

2.3.4   Calibrating a spectrum

To identify peaks with transition energies from a decay you need a relation between
the channels of your spectrum and their corresponding energies (**calibration**). To
calibrate a spectrum, you need to identify two peaks in the spectrum or a calibration
from a calibration spectrum. The energies of these positions can be determined from
a calibration spectrum. If you know two channels and their corresponding energies,
you can define the calibration with the command:

    tv> calibration position enter <**chn**$_1$ > <**energy**$_1$ > <**chn**$_2$ > <**energy**$_2$ >

If you have a calibration in a file you can load it. To load a calibration you can
either use the hotkey $\boxed{\text{E}}$ to load the default calibration file *.cal or you switch to
the text–mode and enter the full command:

    tv> calibration position read <**calfile**>

You can abbreviate this command with:

    tv> c p r <**calfile**>

2.3.5   Plot

Plotting with Tv may not lead to sufficient results at the first attempt. Probably
you will miss the inscription at the y–axis. The reason is that the frame width for
left and right frames is set to 3 by default (see section 3.3 p. 16). There are two
ways to get sufficient plots.

The first way is to create a window with the name plot which has frame widths
of 9 for the left and right border. To create the window enter the command:

    tv> window create simple plot

Another way is to change the frame widths of your window. You can do this with
the command:

    tv> window setup frame width <**l**> <**r**> <**b**> <**t**>

The most convenient format to plot your figure is the xfig–format because you can
process and convert plot files with the program **xfig**. The plot is performed by
pressing $\boxed{\text{P}}\,\boxed{\text{x}}$. Alternatively you can use the unix–format by pressing $\boxed{\text{P}}\,\boxed{\text{u}}$. In
both cases you will be queried a filename. $\boxed{\text{CR}}$ at the filename prompt forces the
standard filename for plots to be taken, this is composed from the spectrum name
and a suffix defined by the wildcard (see section 3.12 p. 23) for plots which is .fig
by default.

A sample plot exported to ps via xfig is shown in figure 2.3 on page 11.

2.4   Analyzing a $\gamma\gamma$–matrix

2.4.1   Opening a matrix

To analyze a $\gamma\gamma$–matrix you need a matrix written in mfile format [1]. The default
extension of the matrix filename is .mtx. Furthermore you need the projection [2]
on the x– or y–axis with the default extensions .prx or .pry. All three files must be
located in a directory with the same basename as the matrix, e.g. see table 2.1 on
page 11:

To open the matrix for analysis use the hotkey $\boxed{\text{M}}$ and you will be queried the
name of the directory containing the matrix.

---

[1] To convert a matrix to an appropriate format use the program matconv

[2] The projection can be created using the program matproj.
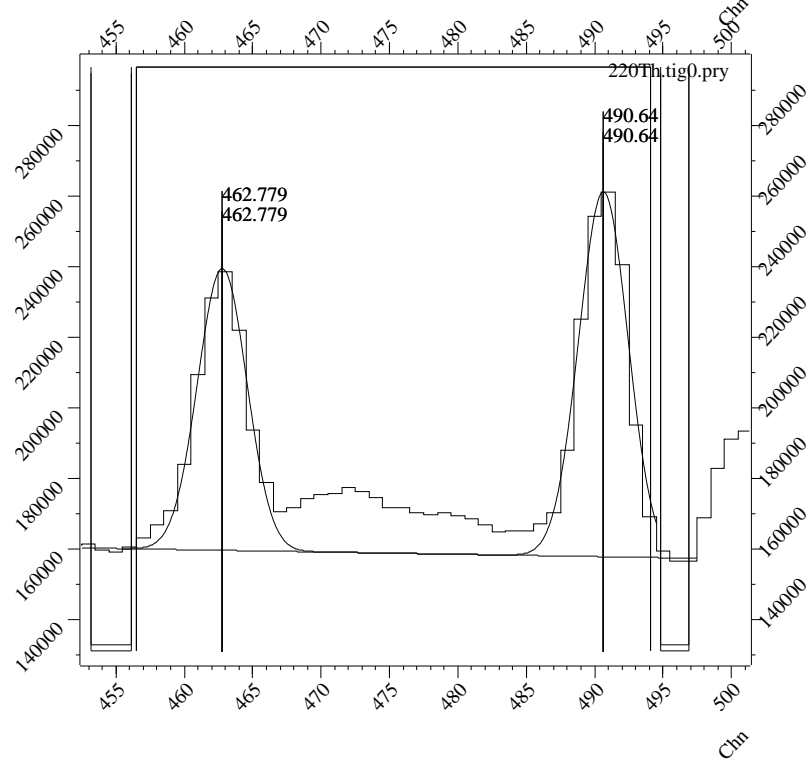
Figure 2.3: Sample plot from Tv, plotted in xfig format and exported to ps.

Matrix directory	220Th
Matrix file	220Th/220Th.mtx
Matrix projection	220Th/220Th.pr[xy]

Table 2.1: Example for a collection of necessary files to examine a γγ–matrix. You need either the projection to the x–axis or y–axis.

2.4.2   Cutting in a matrix

A $\gamma\gamma$–matrix is a twodimensional object which contents are coincident data from an experiment. To examine coincident data you can **cut** out a slice of the matrix which gives you a **cut spectrum**. To actually create a cut spectrum (see chapter 5 p. 47) you must define a **cut gate** using the hotkey $\boxed{c}$. Similar to the definition of the fit–region you define the left and right border of the cut gate by pressing $\boxed{c}$ once for each border. Further gates defined with $\boxed{c}$ are automatically interpreted as **background gates**. If you want additional cut gates, use the hotkey $\boxed{G}\,\boxed{G}$ to define them. Pressing $\boxed{C}$ performs the cut and the cut–spectrum will be displayed in the window. You can delete your cut and background gates by pressing $\boxed{-}\,\boxed{C}$. A gate will then be marked in your projection as shown in figure 2.4 on page 12.
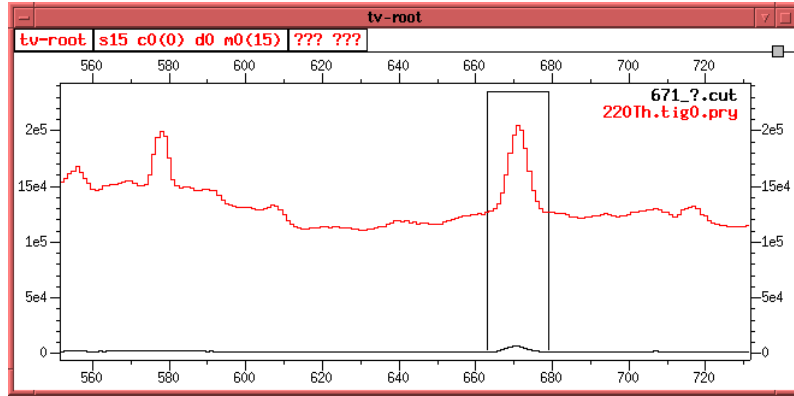
Figure 2.4: Cut gate in the projection of a ^{220}Th spectrum. The peak at channel 671 is marked to find its coincident transitions. The spectrum at the bottom is the **cut spectrum** for channel 671 which has less statistic. Besides you can see that channel 671 is autocoincident.

With a view to greater clarity it is recommended to create the so called **cut–window**. It displays the projection and cut in one window. You can create it with the command:

    tv> **window create cut testcut**

Compare figure 2.4 on page 12 and figure 2.5 on page 13 to see the advantages of the cut–window.
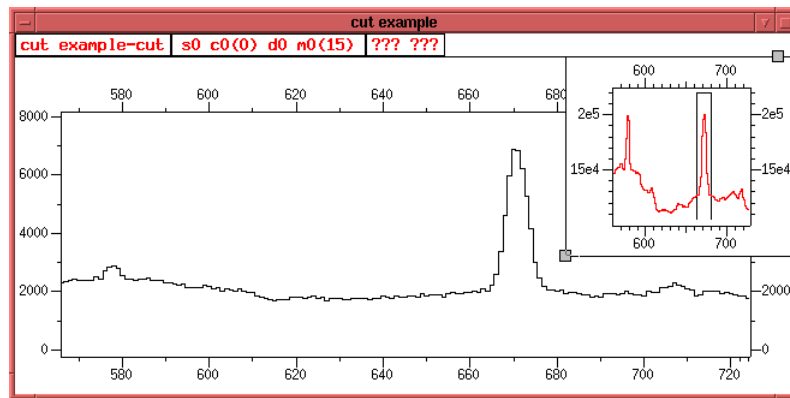
Figure 2.5: The cut in its special window. In the upper right corner the projection is displayed.

Chapter 3

Basic usage

Contents

3.1   Introduction

This chapter describes the user port of Tv. It introduces Tv's basic usage from the commandline arguments, windows and input modes via the usage of the mouse and the command– and menu–concept to the defining of hotkeys, writing of command files and sending commands from outer space.

3.2   Starting Tv

You can start Tv with a number of command–line options which are described in the following section. The usage is:

   tv [-? | -help] [-shm] [-src] [-rc] [-v] [-s <ns>] [-e <string>]
Where:

-? | **-help:** prints a list of allowed command–line options

-shm: Tv performs operations on spectra from shared memory

-src: the evaluation of the systemwide setupfile *site.tvrc* is omitted

-rc: the evaluation of the personal setupfile *~/.tvrc* is omitted

-v: information about the version is printed

-s <ns>: sets the number of spectrabuffer from the default value of 16 to <ns>

-e <string>: interprets the string and executes the commands

3.3   The windows

Two windows appear at your display when you start the program. Their titles are
tv and **tv–root**.

3.3.1   The graphics–window

tv–root (see figure 3.1 p. 16) is the main graphic–window. By default all spectra
loaded are displayed in this window.
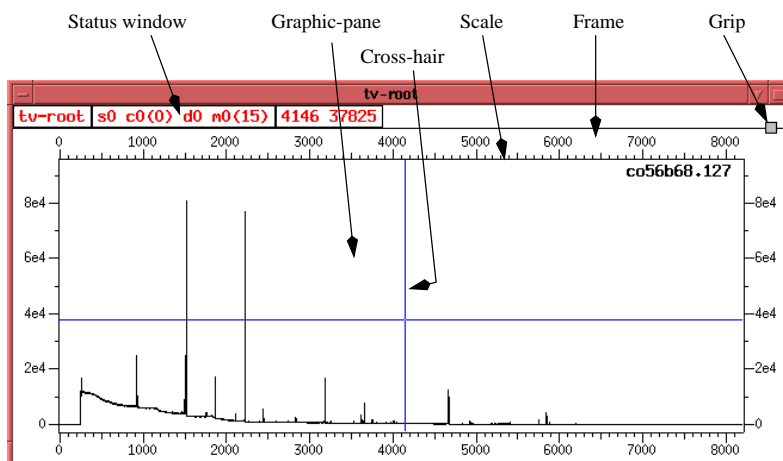
Figure 3.1: Graphic–display window immediately after startup with a ^{56}Co spectrum
loaded.

The graphic–window is composed of two subwindows.

The **status–window** displays the window name as well as the buffernumbers
of the active spectrum, cut, directory and matrix. In parentheses the buffernumber
for the spectrum where the cut is saved is printed as well as the buffernumber of the
projection attached to the active matrix. In the case that a spectrum is loaded the
x– and y–position are displayed, otherwise questionmarks are printed at the place
of the coordinates. In windows with more then one pane (see section 4.2.1 p. 28)
the name of the currently active pane is displayed.

The lower subwindow or subwindows called **graphic–pane(s)** display the spec-
tra and show their according names which are printed in the upper right corner.
In this window you can perform operations with the mouse as described in section
3.5 on page 19. The graphic–pane is surrounded by **scales** at all sides which show

the counts at the left and right side (**y–scale**) as well as the x–position in channels at the top and energy (see section 4.8 p. 34) at the bottom (**x–scale**) if an energy calibration is loaded. The scales are positioned in the **frame**. The **grip** is used to move the line between the status–window and the top frame.

You can switch the y–scale between **linear, logarithmic and squared scale** with the aliases:

    `tv>` **lin**

    `tv>` **log**

    `tv>` **qua**

You can change the **viewport** (this is the visible part of the spectra) in many ways with the hotkeys listed in table 3.1 on page 17.

Key	Tv–commands	Function
Spacebar	tv> window mark vertical enter cursor;	Set vertical markers to define x–range to expand
- 1 — - 9	tv> window view full y; tv> window view stretch (-1 to -9) cursor 0;	Zoom out see appendix A on page 83
,	tv> window view full y; tv> window view shift -70.0 0.0;	Move the viewport to the left by 70% of the visible channels
.	tv> window view full y; tv> window view shift 70.0 0.0;	Move the viewport to the right by 70% of the visible channels
0	tv> window view full y; tv> window view stretch -1 cursor 0;	Zoom out
1 — 9	tv> window view full y; tv> window view stretch (1 to 9) cursor 0;	Zoom in see appendix A on page 83
<	tv> window view full y; tv> window view shift -70.0 0.0;	Move the cursor to the left by 70% of the visible channels
>	tv> window view full y; tv> window view shift 70.0 0.0;	Move the cursor to the right by 70% of the visible channels
e	tv> window view expand;	Expand viewport to the x–range between markers set with Spacebar
f	tv> window view full both; tv> window view expand;	Expand visible part to full size
x	tv> window view full x; tv> window view expand;	Expand viewport to full size in x–direction
y	tv> window view full y; tv> window view expand;	Expand viewport to full size in y–direction
|	tv> window view full y; tv> window view center x cursor;	Center the spectra around position defined by cursor

Table 3.1: Hotkeys to change the viewport of the graphic–pane.

Some of these operations can be executed as described in section 3.5 on page 19.

3.3.2 The text–window

tv (figure 3.2 p. 18) is a text window and composed of three subwindows.

The **bufferlist–window** shows the buffernumbers and the names of the according spectra.

Scrollbar   Tvkeys-window   Bufferlist-window   Grip
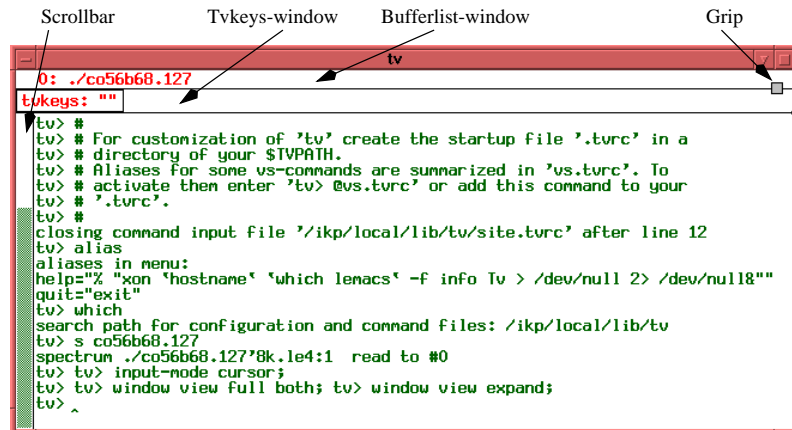
Figure 3.2: Text–display immediately after startup. The bufferlist–window shows a ^{56}Co spectrum loaded to buffer 0.

The **tvkeys–window** echoes the keys you have entered in **cursor–mode** (see section 3.4 p. 18).

All commands you enter in **text–mode** in any Tv window are echoed in the **text–window**. They will be evaluated after pressing CR and the results are displayed in the text– or graphic–windows.

The text–window is **full screen editable**, i.e. you can move the cursor with the cursor–keys or the mouse to any previous executed command and execute it again, or edit it and execute the new command.

If you want to perform commands that have been executed many commands before you can use the **scrollbar** to accelerate your search and place the cursor with a **mouseclick** at the position wanted. Alternatively you can search previous executed commands by a **search context**. Therefore use Strg r or Strg s . You can also **cut an paste** text blocks with the mouse.

3.4   Input modes

There are two input–modes between which you can toggle with the **input–mode** command in the text–window or by pressing Esc in the graphics–window.

3.4.1   Edit–mode

This mode is indicated by a **text–cursor** I in the **tv–root** window. All keys you enter in this mode in any of Tv's windows are echoed in the text–window. When you enter CR , Tv will try to execute the entire line.

You can enter "**?**" **(question mark)** at any time to receive a list of possible commands. Furthermore it is sufficient to enter just the first few letters of a command (see section 3.7 p. 20) and press CR . Tv will try to complete the command and execute it. In general it is sufficient to enter just the first letter of the command. If the command is not yet completed Tv requests further input.

3.4.2   Cursor–mode

This mode is indicated by a **pirate–cursor** ☠ in the **tv–root** window. It allows you to shortcut commands by using "hotkeys" which are displayed in the tvkeys–

window when pressed. Keys you press in graphic–pane of **tv–root** are looked up in the hotkey–table (see section 3.11 p. 22) and the bound commands are executed as soon as the hotkey–combination is definite. As an example the following commands are bound to key $\boxed{\texttt{f}}$:

> tv> **window view full both**; tv> **window view expand**;

and when you press it in the graphic–pane, the viewport is expanded to full size. If you press "**?**" (**question mark**) in this mode a list of all hotkeys is printed in the text–window.

3.5   Mouse

The mouse is used for two purposes. First you can change the viewport of the spectrum. To do so, you can set viewport–markers with mouse clicks. Second you can use it for cut and paste text as well as positioning the cursor in the text–window.

While the cursor is placed in the graphic–pane (cross–hair cursor), the left mouse button (#1) sets a marker and the other buttons perform certain actions each. The middle button (#2) expands the region between the marker and the border in which the cursor is positioned and the right button (#3) expands the region between the marker set with #1 and the current cursor position.

You may set a vertical marker (single click #1), a horizontal marker (double click #1) or a corner marker (triple click #1). A corner marker is a combination of a vertical and a horizontal marker and defines a corner of a rectangle. Leaving the **tv–root** window deletes all viewport–markers set.

With the cursor on the scales **gumby–cursor** 🐾 the left mouse button shifts the displayed spectra to the left and the right button to the right.

In combination with the keys $\boxed{\texttt{Strg}}$ $\boxed{\texttt{Shift}}$ #1 shifts the viewport to the left, #2 centers the viewport at cursor position and #3 shifts the viewport to the right.

The configuration of the mouse buttons can be defined in your .Xresources file (see appendix B p. 91) which will overwrite the default settings.

The x– and y–position of the cursor is displayed in the status bar of the window the cursor is in (see 3.1 p. 16). If a calibration is loaded, the x–position is displayed both calibrated and uncalibrated (in parenthesis).

3.6   Commands and Menus

Tv's commands are organized in a menu structure. This leads to a kind of object orientation because all commands that are allowed to an object are in its menu. To see all available commands for the current menu enter $\boxed{\texttt{?}}$ $\boxed{\texttt{CR}}$. In the root menu, which is indicated by the prompt **tv>**, the available commands are separated in three groups. There are **aliases**, **main command group** and **miscellaneous**. Commands can be combinations of subcommands, for example:

> tv> **window setup function–y normalization off**

Leaving out at least one argument of the complete command will put you in another menu. For example, if you only enter

> tv> **window**

Tv will be in the menu `window` afterwards. This is indicated by changing the command prompt **tv>** to the name of the menu:

> `window`

All commands are interpreted **relative** to the menu you are in, i.e. they are appended to the menu name. For example

> `window` **spectrum get co56b68.127**

will result in the error

```
    fatal:   illegal input (window) 'spectrum get co56b68.127'
```
since **spectrum** is not an available command to the window menu.

Tv accepts **absolute** commands from any menu if preceded by **tv>**. You will be in the same menu after the evaluation of your command. To load the spectrum from the example above enter:

    window **tv> spectrum get co56b68.127**

To see a list of all arguments available to a command, for example the window command, enter:

    tv> window ?

Some menus have **default commands** which do not need to be entered. These are underlined in the manual. For example the default command to the **spectrum menu** is **get**. To load a spectrum you can enter:

    tv> spectrum <u>get</u> co56b68.127

    tv> spectrum co56b68.127

or go to the spectrum menu

    tv> spectrum

and enter the spectrum's name

    spectrum **co56b68.127**

To leave a menu and go to the menu one level higher, enter $\boxed{\text{CR}}$.

3.7   Abbreviations

Every command can be **abbreviated**. Note that, if the abbreviation is ambiguous, Tv takes the first matching alias or if not existant the first matching command. The order of the builtin commands is in most cases alphabetical, but not necessarily. The order of the aliases is given by the order of their definition. To see the order of commands and aliases in a given menu, type $\boxed{?}$.

For example if you plot a spectrum and therefore want to change the frame width to get inscriptions at the y–scale, you can set them to 9, 9, 3 and 3 for the left, right, top and bottom frame with the command:

    tv> window setup frame width 9 9 3 3

Your minimum set of characters to abbreviate this command is:

    tv> w se f w 9 9 3 3

By the way these are the default frame width values for the plot window which you can create with:

    tv> w c s plot

3.8   Special characters

To define aliases, keyaliases or write commandfiles, you need a set of **control characters**. These are used to replace keys like $\boxed{\text{CR}}$ or $\boxed{\text{Esc}}$, which cannot be inserted in files. Tv's control characters are listed in table 3.2 on page 21.

3.9   Aliases

You may configure your own menu structure or simply create, redefine or rename certain commands by using the **alias**–mechanism. An alias is a named string known to a menu. When parsing the command line Tv checks each word whether there is an alias defined and substitutes the according definition in the command line.

Aliases are defined in the menu you are in. You can not define aliases in menus other than the current menu. Since you destroy built-in commands if you redefine

%	Execute system command.
@	Execute command file.
#,!	Comment.
;	Carriage Return.
\	Escape.
?	Give a listing of available commands.
" "	String, contents will not be evaluated.
()	Parenthesize.
$(n)	Insert value of argument # n.
$(ENVIRONMENT)	Insert content of environment variable ENVIRONMENT.
`unix command`	The result of the unix command will be inserted at the place of this expression. E.g. look at the alias for help.
ESC	Toggle between edit– and cursormode.
Ctrl-C	Leave Tv.
*, ?, [,]	filename evaluation like /bin/ls see spectrum get.

Table 3.2: Summary of the command line interpreter's special characters.

them, you must not do that if you want to use them again. For example to define
an alias to switch the y–scale to logarithmic enter:

> **tv>** **alias lin** "**window setup function–y logarithmic ;**
> **window setup function–y normalization on ;**
> **window setup function–y normalization off;**"
> "**<no arguments>**"

Pay attention to enclose the second and third argument in double quotes ("") if you
use definitions consisting of more than one word. The third argument is appended to
the usage message printed by **lin ?**. **alias** without arguments prints all aliases known
to the current menu. **unalias lin** deletes the alias lin. You may use **commandline
arguments** or **environment variables** in the second argument following the same
convention as in common shells like the **bash**. To program an alias to switch the
y–scale according to a given commandline argument enter

> **tv>** **alias yscale** "**window setup function–y $(1) ;**
> **window setup function–y normalization on ;**
> **window setup function–y normalization off;**"
> "**<linear | logarithmic | squared>**"

where **$(1)** denotes the first argument passed to the **yscale** command. The paren-
theses in $(1) must be entered.

alias and **unalias** are builtin commands of the command line interpreter and there-
fore available in every menu and not listed by $\boxed{?}$.

Aliases defined by default in Tv's root menu are **help**, **quit**, **lin** and **log** and their
definitions listed by the alias command are:

```
tv> alias
aliases in menu:
help="% "xon `hostname` `which lemacs` -f info Tv > /dev/null 2> /dev/null&""
lin="wi sc fu li ; wi sc fu no on ; wi sc fu no off ;"
log="wi sc fu lo ; wi sc fu no on ; wi sc fu no off ;"
quit="exit"
```

3.10   Command files

Frequently used commands, definitions or setups can be stored in so called **com-
mand files**. To execute them, enter their names preceded by the special character

@. Be sure to be in a menu suitable for your command file. This is usually the root menu. A command file must always end with a `newline`.

If you do not use abbreviations shorter than 4 characters, your command files are guaranteed to work with every version of Tv.

Some useful command files to compare matrices, cuts and spectra are in the directory */ikp/local/lib/tv*. These are listed in table 3.3 on page 22.

cmp2cut	Compare two cuts.
cmp2mat	Compare two matrices.
cmp2spc	Compare two spectra.
cmp3mat	Compare three matrices.
cmp4mat	Compare four matrices.

Table 3.3: Sample command files.

As an example the contents of *cmp2mat* will be shown here in an abbreviated form. To understand the meaning of the commands take a look at chapter 6 on page 55.

```
wind dele projections
wind dele comparison

cut acti 0;
cut atta matr 0 dire 0;
cut matr atta 0;
cut atta spec 4;

cut acti 1;
cut atta matr 1 dire 1;
cut matr atta 1;
cut atta spec 5

wind crea simp projections;
wind setu keyb projections;
wind show spec 0 1;
wind crea simp comparison;
wind setu keyb comparison;
wind show spec 4 5;
wind setu keyb none;
keya "C" "cut acti 0; cut crea cut; cut acti 1; cut crea cut;";

cut acti 0;
cut envi $(1);
cut acti 1;
cut envi $(2);
```

3.11   Hotkeys

You can define **hotkeys**, which work in cursor–mode as explained before (see section 3.4.2 p. 18), by using the **keyalias** command. For example to bind the commands

   tv> **edit**; tv> **spectrum get**;

to the key $\boxed{\text{g}}$, you have to enter:

   tv> **keyalias g** "tv> **edit**; tv> **spectrum get**;"

You can print a complete list of hotkeys with the command:

   tv> **keyalias**

To destroy the just defined keyalias for key ⌷g⌷ use the command:

   tv> **keyunalias**

At startup Tv reads the file */ikp/local/lib/tv/.tvkeys* which contains the hotkey definitions for the cursor–mode. The contents of this file is shown in appendix A on page 83.

3.12 Wildcards

Tv supports a wildcard mechanism for filenames which have special meaning for certain commands and datatypes.

3.12.1 Wildcard concept

If you want for example to analyse a **matrix** you need the matrix itself, its **projection** and a **directory** where the results will be saved. Using wildcards you need not to enter all three pathnames, but only one basename. Therefore the three components have to follow a certain naming convention defined by the according wildcard. A matrix filename must have the extension .mtx by default and furthermore it must be saved in a directory that has the same name as the basename of the matrix. If you want to load the matrix *220Th.mtx* it must be in the directory *./220Th*. The default wildcard for matrices is

   *mtx → !f/!t.mtx

which is resolved in table 3.4 on page 23 for the matrix

   /matrix1/220Th/220Th.mtx

!f	/matrix1/220Th/220Th
/	/
!t	220Th
.mtx	.mtx

Table 3.4: Resolved wildcards for the matrix */matrix1/220Th/220h.mtx*.

Besides the matrix you need the projection to the y–axis which must obey the wildcard *prospc and is according to the default values for the example above:

   /matrix1/220Th/220Th.pry

The directory must obey the wildcard *cutdir (see table 3.6 p. 24) and will be made by Tv if not existing. Its pathname is:

   /matrix1/220Th/220Th.cutdir

If you have the necessary files you can start analysing your matrix with the command

   tv> **cut env** <path>

which performs the opening and loading of your data automatically.

The filename for matrices (*mtx) is composed of the full absolute directory name (!f), a slash (/), matrix basename (!t) and the suffix .mtx.

For the meaning of the wildcard characters see table 3.5 on page 24. The wildcard characters mus be preceded by an **! (exclamation mark)**.

The wildcards which are defined by default are listed in table 3.6 on page 24. You can print a list of all wildcards with the command:

   tv> **wildcard status**.

You can combine any entries of table 3.5 by using the pipe character "\". This character works like the pipe character | in unix shells, as shown in the following

Wildcard	Meaning	Example
f	full filename	/matrix1/220Th/220Th.tig0
t	filename (tail)	220Th.tig0
h	absolute path (!f-!t)	/matrix1/220Th/
r	basename (without last ext)	220Th
e	extension	.mtx

Table 3.5: Wildcard characters.

Wildcard	Definition	Datatype
*cutdir	!f/!t.cutdir	Cut directory
*cutcal	!f/!t.cut.cal	Cut calibration
*cutgat	!f/!t.gate	Cut gate marker
*cutspc	!f/!t.cut	Cut spectrum
*cutspcgat	!f/!t.gt	Cut spectrum gate marker
*cutspcbg	!f/!t.bg	Cut spectrum background gate marker
*cutspccl	!f/!t.cl	
*cutspccr	!f/!t.c	
*fit	!h/!t.fit	Fit marker
*cal	!r.cal	Calibration
*plt	!r.fig	Plot
*rcl	!r.rcl	Recalibration marker
*mtx	!f/!t.mtx	Matrix
*prospc	!f/!t.pry	Projection spectrum
*procal	!f/!t.pry.cal	Calibration for projection spectrum

Table 3.6: Default wildcard definitions.

example, where the left wildcard expression will be evaluated and given as input to the right expression. If you are in the directory */matrix1/220Th* and enter

> tv> **cut env 220Th.test**

the modified wildcard for matrices

> ***mtx** → !f/!t\!r.mtx

will result in a matrix filename:

> */matrix1/220Th/220Th.test/220Th.mtx.*

You can define own wildcards or alter existing settings. For example to set the modified matrix wildcard from the example above enter:

> tv> **wildcard enter *mtx "!f/!t\!r.mtx"**

The second argument should contain the leading **asterisk (*)**. Be sure to encapsulate the second argument in double quotes. Also you can change existing wildcards with the **wildcard enter** command. To delete a wildcard use the command:

> tv> **wildcard delete <wildcard>**

3.12.2 Example for user defined wildcards

For example you can use wildcards for the input and output of multiple spectra. To write a group of calibrated spectra you can define the wildcard *calspc

> tv> **wildcard enter *calspc "!r.cal_spc"**

and write the spectra with the command:

> tv> **spectrum write *calspc all**

For example if you have the spectra prge0.0121 and prge1.0121 loaded, this command saves the files:

> *prge0.cal_spc* and *prge1.cal_spc*

The input and output of spectra can also be performed using the ls–file–mechanism described in section 4.3.2 on page 31.

To delete the just defined wildcard *calspc use the command:

> tv> **wildcard delete *calspc**

3.13 Configuration files

Tv allows you to configure its behaviour in a wide range by using aliases (see section 3.9 p. 20), changing the window appearence (see appendix B p. 91), i.e. colours and cursorforms, and defining the hotkeys for the cursor–mode (see section 3.11 p. 22). For all these purposes you can use configuration files which are evaluated by Tv automatically at startup or which you have to start from automatically evaluated files or by hand.

If the environment variable TVPATH is not set Tv looks for configuration files in the following order. Only one *.tvrc* is evaluated.

1. */ikp/local/lib/tv/site.tvrc*

2. */ikp/local/lib/tv/.tvrc*

3. *<current working directory>/.tvrc*

If TVPATH is set only the directories defined in this variable are scanned. First for *site.tvrc* and afterwards for *.tvrc*. Set TVPATH in your *.bashrc* with:

> export TVPATH="$TVPATH:/ikp/local/lib/tv:~/.:."

3.14   Remote control

Since Tv does not support any **control structures** like loops or branchings an
extension was programmed to send commands to Tv from any script language.
The program that fulfills this is called STV (stands for send tv). For example

```
i=1;
while 'test ${i} != 8';
  do
    stv "tv> s ${i}.spc";
    i='expr ${i} + 1';
  done
```

started from a shell, loads 7 spectra. STV sends commands only to Tv's which are
owned by the user running STV and waits for Tv to finish the command before
returning. Only the last one started Tv receives the commands or you must ex-
plicitely define a portnumber. This is printed immediately after startup of Tv and
saved in the file ˜/.tv-port. If you want to send commands to Tv with portnumber
10000 you enter for example:

```
i=1;
while 'test ${i} != 8';
  do
    stv "tv> s ${i}.spc" 10000;
    i='expr ${i} + 1';
  done
```

Tv shell commands like "%rm -rf ˜" sent by STV are ignored.

Chapter 4

Spectra analysis

Contents

4.1 Introduction

The main topic of Tv is to display and analyze spectra. With respect to that matter in this chapter it will be explained how to load one or more spectra and perform all necessary operations to analyze the data, as there are the choosing of the best window type, calibration, recalibration, normalization, arithmetic operations, integration and fit as well as peaksearch and finally the plotting of spectra.

4.2   Graphic window

4.2.1   Graphic window types

Tv offers several types of windows suitable for special purposes.

First there is the basic window type **simple** which consists of just one pane. You can use it to display an arbitrary number of spectra.

There is one special window of window type simple, the plot window. Since the default frame width for the left and right side is not large enough to see the scale's inscriptions on your plot you can enlarge them (see section 4.6 p. 34) or create a new window named **plot** which is suitable for the plotting of spectra.

> **tv> window create simple plot**

Besides the simple window there are three different **paned** windows, the xy–, x– and y–paned. They are composed of one or more panes, which are arranged one above the other. All panes share a common status line. These window types provide a y–scale per pane and a total of two x–scales. In an xy–paned window the x–scales are independent as well as the y–scales. The scale at the top of the window belongs to the uppermost spectrum shown and the one at the bottom to the lowest. So with more than two panes, you have spectra without x–scale. In x– or y–paned windows, the x–scales are coupled to all spectra shown. So if you set x–view markers with
$\boxed{\text{Spacebar}}$ in one pane and expand with $\boxed{e}$, the viewport in all panes will be changed. In a y–paned window (see figure 4.2 p. 31) all y–scales are coupled, too.

> **tv> window create {paned | xpaned | ypaned} <name> <number>**

Paned windows are good means for the comparison of many spectra. For example you can create an x–paned window for the comparison of four distances of a lifetime experiment with the command:

> **tv> window create xpaned lifetime 4**

The remaining window types are composed of two graphic subwindows. The **fit** window is an x–paned window in which lower pane the residuum of the fit is shown. The **cut** window has a small subwindow in its upper right corner where the position of the cut marker in the projection is shown (see figure 2.5 p. 13).

> **tv> window create {fit | cut} <name>**

Figure 4.1 on page 28 shows an example for the fit window that was created with the command:

> **tv> window create fit "fit example"**

In the lower pane the residuum is displayed.
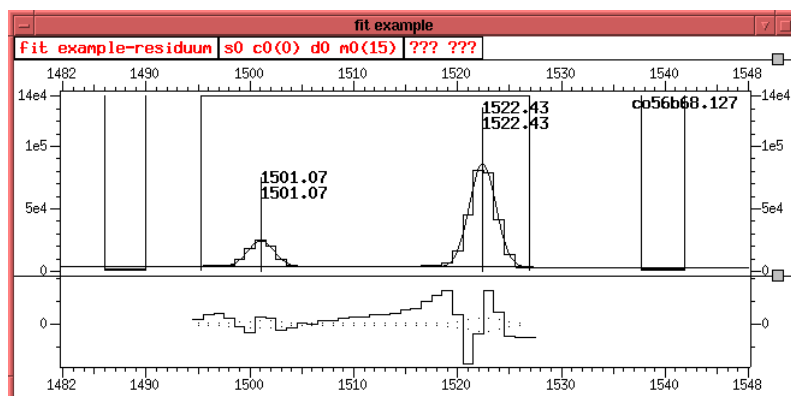
Figure 4.1: A fit of two peaks in a ^{56}Co spectrum in the special fit window. In the lower pane the residuum is displayed.

To delete the graphic window named "fit example" use the command:

> tv> **window delete "fit example"**

4.2.2 Configuration of graphic windows

When more than one graphic–window is existent it is not definite where commands entered in the text–window are evaluated. A focus to the window must be set where commands concerning the window should be evaluated. These are for example commands to plot a window, change the y–scaling or change its habits, e.g. frame width or tic length. All commands to configure the graphic window are listed in section 6.9 on page 74 of chapter 6.

By default to set the focus to a window you have to move the mouse into the window. You can set the keyboard–focus to a certain window named "lifetime" with the following command, assuming it exists:

> tv> **window setup keyboard–focus <u>title</u> lifetime**

and switch back to default behaviour by pressing hotkey $\boxed{\text{L}}$ or with the command:

> tv> **window setup keyboard–focus cursor**

There are three values for the short, medium and long tic lengths in this order and another one for the distance between the axis–label and the tic. The command to change these values which are given in the unit character size is:

> tv> **window setup frame length 0.2 0.4 0.6 0.75**

Another group of commands configures the y–scale of the windows. For example to set the y–scale to logarithmic scale you have to enter the commands:

> tv> **window scaling function–y logarithmic**
> tv> **window scaling function–y normalization on**
> tv> **window scaling function–y normalization off**

These three commands are defined as alias **log** by default.

You can switch off the graphic display to omit graphical output and thereby accelerate the execution of command files. To do this enter

> tv> **graphic suspend**

and to turn it on again enter

> tv> **graphic resume**

4.3 Loading and saving of spectra (I/O)

Tv supports several functions for spectrum input and output, as there are:

> tv> **s read <fname['fmt]> {{<ind>...} | all | shown | active}**
> tv> **s <u>get</u> <fname['fmt] ...>**
> tv> **s write {<fname['fmt]> | <wc>} {{<ind>...} | all | shown | active}**
> tv> **s format {input | output} <fmt>**

<fname> is the name of the spectrum file. **<fmt>** is the format of the spectrum file (see below). **<ind>...** are the buffernumbers for the spectrum. **<wc>** is a wildcard defined according to section 3.12.2 on page 25. The commands to the spectrum menu are described now.

read loads the spectrum to the first defined buffer and copies it to the other buffers specified. Tv displays the same buffer always in the same color. If you want to see a special spectrum in a certain color you can ensure this by using the read command.

get reads spectra, too and is the default command to the spectrum menu, meaning it can be omitted to enter it.

The difference between **read** and **get** is that **read** loads one spectrum into one or more buffers you have to specify manually, whereas **get** loads them to the next free buffers automatically selected by Tv.

write saves the spectra to the defined files. It operates on **all**, only the **shown** or the **active** spectra or on spectra defined by the **index** list.

format sets the format for the input or output of spectra. **input** is the default value to this command. For a list of allowed formats see appendix C on page 101.

Tv performs I/O operations using the mfile library[1]. If you want to read spectra not written in mfile format, it may be necessary to specify the spectrum format explicitely.

For example you want to load the spectrum *prge0.0121* to buffers #0 to #3 and display them for comparison in a y–paned window (see section 4.2.1 p. 28). You create the window with the command

> tv> **window create ypaned ypwindow 4**

and Tv prints which spectrum and cut is active in each pane:

```
yp--window 0:   spectrum #0 active
yp--window 0:   cut #0 active
yp--window 1:   spectrum #1 active
                    ⋮
```

If the window exists load the spectrum with the command

> tv> **spectrum read /home/fitz/spec/0121/prge0.0121 0 1 2 3**

and Tv prints to which buffer the spectrum is loaded and copies it to the other buffers.

```
tv> spectrum ./prge0.0121'8k.lc:2 read to #0
tv> spectrum#0 copied to #1
                    ⋮
```

This results in the window shown in figure 4.2 on page 31.

As an example for the get command we load the spectrum *ge0.0121* after the above operation. Tv places it to the next free buffer.

> tv> **spectrum /home/fitz/spec/0121/prge0.0121**

```
tv> spectrum ./prge0.0121'8k.lc:2 read to #4
```

This loads the 8k spectrum which is saved in mfile format (see appendix C p. 101) with linecompress mode 2 to buffer #4.

To force Tv to use a certain format, you have to specify it either while loading a spectrum with get or read, or by setting the default format with the format command. Choosing the wrong format

> tv> **spectrum prge0.0121'16k.lc:2**

will result in an error:

```
fatal:  cannot read spectrum prge0.0121'16k.lc:2
vsSpectra:  could not set mfile format
fatal:  illegal input (spectrum) ''
```

Here is an example for the format command:

> tv> **spectrum format input 16k.lc:2**

The maximum number of buffers is set to 16 by default and may be specified in the command line (see section 3.2 p. 15) when Tv is started.

---

[1] The mfile library was programmed by S. Esser as diploma thesis for the IKP and is a tool for the saving and loading of compressed spectra. The compression factor is about 1:5.
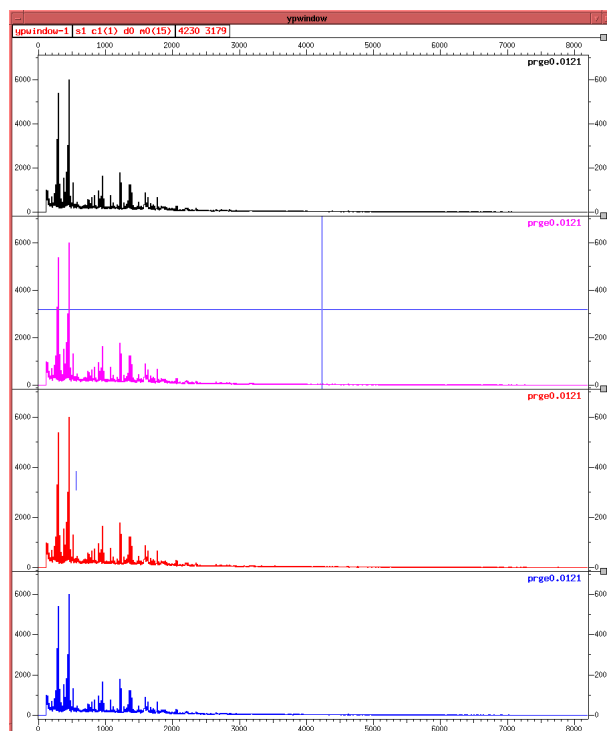
Figure 4.2: Ypaned window with the a spectrum loaded to all four panes.

4.3.1   I/O with multiple spectra

To load multiple spectra with the get command, you may use filename patterns. Tv expands the patterns (like a shell) and loads all matching files. As example we load spectra *prge0.0121 ... prge5.0121.*

    tv> spectrum prge[012345].*

Tv distributes them to buffers #0 to #5 automatically if these are free:

```
tv> spectrum ./prge0.0121'8k.lc:2 read to #0
tv> spectrum ./prge1.0121'8k.lc:2 read to #1
                    ⋮
```

    Tv has read the spectrum files to buffers #0 to #5, using the format 8k.lc:2. The second, more general way to load multiple spectra is the ls–file–mechanism (see section 4.3.2 p. 31), which loads (or performs other operations on) spectra, listed in a special file (the ls–file).The third alternative is the loading of multiple spectra using wildcards (see section 3.12.2 p. 25).

4.3.2   ls–file–mechanism

In order to perform an operation on a group of spectra that you want to load, write their names in a so called ls–file. You can generate an ls–file from within Tv, for example for the spectra from the last section, with the command:

    tv> % "ls prge[012345].* > spc–file"

That is where it got its name from. The syntax for the ls–file command is:

    tv> s ls <operation> <option>

    All operations listed in table 4.1 on page 32 are possible:

add	Add several spectra to a specified buffer.
commandget	Load spectra and execute command file for each spectrum loaded.
get	Load several spectra.
subtract	Subtract several spectra from a specified buffer.

Table 4.1: Allowed operations for the ls–file command.

Note that, if there is a position calibration for all spectra involved in the ls–file operation, it will be applied before the operations **add** and **subtract**. If you want the uncalibrated spectra to be added or subtracted you have to unload the calibration for at least one spectrum.

To access the spectra listed in the ls–file, all operations mentioned above have the options listed in table 4.2 on page 32.

ls–file	Open an ls–file.
+	Take the next spectrum in the list.
++	Take all following spectra up to the end of file.
–	Take the previous spectrum in the list.
– –	Take all previous spectra.
line–index	Take spectrum at line–index.

Table 4.2: Options to the ls–file operations to define which files will be operated.

To use the ls–file in Tv it must be opened first with:

   tv> spectrum ls–file open spc–file

To list the contents of an ls–file you can use:

   tv> spectrum ls–file list [<filename>] [+ | ++ | – | – – | <index>]

The pointer is set to the next file if a file is listed. If another ls–file is open than the one given by <filename>, the old one will be closed and the new one opened, but if no ls–file is opened, an error occurs. The list command without arguments shows the index of the next file.

To set the pointer in the ls–file to position <number> enter:

   tv> spectrum ls–file position <number>

In order to load all spectra from the current file position to the end of the ls–file, use the command:

   tv> spectrum ls–file get ++

You may close the ls–file with:

   tv> spectrum ls–file close

As an example we load the spectra from the last section. Imagine that no ls–file is opened,so we have to open the file *spc-file* first with the command:

   tv> spectrum ls–file open spc–file

Now you can load the spectra with the command:

   tv> spectrum ls–file get ++

Of course not more spectra can be loaded then buffers are available.

For the syntax of all ls–file commands see section 6.8.10 on page 72. A list of hotkeys for ls–file operations is shown in table 4.3 on page 33.

Key	Tv–commands	Function
l +	tv> spectrum delete all; tv> spectrum ls-file get ++;	Deletes all spectra and reads according to the ls–file rules (see section 6.8.10)
l -	tv> spectrum delete all; tv> spectrum ls-file get --;	Deletes all spectra and reads according to the ls–file rules (see section 6.8.10)
l g	tv> edit; tv> spectrum ls-file get;	Asks for spectra to be loaded and reads according to the ls–file rules (see section 6.8.10)
l o	tv> edit; tv> spectrum ls-file open;	Queries an ls–file name and opens it

Table 4.3: Hotkeys for ls–file operations.

4.4 Buffer operations

Tv provides three commands to copy, create and delete spectra in memory, as there are:

> tv> **spectrum copy** **<source index>** {**<destination index>**... }
> tv> **spectrum create** **<name>** **<resolution>** **<index>**
> tv> **spectrum delete** {**<index>**... }

copy copies buffer <source index> to buffers <destination index>.... **create** creates an **empty spectrum** with name <name> and resolution <resolution> in buffer <index>. **delete** removes spectra from buffers <index> .... You can perform the delete operation with hotkey k .

4.5 Using the mouse

The usage of the mouse is described in section 3.5 on page 19.

4.6 Plot and labels

Labels are used to mark points in a spectrum, e.g. a peak. They have no effect to any operations in Tv.

Labels consist of an **info string** and a **position** on one axis of a spectrum. The **info string** is normally generated automatically by Tv and contains the energy (calibrated or not). Labels allow the user to define an own info string. The **position** may be on the x–axis (labels and vertical markers) or on the y–axis (horizontal markers). The position of the **info string** on the other axis is calculated by Tv.

You can set labels for **peaks**, **fits** and **cuts** as well as **user** markers whereat you can define any combination of the calibrated and uncalibrated position and the info string to be printed. You define a label with the command (which takes a position as argument by default):

> tv> **label user enter** {**cursor** | **value** | **offset**}

The combination of data to be printed at the label can be defined with the following command where pairs of arguments function like switches. For example **calibrated** switches the printing of the calibrated position on while **nocalibrated** turns it off.

> tv> **label** {**cut** | **fit** | **peak** | **user**} {**libr** | **cali** | **noca** | **unca** | **noun** | **stri** | **nost**}

You can manage several labellists at once. The commands for creating a new labellist and activating an existent one are:

tv> label user create <title>

tv> label user activate <title>

You can use labels to mark peaks in plotted spectra. To mark a peak with an arbitrary string the following command is used:

tv> label user enter <position> <colorindex> <"string>

The colors are given in table 4.4.

Colorindex	Color	Colorindex	Color
0	yellow	7	pink
1	magenta	8	MediumOrchid
2	red	9	seashell4
3	blue	10	firebrick
4	white	11	LightSlateBlue
5	wheat	12	aquamarine
6	cyan	13 ...	yellow

Table 4.4: Colorindices for labels.

Sometimes markers are disturbing in printouts. Therefore they can be switched off and turned on with the following commands:

tv> window hide labellist user

tv> window show labellist user

Since you want to plot paned windows sometimes and the simple window named plot has one frame only you can change the frame widths of the window panes. For example create a paned window with three frames, move the cursor to the upper frame and enter:

tv> window setup frame width 9 9 3 0

Then move the cursor to the center pane and enter:

tv> window setup frame width 9 9 0 0

And finally move the cursor to the bottom pane and enter:

tv> window setup frame width 9 9 0 3

Then you can plot your paned window with inscriptions at the y–axes as usually.

4.7   Marker

Several Tv commands, e.g. fit or recalibration, do not work on the whole spectrum but on regions only. Such regions can be defined using markers. There are hotkeys for all kind of markers, which are listed in table 4.5 on page 35.

Markers are set in a window and remain there, even if the spectrum displayed is exchanged by another. So you can operate with one set of markers on many spectra.

4.8   Calibration

Tv offers commands to calibrate various parameters, as there are **position**, **width**, **efficiency**, **lefttail** and **righttail**. You can set **startchannel** to define the energy associated with channel 0 and define the unit of the x–scale by setting **unit**.

For example to calibrate the position and width of peaks in a spectrum in buffer 0 in keV where the energy for channel 0 is 17 keV, you set the startchannel to 17, the unit to keV and perform the calibrations as shown below.

tv> calibration startchannel enter 17

tv> calibration unit enter keV

Hotkey	Tv–commands	Function
b	tv> fit mark bg-region enter cursor;	Set background–region marker at cursor position
c	tv> cut mark cut enter cursor;	Set cut marker at cursor position
G G	tv> cut marker gate enter cursor;	Defines a gate marker.
h	tv> window mark horizontal enter cursor;	Set a horizontal marker
o	tv> normalization mark enter cursor;	Set normalization marker at cursor position
p	tv> fit mark peak enter cursor;	Set a peak marker at cursor position
r	tv> fit mark region enter cursor;	Set fit–region marker at cursor position
R r	tv> recalibration marker enter cursor;	Set a recalibration marker at cursor position
- b	tv> fit mark bg-region delete cursor;	Delete background marker closest to the cursor
- c g	tv> cut mark gate delete cursor;	Delete gate marker closest to cursor
- c b	tv> cut mark bg-gate delete cursor;	Delete background–gate marker closest to cursor
- n	tv> normalization mark delete;	Delete the normalization markers
- p	tv> fit mark peak delete cursor;	Delete peak marker closest to cursor
- r	tv> fit mark region delete cursor;	Delete fit–region marker closest to cursor
- A	tv> cut mark cut delete; tv> window hide mark cut; tv> fit mark fit delete; tv> fit delete; tv> window hide mark fit; tv> label user delete all; tv> normalization mark delete; tv> recalibration marker delete all;	Delete all markers. You will still see markers which are part of the autonomous fit
- B	tv> fit mark bg-region delete all;	Delete all background markers
- C	tv> cut mark cut delete; tv> window hide mark cut;	Delete cut markers
- F	tv> fit mark fit delete; tv> fit delete; tv> fit delete activ; tv> window hide mark fit;	Delete the current fit and all its markers
- P	tv> fit mark peak delete 0 uncalib 8192 uncalib;	Delete all peak markers
- R	tv> recalibration marker delete cursor;	Delete recalibration marker closest to cursor

Table 4.5: Hotkey definitions for all markers known by Tv.

   `tv>` **calibration** <u>**position**</u> **enter 1204 1197 1579 1563**
   `tv>` **calibration** <u>**width**</u> **enter 1213 4.26 1258 4.63**
Tv prints the parameters for the position calibration with the command:
   `tv>` **calibration** <u>**position**</u> **list 0**
and for the width calibration with the command:
   `tv>` **calibration width list 0**
You can list the results for the complete set of calibrations at once with the command:
   `tv>` **calibration set list 0**
You can save the energy calibration to a file with standard filename according to the wildcard definition *cal or another file you have to enter with the command
   `tv>` **calibration** <u>**position**</u> **write**
and load from file *cal or any other file by pressing hotkey $\boxed{\text{E}}$ or by entering the full command:
   `tv>` **calibration** <u>**position**</u> **read**
You can copy the calibration to spectra in other buffers with
   `tv>` **cali** <u>**position**</u> **copy <src ind> {{<ind>... } | acti | all | show | visi}**
All calibrations entered or read are copied to all buffers automatically. For the syntax of all calibration commands see section 6.2 on page 58.

4.9   Recalibration

Due to electrical effects during the data aquisition, peaks may be shifted in some spectra which means that peaks that should be at the same position are not. Shifting spectra to have their according peaks in the same channels is provided by Tv with the **recalibration** commands. Basically this is a linear calibration.

For example load two spectra *prge[35].0121* which are shifted. In order to perform the recalibration you have to define at least one recalibration region. The region is defined by pressing the hotkey $\boxed{\text{R}}\boxed{\text{r}}$ once for the left border and once again for the right side. To perform the recalibration enter
   `tv>` **recalibration create**
in the text window. Tv will query you for further parameters, as there are the **reference spectrum** and a list of spectra to be recalibrated. By default the type of recalibration is set to squared–distance. You can change this for example to center–of–mass with the command:
   `tv>` **recalibration type center–of–mass**
Redisplay the window to see the results by pressing hotkey $\boxed{\text{CR}}$ or with the command:
   `tv>` **window redisplay**
You can print a list of recalibration parameters with the command:
   `tv>` **recalibration list**

A list of all recalibration commands is printed in section 6.7 on page 70.

4.10   Normalization

For the comparison of spectra **normalization** is often useful to bring their peaks to comparable heights.

To perform a normalization for a specific peak, you have to set markers with the hotkey $\boxed{\text{o}}$, one at the right side of the peak and one at the left border, which define the normalization region. You can set normalization background markers by using the command:

    `tv>` **normalization marker enter** {**cursor** | **value** | **offset**}

Spectra are normalized in Tv by using the command:

    `tv>` **spectrum norm**

To identify normalized spectra, an asterisk (*) will be applied to the spectrum status (see section 6.8.18 p. 73).

An example of recalibrated and normalized spectra can be seen in figure 4.3 on page 37, where the region of the right peak was marked before normalization.
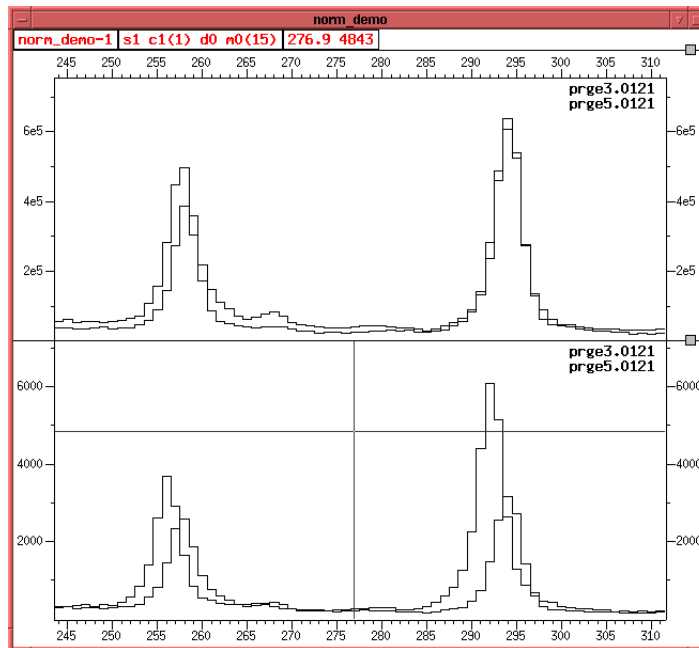
Figure 4.3: An example for recalibrated and normalized spectra. In the lower pane the original data is shown.

A complete list of all normalization commands can be found in section 6.6 on page 69.

4.11 Arithmetic operations

Tv provides five commands to perform arithmetic operation on spectra, as there are:

    `tv>` **s add** <**idx**> {<**fname**>['<**fmt**>] | {<**idx**>...} | **acti** | **all** | **show** | **visi**}

    `tv>` **s subt** <**idx**> {<**fname**>['<**fmt**>] | {<**idx**>...} | **acti** | **all** | **show** | **visi**}

    `tv>` **s mult** <**fact**> {{<**idx**>...} | **acti** | **all** | **show** | **visi**}

    `tv>` **s maxi** <**idx**> {{<**idx**>...}}| **acti** | **all** | **show** | **visi**}

    `tv>` **s mini** <**idx**> {{<**idx**>...}}| **acti** | **all** | **show** | **visi**}

add summarizes the contents of all spectra specified by the second argument to the buffer defined by the first argument <idx>. As second argument it accepts an integer value by default which is interpreted as buffer number, or alternatively a text value which is taken as filename of a spectrum. The argument all makes Tv to add all spectra but the one in the destination buffer. If the destination buffer was empty before, the spectrum will get the name of the first spectrum added.

subtract works analogous to add, but subtracts spectra.

multiply multiplies the spectra given by the last argument with the factor <fact>. The second argument takes an integer argument as default value.

maximum takes the maximum for each channel of all spectra defined by the last argument and puts this value in the spectrum specified by the second argument. The last argument takes integer values as default, which give the buffer–numbers for comparison.

minimum works analogous to maximum, but takes the minimum of each channel.

4.12   Fit and integration

The interesting results of the quantitative analysis of spectra are position, volume and shape of the observed peaks. The analysis must comprise a proper description of the background. Since the background has a more global structure, whereas the peaks in a gamma spectrum are local structures, the former can be approximated by a polynom, usually of second degree. On the contrary, particle spectra have very broad peaks. Therefore it is necessary to use a more physical description of the background which is provided by an exponential term besides the polynom. This background function may be useful in the upper part of a Ge–detector's spectrum, too since the exponential term reproduces approximately the detector's efficiency.

Tv offers three methods for the analysis of spectra as there are integration, peaksearch and fit, whereat two different functions to fit the peaks and two different functions to approximate the background can be used. The methods are explained in the following sections while the definitions are given in appendix D on page 103.

4.12.1   Integration

The main advantage of integration in comparison with fitting is that no hypothesis on the shape of the peak is necessary. Tv calculates volume, center of mass, width (corrected by a factor $2 \cdot \sqrt{2 \cdot \ln 2}$ to obtain the FWHM), and skewness (asymmetry). Tv takes into account existing background fits and calculates background corrected values, too. If background regions are defined but not fitted Tv determines the mean background and uses this for the background correction. The results of the integration increasingly depend from volume to skewness, on a good description of the background. Integration is appropriate for γ –spectra with not well determined peak shapes.

You can only analyze well separated peaks by integration. Otherwise it is necessary to make assumptions about the line–shape and to use a fit for decomposition of overlapping peaks. Furthermore it needs a lot of effort to integrate a large number of peaks.

To perform an integration you have to define a fit–region by pressing the hotkey $\boxed{\text{r}}$ once for the left and once for the right border of this region. You can also define background–regions by using the hotkey $\boxed{\text{b}}$ and create the background–fit with $\boxed{\text{B}}$ or the command:

   tv> **fit background–create**

The integration is created and a short status printed by pressing hotkey $\boxed{\text{I}}$. If you just want to create the integration use the command:

   tv> **fit integration–create**

and print the results of integration with:

   tv> **fit status {full | short}**

where **full** status gives detailed information about the background and all parameters of the peak.

4.12.2 Peaksearch

To obtain a list of all peaks in a spectrum Tv offers a peaksearch function. The fit function used is a simple gaussian with constant background. Before performing the peaksearch it is necessary to apply a width calibration to determine the width of the gaussian. To get the two widhts you need for width calibration you can use the quickfit and to perform the width calibration use:

> **tv> calibration width <chnl$_0$ > <width$_0$ > <chnl$_1$ > <width$_1$ >**

The amplitudes of the gaussian and background are fitted and the probability integral is compared with a given limit which is 99.99% by default. You can determine the current probability limit and set a new value with the command:

> **tv> fit psearch probability [<limit>]**

If the probability integral exceeds the limit, the peak position is determined to a precision of 1/10 of a channel and the peak is added to a list. The peaksearch is performed by the command:

> **tv> fit psearch create**

which prints the number of peaks found. You will get a list of all peaks found with their positions, widths and volumes with the command:

> **tv> fit psearch list**

If you do not want to search for peaks in the whole spectrum, you can set markers to limit the range. To set a marker enter

> **tv> fit psearch marker enter {cursor | <u>value</u> | offset}**

If you found a pair of good markers that you want to keep in mind while looking for a better pair you can store and restore them with:

> **tv> fit psearch marker store**
> **tv> fit psearch marker restore**

Saving peaklists

The results of a peaksearch can be saved either as binary with

> **tv> fit psearch write**

to the file <spectrumname>.fit or as asciitext using one of the following commands

> **tv> fit print**
> **tv> fit psearch print**
> **tv> fit peaklist print**

to a file open with one of the following commands

> **tv> fit result–file open psearch.demo**
> **tv> fit result–file append–open psearch.demo**

The asciifile will be closed by using the command or at exiting Tv.

Loading peaklists

Peaklists can be loaded from binaryfiles only by using the command

> **tv> fit psearch read**.

By use of the command

> **tv> fit psearch open**

besides the peaklist eventually saved fits are loaded too.

4.12.3 Fitting

A **fit** consists of **markers** and **parameters** and is autonomous, i.e. if you delete the spectrum you have fitted, the fit will still be there. So you may load or activate another spectrum while keeping the markers and use the same fit on it.

To obtain best fit results you have to define **fit– and background–regions** as well as **peak markers**. To define the fit–region, i.e. the region of the spectrum

containing the peak(s) to be fitted, press $\boxed{\text{r}}$ for the left border and press $\boxed{\text{r}}$ again for the right border of the fit–region. Now mark all peaks to be fitted inside the fit–region by pressing $\boxed{\text{p}}$ over each peak. Then you can **perform the fit** by pressing $\boxed{\text{F}}$ or with the command:

> tv> **fit region–create**

Optionally you can further improve the fit by defining an arbitrary number of **background–regions** using the hotkey $\boxed{\text{b}}$ once for the left and once for the right border of each background–region. Press $\boxed{\text{B}}$ to fit spectral background inside the background–regions and $\boxed{\text{F}}$ afterwards to fit peak(s). The background fitted to the background–regions will be used for the fit inside the fit–region. Note, though Tv supports only one fit–region, any number of background–regions can be defined for the fit. Figure 4.4 on page 40 shows spectra with fits. In its upper pane no fit parameters have been changed and the fit does not suit very good as you can see.

If you want to save your fit results to a file it must be opened first. To open the file enter the command:

> tv> **fit result–file open fit–result.demo**

and write your results with:

> tv> **fit print**

After finishing your work you must close the file with the command:

> tv> **fit result–file close**

You can store your fit markers to memory and restore them with the commands:

> tv> **fit store**

> tv> **fit restore**

If you want to set new markers to perform another fit, you have to delete the old markers first. Best you can do this by pressing $\boxed{\text{-}}$ $\boxed{\text{F}}$. You can recover fit markers that have been deleted accidentally with the command:

> tv> **fit recover–backup**
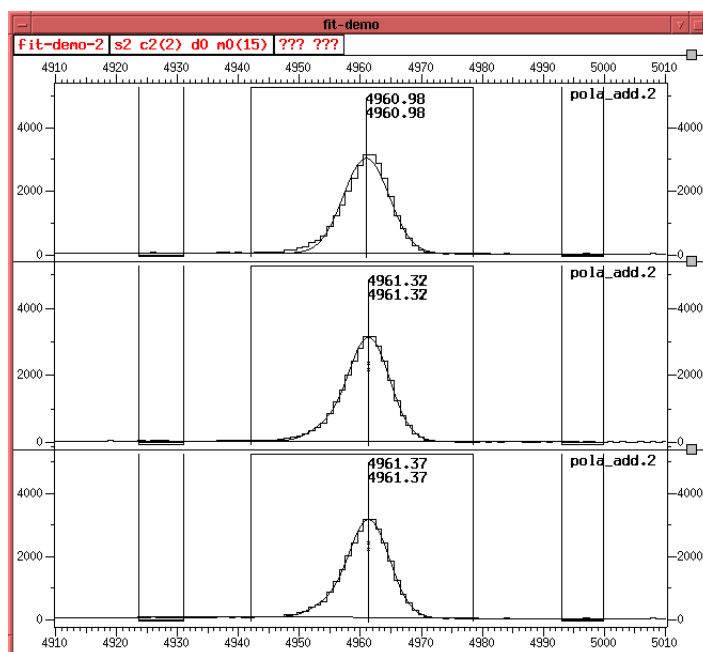
Figure 4.4: Graphic–display window with a ^{226}Ra spectrum loaded and a fit attached. **Upper:** The fit parameters are set to their default values. **Middle:** Left tail parameter is set free. **Lower:** Left tail and step height parameters are set free.

In order to consider the left tail you must allow Tv to fit it. The command to do that is:

> **tv> fit parameter tl free**

The result for the fit with tail is shown in the middle pane of figure 4.4 on page 40 and the difference is evident. You can furthermore consider the background form in various ways. Since there can be a loss of energy in electronics during data aquisition some events are shifted to lower energies and a step in the background will occur. You can fit height as well as width of these steps. To allow Tv to fit the height of a step you enter:

> **tv> fit parameter sh free**

The lower pane of figure 4.4 on page 40 shows the difference in the background form. The step is fitted as arctan but you can also consider an erf–function (see section D.1 p. 103). You can do this with the command:

> **tv> fit function peak activate additive–tail/erf–step**

A list of all fit parameter commands is printed in section 6.4.12 on page 63. Table 4.6 on page 42 shows all hotkeys related to the fit commands.

4.12.4 Decomposition

A spectrum does not only consist of well separated peaks but they often appear as multiplets (a line in a spectrum composed of a group of related lines). For the decomposition of multiplets a fit must be used. Therefore a description of the observed line shapes is necessary. Otherwise it is impossible to find the correct number of overlapping lines and to obtain a good description of the spectrum.

As an example what can happen if you fit a multiplet in the wrong way figure 4.5 on page 43 shows in its upper pane a quickfit. As you can see, it can not decompose the two peaks. In the middle pane a normal fit has been performed with two peakmarkers set and in the lower pane the decomposition is shown. You can show or hide the decomposition with the commands:

> **tv> window show fit decomposition**
> **tv> window hide fit decomposition**

or the hotkeys `m` `d` or `u` `d`.

Tv uses a modified gaussian with left tail and right tail and an underlying step. Parameters of the gaussian are position, volume and width as well as left and right tail and finally width and height of the step (see section D.1 p. 103). These seven parameters are associated with each peak fitted. To reduce the number of parameters it is possible to use calibrations for single parameters or to correlate them, e.g. to equal widths. Step and tail parameters sometimes can be neglected. The background parameters can be simultaneously fitted with the peaks or can be fixed in advance by a fit in separate background regions. To select the peak function or background function to be used or whether parameters are to be fitted or not use the following commands (for all **fit function** and **fit parameter** commands see section 6.4.7 on page 62 and section 6.4.12 on page 63).

To activate a certain peak function (by default continuous–exp–tail/arctan–step is active) enter:

> **tv> fit function peak activate {cont | additive–tail/erf–step}**

You can generally set parameters to a value and hold this value, i.e. disallow Tv to fit the parameter or set the parameter free and allow Tv to fit it. If you do not want Tv to fit a parameter after you set it you should set it to hold.

The degree of the background polynom (default value is 2) is set with:

> **tv> fit parameter background degree <degree>**
> **tv> fit parameter background hold**

and the exponential term is activated by setting the parameter FAC to nonzero with:

Hotkey	Tv–commands	Function
b	tv> fit mark bg-region enter cursor;	Set background–region marker at cursor position
p	tv> fit mark peak enter cursor;	Set a peak marker at cursor position
r	tv> fit mark region enter cursor;	Set fit–region marker at cursor position
- b	tv> fit mark bg-region delete cursor;	Delete background marker closest to the cursor
- p	tv> fit mark peak delete cursor;	Delete peak marker closest to cursor
- r	tv> fit mark region delete cursor;	Delete fit–region marker closest to cursor
- A	tv> cut mark cut delete; tv> window hide mark cut; tv> fit mark fit delete; tv> fit delete; tv> window hide mark fit; tv> label user delete all; tv> normalization mark delete; tv> recalibration marker delete all;	Delete all markers. You will still see markers which are part of the autonomous fit
- B	tv> fit mark bg-region delete all;	Delete all background markers
- F	tv> fit mark fit delete; tv> fit delete; tv> fit delete activ; tv> window hide mark fit;	Delete the current fit and all its markers
Q	tv> fit mark fit delete; tv> fit delete; tv> fit mark peak enter cursor; tv> fit mark region enter offset -10 offset 20; tv> window view full y; tv> fit region-create; tv> fit status short; tv> window show fit function/marker;	Creates a quick fit.
- P	tv> fit mark peak delete 0 uncalib 8192 uncalib;	Delete all peak markers
H P	tv> fit parameter peak hold;	Prevent peak position from being fitted
H W	tv> fit parameter width hold;	Prevent peak width from being fitted
P b h	tv> fit param backgr hold; tv> fit param exponent hold; tv> fit param factor hold;	Prevent background–parameters from being fitted
P b f	tv> fit param backgr free; tv> fit param exponent free; tv> fit param factor free;	Allow background–parameters to be fitted
P w e	tv> fit parameter width equal;	
w e	tv> fit parameter width equal;	
w f	tv> fit parameter width free;	Allow width to be fitted

Table 4.6: Hotkey definitions for fit markers and commands known by Tv.
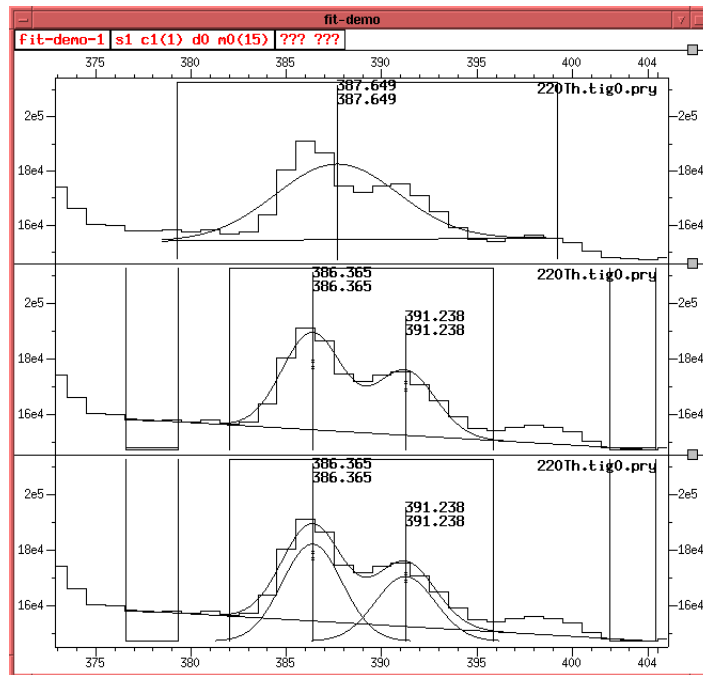
Figure 4.5: Graphic–display window with a ^{220}Th spectrum loaded and a fit attached. **Upper:** A quickfit has been performed. The peaks are not decomposited. **Middle:** A normal fit is performed with two peaks marked. **Lower:** The fit function and its decomposition is shown.

> **tv>** **fit parameter factor–background number <number>**
> **tv>** **fit parameter factor–background hold**

The scaling of the exponential term is set with:

> **tv>** **fit parameter exponent–background number <number>**
> **tv>** **fit parameter exponent–background hold**

All parameters which can be influenced by the **fit parameter** command are listed in table 4.7 on page 43.

Parameter	Meaning
background	Degree of background function.
exponent–background	Scaling of exponential term (see section D.2 p. 105).
factor–background	Factor of exponential term (see section D.2 p. 105).
position	Position.
sh	Height of step.
sw	Width of step.
tl	Left tail.
tr	Right tail.
volume	Volume.
width	Width.

Table 4.7: Fit parameters and their meaning.

It is important to choose the optimal procedure to optimize the parameters. The correct fit function gives the expectation value for the contents of each channel fitted. Since the measured values follow a probability distribution, their probability

with respect to the expected function is known for each channel. For a certain set of parameters a total probability can be calculated as product of these channel probabilities. The optimal set of parameters is assumed to be most likely. It is determined by searching the maximum of this product in dependence of the free parameters (maximum likelihood).

By default it is assumed that the channel contents follow a gaussian distribution. In this case maximum likelihood is equivalent to the commonly used χ^2–minimization (see section D.3 p. 105). Unfortunately this is not applicable if the expectation values for the contents of single channels are in the order of one. In this case the values are described by a poisson distribution. Obviously the fit function misestimates the contents of the spectrum. This does not depend on the integral of the fitted data but only on the amplitudes, i.e. it does not matter how large the fitted region is.

Besides the χ^2–minimization Tv can do the maximization of the poisson distribution (see section D.3 p. 105). The results determined by this method are correct as long as the spectra have only be incremented or added. For normalized or subtracted spectra the variance of the data following the poisson distribution is not defined and therefore this method is not usable for those spectra. For data distributed according to the gaussian distribution these operations are possible. As a conclusion from the above one can say that none of the measure functions determines correct results, if for example gates with low statistics have been subtracted.

In order to switch between the measure functions use the command:

tv> **fit measure activate** {**dy**–χ^2 | **y**–χ^2 | **poisson**}

4.12.5   Holding parameters constant

Sometimes it is useful to do not fit parameters but keep them constant or in correlation with other parameters. One example is if the energy of one peak in a doublet is known and the other should be fitted, or to set the width of several peaks equal. To do this all steps have to be gone as with the fitting of separated peaks. I.e. set the fit region, background- and peakmarkers and perform the fit.

The short status of the fit (tv> **fit status short**) for two peaks looks as follows:
```
spectrum #15:   ./3_3_11/3_3_11.pry [keV] 2
#0 pos 716.937(25) wdt 2.298(33) vol 7829(144)
#1 pos 720.160(22) wdt 2.298(33) cor[0] vol 8818(149)
```

From this one can see that the positions are not held and the widths are correlated, the width of peak1 is set equal to the width of peak0. Now lets set the energy of peak0 to its known value of 717.3 keV and hold this position while the second peak will be fitted. Use the following commands to set and hold the position:

tv> **fit parameter position0 = 717.3**

tv> **fit parameter position0 hold**

The hotkeys to do this are $\boxed{P}\,\boxed{p}\,\boxed{=}\,\boxed{0}$ and $\boxed{P}\,\boxed{p}\,\boxed{h}\,\boxed{0}$. The short status of this fit looks as follows, which shows that position of peak0 is held:
```
spectrum #15:   ./3_3_11/3_3_11.pry [keV]
#0 pos 717.3(0) hol wdt 2.298(33) vol 7829(144)
#1 pos 720.251(22) wdt 2.298(33) cor[0] vol 8818(149)
```

In the above example the widths of the peaks are correlated. To set correlations the command **equal** is used and the according command is:

---

[2] Thank you Constantin Chitu (Ayak).

> tv> **fit parameter width1 equal width0**

To set all widths free for fitting the following command is used:
tv> **fit parameter width free**

or the hotkey ⟨w⟩⟨f⟩ respectivelly.

All hotkeys to hold, free and set parameters are given in table 4.8.

⟨P⟩⟨p⟩⟨h⟩⟨0⟩	tv> fit para position0 hold;	Hold position of peak 0. This hotkey is available for peaks 1 to 4 either.
⟨P⟩⟨p⟩⟨f⟩⟨0⟩	tv> fit para position0 free;	The position of peak 0 is set free for fitting. This hotkey is available for peaks 1 to 4 either.
⟨P⟩⟨p⟩⟨n⟩⟨0⟩	tv> fit para position0 none;	Peak 0 is removed from the peaklist. This hotkey is available for peaks 1 to 4 either.
⟨P⟩⟨p⟩⟨=⟩⟨0⟩	tv> edit; tv> fit para position0 = ?	The position of peak 0 is set to the queried value. This hotkey is available for peaks 1 to 4 either.
⟨P⟩⟨b⟩⟨h⟩	tv> fit param backgr hold; tv> fit param exponent hold; tv> fit param factor hold;	The background–parameters will not be fitted
⟨P⟩⟨b⟩⟨f⟩	tv> fit param backgr free; tv> fit param exponent free; tv> fit param factor free;	The backfround–parameters will be fitted
⟨P⟩⟨w⟩⟨e⟩	tv> fit parameter width equal;	The widths of all peaks will be correlated
⟨H⟩⟨P⟩	tv> fit parameter position hold;	The peak position will not be fitted
⟨H⟩⟨W⟩	tv> fit parameter width hold;	The peak width will not be fitted
⟨w⟩⟨e⟩	tv> fit parameter width equal;	The widths of all peaks will be correlated
⟨w⟩⟨f⟩	tv> fit parameter width free;	The widths of all peaks will be fitted

Table 4.8: Hotkeys to hold, free and set parameters.

4.13 Saving and loading fits

The results from fits and integrations can be saved to a file. As with the peaksearch one can decide to save the data as binary or asciitext. Again data can be loaded from binaryfiles only.
To save data to the binaryfile <spectrumname>.fit the following command
    tv> **fit write**
will be used or the hotkey ⟨S⟩⟨f⟩. To read data from this file the following commands can be used:
    tv> **fit read bin {position | first | last | next | previous | index}**
    tv> **fit read fit {first | last | next | previous}**
    tv> **fit read integration {first | last | next | previous}**

tv> fit read record {first | last | next | previous}

The first command scans the file <spectrumname>.fit for saved fits and loads them. A fit can be loaded for a certain **position** given by the cursorposition or a channelnumber. To load the first set of information one will usually use the option **first** (hotkey R f f). The options **first** and **last** load all entries from the file "energetically" sorted. If one wants to load a certain fit and knows the position in the file, one can use the option **index** and give the according number as an argument. Besides the markers and parameters for the selected fit, the positions for all saved fits and integrations are read and displayed in the graphic window by their **bin** markers at the upper frame. Using these bins further fits can be loaded easily with the following command:

tv> fit read bin position cursor

For this operation the hotkey R b c can be used either.

The command **tv> fit read fit ...** is intended to load fits only, but it doesn't offer the easy way with the bins.

The third command does the same with integrations.

To load fits and integrations the latter command ca be used. With this command the entries are read from the file in the order they were saved.

All hotkeys for saving and loading fits and integrations are given in table 4.9.

R b c	tv> fit read bin position cursor;	Reads the fit at the cursorposition.
R b f	tv> fit read bin first;	Reads the first fit from the fitfile.
R b n	tv> fit read bin next;	Reads the next fit from the fitfile.
R b p	tv> fit read bin position cursor;	Reads the fit at the cursorposition.
R f f	tv> fit read fit first;	Reads the first fit from the fitfile.
R f n	tv> fit read fit next;	Reads the next fit from the fitfile.
R i f	tv> fit read integration first;	Reads the first integration from the fitfile.
R i n	tv> fit read integration next;	Reads the next integration from the fitfile.
S f	tv> fit write;	Writes a fit or integration to the fitfile.

Table 4.9: Hotkeys for saving and loading of fits and integrations.

To write the results from a fit or integration as asciitext to a protocolfile, the file must be opened first with one of the commands:

tv> fit result–file open <resultfile>

tv> fit result–file append–open <resultfile>

The results are saved with the command

tv> fit print

After writing all results, the file can be closed with the command

tv> fit result–file close

or it will be closed automatically at leaving Tv.

Chapter 5

Matrix analysis

Contents

5.1 Introduction

The analysis of coincident data is important since most *gamma* −spectroscopy experiments yield such data. This chapter explains all necessary operations to open the matrix, attach the according cuts, projections, spectra and buffers. Furthermore it is shown how to cut in a single matrix or in several matrices simultaneous.

5.2 Setup

To create a cut you need a matrix, its x− and y−projection, a set of markers and a buffer for the cut spectrum.

Tv can manage an arbitrary number of matrices, projections, cuts and spectra. By default Tv offers 16 buffers (see section 3.2 p. 15) for storing matrices, cuts, spectra and directories. Their names are composed of the first character of the buffer type and the buffer number, e.g. c0,...,c15 for the cut buffers.

5.2.1   Attachments

Tv needs relations between matrix, spectrum, cut and directory buffers which are called **attachments**.

For example if you create a cut, the associated gates will be applied to the attached matrix and the cut spectrum will be written to the attached spectrum buffer. If you save the cut, the cut spectrum will be written into the attached directory. By default Tv has the attachments listed in table 5.1 on page 48.

c<n>	m0	Matrix m0 is attached to all cuts.
c<n>	s<n>	Spectrum s<n> is attached to cut <n>.
c<n>	d0	Directory d0 is attached to all cuts.
s15	m0	Spectrum buffer 15 contains the projection for matrix 0.

Table 5.1: Tv's default attachments.

5.2.2   Changing the attachments

All changes are for the active cut which is c0 by default. You can select the active cut with the command:

    tv> cut activate <index>

The attachments can be changed with:

    tv> cut attach [matrix <n>] [spectrum <n>] [directory <n>]

The projection attached to a matrix can be changed with:

    tv> cut matrix attach–projection <s>

where <s> is the spectrum buffer containing the projection.

5.2.3   Open a matrix

Now we have defined the attachments between all necessary buffers. They can be loaded with the according data in the following way. To open a matrix enter its index and filename:

    tv> cut matrix open <mtx-idx> <filename>

Load the projection <filename> in buffer <n> with:

    tv> spectrum read <filename> <n>

Define the pathname of the cut directory with:

    tv> cut directory open <dir-idx> <pathname>

5.2.4   Example setup

As an example cut 0 is activated and the according attachments are made to matrix 1, projection 2, spectrum 3 and directory 4.

    tv> cut activate 0
    tv> cut attach matrix 1
    tv> cut matrix attach–projection 2
    tv> cut attach spectrum 3
    tv> cut attach directory 4

Now change to a suitable directory and open the matrix as well as the directory and load the projection.

    tv> cd /matrix1
    tv> cut directory open 4 /220Th/220Th.cutdir
    tv> cut matrix open /220Th/220Th.mtx
    tv> spectrum read /220Th/220Th.pry 2

Examples for the comparison of matrices, where attachments for several matrices are made, are given in chapter 5.6.

5.3 Fast setup

Since the files needed for matrix analysis are always of the same structure, Tv offers a method to make the setup much easier. It is called **cut environment** and uses the wildcards described in section 3.12.2 on page 25, i.e. the files must have standard filenames to be loaded with the environment command.

5.3.1 Definition of environment

The **cut environment** defines a file structure that simplifies the analysis of matrices, i.e. you only have to specify the pathname to access the matrix, load its projection and open the cut directory.

For example to load the environment from the directory */matrix1/220Th* it must contain the files listed in table 5.2 on page 49.

220Th.mtx	The matrix file.
220Th.pr[xy]	The projection files.

Table 5.2: Files that must exist for the **cut environment** command.

5.3.2 Load the environment

If all files exist that are mentioned in the previous section you can load the environment that you want to work with by pressing the hotkey $\boxed{\text{E}}$ and entering the pathname or with the command:

    tv> cut environment /matrix1/220Th

Tv loads the environment to the buffers that are by default set to the values listed in table 5.3. Using the commands from chapter 5.2 one can define more than one environment and perform cuts in several matrices simultaneously.

Buffer type	Default
Cut	0
Matrix	0
Projection	15
Spectrum	0

Table 5.3: Default buffers for the **cut environment** command.

The single commands executed by the cut environment command are:

    tv> cut matrix open /matrix1/220Th/220Th.mtx
    tv> spectrum read /matrix1/220Th/220Th.pry
    tv> cut directory open /matrix1/220Th/220Th.cutdir

5.4 Creating cuts

To create a cut you have to define a set of gates. You can do this in several ways, e.g. with the hotkey $\boxed{\text{c}}$ or the according command:

    tv> cut marker cut enter cursor

The first two markers define the positive gate, whereas the following marker pairs define background gates.

Another way is to enter the markers by their channel or energy. You can switch between channel and energy with the commands:

> tv> **cut scale marker default**
>
> tv> **cut scale marker calibrated**
>
> tv> **cut scale marker uncalibrated**

and the marker is set with the command:

> tv> **cut marker cut enter <u>value</u>**

Since you can set only one positive gate by using the hotkey $\boxed{C}$ Tv offers a possibility to set further gates with the hotkey $\boxed{G}\ \boxed{G}$ or the according command:

> tv> **cut marker gate enter cursor**

Another way to define background–gates is the command:

> tv> **cut marker bg–gate enter <u>value</u>**

To perform the cut press the hotkey $\boxed{C}$ or enter the command:

> tv> **cut create cut**

The background gates are weighted against the positive gates. The more background you mark the better it is, because it reduces the statistical error. The background is weighted by gatewidth by default and can be weighted by volume, too. The commands to switch the weight are:

> tv> **cut weight gatewidth**
>
> tv> **cut weight fit**

It is evident how important it is to choose correct background gates as is shown in figures 5.1 on page 50, 5.2 on page 51 and 5.3 on page 51.
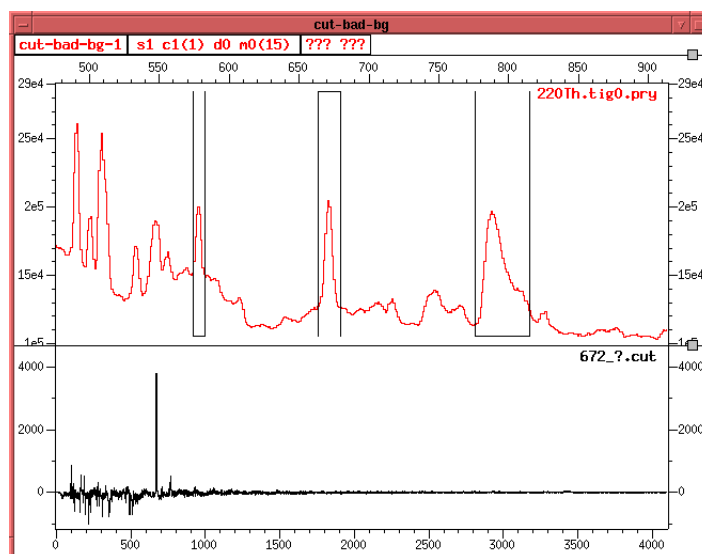
Figure 5.1: Cut without background gates.

If you want to use the cut markers from another cut in the active cut you can do this with the command:

> tv> **cut use–marker <index>**

which copies the gates from cut <index> to the active cut.

If you have found a good set of gates but want to look for an even better one, you can store the current gates and background–gates with the **store** commands:

> tv> **cut marker gate store**
>
> tv> **cut marker bg–gate store**
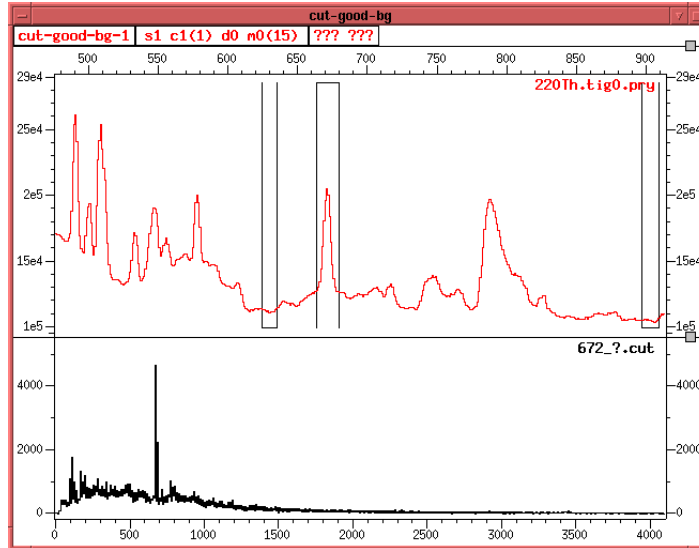
Figure 5.2: Cut with good background gates.
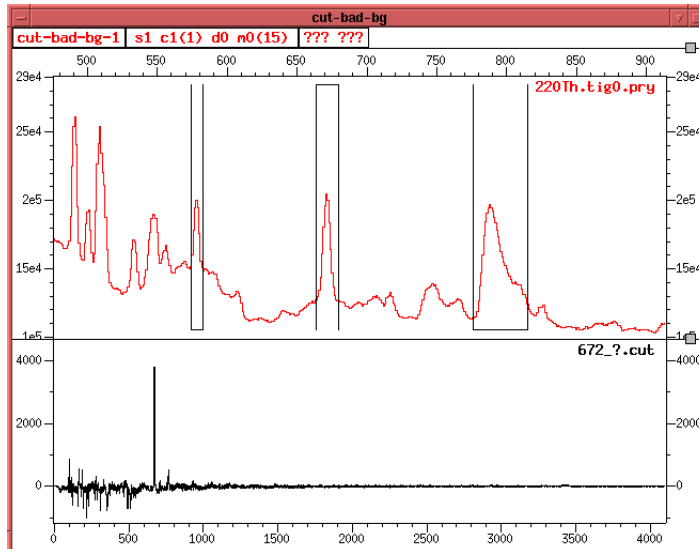
Figure 5.3: Cut with bad background gates.

and resume them with the according **restore** commands.

To perform a new cut you have to delete the gates from the previous cut first. Again you have to separately do it for the gates and background–gates. If you want to delete both you can use the hotkey $\boxed{-}\,\boxed{\text{C}}$.

5.5 Saving and loading cuts

Cuts can be saved to and loaded from disk. Cuts are saved in the cutdirectory where a subdirectory is created for each cut. The name of the subdirectory is given by the energy or channel where the cut is performed and an attached underscore and index. By means of the index more than one cut can be performed at one position. A cut performed at channel 1105 of the ^{220}Th matrix will be saved with the command

  tv> **cut write cut**

or the according hotkey $\boxed{\text{S}}\,\boxed{\text{c}}$ in the cutdirectory *220Th/220Th.cutdir/1105_0*. The markers are written to file *1105_0.gate* and the cutspectrum is saved in file *1105_0.cut*.

To load cuts, different commands are given. One can choose to only load markers or to load markers and spectrum at once. In the first case the loaded markers are used to perform a cut.

The commands to load markers only or markers and spectrum together at the position defined by the cursor are

  tv> **tv> cut read head cursor**
  tv> **tv> cut read cut cursor**

All hotkeys for saving and loading of markers and spectra are given in table 5.4.

|		
$\boxed{\text{R}}\,\boxed{\text{c}}\,\boxed{\text{c}}$	tv> cut read cut cursor;	Reads cutspectrum and set of cutmarkers at cursorposition.
$\boxed{\text{R}}\,\boxed{\text{c}}\,\boxed{\text{f}}$	tv> cut read cut first;	Reads the first cutspecttrum and the first set of cutmarkers.
$\boxed{\text{R}}\,\boxed{\text{c}}\,\boxed{\text{n}}$	tv> cut read cut next;	Reads the next cutspectrum and the next set of cutmarkers.
$\boxed{\text{R}}\,\boxed{\text{h}}\,\boxed{\text{c}}$	tv> cut read head cursor;	Reads the set of cutmarkers at the cursorposition and performs the cut.
$\boxed{\text{R}}\,\boxed{\text{h}}\,\boxed{\text{f}}$	tv> cut read head first;	Reads the first set of cutmarkers and performs the cut.
$\boxed{\text{R}}\,\boxed{\text{h}}\,\boxed{\text{n}}$	tv> cut read head next;	Reads the next set of cutmarkers and performs the cut.
$\boxed{\text{S}}\,\boxed{\text{c}}$	tv> cut write cut;	Writes the cutspectrum and the set of cutmarkers to disk.

Table 5.4: Hotkeys for saving and loading of cutmarkers and cutspectra.

5.6 Comparing matrices

Tv offers some scripts listed in table 5.5 on page 53 to compare spectra, matrices or cuts. Each of these is described in the following subsections. For example to compare two spectra enter:

  tv> **@cmp2spc**

The path of these files is by default */ikp/local/lib/tv* and on some systems */usr/local/lib/tv*.
To find the location of the file for example use the command:

   tv> which cmp2spc

cmp2cut	→ Compare two cuts.
cmp2mat	→ Compare two matrices.
cmp2spc	→ Compare two spectra.
cmp3mat	→ Compare three matrices.
cmp4mat	→ Compare four matrices.

Table 5.5: Sample batchfiles.

5.6.1   cmp2spc

cmp2spc makes attachments for cuts 0 and 1. Matrix and directory 0 as well
as the projection 5 and spectra 0 and 1 are attached. A simple window named
normalized spectra comparison is created. The following hotkeys are defined
(see table 5.6 p. 53).

Hotkey	Tv–commands	Function
N s c	window scaling function–y normalization on;	
	window show spectrum 0 1;	
	window show spectrum - 5;	
	window show spectrum names;	
	window setup histogram–compression each;	
G n	cut activate 1;	
	cut read cut position cursor;	
	cut activate 0;	

Table 5.6: Hotkeys for **cmp2spc**.

5.6.2   cmp2mat

cmp2mat makes attachments for cuts 0 and 1. Matrices, directories and pro-
jections 0 and 1 are attached as well as spectra 4 and 5. The simple windows
projections which shows spectra 0 and 1, and **comparison** which shows spectra
4 and 5 are created. One hotkey is defined (see table 5.7 p. 53) which creates the
cuts for both matrices.

Hotkey	Tv–commands	Function
C	cut activate 0;	
	cut create cut;	
	cut activate 1;	
	cut create cut;	

Table 5.7: Hotkey for **cmp2mat**.

Finally cmp2mat opens the environments for the two matrices given as command
line arguments. For example:

   tv> @cmp2mat 220Th.tig0 220Th.tig1

The scripts **cmp3mat** and **cmp4mat** perform the same operations but for more matrices. They take more arguments, too.

5.6.3   cmp2cut

cmp2cut makes attachments for cuts 0 and 1. Matrices and directories 0 as well as projection 5 and spectra 0 and 1 are attached. Cut windows CUT0 for spectrum 0 and CUT1 for spectrum 1 are created. The projection is shown in tv–root. Some hotkeys are defined as shown in table 5.8 on page 54.

Hotkey	Tv–commands	Function
A 0	cut activate 0; cut create cut;	
A 1	cut activate 1; cut create cut;	
S a	cut activate 0; cut write cut; cut activate 1; cut write cut;	
q	edit; cut activate	

Table 5.8: Hotkey for **cmp2cut**.

Chapter 6

Command summary

Contents

6.1 Introduction

This chapter lists all commands in alphabetic order and gives a short description for each of them. Underlined words are default commands and only their arguments must be entered. Arguments to the <u>value</u> command can be energies (calibrated) or channels (uncalibrated). The current interpretation of the entered value is printed

and you can overwrite it by entering the appropriate word (cal or uncal) after the value.

Commands in Tv are mostly object oriented, i.e. you have to put the object you want to work with to the beginning of the command.

6.2   Calibration

Table 6.1 shows the commands allowed to the calibration menu and their according arguments. In the subsections the commands allowed to the calibration commands are listed.

Command	Arguments
copy	<source index> {{<index>...} | active | all | shown | visible}
delete	{<index>...} | active | all | shown | visible
enter	$<channel_0>$ $<energy_0>$ $<channel_1>$ $<energy_1>$
list	{<index>...}
read	<filename>
write	<filename>

Table 6.1: Commands allowed to the **calibration** menu.

6.2.1   Calibration efficiency

tv> **calibration efficiency {copy | delete | list | <u>read</u> | write}**
Performs an operation on the efficiency's calibration.

6.2.2   Calibration lefttail

tv> **calibration lefttail {copy | delete | enter | list | <u>read</u> | write}**
Performs an operation on the lefttail's calibration.

6.2.3   Calibration <u>position</u>

tv> **calibration position {copy | delete | enter | list | <u>read</u> | write}**
Performs an operation on the position's calibration.

6.2.4   Calibration righttail

tv> **calibration righttail {copy | delete | enter | list | <u>read</u> | write}**
Performs an operation on the righttail's calibration.

6.2.5   Calibration set

tv> **calibration set {copy | delete | list | <u>read</u> | write}**
Performs the operations on the complete **set** of calibrated parameters.

6.2.6   Calibration startchannel

tv> **calibration startchannel {copy | delete | enter}**
Sets startchannel for calibration.

6.2.7 Calibration unit

`tv>` **calibration unit {copy | delete | enter <unit>}**
Sets unit of x–scale, e.g. keV.

6.2.8 Calibration width

`tv>` **calibration width {copy | delete | enter | list | read | write}**
Performs an operation on the width's calibration.

6.3 Cut

6.3.1 Cut activate

`tv>` **cut activate <index>**
Activates cut <index> for following cut commands.

6.3.2 Cut attach

`tv>` **cut attach directory <index>**
Attach directory <index> to the active cut.

`tv>` **cut attach matrix <index>**
Attach matrix <index> to the active cut.

`tv>` **cut attach spectrum <index>**
Attach spectrum <index> to the active cut.

6.3.3 Cut create

`tv>` **cut create <u>cut</u>**
Create head and spectrum.

`tv>` **cut create head**
Create head, i.e. do the gate weight calculation.

`tv>` **cut create spectrum**
Creates a cut spectrum and copies it to the buffer attached to the active cut.
The cut head must be created first.

6.3.4 Cut directory

`tv>` **cut directory close <index>**
Closes cut directory.

`tv>` **cut directory format <format>**
Defines format to save spectra (see section C.2 p. 101).

`tv>` **cut directory <u>open</u> <index> <filename>**
Opens directory <filename> for cut <index>.

`tv>` **cut directory status**
Prints attachments between directories and buffers.

6.3.5 Cut environment

`tv>` **cut environment <pathname>**
Opens cut environment (see section 5.3 p. 49).

6.3.6   Cut list

tv> cut list all
Lists all cuts in the attached cut directory.

tv> cut list range {cursor | <u>value</u> | offset}
Set markers to define a range for which you want to list the cuts.

6.3.7   Cut marker

Command	Function
delete	Delete marker.
enter {cursor | <u>value</u> | offset}	Set new marker.
list	List all markers.
read <filename>	Read list of markers from file.
restore	Restore markers from memory.
store	Store markers to memory.
write <filename>	Save list of markers to file.

Table 6.2: Commands allowed to the **cut marker** menu.

tv> cut marker bg-gate {delete | enter | list | read | restore | store | write}
Performs an operation according to table 6.2 on background–gates.

tv> cut marker cut {delete | enter}
Performs an operation according to table 6.2 on cuts.

tv> cut marker gate {delete | enter | list | read | restore | store | write}
Performs an operation according to table 6.2 on gates.

6.3.8   Cut matrix

tv> cut matrix attach-projection <spectrum>
Attaches spectrum in buffer <spectrum> to the active matrix as its projection.

tv> cut matrix close <index>
Close matrix in buffer <index>.

tv> cut matrix format <format>
Queries the format of the matrix to load (see section C.2).

tv> cut matrix open <index> <filename>
Open matrix for buffer <index>.

tv> cut matrix status
Show a list of all loaded matrices.

6.3.9   Cut read

tv> cut read cut {cursor | <u>entry</u> | <u>position</u> | first | last | next | previous}
Performs the read operation for the cut according to table 6.3.

tv> cut read head {cursor | <u>entry</u> | <u>position</u> | first | last | next | previous}
Performs the read operation for the head according to table 6.3.

Command	Function
cursor	Reads the cut at the cursorposition.
entry	Queries for a directoryname to read data from.
position {cursor | value}	Reads the cut or head it queries for.
first	Reads the first cut in the cudirectory.
last	Reads the last cut in the cutdirectory.
next	Reads the next cut in the cutdirectory.
previous	Reads the previous cut in the cutdirectory.

Table 6.3: Commands allowed to the **cut read** command.

6.3.10   Cut rm

tv> cut rm cut <filename>
Remove cut <filename> from the attached cut directory.

tv> cut rm fit <filename>
Remove fit <filename> from the attached cut directory.

tv> cut rm spectrum <filename>
Remove spectrum <filename> from the attached cut directory.

6.3.11   Cut status

tv> cut status
Prints status information about all cuts and their attachments.

6.3.12   Cut scale

tv> cut scale i/o {calibrated \| default \| uncalibrated}
Adjusts the scale for writing or reading of markers.

tv> cut scale marker {calibrated \| default \| uncalibrated}
Adjusts the scale for marker queries.

6.3.13   Cut use–marker

tv> cut use–marker <index>
The markers from cut <index> will be used for the active cut.

6.3.14   Cut write

tv> cut write cut
Writes the active cutmarkers and cutspectrum.

tv> cut write head
Writes the active cutmarkers.

6.3.15   Cut weight

tv> cut weight gatewidth
The gatewidths are used for the weight calculation.

tv> cut weight fit
The background volumes are used for the weight calculation.

6.4 Fit

6.4.1 Fit activate

`tv>` **fit activate {<u>index <index></u> | next | previous | status}**
Activates spectrum in buffer <index>, the next or previous buffer or prints
status information about loaded buffers and active fits and cuts.

6.4.2 Fit background–create

`tv>` **fit background–create**
Creates a background fit (see section D.2).

6.4.3 Fit bg–function

`tv>` **fit bg–function definition**
Prints the definition of the background–function (see section D.2).

6.4.4 Fit <u>region–create</u>

`tv>` **fit <u>region–create</u>**
Creates a fit for the region defined by region markers.

6.4.5 Fit integration–create

`tv>` **fit integration–create**
Integrates the region defined by region markers.

6.4.6 Fit delete

`tv>` **fit delete**
Deletes the fit and all its markers.

6.4.7 Fit function

`tv>` **fit function background definition**
Prints the definition of the background–function.

`tv>` **fit function peak**

> `tv>` **fit function peak activate {continuous | additive}**
> Queries the fit–function to use (see D).
>
> `tv>` **fit function peak definition {continuous | additive}**
> Prints the definition of the fit–function.
>
> `tv>` **fit function peak exponent {tl | tr}**
> Queries the exponent for the left or right tail term of the fit–function.

6.4.8 Fit list

`tv>` **fit list**
Lists all fits and prints some information about them.

Command	Function
delete	Delete marker.
enter {cursor | <u>value</u> | offset}	Set new marker.
list	List all markers.
read <filename>	Read list of markers from file.
restore	Restore markers from memory.
number <multiple | single>	
store	Store markers to memory.
write <filename>	Save list of markers to file.

Table 6.4: Commands allowed to the **fit marker** menu.

6.4.9 Fit marker

tv> fit marker bin {delete | enter | list | read | restore | store | write}
Performs an operation on bin markers.

tv> fit marker bg–region {delete | enter | list | read | restore | store | write}
Performs an operation on background markers.

tv> fit marker fit delete
Deletes all markers for the active fit.

tv> fit marker peak {delete | enter}
Performs an operation on peak markers.

tv> fit marker region {delete | enter | list | read | restore | number | store | write}
Performs an operation on fit region markers.

6.4.10 Fit measure

tv> fit measure activate {dy–χ^2 | y–χ^2 | poisson}
Activates the selected measure function (see section D.3 p. 105).

tv> fit measure definition {dy–χ^2 | y–χ^2 | poisson}
Prints the definition for the measure function (see section D.3 p. 105).

6.4.11 Fit open

tv> fit open
Opens the file <specfile>.fit and reads all fits and intergrations.

6.4.12 Fit parameter.

tv> fit parameter background {degree | free | hold}
Defines how to fit background (see table 6.5 p. 64).

tv> fit parameter exponent–background {number | free | hold}
Defines how to fit exponent background exponent.

tv> fit parameter factor–background {number | free | hold}
Defines how to fit background factor.

tv> fit parameter position {= | free | hold | none | spaced}
Defines how to fit position.

Command	Argument	Function
degree	\<degree\>	Set degree of background–polynom (see section D.2 p. 105). Default value is 2.
number	{\<value\>}	Set factor or scaling of the exponential term of the background function (see section D.2 p. 105). Default values are 0 for each of them.
free		Set parameter to be fitted.
hold		Hold this parameters value.
=	\<expression\>	Set position to the result from \<expression\>.
none		When fitting position the peak will be removed from peaklist the fit of the tail will be suppressed.
spaced	\<parameter\> \<index\>	?????
calibrated		?????
equal	\<parameter\> \<index\>	Take value from parameter from buffer \<index\>.

Table 6.5: Commands allowed to the **fit parameter** command.

tv> **fit parameter sh** {= | **calibrated** | **free** | **hold** | **equal** | **none**}
Defines how to fit step height.

tv> **fit parameter sw** {= | **calibrated** | **free** | **hold** | **equal** | **none**}
Define how to fit step width.

tv> **fit parameter tl** {= | **calibrated** | **free** | **hold** | **equal** | **none**}
Defines how to fit left tail.

tv> **fit parameter tr** {= | **calibrated** | **free** | **hold** | **equal** | **none**}
Defines how to fit right tail.

tv> **fit parameter volume** {= | **free** | **hold**}
Defines how to fit volume.

tv> **fit parameter width** {= | **calibrated** | **free** | **hold** | **equal**}
Defines how to fit width.

6.4.13   Fit peaklist

tv> **fit peaklist list**
Lists peaklist in fit file.

tv> **fit peaklist print**
Prints peaklist to protocol file.

tv> **fit peaklist read**
Reads peaklist from fit file.

6.4.14   Fit print

tv> **fit print**
Writes fit results to protocol file (see **fit result-file**).

6.4.15   Fit psearch

tv> **fit psearch list**
List all peaks found by the last peaksearch.

`tv>` **fit psearch create**
Create a peaksearch for the entire spectrum if no markers are set.

`tv>` **fit psearch marker {delete | enter {cursor | <u>value</u> | offset} | list | read <filename> | restore | store | write <filename>}**
Handle markers for the peaksearch.

`tv>` **fit psearch open**
Opens the file <specname>.fit and reads the list of peaks and fits saved in it.

`tv>` **fit psearch print [<index>]**
Prints the number of peaks found by the last peaksearch for the spectrum in buffer <index>.

`tv>` **fit psearch probability [<limit>]**
Determine or set the probability limit for the peaksearch.

`tv>` **fit psearch read {<index> | all | shown | active}**

`tv>` **fit psearch scale [calibrated | default | uncalibrated]**
Determine or set if the peaksearch is performed at the calibrated or uncalibrated scale.

`tv>` **fit psearch status**
Prints the number of peaks found by the last peaksearch for all buffers.

`tv>` **fit psearch write**
Writes results from the last peaksearch to the file <specname>.fit.

6.4.16 Fit read

Command	Function
<u>position</u> {cursor	<u>value</u>	offset}	Reads the fit at position.
first	Reads the first fit.
last	Reads the last fit.
next	Reads the next fit.
previous	Reads the previous fit.
<u>index</u>	Reads the fit number <index>.

Table 6.6: Commands in the **fit read** menus.

`tv>` **fit read <u>bin</u> {<u>position</u> | first | last | next | previous | <u>index</u>}**
Reads a fit from file <specname>.fit according table 6.6.

`tv>` **fit read fit {first | last | next | previous}**
Reads a fit from file <specname>.fit according table 6.6.

`tv>` **fit read integration {first | last | next | previous}**
Reads an integration from file <specname>.fit according table 6.6.

`tv>` **fit read record {first | last | next | previous | index}**
Reads an entry from file <specname>.fit according table 6.6.

6.4.17 Fit recover–backup

`tv>` **fit recover–backup {<index> | all | shown | <u>active</u>}**
Recovers fit for the buffers defined by the additional argument. The default value is the active buffer.

6.4.18   Fit restore

tv> **fit restore**
Restores fit from memory (see **fit restore**)

6.4.19   Fit result–file

tv> **fit result–file**

tv> **fit result–file append–open** <**filename**>
Open protocol file <filename> to append fit results printed with **fit print**.

tv> **fit result–file close**
Close protocol file.

tv> **fit result–file format**

Argument	Function
bg–polynom	Format to save the background–polynom to a file.
bg–exponential	Format to save the exponential part of the background to a file.
fit	Format to save the results from a fit to a file.
integration	Format to save the results of an integration to a file.
mark	Format to save markers to a file.
peak–fit	Format to save the results of a peak–fit to a file.
peak–integration	Format to save the results of a peak–integration to a file.
peak–search	Format to save the results of a peaksearch to a file.

Table 6.7: Arguments for the **result–file format** command.

tv> **fit result–file format default** <**arg**>
Sets the format for <arg> to the default.

tv> **fit result–file format enter** <**arg**>
Queries the format for <arg>.

tv> **fit result–file format read** <**arg**> <**filename**>
Reads the format for <arg> from file <filename>.

tv> **fit result–file format status**
Prints the format for all parameters.

tv> **fit result–file format write** <**arg**> <**filename**>
Saves the format for <arg> to the file <filename>.

tv> **fit result–file open** <**filename**>
Open <filename> to write fit results printed with **fit print** or **fit peaklist print**.

6.4.20   Fit status

tv> **fit status full**
Prints detailed status informations about the current fit.

tv> **fit status short**
Prints short status informations about the current fit.

6.4.21 Fit scale

tv> **fit scale input** {**calibrated** | **default** | **uncalibrated**}
Defines how read values are interpreted.

tv> **fit scale marker** {**calibrated** | **default** | **uncalibrated**}
Defines how marker values are interpreted.

tv> **fit scale output** {**calibrated** | **default** | **uncalibrated**}
Defines the scale for writing markers.

6.4.22 Fit store

tv> **fit store**
Stores the current fit to memory (see **fit restore**).

6.4.23 Fit use–fitdata

tv> **fit use–fitdata** **<index>** | **[active]**
The markers from buffer <index> will be used for the active buffer.

6.4.24 Fit write

tv> **fit write**
Saves the fit with all its parameters und markers to the file <specname>.fit.

6.5 Label

Argument	Function
library	?????
calibrated	Calibrated energy is printed.
nocalibrated	Calibrated energy is not printed.
uncalibrated	Uncalibrated energy is printed.
nouncalibrated	Uncalibrated energy is not printed.
string	The info string is printed.
nostring	The info string is not printed.

Table 6.8: Possible arguments for the **parameter** command.

6.5.1 Label cut

tv> **label cut parameter** **<arg>**
Selects which information is printed at a cut label. <arg> is a list of entries
from table 6.8 on page 67.

tv> **label cut status**
Lists which values are printed at cut labels.

tv> **label cut write** **<filename>**
Writes the current list of cut labels to <filename>

6.5.2   Label fit

`tv>` **label fit parameter** **<arg>**
Selects which information is printed at a fit label. <arg> is a list of entries
from table 6.8 on page 67.

`tv>` **label fit status**
Lists which values are printed at fit labels.

`tv>` **label fit write** **<filename>**
Writes the current list of fit labels to <filename>.

6.5.3   Label peaklist

`tv>` **label peaklist parameter** **<arg>**
Selects which information is printed at a peaklist label. <arg> is a list of
entries from table 6.8 on page 67.

`tv>` **label peaklist status**
Lists which values are printed at peak labels.

`tv>` **label peaklist write** **<filename>**
Writes the current list of peak labels to <filename>.

6.5.4   Label user

`tv>` **label user activate** **<title>**
Activate user list <title>

`tv>` **label user create** **<titel>** **[calibrated | uncalibrated]**
Creates labellist <title>.

`tv>` **label user delete** **{all | cursor | list <title>}**
Delete labels from active list or entire list.

`tv>` **label user enter** **{cursor | value | offset}** **[<coloridx> ["string"]]**
Enter a new label at cursor position or by value. The value is interpreted
as energy or channel, depending on how the list was created (see **label user
create** and **label user scale**). The color of the marker can be selected
with the argument <coloridx> (see 4.4). An arbitrary string can be printed
as marker when given as the final argument and selected with **label user
parameter string**

`tv>` **label user parameter** **<arg>**
Selects which information is printed at a user label. <arg> is a list of entries
from table 6.8 on page 67.

`tv>` **label user read**

   `tv>` **label user read merge** **<filename>** **[<title>]**
   Reads a labellist from <filename> and merges it to <title> or the active
   labellist if no title is specified.

   `tv>` **label user read replace** **<filename>** **[<title>]**
   Read a labellist from <filename> and replaces <title> or the active
   labellist if no title is specified.

`tv>` **label user scale** **<title>** **[calibrated | uncalibrated]**
Defines how values are interpreted for user labellists.

`tv>` **label user status {active | existing}**
Lists which values are printed at user labels.

`tv>` **label user write <filename> <title>**
Writes labellist <title> to <filename>.

6.6 Normalization

The normalization as it, is performed with the command (see section 6.8.14 on page 73):
   `tv>` **spectrum norm {{<u><index>...</u>} | active | all | shown | visible}**

6.6.1 Normalization marker

`tv>` **normalization marker delete**
Deletes set of normalization markers.

`tv>` **normalization marker enter {cursor | <u>value</u> | offset}**
The first two markers define the normalization area, pairs of following markers define normalization background areas.

`tv>` **normalization marker list**
Lists all defined normalization markers.

`tv>` **normalization marker read <filename>**
Reads set of normalization markers from file.

`tv>` **normalization marker restore**
Restores previously saved set of normalization markers from memory.

`tv>` **normalization marker store**
Stores set of normalization markers to memory.

`tv>` **normalization marker write <filename>**
Saves set of normalization markers to file.

6.6.2 Normalization off

`tv>` **normalization off**
Turns off normalization so that **spectrum norm** has no effect on any spectrum.

6.6.3 Normalization scale

`tv>` **normalization scale [calibrated | default | uncalibrated]**
Defines whether values are interpreted as energy or channel for normalization markers.

6.6.4 Normalization status

`tv>` **normalization status**
Prints status information how spectra will be normalized.

6.6.5   Normalization <u>type</u>

tv> **normalization type factor** <**factor**> {{<**index**>. . . } | **active** | **all** | **shown** | **visible**}
?????

tv> **normalization type maximum** {{<**index**>. . . } | **active** | **all** | **shown** | **visible**}
Normalize spectra defined by the last argument with reference to the spectrum with maximum channel.

tv> **normalization type reference** <**reference**> {{<**index**>. . . } | **active** | **all** | **shown** | **visible**}
Normalize spectra defined by the last argument with reference to the spectrum in buffer <reference>.

tv> **normalization type unity** {{<**index**>. . . } | **active** | **all** | **shown** | **visible**}
Normalize spectra defined by the last argument with reference to the spectrum with maximum integral.

6.7   Recalibration

6.7.1   Recalibration append

tv> **recalibration append** <**filename**> {<**index**>. . . }
Append recalibration data of buffers <index>. . . to file <filename>.

6.7.2   Recalibration create

tv> **recalibration create** <**reference**> {{<**index**>. . . } | **active** | **all** | **shown** | **visible**}
Shifts spectra defined by the last argument with respect to spectrum in buffer <reference>. All shifted spectra are marked with the character < in the spectrum status.

6.7.3   Recalibration delete

tv> **recalibration delete** {{<u><**index**>. . .</u>} | **active** | **all** | **shown** | **visible**}
Deletes recalibrations for the spectra defined by the argument.

6.7.4   Recalibration marker

tv> **recalibration marker delete** {**all** | <u>**range**</u> {**cursor** | <u>**value**</u> | **offset**}
Delete recalibration region marker.

tv> **recalibration marker** {**enter** {**cursor** | <u>**value**</u> | **offset**} | **list** | **read** <**filename**> | **restore** | **store** | **write** <**filename**>}
Handle recalibration region marker.

6.7.5   Recalibration list

tv> **recalibration list** {{<u><**index**>. . .</u>} | **active** | **all** | **shown** | **visible**}
Lists recalibrations for spectra in buffers defined by the argument.

6.7.6 Recalibration read

tv> recalibration read <filename> {{<u><index></u>...} | active | all | shown | visible}
Reads recalibrations from file <filename> to spectra defined by the last argument.

6.7.7 Recalibration type

tv> recalibration type {center–of–mass | squared–distance | pair | polynom}
Defines which method is used to recalibrate spectra.

6.7.8 Recalibration write

tv> recalibration write <filename> {{<u><index></u>...} | active | all | shown | visible}
Writes recalibrations for spectra defined by the last argument to file <filename>.

6.8 Spectrum

6.8.1 Spectrum activate

tv> spectrum activate {<index> | next | previous | status Activates the spectrum defined by the last argument.

6.8.2 Spectrum add

tv> spectrum add <index> {<u><filename>['<format>]</u> | {<u><index></u>...} | active | all | shown | visible}
Adds buffers defined by the last argument to buffer <index>. This command has one text default or one integer default.

6.8.3 Spectrum analysator

tv> spectrum analysator <hostname>
Read data from host <hostname> which is bibo by default.

6.8.4 Spectrum copy

tv> spectrum copy <source> {<destination> ...}
Copies spectrum from buffer <source> to buffers <destination> ....

6.8.5 Spectrum create

tv> spectrum create <name> <resolution> <index>
Creates an empty spectrum <name> with resolution <resolution> in buffer <index>.

6.8.6 Spectrum delete

tv> spectrum delete {<index> ...}
Prints a list of spectra and asks for a list of buffers to delete.

6.8.7 Spectrum enter

tv> spectrum enter <index> <channel> <value>
Set channel <channel> of spectrum in buffer <index> to value <value>.

6.8.8 Spectrum format

tv> spectrum format input <format>
Set input format of spectra to <format>.

tv> spectrum format output <format>
Set output format of spectra to <format>.

6.8.9 Spectrum get

tv> spectrum get <filename>
Reads spectra from file(s) <filename> to the next free buffer(s).

6.8.10 Spectrum ls–file

To address a file in an ls–file the following commands are possible:

Command	Function
ls–filename	Name of ls–file to process.
+	Use next spectrum in list.
++	Use all spectra from here to the end of the ls–file.
-	Use previous spectrum.
--	Use all previous spectra to the beginning of the ls–file.
line–index	Use spectrum at position line–index.

Table 6.9: Possible arguments for the **ls–file** commands.

tv> spectrum ls–file add <index> <arg>
Add the spectra according to <arg> to buffer <index>.

tv> spectrum ls–file close
Close currently open ls–file.

tv> spectrum ls–file commandget <commandfile> <arg>
Perform commands from <commandfile> to the spectra defined by <arg>.

tv> spectrum ls–file get <arg>
Load the spectra addressed with <arg>.

tv> spectrum ls–file list
Show current pointer–position in ls–file.

tv> spectrum ls–file open <ls–filename>
Open the ls–file <ls–filename> for operation.

tv> spectrum ls–file position <index>
Position the pointer in the opened ls–file to <index>.

tv> spectrum ls–file subtract <index> <arg>
Subtract spectra addressed by <arg> from spectrum in buffer <index>.

6.8.11 Spectrum maximum

tv> spectrum maximum \<index\> {{<u>\<index\></u>...} | active | all | shown | visible}
Sorts spectrum with maximum peak height from spectra defined by the last argument to the spectrum in buffer \<index\>.

6.8.12 Spectrum minimum

tv> spectrum minimum \<index\> {{<u>\<index\></u>...} | active | all | shown | visible}
Sorts spectrum with minimum peak height from spectra defined by the last argument to the spectrum in buffer \<index\>.

6.8.13 Spectrum multiply

tv> spectrum multiply \<factor\> {{<u>\<index\></u>...} | active | all | shown | visible}
Multiply spectra defined by the last argument with the factor \<factor\>.

6.8.14 Spectrum norm

tv> spectrum norm {{<u>\<index\></u>...} | active | all | shown | visible}
Normalizes the spectra defined by the argument (see also 6.6). All normalized spectra are marked in the spectrum status with the character *.

6.8.15 Spectrum offset

tv> spectrum offset \<offset\> {{<u>\<index\></u>...} | active | all | shown | visible}
Adds offset \<offset\> to spectra defined by the last argument.

6.8.16 Spectrum read

tv> spectrum read \<filename\> {{<u>\<index\></u>...} | active | all | shown | visible}
Reads a single spectrum \<filename\> to buffers defined by the last argument.

6.8.17 Spectrum resolution

tv> spectrum resolution \<index\> \<resolution\>
Changes resolution of spectrum in buffer \<index\> to \<resolution\>.

6.8.18 Spectrum status

tv> spectrum status
Prints status information about all loaded spectra. The last coloumn informs about which operations have been performed on the spectra. The meaning of the special characters used there are listed in table 6.10.

Tag	Meaning
*	Spectrum has been normalized or multiplied.
<	Spectrum has been recalibrated.
+	Spectrum has been added, subtracted or the offset operation has been performed.
&	Spectrum has been used for maximization.

Table 6.10: Meaning of the special characters used in spectrum status output.

6.8.19   Spectrum subtract

tv> **spectrum subtract** <**index**> {<**filename**>['<**format**>] | {<u><**index**></u>...}
| **active** | **all** | **shown** | **visible**}
Subtracts buffers defined by the last argument from buffer <index>. This
command has one text default or one integer default.

6.8.20   Spectrum update

tv> **spectrum update** {<**index**> | <u>**all**</u> | **shown** | **active**}
!!!!!!!!!!!!!! ATTENTION !!!!!!!!!!!!!!!
Reloads all queried spectra from harddisk without asking.

6.8.21   Spectrum write

tv> **spectrum write** <**filename**> {{<u><**index**></u>...} | **active** | **all** | **shown** |
visible}
Writes buffers defined by the last argument to the single spectrum <filename>.

6.9   Window

6.9.1   Window create

See section 4.2.1 on page 28 for a desription of window types, which are created
with this command.

tv> **window create simple** <**window name**>

tv> **window create cut** <**window name**>

tv> **window create fit** <**window name**>

tv> **window create paned** <**window name**> <**# panes**>

tv> **window create xpaned** <**window name**> <**# panes**>

tv> **window create ypaned** <**window name**> <**# panes**>

6.9.2   Window delete

tv> **window delete** <**window name**>
Deletes window <window name>

6.9.3   Window hide

Hides the object in active window, which is given by the additional arguments.

tv> **window hide cut–gate–marker**

tv> **window hide fit** {**bin–list** | **decomposition** | **function/marker**}

`tv>` **window hide marker {cut | fit | normalization | recalibration | psearch}**

`tv>` **window hide labellist {cut | peaklist | user}**

`tv>` **window hide peaklist**

`tv>` **window hide spectrum–names**

6.9.4 Window list

`tv>` **window list**
Prints a list of graphic windows.

6.9.5 Window marker

`tv>` **window marker horizontal {delete | enter {cursor | value} | list | read <filename> | restore | store | write <filename>}**
Handles horizontal markers in active window.

`tv>` **window marker vertical {delete | enter {cursor | value} | list | read <filename> | restore | store | write <filename>}**
Handles vertical markers in active window.

6.9.6 Window plot

`tv>` **window plot create**

> `tv>` **window plot create unix–plot [filename]**
> Creates a plot in unix–plot format and writes the output to <filename>.
> If no filename is defined the filename according to the wildcard setting
> will be used.

> `tv>` **window plot create xfig [filename]**
> Creates a plot in xfig format (protocol 3.2).

`tv>` **window plot format**

> `tv>` **window plot format cm–format <x> <y>**
> Defines the plot size in cm.

> `tv>` **window plot format px–format <x> <y>**
> Defines the plot size in pixels.

> `tv>` **window plot format pts/inch <x> <y>**
> Defines the resolution for the plot in points per inch.

> `tv>` **window plot format fontscale–factor <factor>**
> Defines the scale factor for fonts.

6.9.7 Window raise

`tv>` **window raise <window name>**
Raises window <window name>

6.9.8 Window redisplay

`tv>` **window redisplay**
Refreshes the active window.

6.9.9   Window scaling

tv> window scaling function–y

> **tv> window scaling function–y linear**
> This command prepares linear y–scales.
>
> **tv> window scaling function–y logarithmic**
> This command prepares logarithmic y–scales.
>
> **tv> window scaling function–y normalization {on | off}**
> This command switches to prepared y–scales.
>
> **tv> window scaling function–y squared**
> This command prepares quadratic y–scales.

tv> window scaling modification–y

> **tv> window scaling modification–y no–modification**
> Cancels all y–scale modifications.
>
> **tv> window scaling modification–y efficiency**
> Stretches y–scale according to the efficiency calibration?????
>
> **tv> window scaling modification–y multiplied**
> Multiplies the contents of each channel with the channel number.

tv> window scaling scale–x [calibrated | default | uncalibrated]
Defines whether entered values are interpreted as energies or channels.

6.9.10   Window setup

tv> window setup cursor

> **tv> window setup cursor cross–hair**
> Sets graphic cursor to cross–hair.
>
> **tv> window setup cursor vertical**
> Changes nothing.
>
> **tv> window setup cursor subwindow**
>
> > **tv> window setup cursor subwindow cross–hair**
> > Sets graphic cursor to cross–hair.
> >
> > **tv> window setup cursor subwindow doppler <factor>**
> > Changes cursor to a grid with a number of elements of 1 and offset
> > depending on cursor position. The default value for <factor> is 0.
> >
> > **tv> window setup cursor subwindow grid <number> <offset>**
> > Changes cursor to a grid with <number> of lines to the right and
> > to the left of the vertical cursor line. The distance between the
> > lines is <offset> scale units. The default values for <number> and
> > <offset> are 0.

tv> window setup frame

> **tv> window setup frame distances <tic> <label>**
> Defines the minimum distances between tics and labels. Default values
> are 0.3 characters (chars) for tics and 3 chars for labels.
>
> **tv> window setup frame lengths <short> <medium> <long> <dist>**
> Defines the short, medium and long tic lengths and the distance between
> tic and axis–label. Default values are 0.2, 0.4 and 0.6 chars for the tic
> lengths and 0.75 chars for the distance.

`tv>` **window setup frame widths <left> <right> <bottom> <top>**
Defines the widths of the left, right, bottom and top frame. Default values are 3, 3, 2 and 2 chars in the order of arguments.

`tv>` **window setup histogram–compression**

`tv>` **window setup histogram–compression each**
Compresses a group of channels by taking the average of all these channels.

`tv>` **window setup histogram–compression first–found**
Compresses a group of channels by taking only the first of these channels.

`tv>` **window setup histogram–compression min–max**
Compresses a group of channels by taking only the channel with the minimum or maximum number of counts depending on the sign.

`tv>` **window setup keyboard–focus {<u>title <subwindow></u> | cursor | none}**
Selects the window where the input entered in the text–window will be evaluated. It can be a <subwindow>, the window where the cursor is positioned or none.

6.9.11 Window show

`tv>` **window show cut**

`tv>` **window show cut spectrum <index>**
Display the spectrum attached to the active cut.

`tv>` **window show cut projection**
Display the projection attached to the active cut.

`tv>` **window show cut gate–marker**
Display the gate–markers for the activ cut.

`tv>` **window show fit**

`tv>` **window show fit bin–list**

`tv>` **window show fit decomposition**
Display the decomposition of the fit.

`tv>` **window show fit function/marker**
Display the fit function and markers.

`tv>` **window show labellist**

`tv>` **window show labellist cut**
Display the labellist for the active cut.

`tv>` **window show labellist peaklist**
Display the labellist for all peaks in the active peaklist.

`tv>` **window show labellist user**
Display all labels from the user labellist.

`tv>` **window show marker**

`tv>` **window show marker cut**
Display markers for the active cut.

`tv>` **window show marker fit**
Display markers for the active fit.

> `tv>` **window show marker normalization**
> Display normalization markers.

> `tv>` **window show marker recalibration**
> Display recalibration markers.

> `tv>` **window show marker psearch**
> Display peaksearch markers.

`tv>` **window show peaklist**

`tv>` **window show spectrum**

> `tv>` **window show spectrum index <index>**
> Show spectrum in buffer <index>.

> `tv>` **window show spectrum next**
> Show next spectrum.

> `tv>` **window show spectrum previous**
> Show previous spectrum.

> `tv>` **window show spectrum + {<index>...}**
> Show additional spectrum.

> `tv>` **window show spectrum - {<index>...}**
> Hide spectrum.

> `tv>` **window show spectrum names**
> Show one spectrum (start with spectrum in lowest buffer).

`tv>` **window show status**
Prints the number of active spectrum and cut as well as a list of all spectra
shown in the active window.

6.9.12   Window status

`tv>` **window status**
Prints a list of graphic windows.

6.9.13   Window view

`tv>` **window view center**

> `tv>` **window view center both <x–value> <y>**
> Centers displayed spectra in the current window section around the en-
> ergy or channel <x–value> and the counts <y>.

> `tv>` **window view center x <x–value>**
> Centers displayed spectra in the current window section around the en-
> ergy or channel <x–value>.

> `tv>` **window view center y <y>**
> Centers displayed spectra in the current window section around the
> counts <y>.

`tv>` **window view expand**
Expands region between markers set with **window mark vertical enter** or
the **window view full** commands.

`tv>` **window view full**

> `tv>` **window view full both**
> Prepares window to display spectra in full x– and y–size.
>
> `tv>` **window view full x**
> Prepares window to display spectra in full x–size.
>
> `tv>` **window view full y**
> Prepares window to display spectra in full y–size.

`tv>` **window view position <width> <position>**
Centers displayed spectra in window around the energy or channel <position>. The number of displayed channels is given by <width>.

`tv>` **window view shift <x–percent> <y–percent>**
Shifts window section by <x–percent> along the x–axis and <y–percent> along the y–axis.

`tv>` **window view stretch <-log$_2$(x–fac)> <cursor | x–value> <-log$_2$(y–fac)> <cursor | y–value>**
Stretches window section by <-log$_2$(x–fac)> in x–direction where the center is given by <x–value> or cursor position and by <-log$_2$(y–fac)> in y–direction where the center is given by <y–value> or the cursor position.

`tv>` **window view reset**

> `tv>` **window view reset both**
>
> `tv>` **window view reset x**
>
> `tv>` **window view reset y**

6.10 Miscellanous

6.10.1 alias

`tv>` **alias [<command> [<alias> [<arg–list>]]]**
Execute <alias> if <command> is entered. <arg–list> is printed if a question mark is entered as argument to <command>. If no argument is given to alias, all alias definitions are printed.

6.10.2 ReadMe

`tv>` **ReadMe**
Prints a copyright information and a list of allowed command line arguments.

6.10.3 cd

`tv>` **cd <pathname>**
Changes the current working directory to <pathname>. If no pathname is entered you are queried for input and the current working directory's pathname is used as default.

6.10.4 command–file

`tv>` **command–file close**
Closes a command–file.

`tv>` **command–file open <filename>**
Opens the command–file speccified by <filename>.

tv> command–file resume
Resumes the graphic and executes the commands from an open command–file.

tv> command–file suspend
Suspends the graphic and executes the commands from an opened command–file.

6.10.5   edit–lock

tv> edit–lock

Some commands need alphanumerical input which is not possible in cursor mode. edit–lock escapes from cursor mode until the next $\boxed{\text{RETURN}}$.

6.10.6   graphic

tv> graphic resume

tv> graphic resume–forced

tv> graphic suspend

You may switch off the graphical output to accelerate automated operations, e.g. the time shifting of projections. suspend and resume have to be used in pairs, i.e. if you have suspended the graphic n times you have to resume it n times. The suspend counter can be resetted with resume–forced.

6.10.7   input–mode

tv> input–mode cursor
Switches the input–mode to cursor–mode (see section 3.4.2 p. 18).

tv> input–mode edit
Switches the input–mode to edit–mode (see section 3.4.1 p. 18)

6.10.8   keyalias

tv> keyalias [<keysequence> [<alias>]]
Without arguments the entire table of hotkeys (see chapter A p. 83) is shown. If a keysequence is given as argument only the alias for this hotkey is shown and if an alias is given besides the keysequence it will be added to the hotkey table if not existent or redefined in the other case.

6.10.9   keyunalias

tv> keyunalias <keysequence>
The <keysequence> is deleted from the hotkey table.

6.10.10   keytable

tv> keytable activate <title>
Deactivates the currently used keytable and activates the keytable named <title>.

tv> keytable ascii [<char–value> [<char–value>]]
Prints a list of ascii codes in five formats. The values printed are decimal, hexadecimal, octal, a 'symbolic name' and the key to be pressed. The circumflex stands for the control key.

`tv>` **keytable create <title>**
Creates an empty keytable.

`tv>` **keytable delete <title>**
Deletes the keytable named <title>.

`tv>` **keytable list**
Prints the active hotkey table.

`tv>` **keytable read**

> `tv>` **keytable read merge <filename> [<title>]**
> Merges the keytable read from <filename> to the keytable <title> or
> the active keytable if no title is specified.

> `tv>` **keytable read replace <filename> [<title>]**
> Replaces the keytable specified by <title>, or the active one if no title
> is specified, with the one read from <filename>

`tv>` **keytable status**
Prints a list of existing keytables and the files they are saved in.

`tv>` **keytable write <filename>**
Writes active keytable to <filename>.

6.10.11 precision

`tv>` **precision <digits>**
Sets the precision of marker output format, i.e. the number of digits after the
comma. It must be an integer value.

6.10.12 protocol

`tv>` **protocol command {close | open <filename>}**
All commands entered are recorded to the protocol file <filename>.

`tv>` **protocol session {close | open <filename>}**
All commands entered as well as all output from Tv are recorded to the
protocol file <filename>.

`tv>` **protocol standard–output {close | open}**
All commands entered as well as all output from Tv are printed to standard–
output.

6.10.13 which

`tv>` **which [config–file | command–file]**
which without arguments prints the search path for configuration– and command–
files. If a filename is given as argument, **which** prints the full path of the file
used by Tv.

6.10.14 wildcard

`tv>` **wildcard delete <wildcard>**
Deletes <wildcard> from list.

`tv>` **wildcard enter <wildcard> <expression>**
Adds a new wildcard to the list or replaces an existing one.

tv> wildcard status
Prints a list of defined wildcards.

See section 3.12.2 on page 25 for further explanations.

6.10.15   exit

6.11   Default aliases

6.11.1   quit

tv> quit
Performs the command:

tv> exit

6.11.2   lin

tv> lin
Changes the y–scale to linear units by performing:

tv> window scaling function–y linear;

tv> window scaling function–y normalization on;

tv> window scaling function–y normalization off;

6.11.3   log

tv> log
Changes the y–scale to logarithmic units by performing:

tv> window scaling function–y logarithmic;

tv> window scaling function–y normalization on;

tv> window scaling function–y normalization off;

tv> ysqr
Changes the y–scale to squared units by performing:

tv> window scaling function–y squared;

tv> window scaling function–y normalization on;

tv> window scaling function–y normalization off;

Appendix A

The default hotkey table

By default the file *.tvkeys* has the following contents.

Hotkey	Tv–commands	Function
Strg c	tv> input–mode edit;	Switch to edit–mode when in cursor–mode, quit Tv else
Strg l	tv> window redisplay;	Refresh display
Strg o	tv> window redisplay;	Refresh display
Esc	tv> input-mode edit;	Switch between edit– and cursor–mode
Spacebar	tv> window mark vertical enter cursor;	Set vertical markers to define x–range to expand
- 1	tv> window view full y; tv> window view stretch -1 cursor 0;	Zoom out, show 50% of channels
- 2	tv> window view full y; tv> window view stretch -2 cursor 0;	Zoom out, show 25% of channels
- 3	tv> window view full y; tv> window view stretch -3 cursor 0;	Zoom out, show 12.5% of channels
- 4	tv> window view full y; tv> window view stretch -4 cursor 0;	
- 5	tv> window view full y; tv> window view stretch -5 cursor 0;	
- 6	tv> window view full y; tv> window view stretch -6 cursor 0;	
- 7	tv> window view full y; tv> window view stretch -7 cursor 0;	
- 8	tv> window view full y; tv> window view stretch -8 cursor 0;	
- 9	tv> window view full y; tv> window view stretch -9 cursor 0;	
+	tv> label user enter cursor;	
- A	tv> cut mark cut delete; tv> window hide mark cut; tv> fit mark fit delete; tv> fit delete; tv> window hide mark fit; tv> label user delete all; tv> normalization mark delete;	Delete all markers. You will still see markers which are part of the autonomous fit
To be continued ...		

Hotkey	Tv-commands	Function
	tv> recalibration marker delete all;	
- C	tv> cut mark cut delete; tv> window hide mark cut;	Delete cut markers
- D	tv> window hide fit decomposition;	Hides the decomposition of a fit.
- F	tv> fit mark fit delete; tv> fit delete; tv> fit delete activ; tv> window hide mark fit;	Delete the current fit and all its markers
- P	tv> fit mark peak delete 0 uncalib 8192 un-calib;	Delete all peak markers
- p	tv> fit mark peak delete cursor;	Delete peak marker closest to cursor
- B	tv> fit mark bg-region delete all;	Delete all background markers
- b	tv> fit mark bg-region delete cursor;	Delete background marker closest to the cursor
- c g	tv> cut mark gate delete cursor;	Delete gate marker closest to cursor
- c b	tv> cut mark bg-gate delete cursor;	Delete background-gate marker closest to cursor
- l	tv> label user delete cursor;	Delete user label
- n	tv> normalization mark delete;	Delete the normalization markers
- r	tv> fit mark region delete cursor;	Delete fit-region marker closest to cursor
- R	tv> recalibration marker delete cursor;	Delete recalibration marker closest to cursor
- w n	tv> window scaling function-y normalization off;	
,	tv> window view full y; tv> window view shift -70.0 0.0;	Move the viewport to the left by 70% of the visible channels
.	tv> window view full y; tv> window view shift 70.0 0.0;	Move the viewport to the right by 70% of the visible channels
0	tv> window view full y; tv> window view stretch -1 cursor 0;	Zoom out, show 50% of channels
1	tv> window view full y; tv> window view stretch 1 cursor 0;	Zoom in, show 200% of channels
2	tv> window view full y; tv> window view stretch 2 cursor 0;	Zoom in, show 400 % of channels
3	tv> window view full y; tv> window view stretch 3 cursor 0;	
4	tv> window view full y; tv> window view stretch 4 cursor 0;	
5	tv> window view full y; tv> window view stretch 5 cursor 0;	
6	tv> window view full y; tv> window view stretch 6 cursor 0;	
7	tv> window view full y; tv> window view stretch 7 cursor 0;	
8	tv> window view full y; tv> window view stretch 8 cursor 0;	
To be continued ...		

Hotkey	Tv-commands	Function
9	tv> window view full y; tv> window view stretch 9 cursor 0;	
<	tv> window view full y; tv> window view shift -70.0 0.0;	
=	tv> spectrum update all;	
>	tv> window view full y; tv> window view shift 70.0 0.0;	
?	tv> keyalias;	List all hotkeys in text-window
B	tv> fit param backgr free; tv> fit param expo free; tv> fit param fact free; tv> fit background-create; tv> fit param backgr hold; tv> fit param expo hold; tv> fit param fact hold; tv> window show fit function/marker;	
C	tv> cut create cut;	
D	tv> window show fit decomposition;	
E	tv> edit; tv> calibration position read *cal;	Reads a calibration files for all spectra loaded.
F	tv> fit region-create active; tv> fit status short; tv> window show fit function/marker;	
G f	tv> fit read bin position cursor; tv> window show fit function/marker;	
G c	tv> cut read cut position cursor;	
G h	tv> cut read head position cursor;	
G G	tv> cut marker gate enter cursor;	Defines a gate marker.
H P	tv> fit parameter position hold;	Prevent peak position from being fitted
H W	tv> fit parameter width hold;	Prevent peak width from being fitted
I	tv> fit integration-create active; tv> fit status short;	Integrate active spectrum and print short status
L	tv> window setup keyboard-focus cursor;	Set keyboard to graphic-window selected by cursor
M	tv> edit; tv> window create cut Cut; tv> cut environment	Opens a cut-window named Cut and queries an environment to load.
N n tv> window redisplay;	tv> window show spectrum next;	Display next spectrum in buffer list
N p tv> window redisplay;	tv> window show spectrum previous;	Display previous spectrum in buffer list
P b h	tv> fit param backgr hold;	The background-parameters
To be continued ...		

Hotkey	Tv–commands	Function
	tv> fit param exponent hold; tv> fit param factor hold;	will not be fitted
`P` `b` `f`	tv> fit param backgr free; tv> fit param exponent free; tv> fit param factor free;	The background–parameters will be fitted
`P` `p` `h` `0`	tv> fit para position0 hold;	Hold position of peak 0. This hotkey is available for peaks 1 to 4 either.
`P` `p` `f` `0`	tv> fit para position0 free;	The position of peak 0 is set free for fitting. This hotkey is available for peaks 1 to 4 either.
`P` `p` `n` `0`	tv> fit para position0 none;	Peak 0 is removed from the peak-list. This hotkey is available for peaks 1 to 4 either.
`P` `p` `=` `0`	tv> edit; tv> fit para position0 = ?	The position of peak 0 is set to the queried value. This hotkey is available for peaks 1 to 4 either.
`P` `w` `e`	tv> fit parameter width equal;	
`P` `u`	tv> edit; tv> window plot create unix-plot	Plots a unix–plot figure to a selectable file.
`P` `x`	tv> edit; tv> window plot create xfig	Plots a xfig v2.1 figure to a selectable file.
`Q`	tv> fit mark fit delete; tv> fit delete; tv> fit mark peak enter cursor; tv> fit mark region enter offset -10 offset 20; tv> window view full y; tv> fit region-create; tv> fit status short; tv> window show fit function/marker;	Creates a quick fit.
`R` `b` `c`	tv> fit read bin position cursor;	Reads fit at cursor position.
`R` `b` `f`	tv> fit read bin first;	Reads first fit from fitfile.
`R` `b` `n`	tv> fit read bin next;	Reads next fit from fitfile.
`R` `b` `p`	tv> fit read bin position cursor;	Reads fit at cursor position.
`R` `c` `c`	tv> cut read cut cursor;	Reads cut spectrum and a set of cut markers at the cursor position.
`R` `c` `f`	tv> cut read cut first;	Read the first cut spectrum and the first set of cut markers.
`R` `c` `n`	tv> cut read cut next;	Reads the next cut spectrum and the next set of cut markers.
`R` `f` `f`	tv> fit read fit first;	Read the first fit from fitfile.
`R` `f` `n`	tv> fit read fit next;	Reads the next fit from fitfile.
`R` `h` `c`	tv> cut read head cursor;	Reads the set of cut markers at cursor position and performs the cut.
`R` `h` `f`	tv> cut read head first;	Reads the first set of cut markers and performs the cut.
`R` `h` `n`	tv> cut read head next;	Reads the next set of cut markers and performs the cut.
To be continued ...		

Hotkey	Tv–commands	Function
`R` `i` `f`	tv> fit read integration first;	Reads the first integration from fitfile.
`R` `i` `n`	tv> fit read integration next;	Reads the next integration from fitfile.
`R` `r`	tv> recalibration marker enter cursor;	Set a recalibration marker at cursor position
`R` `R`	tv> edit; tv> recalibration create;	Queries a reference buffer and creates the recalibration
`S` `f`	tv> fit write;	
`S` `c`	tv> cut write cut;	Writes a cut to file
`U`	tv> window setup keyboard-focus none;	
`a`	tv> edit; tv> spectrum activate;	Queries a buffer number and activates the according spectrum
`b`	tv> fit mark bg-region enter cursor;	Set background–region marker at cursor position
`c`	tv> cut mark cut enter cursor;	Set cut marker at cursor position
`e`	tv> window view expand;	Expand viewport to the x–range between markers set with `Spacebar`
`f`	tv> window view full both; tv> window view expand;	Expand visible part to full size
`g`	tv> edit; tv> spectrum get	Queries a spectrum name and reads it
`h`	tv> window mark horizontal enter cursor;	Set a horizontal marker
`i`	tv> edit; tv> window view full y; tv> window view position 100;	Queries a position by energy and displays 100 channels with the given position in center
`k`	tv> edit; tv> spectrum delete;	Queries a buffer number and deletes the according spectrum
`l` `+`	tv> spectrum delete all; tv> spectrum ls-file get ++;	Deletes all spectra and reads according to the ls–file rules (see 6.8.10 p. 72)
`l` `-`	tv> spectrum delete all; tv> spectrum ls-file get --;	Deletes all spectra and reads according to the ls–file rules (see 6.8.10 p. 72)
`l` `g`	tv> edit; tv> spectrum ls-file get;	Asks for spectra to be loaded and reads according to the ls–file rules (see 6.8.10 p. 72)
`l` `o`	tv> edit; tv> spectrum ls-file open;	Queries an ls–file name and opens it
`m` `P`	tv> window show peaklist; window show labellist peaklist; tv> fit open active;	
`m` `b`	tv> window show fit bin-list; tv> fit open active;	Display bin markers
`m` `d`	tv> window show fit decomposition; tv> fit open active;	Display decomposition of the fit for the active spectrum
`m` `f`	tv> window show fit function/marker;	Display fit function and marker
To be continued …		

Hotkey	Tv–commands	Function
m n	tv> window show spectrum next;	Display next spectrum in bufferlist
m p	tv> window show spectrum previous;	Display previous spectrum
m s	tv> edit; tv> window show spectrum index;	Queries a list of buffer numbers and displays the according spectra
n	tv> edit; tv> window show spectrum index;	Queries a list of buffer numbers and displays the according spectra
o	tv> normalization mark enter cursor;	Set normalization marker at cursor position
p	tv> fit mark peak enter cursor;	Set a peak marker at cursor position
r	tv> fit mark region enter cursor;	Set fit–region marker at cursor position
s	tv> fit status full;	Print all status information about the fit
u P	tv> window hide peaklist; window hide labellist peaklist;	
u b	tv> window hide fit bin–list;	Do not display bin markers
u c	tv> window hide cut–gate–marker;	Do not display cut gate markers
u d	tv> window hide fit decomposition;	Do not display the decomposition of the fit
u f	tv> window hide fit function/marker;	Do not display fit function and markers
u m c	tv> window hide marker cut;	Do not display cut markers
u m f	tv> window hide marker fit;	Do not display fit–region markers
u m n	tv> window hide marker normalization;	Do not display normalization markers
u m r	tv> window hide marker recalibration;	Do not display recalibration markers
u m s	tv> window hide marker psearch;	Do not display peaksearch– region markers
w e	tv> fit parameter width equal;	The widths of all peaks will be correlated
w f	tv> fit parameter width free;	Allow width to be fitted
w i	tv> window scaling function-y linear; tv> window scaling function-y normalization on; tv> window scaling function-y normalization off;	Switch y–axis to linear scaling.
w l	tv> window scaling function-y logarithmic; tv> window scaling function-y normalization on; tv> window scaling function-y normalization off;	Switch y–axis to logarithmic scaling.
w n	tv> window scaling function-y normalization on;	
To be continued ...		

Hotkey	Tv-commands	Function
w s	tv> window scaling function-y squared; tv> window scaling function-y normalization on; tv> window scaling function-y normalization off;	Switch y-axis to quadratic scaling.
x	tv> window view full x; tv> window view expand;	Expand viewport to full size in x-direction
y	tv> window view full y; tv> window view expand;	Expand viewport to full size in y-direction
		tv> window view full y; tv> window view center x cursor;	Center the spectra around position defined by cursor

Appendix B

Xresources

B.1 Application preferences

By the use of Xresources **application preferences** (e.g. colors) for Tv can be set
quickly and easily. For a detailed description of handling Xresources read the **X(1)**
manpage. The configuration is read from the files

1. app–defaults/Xtv

2. ~/.Xdefaults

3. ~/.Xresources

After changing one of these files you have to reread it to let the changes come to
effect.

B.2 Resources

B.2.1 Sample Xtv file

```
Xtv*xrdb-class:                 CLASS
Xtv*xrdb-planes:                PLANES
Xtv*Font:                       -misc-*-bold-r-*-*-13-*-*-*-*-80-*-*
Xtv*allowResize:                True
Xtv*Linewidth:                  0
Xtv*monochrome*Foreground:      white
Xtv*monochrome*Background:      black
Xtv*gray*Foreground:            white
Xtv*gray*Background:            black
Xtv*colored*Foreground:         yellow
Xtv*colored*Background:         black
tv.geometry:                    650x300-0-0
Xtv*geometry:                   500x500-0+0
Xtv*NumGCs:                     16
Xtv.comparison.geometry:                        500x500+0+0
Xtv.comparison*colored*spectrum.foreground0:    yellow
Xtv.comparison*colored*spectrum.foreground1:    magenta
Xtv.projections.geometry:                       500x500-0+0
Xtv.projections*colored*spectrum.foreground4:   yellow
Xtv.projections*colored*spectrum.foreground5:   magenta
Xtv*cut*XFrame:         1
```

```
Xtv*cut*YFrame:          1
!#
!# Default values for plot window
!#
Xtv*plot.geometry:                500x500+0+0
Xtv*plot*XFrame:                  3
Xtv*plot*YFrame:                  9
Xtv*plot*Font:                    fixed
Xtv*GowFrameWidget.Font:          fixed
Xtv*unix.geometry:                900x900+0+0
Xtv*unix*XFrame:                  6
Xtv*unix*YFrame:                  6
Xtv*unix*Font:            fixed
Xtv*unix*LineSpace:       1.3
Xtv*unix*VLabel.labelDistance:  1.0
Xtv*unix*labelLine:       1.2
Xtv*unix*ticLabel:                5.0
Xtv*unix*ticTic:                  0.5
Xtv*unix*ticLong:                 0.9
Xtv*unix*ticMedium:       0.6
Xtv*unix*ticShort:                0.3
Xtv*unix*ticSpace:                1.2
Xtv*unix*spectrum.linestyle4:    shortdashed
Xtv*unix*spectrum.unixLinestyle4:        shortdashed
Xtv*xfig.geometry:                500x500+0+0
Xtv*xfig*XFrame:                  3
Xtv*xfig*YFrame:                  9
Xtv*xfig*Font:                    fixed
Xtv*XfigFontsize:         20
Xtv*UnixFontsize:         20
Xtv*UnixFontname:         times-roman
Xtv*UnixLinestyle:        solid
Xtv*UnixColor:            0,0,0
!#
!# Default values for all window frames
!#
Xtv*XFrame:                       2
Xtv*YFrame:                       3
Xtv*ticLong:                      0.6
Xtv*ticMedium:                    0.4
Xtv*ticShort:                     0.2
Xtv*ticSpace:                     0.75
Xtv*ticLabel:                     3.0
Xtv*ticTic:                       0.3
Xtv*VLabel.labelDistance:         2.0
Xtv*VLabel.labelLine:             1.0
Xtv*VLabel.topSpectrum:           True
Xtv*VLabel.channelRadius:         2
Xtv*VLabel.libraryRadius:         1.0
!#
!# Default values for cursors
!#
Xtv*GowFrameWidget.cursor:        gumby
Xtv*Paned.gripCursor:             sb_v_double_arrow
```

```
Xtv*GowCutWidget.gripCursor:      dot
Xtv*GowCutWidget.adjustCursor:    bottom_left_corner
!special named widgets
Xtv*fit.geometry:                 780x570+0+0
Xtv*cut.geometry:                 300x300-0-0
Xtv*pro.geometry:                 100x100-0-0
Xtv*xpane.geometry:                300x300+0-0
!Xtv*pane-1*compressMode:         firstfound
Xtv*fit.height:                   200
Xtv*residuum.height:              100
Xtv*residuum.skipAdjust:          True
Xtv*CrossFraction:                0.01

Xtv*bin-list.vBarPosition:        bottom
Xtv*bin-list.upperFraction:       0.00
Xtv*bin-list.lowerFraction:       0.99

Xtv*fit-region.vBarPosition:      top
Xtv*fit-region.upperFraction:     0.02
Xtv*fit-region.lowerFraction:     0.02

Xtv*bg-region.vBarPosition:       bottom
Xtv*bg-region.upperFraction:      0.02
Xtv*bg-region.lowerFraction:      0.02
Xtv*peaklist-label.topSpectrum:         true
Xtv*peaklist-label.upperFraction:       0.01
Xtv*peaklist-label.lowerFraction:       0.01
Xtv*fit-label.topSpectrum:              false
Xtv*fit-label.upperFraction:            0.01
Xtv*fit-label.lowerFraction:            0.01
Xtv*peak-label.topSpectrum:             false
Xtv*peak-label.upperFraction:           0.01
Xtv*peak-label.lowerFraction:           0.01
Xtv*fit-bg-marker.vBarPosition:         bottom
Xtv*fit-bg-marker.upperFraction:        0.03
Xtv*fit-bg-marker.lowerFraction:        0.03
Xtv*fit-region-marker.vBarPosition:     top
Xtv*fit-region-marker.upperFraction:    0.02
Xtv*fit-region-marker.lowerFraction:    0.02
Xtv*fit-bin-marker.vBarPosition:        bottom
Xtv*fit-bin-marker.upperFraction:       0.00
Xtv*fit-bin-marker.lowerFraction:       0.99
Xtv*vertical-marker.vBarPosition:       none
Xtv*vertical-marker.upperFraction:      0.02
Xtv*vertical-marker.lowerFraction:      0.02
Xtv*gate-marker.vBarPosition:                 top
Xtv*gate-marker.upperFraction:                0.03
Xtv*gate-marker.lowerFraction:                0.03
Xtv*bg-gate-marker.vBarPosition:              bottom
Xtv*bg-gate-marker.upperFraction:             0.03
Xtv*bg-gate-marker.lowerFraction:             0.03
Xtv*cut-gate.vBarPosition:                    top
Xtv*cut-gate.upperFraction:                   0.03
Xtv*cut-gate.lowerFraction:                   0.03
```

```
Xtv*cut-bg-gate.vBarPosition:                         bottom
Xtv*cut-bg-gate.upperFraction:                        0.03
Xtv*cut-bg-gate.lowerFraction:                        0.03
Xtv*gray*BorderColor:            gray75
Xtv*gray*grip.foreground:        white
Xtv*gray*grip.background:        white
Xtv*monochrome*BorderColor:      white
Xtv*monochrome*grip.foreground: white
Xtv*monochrome*grip.background: white
Xtv*colored*BorderColor:         gray75
Xtv*colored*grip.foreground:     white
Xtv*colored*grip.background:     white
Xtv*borderWidth:                 1
Xtv*io.grip.height:              1
Xtv*io.grip.width:               1
Xtv*io.gripIndent:               0
Xtv*io.status.vSpace:            0
Xtv*io.status.hSpace:            0
Xtv*io.status.allowResize:       True
Xtv*paned-status.position.label:        ??? ???
Xtv*paned-status.vSpace:                0
Xtv*paned-status.hSpace:                0
Xtv*paned-status.allowResize:           False
!#
!# children of Paned
!#
Xtv*monochrome*GowVsWidget.cursorForeground:    white
Xtv*monochrome*GowVsWidget.cursorBackground:    black
Xtv*gray*GowVsWidget.cursorForeground:          white
Xtv*gray*GowVsWidget.cursorBackground:          black
Xtv*colored*GowVsWidget.cursorForeground:       yellow
Xtv*colored*GowVsWidget.cursorBackground:       black
Xtv*GowVsWidget.height:                 100
Xtv*GowVsWidget.cursor:                 pirate
Xtv*GowVsWidget.cursorLinestyle:        solid
Xtv*GowVsWidget.cursorLinewidth:        0
Xtv*GowVsWidget.xClipMin:               0.0
Xtv*GowVsWidget.xClipMax:               0.0
Xtv*GowVsWidget.yClipMin:               0.02
Xtv*GowVsWidget.yClipMax:               0.15
!#
!# viewport modifications by mouse buttons do not affect the actual input text
!# (Shift <Btn> is normally used by the window manager)
!# mouse leaving window cancels pending expand
!#
!#      center(x|y|xy)
!#      expand(), expand-x(), expand-xy(), expand-y(), cancel-expand()
!#      full(x|y|xy)
!#      scalereset(x|y|xy)
!#      shift(<x-percent> <y-percent>)
!#      stretch(<log2(xfac)> <log2(yfac)>)
!#      execute-string("<commandstring>")
!#      execute-keysequence(), cancel-keysequence()
!#      cancel-keyrequest()
```

```
!#
Xtv*GowFrameWidget.Translations: #override \n\
        !<Btn1Down>(1):          full(y) shift(-70.0 0.0) \n\
        !<Btn3Down>(1):          full(y) shift(70.0 0.0)
Xtv*GowVsWidget.Translations:    #override \n\
        !<Btn1Down>(1):          expand-x() \n\
        !<Btn1Down>(2):          expand-y() \n\
        !<Btn1Down>(3):          expand-xy() \n\
        !<Btn1Down>(4):          cancel-expand() \n\
        !Ctrl<Btn1Down>(1):      full(y) \n\
        !Ctrl<Btn1Down>(2):      full(xy) \n\
        !Ctrl<Btn1Down>(3):      full(x) \n\
        !Ctrl<Btn1Down>(4):      scalereset(xy) \n\
        !<Btn2Down>:             expand-to-limit() \n\
        !<Btn3Down>:             expand() \n\
        !Ctrl<Btn3Down>:         cancel-expand() scalereset(xy) \n\
        <Leave>:                 cancel-expand() \n\
        Shift Ctrl<Btn1Down>:    full(y) shift(-70.0 0.0) \n\
        Shift Ctrl<Btn2Down>:    full(y) center(x) \n\
        Shift Ctrl<Btn3Down>:    full(y) shift(70.0 0.0) \n\
        Ctrl<Key>G: cancel-keysequence() cancel-keyrequest() cancel-expand()\n\
        <Key>Escape:     execute-string("tv> input-mode edit;\n") \n\
        <Key>:           execute-keysequence()
Xtv*spectralist.Translations:           #replace
Xtv*spectralist.internalHeight:         0
Xtv*spectralist.internalWidth:          0
Xtv*spectralist-view.allowVert:         False
Xtv*spectralist-view.allowResize:       True
Xtv*spectralist-view.skipAdjust:        False
!Xtv*io.skipAdjust:                     True
!Xtv*io.view.height:                    100
!Xtv*io.allowResize:                    True
Xtv*io*borderWidth:                     1
Xtv*io*text*search*scrollVertical:      False
Xtv*io*text*search*scrollHorizontal:    False
Xtv*io*text*search*resize:              True
!#
!# two optional scrollbars for text widget:
!#
!# athena viewport widget is not able to trace the position of insertion point
!#
Xtv*io*allowHoriz:              False
Xtv*io*useBottom:               True
Xtv*io*allowVert:               False
Xtv*io*useRight:                False
Xtv*io*forceBars:               True
!#
!# athena text widget forces visibility of insertion point on keyboard
!# events but the position of the last line is sometimes below window ...
!#
Xtv*io*text*scrollVertical:     always
Xtv*io*text*scrollHorizontal:   never
Xtv*io*text*resize:             never
Xtv*io*text*displayCaret:       True
```

```
Xtv*io*text*wrap:                       word
Xtv*monochrome*io*text*background:                  white
Xtv*monochrome*io*text*foreground:                  black
Xtv*gray*io*text*background:                white
Xtv*gray*io*text*foreground:                black
Xtv*colored*io*text*background:             wheat
Xtv*colored*io*text*foreground:             DarkGreen
Xtv*io*text*Translations:       #override \n\
        Ctrl<Key>G:     cancel-keysequence() cancel-keyrequest() \n\
        <Key>Escape:    execute-string("tv> input-mode cursor;\n") \n\
        <Key>Return:    execute-command() edit-unlock() \n\
        <Key>Linefeed:  execute-command() \n\
        <Key>KP_Enter:  execute-command() \n\
        Ctrl<Key>J:     execute-command() \n\
        Ctrl<Key>M:     execute-command() \n\
        Ctrl<Key>O:     execute-command() \
        Ctrl<Key>C:     end-of-file() \
                        newline() \
                        insert-char() \
                        execute-command() \n\
        <Btn1Down>:     select-start() \n\
        <Btn1Motion>:   extend-adjust() \n\
        <Btn1Up>:       extend-end(PRIMARY, CUT_BUFFER0) \n\
        <Btn2Down>:     insert-selection(PRIMARY, CUT_BUFFER0) \n\
        <Btn3Down>:     extend-end(PRIMARY, CUT_BUFFER0)
Xtv*Spectrum.lineSpace:         1.1
!Xtv*spectrum.foreground0:      white
!Xtv*spectrum.background0:      black
!Xtv*spectrum.linewidth0:       0
!Xtv*fit-function.foreground0:  white
!Xtv*bg-function.foreground0:   white
Xtv*monochrome*spectrum.Foreground:     white
!Xtv*monochrome*spectrum.Linewidth:     1
Xtv*monochrome*spectrum.linewidth0:     0
Xtv*monochrome*spectrum.linewidth7:     0
!Xtv*monochrome*spectrum.linestyle0:    solid
!Xtv*monochrome*spectrum.linestyle1:    longdashed
!Xtv*monochrome*spectrum.linestyle2:    disconnected
!Xtv*monochrome*spectrum.linestyle3:    dotdashed
!Xtv*monochrome*spectrum.linestyle4:    dotted
!Xtv*monochrome*spectrum.linestyle5:    shortdashed
!Xtv*monochrome*spectrum.linestyle6:    odddashed
!Xtv*monochrome*spectrum.linestyle7:    solid
Xtv*gray*spectrum.Foreground:   white
!Xtv*gray*spectrum.Linewidth:   1
!Xtv*gray*spectrum.linewidth0:  0
!Xtv*gray*spectrum.linewidth7:  0
!Xtv*gray*spectrum.linestyle0:  solid
!Xtv*gray*spectrum.linestyle1:  longdashed
!Xtv*gray*spectrum.linestyle2:  disconnected
!Xtv*gray*spectrum.linestyle3:  dotdashed
!Xtv*gray*spectrum.linestyle4:  dotted
!Xtv*gray*spectrum.linestyle5:  shortdashed
!Xtv*gray*spectrum.linestyle6:  odddashed
```

```
!Xtv*gray*spectrum.linestyle7:  solid
Xtv*colored*spectrum.foreground0:        yellow
Xtv*colored*spectrum.background0:        red
Xtv*colored*fit-function.foreground0:    gold
Xtv*colored*bg-function.foreground0:     green
Xtv*colored*foreground0:                 yellow
Xtv*colored*foreground1:                 magenta
Xtv*colored*foreground2:                 red
Xtv*colored*foreground3:                 blue
Xtv*colored*foreground4:                 white
Xtv*colored*foreground5:                 wheat
Xtv*colored*foreground6:                 cyan
Xtv*colored*foreground7:                 pink
```

B.2.2   X widget hierachy

```
<application>                       Xtv(application)
   <*** color ***>                  Paned(widget)
      *** standard status ***

      <application>-root            GowFrameWidget(widget)
         <application>-root-sheet   GowVsWidget(widget)
            *** standard GowVsWidget ***

      "spectralist-view"            Viewport(widget)
         "spectralist"              List(widget)
      "io"                          Paned(widget)
         "status"                   Box(widget)
            "keysequence"           Label(widget)
         "view"                     Viewport(widget)
            "text"                  Text(Widget)

   <simple>                         TopLevelShell(widget)
      <*** color ***>               Paned(widget)
         *** standard status ***

         <simple>                   GowFrameWidget(widget)
            <simple>-sheet          GowVsWidget(widget)
               *** standard GowVsWidget ***

   <fit>                            TopLevelShell(widget)
      <*** color ***>               Paned(widget)
         *** standard status ***

         <fit>-fit                  GowFrameWidget(widget)
            <simple>-sheet          GowVsWidget(widget)
               *** standard GowVsWidget ***

         <fit>-residuum             GowFrameWidget(widget)
            <fit>-residuum-sheet    GowVsWidget(widget)
               "residuum"           Residuum(object)

   <cut>                            TopLevelShell(widget)
      <*** color ***>               Paned(widget)
```

```
            *** standard status ***

            "cut"                       GowCutWidget(widget)
                <cut>-cut               GowFrameWidget(widget)
                    <simple>-sheet      GowVsWidget(widget)
                        *** standard GowVsWidget ***

                <cut>-projection        GowFrameWidget(widget)
                    <cut>-projection-sheet  GowVsWidget(widget)
                        *** standard GowVsWidget ***

    <paned>                             TopLevelShell(widget)
        <*** color ***>                 Paned(widget)
            *** standard status ***

            "pane"-<index>              GowFrameWidget(widget)
                "pane"-<index>-sheet    GowVsWidget(widget)
                    *** standard GowVsWidget ***

*** standard status ***:
        "paned-status"                  Box(widget)
            "focus"                     Label(widget)
            "activ"                     Label(widget)
            "full"                      Label(widget)
            "position"                  Label(widget)

*** standard GowVsWidget ***:
        <name>                          GowVsWidget(widget)
            "bin-list"                  VMark(object)
            "cut-gate"                  VMark(object)
            "cut-bg-gate"               VMark(object)
            "cut-label"                 VLabel(object)
            "decomposition"             Decomposition(object)
            "fit-function"              Fitfunction(object)
            "fit-region"                VMark(object)
            "bg-function"               Fitfunction(object)
            "bg-region"                 VMark(object)
            "fit-label"                 VLabel(object)
            "peak-label"                VLabel(object)
            "user-label"                VLabel(object)
            "position-label"            VLabel(object)
            "horizontal-marker"         HMark(object)
            "vertical-marker"           VMark(object)
            "recalibration-marker"      VMark(object)
            "gate-marker"               VMark(object)
            "bg-gate-marker"            VMark(object)
            "fit-bg-marker"             VMark(object)
            "fit-region-marker"         VMark(object)
            "fit-bin-marker"            VMark(object)
            "integration-marker"        VMark(object)
            "peak-search-marker"        VMark(object)
            "normalization-marker"      VMark(object)
            "normalization-bg-marker"   VMark(object)
            "range-marker"              VMark(object)
```

```
                 "peaklist-label"              VLabel(object)
                 "peaklist-function"           Peaklistfunction(object)
                 "spectrum"                    Spectrum(object)

GowCutWidget:
        heightRatio              HeightRatio      float
        widthToHeight            WidthToHeight    float
GowFrameWidget:
        bottomFrame              XFrame           int
        topFrame                 XFrame           int
        leftFrame                YFrame           int
        rightFrame               YFrame           int
        ticLabel                 TicLabel         float
        ticTic                   TicTic           float
        ticLong                  TicLong          float
        ticMedium                TicMedium        float
        ticShort                 TicShort         float
        ticSpace                 TicSpace         float
GowVsWidget:
        cursorLinestyle          CursorLinestyle <*** linestyle ***>
        cursorLinewidth          CursorLinewidth int
        xClipMin                 Clip             float
        xClipMax                 Clip             float
        yClipMin                 Clip             float
        yClipMax                 Clip             float

VLabel:
        labelLine                LabelLine        float
        labelDistance            LabelDistance    float
        topSpectrum              TopSpectrum      boolean
        channelRadius            ChannelRadius    int
        libraryRadius            LibraryRadius    float
Mark:
        crossFraction            CrossFraction    float
        upperFraction            UpperFraction    float
        lowerFraction            LowerFraction    float
VMark:
        vBarPosition             VBarPosition     bottom,top,none
Residuum:
        normalized               Normalized       boolean
Spectrum:
        compressMode             CompressMode     firstfound,minmax
        lineSpace                LineSpace        float

Values of <*** color ***>:
        "monochrome", "gray" or "colored".

Values of <*** linestyle ***>:
        solid           ----------------------
        longdashed      -------____-------____
        disconnected    -_____-_____-_____-
        dotdashed       -----__-__-----__-__--
        dotted          -____-___-___-___-___-
        shortdashed     ----____----____----__
```

```
        odddashed          --___-__---_--___-__--

Graphic context resources:
        numGCs                  NumGCs
X11-resources:
        font<i>                 Font            <name>
        foreground<i>           Foreground      <color>
        background<i>           Background      <color>
        linestyle<i>            Linestyle       <*** linestyle ***>
        linewidth<i>            Linewidth       int
unix-plot-resources:
        unixFontname<i>         UnixFontname    <name>
        unixFontsize<i>         UnixFontsize    int
        unixColor<i>            UnixColor       <int>,<int>,<int>
        unixLinestyle<i>        UnixLinestyle   <*** linestyle ***>
xfig-resources:
        xfigFont<i>             XfigFont        int
        xfigFontsize<i>         XfigFontsize    int
        xfigColor<i>            XfigColor       int
        xfigLinestyle<i>        XfigLinestyle   <*** linestyle ***>
        xfigLinewidth<i>        XfigLinewidth   int
        xfigFillstyle<i>        XfigFillstyle   int
```

Appendix C

File formats

C.1   Definition of mfile

Mfile is a library developed by Stefan Esser as diploma thesis for the IKP for system
independent reading and writing of n-dimensional spectra (n < 4). It provides a
compressed spectra file format called lc (line compressed). The compression algo-
rithm codes the differences of succesive channels in 4 bits.

C.2   File formats

Table C.1 lists all file formats from the mfile.h header file.

Element format	Compression format
lc	line compressed
le2	2 byte low endian (VAX, DEC (mips), Intel)
le4	4 byte low endian (VAX, DEC (mips), Intel)
he2	2 byte high endian (HP, Motorola)
he4	4 byte high endian (HP, Motorola)
lf4	low endian 4 byte IEEE float
lf8	low endian 8 byte IEEE float
hf4	high endian 4 byte IEEE float
hf8	high endian 8 byte IEEE float
le2s	signed LE2
he2s	signed HE2 matrix file
vaxf	VAX F format 4 byte float
vaxg	VAX G format 8 byte float
mate	PC-Mate spectra format
txt	ASCII spectra, Integer or Double
trixi	trixi save_matrix format
le2t	triagonal LE2
le4t	triagonal LE4
he2t	triagonal HE2
he4t	triagonal HE4

Table C.1: File formats from the mfile header file.

The order of bytes in a data word is described by its endianess. With high (big)
endian byte order the bytes are numbered consecutively from left to right in a word.

101

With little endian byte order the byte numbering is reversed as you go from right to left. The format of a spectrum is given in the following manner:

    <levels>.<lines>.<columns>.<element format>:[<version>]

like for example

    1k.2k.4k.he4

which characterizes a cube with 1024 levels, 2048 lines and 4096 columns with 4–byte elements (total size = 35 GB) coded in high endian format.

Allowed abbreviations in the sizes of levels, lines and columns are:

k	= kilo
m	= mega

Appendix D

The fit– background– and measure functions

Contents

D.1 Fit functions

There are two alternative parametrizations for the peaks. Both have proven to yield comparable results. The latter reduces the correlation between step and tails since the erf step approximates the asymptotic value quite fast in comparison with the arctan. Furthermore the step width of the erf can be fixed to 1.0 (in units of sigma) which reduces the number of parameters by one. Finally the latter function is analytically integrable whereas the integration of the former has to be done numerical.

In both functions the volume but not the amplitude of the peaks is fitted. The volume is the parameter of interest normally and in this way it is not necessary to integrate the resulting function and to estimate the errors of the obtained volume.

    tv> fit function peak definition continuous-exp-tail/arctan-step

$$F(x) = BG(x) + \sum_{i=0}^{peaknumber} PEAK_i(x))$$

BG(x): background function see section D.2 on page 105

  peak function of i–th peak

$$PEAKi(x) = \frac{V_i}{NORM_i} \cdot (GAUSM_i(x - P_i) + STEP_i(x - P_i))$$

  P_i: position of i–th peak (parameter)

  V_i: volume of i–th peak (parameter)

  $NORM_i$: numeric INTEGRAL($GAUSM_i$)

  modified gauss function of i–th peak

$$GAUSMi(dx) = \begin{cases} \exp(\frac{SL_i}{S_i^2} \cdot (dx + \frac{SL_i}{2})) & dx < SL_i \\ \exp(\frac{-dx^2}{2 \cdot S_i^2}) & SL_i < dx < SR_i \\ \exp(\frac{-SR_i}{S_i^2} \cdot (dx - \frac{SR_i}{2})) & SR_i < dx \end{cases}$$

103

dx: x - P_i

S_i: σ of gaussian part of i–th peak (parameter)

SL_i: $TL_i \cdot S_i^{EL_i}$

SR_i: $TR_i \cdot S_i^{ER_i}$

TL_i: left tail of i–th peak (parameter)

TR_i: right tail of i–th peak (parameter)

EL_i: exponent of σ–weight of TL_i [0..2]

ER_i: exponent of σ–weight of TR_i [0..2]

step function of i–th peak

$$STEP_i(dx) = SH_i \cdot (\tfrac{p_i}{2} + \arctan(\tfrac{SW_i \cdot dx}{S_i \cdot \sqrt{2}}))$$

SH_i: step height of i–th peak (parameter)

SW_i: step width of i–th peak (parameter)

`tv>` fit function peak definition additive-tail/erf-step

$$F(x) = BG(x) + \sum_{i=0}^{peaknumber} PEAK_i(x)$$

BG(x): background function see section D.2 on page 105

peak function of i-th peak

$$PEAK_i(x) =$$
$$\frac{V_i}{NORM_i} \cdot (((1 + TAIL_i(x - P_i)) \cdot GAUSS_i(x - P_i)) + STEP_i(x - P_i))$$

P_i: position of i–th peak (parameter)

V_i: volume of i–th peak (parameter)

$NORM_i$: $S_i \cdot (\sqrt{(2 \cdot P_i)} + TL_i + TR_i)$

gauss function of i–th peak

$$GAUSS_i(dx) = \exp(\tfrac{-dx^2}{2 \cdot S_i^2})$$

S_i: σ of gaussian part of i–th peak (parameter)

additional tail factor of i–th peak

$$TAIL_i(dx) = \begin{cases} \frac{TL_i \cdot (\frac{|dx|}{S_i})^{E L_i}}{FACFAC(EL_i)} & dx < 0 \\ \frac{TR_i \cdot (\frac{|dx|}{S_i})^{E R_i}}{FACFAC(ER_i)} & dx \geq 0 \end{cases}$$

TL_i: left tail of i–th peak (parameter)

TR_i: right tail of i–th peak (parameter)

EL_i: exponent of left tail [2..16]

ER_i: exponent of right tail [2..16]

for simplification of integral

$$FACFAC(n) = \begin{cases} (n-1)!! & n \in \{3, 5, 7, \ldots\} \\ (n-1)!! \cdot \sqrt{\tfrac{\pi}{2}} & n \in \{2, 4, 6, \ldots\} \end{cases}$$

step function of i–th peak

$$STEP_i(dx) = \tfrac{1}{2} \cdot SH_i \cdot (1 - erf(\tfrac{dx}{SW_i \cdot S_i \cdot \sqrt{2}}))$$

SH_i: step height of i–th peak (parameter)

SW_i: step width of i–th peak (parameter)

erf: $erf(x) = \tfrac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)dt$

D.2   The background functions

You can take a look at the definitions of the background functions with the command:

> **tv> fit function background definition polynom**
> $$BG(x) = \sum_{n=0}^{N} B_n \cdot (x - x_0)^n$$

B_n: background coefficient (parameter)

x_0: offset for numerical optimization during fit

> **tv> fit function background definition exponential**
> $$BG(x) = \sum_{n=0}^{N} B_n \cdot (x - x_0)^n + FAC \cdot exp(-(x - x_0) \cdot EXP)$$

B_n: background coefficient (parameter)

x_0: offset for numerical optimization during the fit

FAC: factor of exponential term

EXP: scaling of exponential term

D.3   Measure functions

> **tv> fit measure definition dy–chi–square**

Minimization of χ^2 weighted by data errors (Maximization of a gauss distribution).

$$chi(f)^2 = \sum_{i=0}^{numdata} \left(\frac{f(X_i) - Y_i}{dY_i} \right)^2$$

f: fit function

X_i channel

Y_i spectrum–value of channel X_i

dY_i error of Y_i

> **tv> fit measure definition y–chi–square**

Minimization of χ^2 weighted by data (Maximization of a gauss distribution).

$$chi(f)^2 = \sum_{i=0}^{numdata} \frac{(f(X_i) - Y_i)^2}{|Y_i|}$$

f: fit function

X_i channel

Y_i spectrum–value of channel X_i

> **tv> fit measure definition poisson**

Maximization of a poisson distribution.

$$P(f) = \prod_{i=0}^{numdata} \frac{f(X_i)^{Y_i}}{Y_i!} \cdot \exp(-f(X_i))$$

f: fit function

X_i channel

Y_i spectrum–value of channel X_i

The really maximized and printed measure is the function:

$$
\begin{aligned}
G(f) \quad &= \quad \ln(P(f)) - C \\
&= \quad \sum_{i=0}^{numdata} Y_i \cdot (\ln(f(X_i)) - f(X_i))
\end{aligned}
$$

C: $\sum_{i=0}^{numdata} - \ln(Y_i!)$

Appendix E

License Agreement

The authors (**J. Theuerkauf, S. Esser, S. Krink, M. Luig, N. Nicolay, O. Stuch, H. Wolters**) at the **Institute for Nuclear Physics, Cologne** (hereinafter referred to as **IKP**) grant to the user, a non–transferable and non–exclusive license to copy and use Tv under the following terms and conditions and for the period of time identified in Paragraph 9.

1. This license agreement grants to the **user** the right to use Tv within their own home or organization. The **user** may make copies of Tv for use within their own home or organization, but may not further distribute Tv except as provided in paragraph 4.

2. The license agreement is only effective in connection with the subsequent listed license agreements:

   ”**License Agreement for mfile**” with the **Institute for Nuclear Physics, Cologne**

3. If the use of Tv is connected to any form of publication, the authors of Tv must be cited correctly:

   J. Theuerkauf, S. Esser, S. Krink, M. Luig, N. Nicolay, O. Stuch, H. Wolters; Program Tv; Institute for Nuclear Physics, Cologne.

4. The **IKP** intends that Tv be widely distributed and used, but in a manner which preserves the quality and integrity of Tv. The **user** may send a copy of Tv to another home or organization only after either receiving permission from the **IKP** or after seeing written evidence that the other home or organization has signed this agreement and sent a hard copy of it to the **IKP**. If the **user** has made modifications to Tv and wants to distribute that modified copy, the **user** will first obtain permission from the **IKP** by written or electronic communication. Any user which has received such a modified copy can pass it on as received, but must receive further permission for further modifications. All modifications to copies of Tv passed on to other homes or organizations shall be clearly and conspicuously indicated in all such copies. Under no other circumstances than provided in this paragraph shall a modified copy of Tv be represented as Tv.

5. The **user** will ensure that all their copies of Tv, whether modified or not, carry as the first information item the following copyright notice:

   - Copyright **J. Theuerkauf, S. Esser, S. Krink, M. Luig, N. Nicolay, O. Stuch, H. Wolters** at the **Institute for Nuclear Physics,**

Cologne, 1993. All rights reserved. Copying of this file is authorized to users who have executed the true and proper "**License Agreement for Tv**" and "**License Agreement for mfile**" with the **Institute for Nuclear Physics, Cologne**

6. In particular the authors prohibit to use any part of the code or algorithms of this software in other programs without an additionally negotiated license.

7. Title to and ownership of Tv and its copies shall at all times remain with the **IKP** and those admitted by the **IKP** as contributors to the development of Tv. The **user** will return to the **IKP** for further distribution modifications to Tv, modifications being understood to mean changes which increase the speed, reliability and existing functionality of the software delivered to the users. The **user** may make for his own ownership and use enhancements to Tv which add new functionality and applications which employ Tv. Such modules may be returned to the **IKP** at the option of the **user**.

8. Tv is licensed with no warranty of any kind. The **IKP** will not be responsible for the correction of any bugs or other deficiencies. In no event shall the **IKP** be liable for any damages of any kind, including special, indirect or consequential damages, arising out of or in connection with the use or performance of Tv.

9. This license for Tv shall be effective from the date hereof and shall remain in force until the **user** discontinues use of Tv. In the case the **user** neglects or fails to perform or observe any obligations under this agreement, this agreement and the license granted hereunder shall be immediately terminated and the **user** shall certify to the **IKP** in writing that all copies of Tv in whatever form in its possession or under its control have been destroyed.

10. Requests. Tv is provided by the **IKP** in a spirit of friendship and cooperation. The **IKP** asks that people enjoying the use of Tv cooperate in return to help further develop and distribute Tv. Specifically, the **IKP** would like to know which machines Tv gets used on. A brief notice form is appended to this agreement which the **user** is requested to send by email or otherwise. Please send in further notifications at reasonable intervals if you increase the number and type of machines on which Tv is loaded. You may send these notices to another user which is cooperating with the **IKP** for this purpose.

Index