Tv
Benutzer–Handbuch

Andreas Fitzler[1]
Institut für Kernphysik
Universität zu Köln

13. Juli 2000

[1] e–mail: fitz@ikp.uni-koeln.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

Einführung

Tv ist ein Programm zur Analyse von Spektren und Matrizen, das unter dem Betriebssystem UNIX läuft und erfolgreich unter den Systemen SUNOS 5.5.1, Ultrix 4.3A, HP-UX 9.01 und Linux getestet wurde. Dieses Handbuch dokumentiert die Konfiguration und Benutzung des Programms. Es soll sowohl eine Einführung für den Anfänger als auch ein Nachschlagewerk für den fortgeschrittenen Benutzer sein. Diese Dokumentation beschreibt das Programm als solches und soll nicht den physikalischen Hintergrund erklären, wenn auch hier und da auf Grundlagen der Auswertung kernphysikalischer Daten eingegangen wird und etwas zur allgemeinen Form von Spektren gesagt wird.

Dieses Handbuch ist sowohl in Form eines Postscript[1]–Manuskripts als auch in Hypertext erhältlich. Die HTML Version findet man unter der URL:
http://www.ikp.uni–koeln.de/~fitz/Tv_user-manual/Tv_user-manual.html.
Dort ist auch das Postscript–Manuskript zu finden.

Aufteilung des Handbuches

Dem Anfänger wird empfohlen, die Kapitel 1 bis 3 zu lesen, wenn er die Arbeit mit Tv beginnt. Die Kapitel 4 und 5 gehen über die Grundlagen hinaus und sind als weitergehende Einführung für Anfänger und als Referenz für den fortgeschrittenen Benutzer gedacht. Das Kapitel 6 führt alle Tv–Befehle in alphabetischer Reihenfolge auf. Eine detaillierte Beschreibung der Inhalte aller Kapitel ist in der folgenden Liste gegeben

Kapitel 1 Beschreibt die typographischen Konventionen, die verwendet werden, um bestimmte Stellen des Texts, wie z.B. Kommandos und Dateinamen, hervorzuheben.

Kapitel 2 Gibt eine Einführung in die Hauptfunktionen von Tv mit Beispielen. Die grundlegenden Befehle werden hier erklärt und nach der Lektüre dieses Abschnitts kann Tv für einfache Auswertungen verwendet werden.

Kapitel 3 Führt in die Grundlagen der Benutzerschnittstelle von Tv ein, angefangen von den Kommandozeilen–Argumenten über Fenstern und Eingabe–Modi bis zur Benutzung der Maus. Weiter wird das Kommando– und Menu–Konzept, die Definition der Tastaturbelegung, das Erstellen von Kommandodateien und das Senden von Kommandos aus anderen Galaxien beschrieben.

Kapitel 4 Zeigt, wie eine beliebige Anzahl von Spektren geladen und gespeichert wird, alle Möglichkeiten zur Analyse der Daten, wie z.B. der Auswahl des optimalen Fenstertyps, der Kalibrierung, Rekalibrierung, Normalisierung, den

[1] ©Adobe

arithmetischen Operationen, der Integration und dem Fit, wie auch der Möglichkeit, ein Spektrum nach Peaks zu durchsuchen.

Kapitel 5 Erklärt alle möglichen Operationen, um mit Matrizen, Schnitten (Cuts), Projektionen, Spektren und Buffern zu arbeiten.

Kapitel 6 Listet alle Tv–Kommandos mit ihren Argumenten in alphabetischer Reihenfolge auf, wobei zu jedem von ihnen eine kurze Beschreibung seiner Funktion gegeben ist. Dieses Kapitel richtet sich an den erfahrenen Benutzer, der alle Möglichkeiten von Tv ausschöpfen und alle seine Geheimnisse ergründen will.

Kopierrecht

Vor der Installation von Tv muß eine Lizenz der Programmierer vorhanden sein. Um eine solche zu erhalten, füllen Sie das **License Agreement for Tv** (siehe Anhang E p. 113) aus und senden dieses, sowie eine Kopie, unterschrieben an die Autoren. Die Adresse ist in der Vereinbaung zu finden. Der Gebrauch ist kostenlos, registrierte Benutzer werden über neue Versionen informiert.

Kapitel 1

Syntax in diesem Handbuch

1.1 Einführung

Um die Lesbarkeit des Handbuchs zu erleichtern, werden verschiedene Schriftarten verwendet. In diesem Kapitel wird erklärt, wozu welche Schriftart verwendet wird.

1.2 Kommandos

Tv Kommandos werden in der Schriftart **boldfaced sans serife** gedruckt und folgendermaßen formatiert:

`Menu>` **Kommando <Argument1> [<Argument2>] <Argument3 ...>**
`Menu>` **Kommando {<Argument4> | <Argument5>}**

`Menu>` ist der Prompt, der anzeigt, daß Tv eine Eingabe im Menu **Menu** erwartet. Das Kommando kann vollständig oder, wie später beschrieben, in abgekürzter Form (siehe Kapitel 3.7) eingegeben werden. Argumente in eckigen Klammern [ ] sind optional. Punkte hinter einem Argumentnamen zeigen an, daß eine Liste von Argumenten dieses Typs folgen kann. Von Argumenten in geschweiften Klammern { } muß genau eins eingegeben werden. Die vertikale Linie ('|') im obigen Beispiel bedeutet, daß entweder Argument4 oder Argument5 angegeben werden muß.

1.3 Tastenkombinationen — Hotkeys

Oft benutzte Kommandos können im Cursor–Modus durch ein oder zwei Tastendrücke (siehe Kapitel 3.4.2) ausgeführt werden. Diese Hotkeys werden im Handbuch eingerahmt dargestellt, wie zum Beispiel:

$\boxed{\text{Taste}}$

Zum Beispiel ist die Taste, die von einem Ausschnitt ausgehend das ganze Spektrum darstellt:

$\boxed{\text{f}}$

Da es nur eine begrenzte Anzahl von Tasten auf der Tastatur gibt, können auch Tastenkombinationen verwendet werden, um die Anzahl der abkürzbaren Kommandos zu erhöhen. Jede Taste einer Kombination muß in der angegebenen Reihenfolge gedrückt werden. Zum Beispiel druckt die Tastenkombination $\boxed{\text{P}}$ und $\boxed{\text{x}}$ das Spektrum im xfig–Format. Diese Tastenkombination wird im Handbuch folgendermaßen geschrieben:

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

Die Standard–Hotkeys und –Tastenkombinationen sind im Anhang A auf Seite 89 abgedruckt.

1.4 Programm Ausgabe

Die Terminalausgabe von Tv wird in der Schriftart `typewriter`, wie im folgenden Beispiel, ausgedruckt:

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

1.5 Dateinamen

Dateinamen werden in *Italics* gedruckt,zum Beispiel:

site.tvrc

Kapitel 2

Einführung

Inhaltsangabe

2.1 Einführung

Dieses Kapitel soll neuen Benutzern einen Überblick über die Eigenschaften und Fähigkeiten von Tv geben. Das Programm wird von der Shell aus mit

> **tv&**

gestartet, da es aber alle Meldungen auf die Standardausgabe lenkt, verwendet man besser:

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

Zwei Fenster erscheinen auf dem Bildschirm, ein **Textfenster (namens tv)**, in dem die Kommandos, Ergebnisse und Statusinformationen ausgegeben werden, und ein **Graphikfenster (namens tv–root)**, in dem Spektren angezeigt werden. Beide Arten von Fenstern werden in Kapitel 3.3 im Detail beschrieben und in den Abbildungen 3.2 und 3.1 dargestellt.

2.2 Eingabe Modi

Die Eingabemodi von Tv erscheinen auf den ersten Blick kompliziert, erweisen sich aber bei der täglichen Arbeit als nützlich. Tv unterstützt zwei Hauptmodi zur Eingabe von Kommandos, nämlich den **Edit–Modus** und den **Cursor–Modus**.

Standardmäßig startet Tv im Edit–Modus, wo Tastatureingaben auf das Textfenster fokussiert sind, was bedeutet, daß alle Eingaben in diesem Fenster abgebildet und beim Drücken der $\boxed{\text{CR}}$ Taste ausgewertet werden. Immer wenn sich der Cursor im Textfenster befindet, ist man im Edit–Modus.

Im Cursor–Modus werden Tastatureingaben als Hotkeys interpretiert (siehe Kapitel 3.4.2), wenn der Cursor über dem Graphikfenster plaziert ist (der Cursor ist hat dann die Gestalt eines Totenkopfes ☠). Um zwischen dem Edit– und Cursor–Modus umzuschalten, plaziert man den Cursor über dem Graphikfenster und drückt Esc. Die meisten Operationen in dieser Einführung werden im Cursor–Modus durchgeführt.

2.3 Spektrum

2.3.1 Laden eines einzelnen Spektrums

Um ein einzelnes γ–Spektrum zu untersuchen, schaltet man am besten in den Cursor–Modus und drückt g, um das **Spektrum zu laden**. Dann gibt man den Dateinamen des Spektrums ein und drückt CR. Das Spektrum wird im tv–root Fenster dargestellt (siehe Abbildung 2.1).

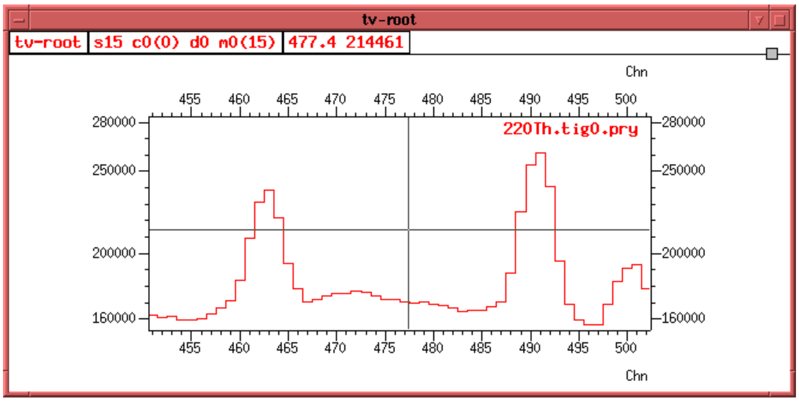

Abbildung 2.1: Graphikfenster mit geladenem ^{220}Th Spektrum.

Tv verwendet die Bibliothek **mfile** für die Ein– und Ausgabe von Spektren. Spektren, die im mfile Format gespeichert sind, haben einen Header, in dem Informationen über das Spektrumformat gespeichert sind. Wenn man ein Spektrum laden möchte, das nicht im mfile Format gespeichert wurde, versuchen die mfile Routinen das Format zu raten. Falls das nicht funktioniert, muß das Format explizit angegeben werden. Zum Beispiel muß man, um ein Spektrum mit 8192 Kanälen zu laden, das im low Endian Format mit 4 Byte Datentiefe gespeichert wurde, folgendes eingeben:

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

In diesem Beispiel steht **s** für **spectrum**. Eine Beschreibung der Möglichkeit Kommandos abzukürzen steht im Kapitel 3.7. Eine detaillierte Beschreibung der Spektrenformate findet man in Anhang C.2.

2.3.2 Bewegen im Spektrum

Die x– und y–Position des Cursors kann man im Statusfenster des Graphikfensters ablesen. Um einen Teil des Spektrums zu **vergrößern**, setzt man zwei Marker durch Drücken der Spacebar für jeden an der verlangten Position und drückt e zum Zoomen. Mit den Tasten 1 bis 9 vergrößert man und mit 0 oder - (1 bis 9) verkleinert man die Darstellung, dabei definiert der Cursor das Zentrum des darzustellenden Ausdrucks. Um den **Ausschnitt des Spektrums (Viewport) zu verschieben**, nach links (oder rechts), drückt man < (oder >), was automatisch die Skalierung der y–Achse anpaßt. Um den Viewport zu verschieben, ohne die Skalierung der y–Achse zu verändern, verwendet man $\boxed{,}$ (oder $\boxed{.}$). Alternativ kann man $\boxed{\text{i}}$ verwenden, um **zu einer bestimmten Position zu springen**, die man dann angeben muß. Diese Position ist die Energie für geeichte Spektren, was dem Kanal für ungeeichte Spektren entspricht. Durch Drücken der Taste $\boxed{\text{f}}$ kann man das ganze Spektrum anzeigen.

Wie man die **Maus** verwendet, um Spektren zu zoomen oder durch sie zu scrollen, wird im Kapitel 3.5 auf Seite 19 beschrieben.

2.3.3 Fitten eines Spektrums

Die einfachste Möglichkeit die Position, Breite und das Volumen eines Peaks zu bestimmen (oder zu **fitten**; siehe 4.12), ist die Quickfit Funktion von Tv. Dazu positioniert man den Cursor über den zu fittenden Peak und drückt $\boxed{\text{Q}}$. Der **Kurzstatus** eines Fits wird wie folgt angegeben:

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

In der letzten Zeile des Ausdrucks sind Position, Breite und Volumen des Peaks angegeben. In Klammern stehen die Fehler der gefitteten Größen.

Der Quickfit rät Position und Fit–Bereich des Peaks, was bei überlappenden Peaks zu Schwierigkeiten führt. Sie werden normalerweise als ein Peak vom Quickfit gefittet. Außdem berücksichtigt der Quickfit keinen Untergrund, was zu einer Fehlabschätzung des Untergrundes führt.

Als Konsequenz aus dem letzten Ansatz müssen **Fit– und Untergrund–Bereiche** und ein **Peak Marker** angegeben werden, um die besten Ergebnisse mit einem Fit zu erzielen. Der Fit–Bereich, das ist der Abschnitt des Spekrums in dem sich die zu fittenden Peaks befinden, wird mit der Taste $\boxed{\text{r}}$ definiert, und zwar müssen eine linke und eine rechte Grenze definiert werden. Alle zu fittenden Peaks innerhalb dieses Bereichs werden mit der Taste $\boxed{\text{p}}$ festgelegt. Dann kann man mit der Taste $\boxed{\text{F}}$ den **Fit durchführen**.

Eventuell kann der Fit durch die Berücksichtigung einer beliebigen Anzahl von **Untergrundbereichen** noch verbessert werden. Diese werden mit der Taste $\boxed{\text{b}}$, wieder jeweils einmal für die linke und rechte Grenze jeden Bereichs, definiert. Mit $\boxed{\text{B}}$ wird die Berechnung der Untergrundfunktion durchgeführt und beim Drücken der Taste $\boxed{\text{F}}$ zur Berechnung des Fits wird diese Untergrundfuktion berücksichtigt. Obwohl Tv nur einen Fit–Bereich erlaubt, kann eine beliebige Anzahl von Untergrundbereichen angegeben werden. Abbildung 2.2 auf Seite 10 zeigt eine gefittete Funktion in einem Spektrum.

Um entweder einen Quickfit oder einen normalen Fit zu löschen, verwendet man die Tastenkombination $\boxed{\text{-}}\boxed{\text{F}}$. Das entfernt alle Bereichmarker aus dem Graphikfenster und löscht den Fit. Man muß das tun bevor man einen weiteren Peak fitten kann.

2.3.4 Eichung eines Spektrums

Um Peaks mit Übergangsenergien eines Zerfalls zu verbinden, benötigt man eine Relation zwischen den Kanälen des Spektrums und den korrespondierenden Energien (**calibration**). Um ein Spektrum zu eichen, muß man zwei Peaks identifizieren oder man benötigt ein Eichspektrum. Falls zwei Kanäle und ihre entsprechenden Energien bekannt sind, kann die Eichung mit folgendem Kommando durchgeführt werden:

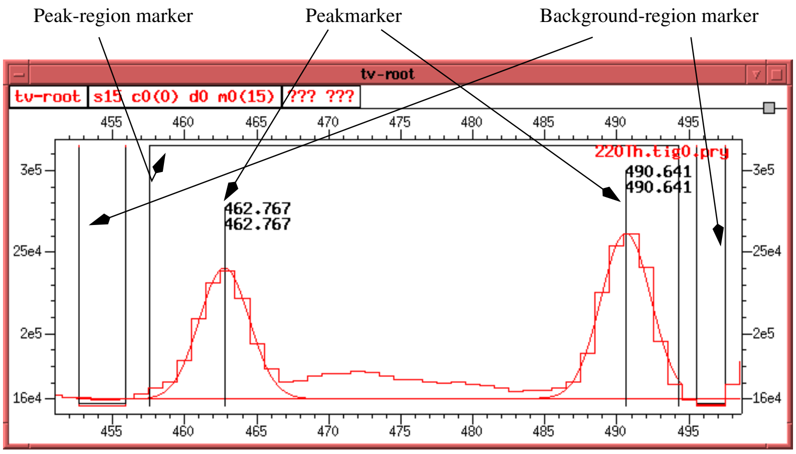

Abbildung 2.2: Graphikfenter mit einer geladenen ^{220}Th Projektion und einem Fit. Man sieht sowohl den Fitbereich, der die mit Markern markierten Peaks umgibt, als auch zwei Untergrundbereiche mit der gefitteten Untergrund–Funktion. Außerdem ist die Fitfunktion mit dargestellt.

tv> **calibration position enter** <**chn**$_1$ > <**energy**$_1$ > <**chn**$_2$ > <**energy**$_2$ >
Man kann eine Eichung auch aus einer Datei laden. Das macht man entweder durch Drücken des Hotkeys E, wodurch die Standardeichung *.cal zu dem entsprechenden Spektrum geladen wird, oder indem man in den Text–Modus umschaltet und das folgende Kommando eingibt:
tv> **calibration position read** <**calfile**>
Das Kommando kann folgendermaßen abgekürzt werden:
tv> **c p r** <**calfile**>

2.3.5 Ausdrucken

Ausdrucken mit Tv führt möglicherweise nicht auf Anhieb zu befriedigenden Ergebnissen, wahrscheinlich wird die Beschriftung der y–Achse fehlen. Der Grund dafür ist, daß die Rahmenbreite des linken und rechten Rahmens standardmäßig auf 3 gesetzt ist (siehe Kapitel 3.3). Es gibt aber zwei Möglichkeiten, vernünftige Ausdrucke zu erhalten.

Die erste Möglichkeit ist, ein Fenster mit dem Namen **plot** zu erzeugen, dessen Rahmenbreite für rechts und links auf 9 eingestellt ist. Dieses Fenster wird mit dem folgenden Befehl erstellt:
tv> **window create simple plot**
Die zweite Möglichkeit ist, die Rahmenbreite des aktiven Fensters mit folgendem Befehl zu vergrößern, wobei als Argumente Zahlenwerte für die linken, rechten, oberen und untere Rahmenbreite einzugeben sind:
tv> **window setup frame width** <**l**> <**r**> <**u**> <**o**>
Das gebräuchliste von Tv unterstützte Format um ein Spektrum auszudrucken ist das xfig–Format, weil das Spektrum dann mit dem Programm **xfig** bearbeitet und konvertiert werden kann. Der Ausdruck wird mit der Tastenkombination P x erzeugt. Alternativ kann auch im unix–Format ausgegeben werden mit der Tastenkombination P u. In beiden Fällen muß ein Dateiname angegeben werden. Drücken von CR bei der Eingabe des Dateinamens erzwingt die Verwendung des Standarddateinamens für Ausdrucke, der zusammengesetzt ist aus dem Namen des Spektrums und einem Suffix, der durch die Wildcard (siehe Kapitel 3.12) für Ausdrucke, die standardmäßig .fig ist, bestimmt ist.

Ein Beispielausdruck, der mit xfig in das Postscriptformat exportiert wurde, ist in Abbildung 2.3 dargestellt.

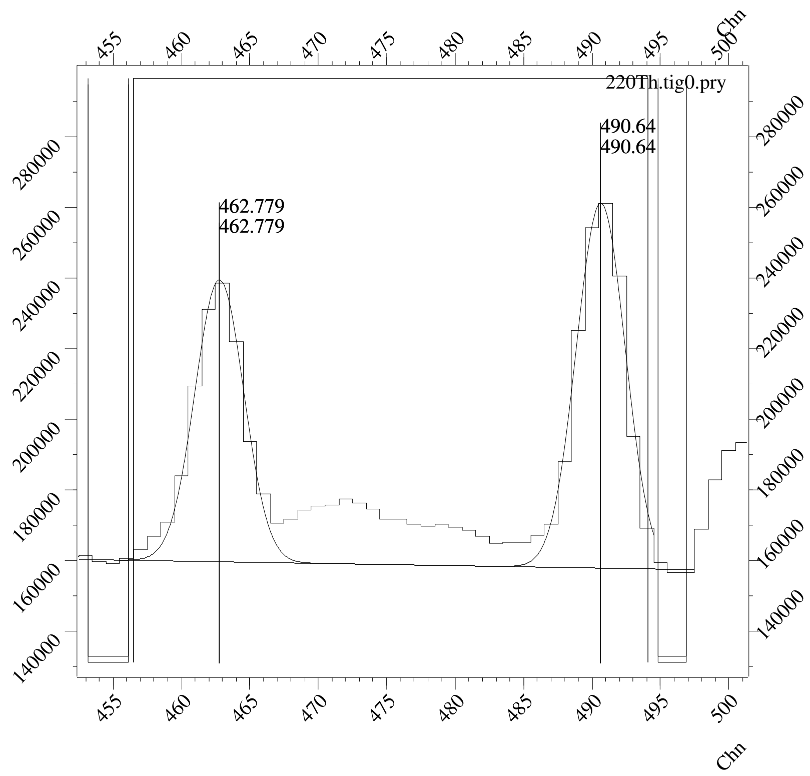

Abbildung 2.3: Beispielausdruck von TV, im xfig–Format ausgegeben und in das Postscriptformat exportiert.

2.4 Analyse einer $\gamma\gamma$–Matrix

2.4.1 Öffnen einer Matrix

Um eine $\gamma\gamma$–Matrix zu analysieren, benötigt man eine im mfile Format geschriebene Matrix. [1] Das Standardsuffix eines Matrixdateinamens ist .mtx. Weiterhin benötigt man Projektionen [2] auf die x– oder y–Achse mit den Standardsuffixes .prx und .pry. Diese drei Dateien müssen sich in einem Verzeichnis befinden, das den gleichen Basisnamen hat wie die Matrix. Als Beispiel siehe Tabelle 2.1.

Matrix Verzeichnis	220Th
Matrix Datei	220Th/220Th.mtx
Matrix Projektion	220Th/220Th.pr[xy]

Tabelle 2.1: Beispiel für eine Liste von Dateien, die zur Analyse einer $\gamma\gamma$–Matrix benötigt werden. Man benötigt entweder die Projektion auf die x– oder die y–Achse.

[1] Um eine Matrix in ein passendes Format umzuwandeln, verwendet man das Programm matconv. Es wurde, wie die mfile–Bibliothek, am IKP programmiert und ist im Paket mit TV erhaältlich.

[2] Eine Projektion kann mit dem Programm matproj erzeugt werden.

Um eine Matrix zur Analyse zu öffnen, verwendet man den Hotkey $\boxed{\text{M}}$, worauf man nach dem Namen des Verzeichnisses gefragt wird, in dem sich die Matrix befindet.

2.4.2 Schneiden in einer Matrix

Eine $\gamma\gamma$–Matrix ist ein zweidimensionales Objekt, dessen Inhalte koinzidente Daten aus Experimenten darstellen. Um die koinzidenten Daten zu untersuchen, kann man eine Scheibe aus einer Matrix **herausschneiden** und man erhält ein **Schnittspektrum**. Um ein Schnittspektrum zu erzeugen (siehe Kapitel 5), muß man ein **positives Schnittfenster** definieren, was mit der Taste $\boxed{\text{c}}$ geschieht. Ähnlich wie bei der Definition des Fit–Bereichs, definiert man eine linke und eine rechte Grenze, durch einmaliges Drücken der Taste $\boxed{\text{c}}$ für jede Grenze. Weitere mit der Taste $\boxed{\text{c}}$ definierte Bereiche werden automatisch als **Untergrundfenster** interpretiert. Zusätzliche **positive Schnittfenster** definiert man mit der Tastenkombination $\boxed{\text{G}}\boxed{\text{G}}$. Die Taste $\boxed{\text{C}}$ führt den Schnitt durch und das Spektrum wird in einem Fenster dargestellt. Die Schnitt– und Untergrundfenster können mit $\boxed{\text{-}}\boxed{\text{C}}$ gelöscht werden.
In der Projektion wird dann ein Fenster angezeigt, wie in Abbildung 2.4.

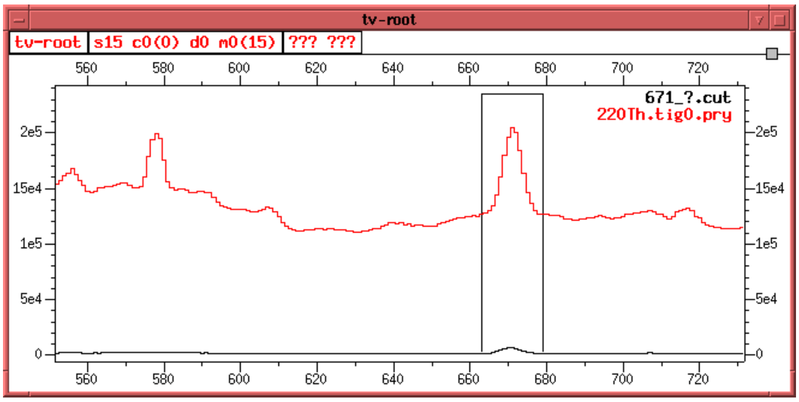

Abbildung 2.4: Positives Schnittfenster in der Projektion eines ^{220}Th Spektrums. Der Peak beim Kanal 671 ist markiert, um seine koinzidenten Übergänge zu finden. Das untere Spektrum ist das **Schnittspektrum** für den Kanal 671, welches weniger Statistik hat. Außerdem erkennt man, daß der Kanal 671 autokoinzident ist.

Um einen besseren Überblick zu haben, wird empfohlen, ein sogenanntes **Schnittfenster** zu erzeugen. Es stellt die Projektion und den Schnitt in einem Fenster dar und wird mit dem folgenden Befehl erzeugt:

`tv>` **window create cut testcut**

Vergleicht man Abbildung 2.4 mit Abbildung 2.5 so erkennt man deutlich, daß die Verwendung eines speziellen Schnittfensters sehr zur Übersichtlichkeit beiträgt.

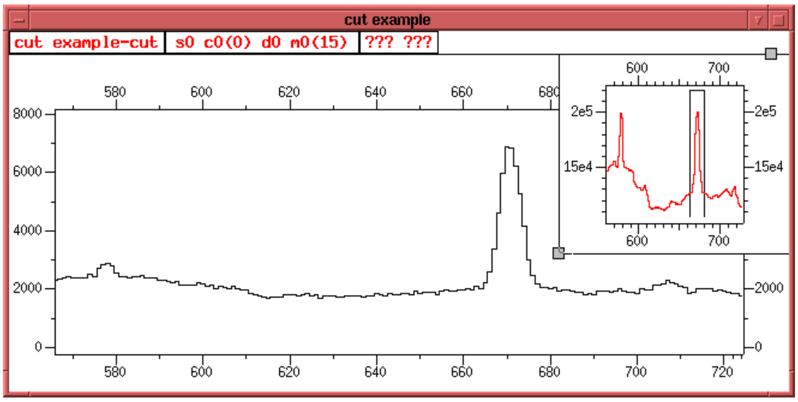

Abbildung 2.5: Ein Schnitt in seinem speziellen Fenster. In der oberen rechten Ecke ist die Projektion dargestellt.

Kapitel 3

Grundlagen

Inhaltsangabe

3.1 Einführung

In diesem Kapitel wird die Benutzerschnittstelle von Tv beschrieben. Die grundlegenden Funktionen zur effektiven Benutzung von Tv werden eingeführt, angefangen bei der Bedeutung der Kommandozeilen–Argumente, der verschiedenen Fenster und Eingabe–Modi. Ferner werden Anleitungen zur Neudefinition der Tastatur und dem Erstellen von Kommando–Dateien bis zur Fernsteuerung von Tv gegeben.

3.2 Starten von Tv

Tv kann mit einer Reihe von Kommandozeilen–Argumenten aufgerufen werden, deren Bedeutung in diesem Abschnitt beschrieben wird.

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

-? | -help: druckt eine Liste aller erlaubten Kommandozeilen–Argumente aus

-src: die Auswertung der systemweiten Initialisierungsdatei *site.tvrc* wird unterdrückt

-rc: die Auswertung der persönlichen Initialisierungsdatei *~/.tvrc* wird unterdrückt

-v: Versionsnummer

-s <ns>: setzt die Anzahl der Spektrenpuffer vom Standardwert 16 auf <ns>

-e <Zeichenkette>: interpretiert den in <Zeichenkette> angegebenen Text und führt das Kommando aus

3.3 Die Fenster

Nach dem Start des Programms erscheinen zwei Fenster auf dem Bildschirm mit Namen **tv** und **tv–root**.

3.3.1 Das Graphik–Fenster

tv–root (siehe Abb. 3.1) ist das Haupt–Graphik–Fenster. In der Regel werden alle geladenen Spektren in diesem Fenster dargestellt.

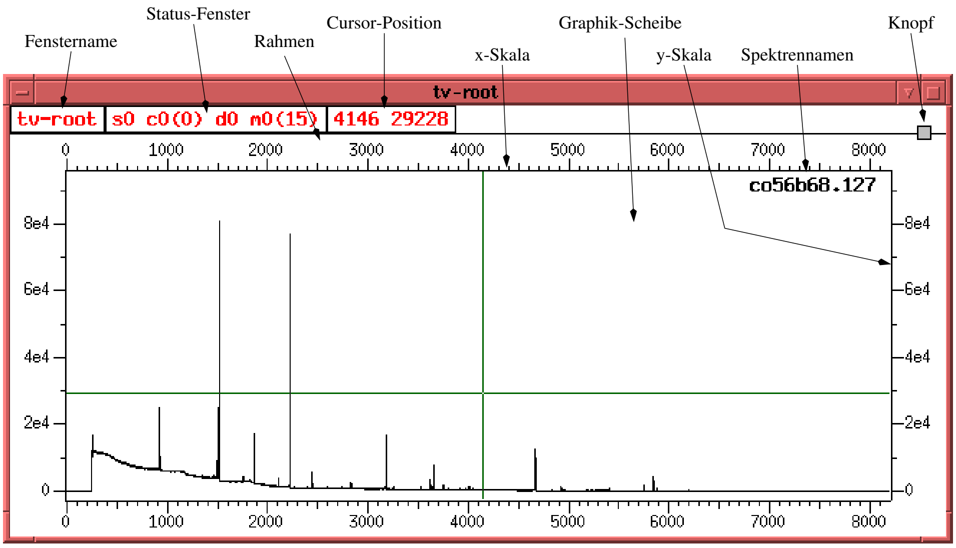

Abbildung 3.1: Graphik–Fenster unmittelbar nach dem Start und mit einem geladenen ^{56}Co Spektrum.

Im oberen Teil des Graphik–Fensters erkennt man das Status–Fenster.

Im Status–Fenster werden sowohl der Name des Fensters als auch die Buffer–Nummern des aktiven Spektrums (s0), des Schnittspektrums (c0), des Verzeichnisses (d0) und der Matrix (m0) angegeben. In Klammern steht die Buffer–Nummer (c0(0)) des Spektrums, in dem das Schnittspektrum abgespeichert ist, sowie die Buffer–Nummer (m0(15)) der zur aktiven Matrix gehörenden Projektion. Falls bereits ein Spektrum geladen ist, werden die x– und y–Koordinaten des Mauszeigers angegeben, sonst stehen Fragezeichen an den entsprechenden Stellen. In Fenstern mit mehreren Rahmen (siehe Kapitel 4.2.1) wird der Name des momentan aktiven Rahmens angezeigt.

In den unteren Teilen des Graphik–Fensters werden die Spektren dargestellt, mit ihren Namen in der oberen rechten Ecke des jeweiligen Fensters. In diesen "Fenster–Scheiben" können Funktionen mit der Maus ausgeführt werden, wie es im Kapitel 3.5 beschrieben wird. Die Spektren sind auf allen Seiten von Skalen umgeben, die auf der linken und rechten Seite (**y–Skala**) die Anzahl der Ereignisse angeben, sowie die x–Position in Kanälen auf der oberen Skala und die Energie (siehe Kapitel 4.8) auf der unteren Skala (**x–Skala**), falls eine Energieeichung durchgeführt oder geladen wurde. Die Skalen sind in den (**Fenster–Rahmen**) untergebracht. Der quadratische **Knopf** wird zum Verschieben der Linie zwischen dem Status–Fenster und dem obersten Spektrum verwendet.

Die Skalierung der y–Skala zwischen **linear, logarithmisch and quadratisch** wird umgeschaltet mit den Befehlen:

`tv>` **lin**
`tv>` **log**
`tv>` **qua**

Der **viewport** (das ist der sichtbare Teil der Spektren) kann auf viele Arten mit verschiedenen Tastenkombinationen, die in der Tabelle 3.1 aufgelistet sind, verändert werden.

Taste	Tv–Kommando	Funktion
Spacebar	tv> window mark vertical enter cursor;	Setze vertikale Marker, um den zu expandierenden x–Bereich zu definieren
- 1 — - 9	tv> window view full y; tv> window view stretch (-1 to -9) cursor 0;	Vergrößern des dargestellten Spektrenausschnitts (siehe Anhang A)
,	tv> window view full y; tv> window view shift -70.0 0.0;	Verschiebe den viewport nach links um 70% der sichtbaren Kanäle
.	tv> window view full y; tv> window view shift 70.0 0.0;	Verschiebe den viewport nach rechts um 70% der sichtbaren Kanäle
0	tv> window view full y; tv> window view stretch -1 cursor 0;	Vergrößern des dargestellten Spektrenausschnitts
1 — 9	tv> window view full y; tv> window view stretch (1 to 9) cursor 0;	Verkleinern des dargestellten Spektrenausschnitts (siehe Anhang A)
<	tv> window view full y; tv> window view shift -70.0 0.0;	Verschiebe den viewport nach links um 70% der sichtbaren Kanäle
>	tv> window view full y; tv> window view shift 70.0 0.0;	Verschiebe den viewport nach rechts um 70% der sichtbaren Kanäle
e	tv> window view expand;	Expandiere den viewport auf den x–Bereich zwischenden mit Spacebar gesetzen Markern
f	tv> window view full both; tv> window view expand;	Zeige das gesamte Spektrum
x	tv> window view full x; tv> window view expand;	Expandiere den viewport auf die volle Größe in x–Richtung
y	tv> window view full y;	Expandiere den viewport auf die

Weiter auf der nächsten Seite

Taste	Tv–Kommando	Funktion
	tv> window view expand;	volle Größe in y–Richtung
$\boxed{\vert}$	tv> window view full y; tv> window view center x cursor;	Zentriere die Spectren um die Position, die durch den cursor definiert wird

Tabelle 3.1: Tastenkombinationen zum Ändern des viewports eines Spektrums.

Einige dieser Operationen können auch mit der Maus ausgeführt werden, wie im Kapitel 3.5 beschrieben.

3.3.2 Das Text–Fenster

tv (Abbildung 3.2) ist ein Text–Fenster und setzt sich aus drei Teilen zusammen.

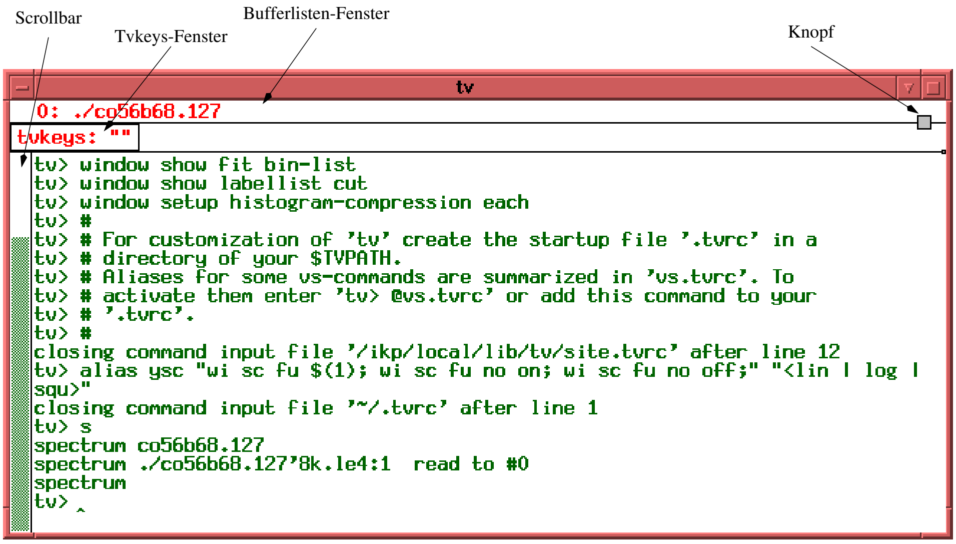

Abbildung 3.2: Text–Fenster unmittelbar nach dem Start von Tv. Das Bufferlisten–Fenster zeigt ein ^{56}Co Spektrum, geladen in den Buffer 0.

Der **Bufferlisten–Teil** zeigt die Buffer–Nummern und die Namen der zugehörigen Spektren.

Der **tvkeys–Teil** zeigt die im Cursor–Modus (siehe Kapitel 3.4) gedrückten Tasten an.

Alle im **Text–Modus** in irgendeinem Fenster eingegebenen Kommandos, werden im **Text–Fenster** angezeigt. Sie werden nach dem Drücken der Taste $\boxed{\text{CR}}$ ausgewertet und die Ergebnisse in Text– oder Graphik–Fenster angezeigt.

Das Text–Fenster ist **Voll Screen editierbar**, d.h., man kann den Cursor mit den Cursortasten an jede beliebige Stelle, auf ein bereits ausgeführtes Kommando bewegen, es editieren und wieder ausführen.

Falls Kommandos wiederholt werden sollen, die vor längerer Zeit ausgeführt wurden, kann die Suche durch Benutzung der **Scrollbar** beschleunigt und der Cursor mit der Maus an der entsprechenden Stelle plaziert werden. Alternativ können vorher benutzte Kommandos in einem **Suchkontext** gesucht werden, was mit den

Tastenkombinationen Strg r oder Strg s initiiert wird. **Cut and paste** von Textblöcken mit der Maus ist ebenso möglich.

3.4 Eingabe–Modi

Es gibt zwei verschiedene Eingabemodi, zwischen denen man im Textmodus mit dem **input–mode** Befehl hin– und herschalten kann, oder durch drücken der Esc Taste im Graphikfenster.

3.4.1 Edit–Modus

Dieser Modus ist am **Textcursor** I im **tv–root** Fenster zu erkennen. Alle Tasten, die in diesem Modus in einem der TV Fenster gedrückt werden, erscheinen im Textfenster. Durch Drücken der CR Taste wird das eingegebene Kommando ausgeführt.

Mit der **"?"** Taste erhält man zu jeder Zeit eine Liste der möglichen Kommandos. Weiterhin ist es ausreichend, nur die ersten Buchstaben eines Kommandos einzugeben (siehe Kapitel 3.7) und CR zu drücken. TV versucht dann das Kommando zu vervollständigen und auszuführen. Meistens ist bereits der erste Buchstabe eines Befehls genügend. Falls das Kommando noch nicht eindeutig ist, fragt TV nach weiterer Eingabe.

3.4.2 Cursor–Modus

Dieser Modus ist am **Piraten–Cursor** im **tv–root** Fenster zu erkennen. Er erlaubt die Abkürzung von Kommandos im Graphikfenster durch Drücken von "Hotkeys", die im tvkeys–Teil angezeigt werden, sobald man sie drückt. Tasten, die im Graphikfenster von **tv–root** gedrückt werden, werden in einer Hotkey–Tabelle (siehe Kapitel 3.11) gesucht und die an sie gebundenen Kommandos ausgeführt, sobald die Hotkey–Kombination vollständig ist. Zum Beispiel sind die folgenden Befehle an die Taste f gebunden:

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

und durch Drücken der Taste f im Graphikfenster wird der viewport auf die volle Größe expandiert. Drücken der Taste ? in diesem Modus liefert eine Liste aller Hotkeys im Textfenster.

3.5 Die Maus

Mit der Maus werden zwei Aufgaben ausgeführt. Zum einen wird mit ihr der viewport verändert und zum anderen kann man mit ihr sowohl Text Ausschneiden und Einfügen als auch den Textcursor im Textfenster positionieren.

Während sich der Cursor im Graphikfenster befindet (am Fadenkreuz zu erkennen), setzt man mit der linken Maustaste (#1) einen Marker und führt mit den anderen Tasten bestimmte Aktionen durch. Die mittlere Maustaste (#2) expandiert den Bereich zwischen dem mit (#1) gesetzten Marker und dem Rahmen, in dem sich der Mauszeiger befindet, wohingegen die rechte Maustaste (#3) den Bereich zwischen einem Marker und der momentanen Cursorposition expandiert.

Es können vertikale Marker (#1), horizontale Marker (Doppelklick #1) oder Eckmarker (Dreifachklick #1) gesetzt werden. Ein Eckmarker ist ein kombinierter vertikaler und horizontaler Marker und definiert eine Ecke eines Rechtecks. Alle viewport–Marker werden beim Verlassen des **tv–root** Fensters gelöscht.

Wenn sich der Cursor auf den Skalen befindet (**gumby–cursor**), verschiebt die linke Maustaste das Spektrum nach links und die rechte Maustaste nach rechts.

In Kombination mit den Tasten Strg Shift #1 verschiebt den viewport nach links, #2 zentriert den viewport um die Cursorposition und #3 verschiebt den viewport nach rechts.

Die Funktionen der Maustasten können in der Datei *.Xresources* (siehe Anhang B) definiert werden, wodurch die Standardeinstellungen überschrieben werden.

Die x– und y–Position des Cursors werden in der Statuszeile des Fensters angezeigt, in dem sich der Cursor gerade befindet(siehe 3.1). Falls eine Energieeichung geladen ist, wird die x–Position sowohl geeicht als auch ungeeicht (in Klammern) angezeigt.

3.6 Kommandos und Menus

Kommandos in Tv sind in einer Menustruktur organisiert. Das führt zu einer Art Objektorientierung, da sich alle Kommandos, die auf einem Objekt erlaubt sind, in seinem Menu befinden. Alle Kommandos im aktuellen Menu werden mit den Tasten ? CR angezeigt. Im Hauptmenu, das durch den Prompt `tv>` angezeigt wird, sind die möglichen Kommandos in drei Gruppen unterteilt. Es gibt die Gruppen **aliases**, **main command** und **miscellaneous**. Kommandos können Gruppen von Subkommandos sein, wie zum Beispiel:

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

Wird wenigstens ein Argument eines kompletten Kommandos ausgelassen, so gelangt man in ein neues Menu. Zum Beispiel gelangt man durch die Eingabe von

`tv>` **window**

in das Menu `window`, was durch den neuen Prompt

`window`

statt `tv>` angezeigt wird.

Alle Kommandos werden **relativ** zum aktuellen Menu interpretiert, d.h. sie werden an den Menunamen abgehängt. Zum Beispiel resultiert das Kommando

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

in der Fehlermeldung:

`fatal: illegal input (window) 'spectrum get co56b68.127'`

weil **spectrum** kein erlaubtes Kommando im Menu `window` ist.

Tv akzeptiert **absolute** Kommandos von jedem Menu aus, falls ihnen **tv>** vorausgeht. Nach der Ausführung des Kommandos befindet man sich im gleichen Menu wie vorher. Um das Spektrum aus dem vorangegangenen Beispiel zu laden, gibt man den folgenden Befehl ein:

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

Eine Liste aller erlaubten Argumente eines Kommandos erhält man, indem man ein Fragezeichen hinter dem Kommando eingibt. Zum Beispiel:

`tv>` **window ?**

Einige Menus haben **Standardkommandos**, die nicht eingegeben werden müssen. In diesem Manual sind sie jeweils unterstrichen. Zum Beispiel ist das Standardkommando im Menu **spectrum** **get**. Folgende Kommandos sind gleichwertig zum Laden eines Spektrums:

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

`tv>` **spectrum co56b68.127**

Ebenso kann man in das **spectrum** Menu gehen

`tv>` **spectrum**

und den Namen des Spektrums eingeben:

`spectrum` **co56b68.127**

Um ein Menu zu verlassen und in ein Menu eine Ebene höher zu gelangen, drückt man CR.

3.7 Abkürzungen

Jedes Kommando kann abgekürzt werden. Wenn eine Abkürzung nicht eindeutig ist, wertet Tv den ersten passenden Alias oder, falls der nicht vorhanden ist, das erste passende Kommando. Die Reihenfolge der eingebauten Kommandos ist in der Regel alphabetisch sortiert, aber nicht notwendigerweise. Die Reihenfolge der Aliases ist so, wie sie definiert wurden. Um die Reihenfolge der Kommandos und Aliases in einem Menu zu sehen, drückt man ?.

Um zum Beispiel ein Spektrum zu plotten und dazu die Rahmenweite auf 9, 9, 3, 3 für den linken, rechten, oberen und unteren Rahmen zu ändern, damit Zahlen an die y–Achse geschrieben werden, gibt man folgendes Kommando ein:

`tv>` **window setup frame width 9 9 3 3**

Dieses Kommando kann folgendermaßen abgekürzt werden:

`tv>` **w se f w 9 9 3 3**

Das sind übrigens die Standardeinstellungen für das **Plotfenster**, das man mit folgendem Kommando erzeugen kann:

`tv>` **w c s plot**

3.8 Steuerzeichen

Um Aliases und Keyaliases zu erzeugen und Kommandodateien zu schreiben, benötigt man einen Satz von **Steuerzeichen**. Diese werden verwendet, um Tasten wie CR oder Esc darzustellen, die nicht in Dateien eingegeben werden können. Tv's Steuerzeichen sind in Tabelle 3.2 aufgelistet.

%	Führe Systemkommando aus.
@	Führe Kommandodatei aus.
#,!	Kommentar.
;	Carriage Return.
\	Escape.
?	Drucke eine Liste der möglichen Kommandos aus.
	Zeichenkette, der Inhalt wird nicht ausgewertet.
()	Klammern.
$(n)	Füge den Wert von Argument #n ein.
$(ENVIRONMENT)	Füge den Wert der Environmentvariablen ENVIRONMENT ein.
‘Unix Kommando‘	Das Ergebnis des Unix Kommandos wird an der Stelle dieses Ausdrucks eingefügt. Der Alias für help ist ein Beispiel.
ESC	Schalte zwischen Edit– und Cursormodus hin und her.
Ctrl-C	Beende Tv.
*, ?, [,]	Wildcards wie in /bin/ls (siehe spectrum get).

Tabelle 3.2: Zusammenfassung der Steuerzeichen des Kommandozeileninterpreters.

3.9 Aliases

Man kann seine eigene Menustruktur konfigurieren oder Kommandos erzeugen, umdefinieren oder umbenennen mit dem **Alias**–Mechanismus. Ein Alias ist eine Zeichenkette, die in einem Menu definiert ist. Beim Interpretieren der Kommandozeile überprüft Tv jedes Wort, ob ein entsprechender Alias definiert ist, und ersetzt die entsprechende Definition in der Kommandozeile.

Aliases werden im aktuellen Menu definiert. Man kann keine Aliases in einem anderen Menu definieren. Mit der Definition von Aliases muß man vorsichtig umgehen, da eingebaute Kommandos dadurch zerstört werden können. Ein Alias, um die y–Achse auf logarithmische Skalierung umzuschalten, ist der folgende:

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

Das zweite und dritte Argument muß in Anführungszeichen eingeschlossen werden, wenn es aus mehr als einem Wort besteht. Das dritte Argument wird an die Benutzungshinweise angehängt, die von **lin ?** ausgedruckt werden. Das Kommando **alias**, ohne Argumente, druckt alle im aktuellen Menu definierten Aliases. Das Kommando **unalias lin** löscht den Alias `lin`. Im zweiten Argument dürfen **Kommandozeilenargumente** oder **Environmentvariablen** verwendet werden, die den gleichen Konventionen wie in Shells, wie zum Beispiel der **bash** folgen. Um einen Alias `yscale` zu definieren, der die y–Achse gemäß einem Kommandozeilenargument skaliert, kann man folgendes eingeben:

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

wobei **$(1)** das erste Argument darstellt, das hinter dem **yscale** Kommando folgt. **alias** und **unalias** sind eingebaute Kommandos des Kommandozeileninterpreters, die in jedem Menu bekannt sind und nicht durch ? angezeigt werden.
Aliases, die standardmäßig im Rootmenu von Tv definiert werden, sind **help**, **quit**, **lin** and **log** und ihre Definitionen, die mit dem Kommando alias ausgegeben werden, sind:

```
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=ëxit"
```

3.10 Kommandodateien

Oft benutzte Kommandos, Definitionen oder Setups können in sogenannten **Kommandodateien** gespeichert werden. Um sie auszuführen gibt man das Sonderzeichen `@` gefolgt von dem jeweiligen Dateinamen ein. Für ihre Ausführung muß man sich in einem Menu befinden, das passend für die jeweilige Datei ist, was normalerweise das Rootmenu ist. Eine Kommandodatei muß immer mit einer `newline` enden.

Wenn keine Abkürzungen mit weniger als 4 Zeichen verwendet werden, laufen die Kommandodateien garantiert mit jeder Version von Tv.

Einige nützliche Kommandodateien um Matrizen, Schnitte und Spektren zu vergleichen, befinden sich im Verzeichnis */ikp/local/lib/tv*. Diese sind in Tabelle 3.3 aufgeführt.

Als Beispiel ist der Inhalt der Kommandodatei *cmp2mat* in abgekürzter Form aufgelistet. Um die Bedeutung der Kommandos zu verstehen, sieht man am besten in Abschnitt 5.2.1 und im Kapitel 6 nach.

```
wind dele projections
```

cmp2cut	Vergleiche zwei Schnitte.
cmp2mat	Vergleiche zwei Matrizen.
cmp2spc	Vergleiche zwei Spectren.
cmp3mat	Vergleiche drei Matrizen.
cmp4mat	Vergleiche vier Matrizen.

Tabelle 3.3: Beispiel Kommandodateien.

```
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

Mit dem **keyalias** Kommando kann man **Hotkeys** definieren, die im Cursor–Modus funktionieren wie oben beschrieben (siehe Kapitel 3.4.2). Um die Kommandos

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

an die Taste $\boxed{g}$ zu binden, ist das folgende Kommando nötig:

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

Man kann eine Liste aller Hotkeys ausgeben, mit dem Kommando:

tv> **keyalias**

Um den eben definierten Keyalias für die Taste $\boxed{g}$ zu löschen, gibt man das folgende Kommando ein:

tv> **keyunalias g**

Beim Start liest Tv die Datei */ikp/local/lib/tv/.tvkeys* ein, die die Hotkeys für den Cursor–Modus enthält. Der Inhalt dieser Datei ist im Anhang A abgebildet.

3.12 Wildcards

TV bietet einen Wildcard–Mechanismus für Dateinamen, die besondere Bedeutung für bestimmte Kommandos und Datentypen haben. Das Konzept der Wildcards ist nicht sehr eingänglich. Im Abschnitt 5.3 ist aber eine Anwendung beschrieben, die sehr zum Verständnis beiträgt.

3.12.1 Wildcard Konzept

Um zum Beispiel eine Matrix zu analysieren, braucht man die Matrix selbst, ihre Projektion und ein Verzeichnis, in dem die Ergebnisse der Analyse gespeichert werden. Durch die Benutzung von Wildcards braucht man nicht alle drei Dateinamen einzugeben, sondern nur einen Basisnamen. Dazu müssen die drei Dateien aber einer bestimmten Namenskonvention gehorchen, die durch die entsprechenden Wildcards festgelegt wird. Eine Matrixdatei muß die Endung .mtx haben, und außerdem muß sie in einem Verzeichnis abgespeichert werden, das den gleichen Namen wie die Matrix hat, allerdings ohne Endung. Um zum Beispiel die Matrix *220Th.mtx* zu laden, muß sie sich im Verzeichnis *./220Th* befinden. Die Standardwildcard für Matrizen ist:

`*mtx` → `!f/!t.mtx`

die in Tabelle 3.4 analysiert wird, für die Matrix

/matrix1/220Th/220Th.mtx

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

Tabelle 3.4: Analysierte Wildcard für die Matrix */matrix1/220Th/220Th.mtx*.

Neben der Matrix benötigt man noch die Projektion auf die y–Achse, deren Name durch die Wildcard *prospc festgelegt wird und die gemäß den Standardwerten, für die obige Matrix folgendermaßen lauten muß:

/matrix1/220Th/220Th.pry

Das Verzeichnis muß der Wildcard *cutdir (siehe Tabelle 3.6) gehorchen und wird von TV erzeugt, falls es nicht existiert. Sein Pfadname ist:

/matrix1/220Th/220Th.cutdir

Wenn alle notwendigen Dateien vorhanden sind, kann man die Analyse starten, mit dem Kommando

`tv>` **cut env <path>**

durch das Öffnen und Laden der Daten automatisch ausgeführt wird.

Der Dateiname für Matrizen (*mtx) setzt sich zusammen aus dem kompletten absoluten Pfadnamen des Verzeichnisses (!f), einem Slash (/), dem Basisnamen der Matrix (!t) und der Endung .mtx.

Die Bedeutung der Wildcards ist in Tabelle 3.5 aufgelistet. Die Wildcards müssen hinter einem **! (Ausrufezeichen)** stehen.

Die standardmäßig definierten Wildcards sind in Tabelle 3.6 aufgeführt. Eine Liste alles Wildcards kann man mit folgendem Befehl ausdrucken:

`tv>` **wildcard status**.

Man kann beliebige Einträge aus der Tabelle 3.5 kombinieren, durch Benutzung der Pipeline "\". Dieses Zeichen arbeitet wie das Pipe Zeichen | in Unix Shells, wie im folgenden Beispiel dargestellt, wo der linke Ausdruck ausgewertet und als Eingabe für den rechten Ausdruck verwendet wird. Wenn man sich im Verzeichnis */matrix1/220Th* befindet und folgendes eingibt,

Wildcard	Bedeutung	Beispiel
f	kompletter Name der Datei	/matrix1/220Th/220Th.tig0
t	Dateiname (tail)	220Th.tig0
h	absoluter Pfadname (!f-!t)	/matrix1/220Th/
r	Basisname (ohne Endung)	220Th
e	Endung	.mtx

Tabelle 3.5: Wildcards.

Wildcard	Definition	Datentype
*cutdir	!f/!t.cutdir	Schnitt Verzeichnis
*cutcal	!f/!t.cut.cal	Schnitt Eichung
*cutgat	!f/!t.gate	Schnitt Gate Marker
*cutspc	!f/!t.cut	Schnitt Spektrum
*cutspcgat	!f/!t.gt	Schnitt Spektrum Gate Marker
*cutspcbg	!f/!t.bg	Schnitt Spektrum Background Gate Marker
*cutspccl	!f/!t.cl	
*cutspccr	!f/!t.c	
*fit	!h/!t.fit	Fit Marker
*cal	!r.cal	Eichung
*plt	!r.fig	Plot
*rcl	!r.rcl	Recalibration Marker
*mtx	!f/!t.mtx	Matrix
*prospc	!f/!t.pry	Projektions Spektrum
*procal	!f/!t.pry.cal	Eichung für Projektions Spektrum

Tabelle 3.6: Standard Wildcard Definitionen.

tv> **cut env 220Th.test**
resultiert die modifizierte Wildcard für Matrizen
*mtx → !f/!t\!r.mtx
in dem Matrixdateinamen:
/matrix1/220Th/220Th.test/220Th.mtx.
Man kann eigene Wildcards definieren, oder bestehende verändern. Um die modifizierte Wildcard aus dem vorigen Beispiel zu definieren, gibt man folgenden Befehl ein:
tv> **wildcard enter *mtx "!f/!t\!r.mtx"**
Das zweite Argument sollte das führende Zeichen **asterisk (*)** enthalten. Außerdem muß das zweite Argument in Anführungszeichen eingeschloßen sein. Mit dem Kommando **wildcard enter** kann man auch bestehende Wildcards ändern. Um eine Wildcard zu löschen gibt man folgenden Befehl ein:
tv> **wildcard delete <wildcard>**

3.12.2 Beispiel für benutzerdefinierte Wildcards

Zum Beispiel kann man Wildcards zur Ein– und Ausgabe von Spektren definieren. Um eine Gruppe von geeichten Spektren zu schreiben, kann man die Wildcard *calspc definieren
tv> **wildcard enter *calspc "!r.cal_spc"**
und die Spektren mit dem Kommando schreiben:
tv> **spectrum write *calspc all**
Wenn zum Beispiel die Spektren prge0.0121 und prge1.0121 geladen sind, speichert das Kommando die Dateien
prge0.cal_spc und *prge1.cal_spc*
Die Ein– und Ausgabe von Dateien kann ebenfalls mit dem ls–file–Mechanismus durchgeführt werden, der im Kapitel 4.3.2 auf Seite 34 beschrieben wird.
Um die eben definierte Wildcard *calspc zu löschen, benutzt man das Kommando:
tv> **wildcard delete *calspc**

3.13 Konfigurationsdateien

Tv kann individuell für den jeweiligen Benutzer konfiguriert werden, zum Beispiel durch die Benutzung von Aliases (siehe Kapitel 3.9), durch Veränderung des Erscheinungsbildes der Fenster (siehe Anhang B, d.h. Farben und Cursorformen), und die Definition von Hotkeys für den Cursor–Modus (siehe Kapitel 3.11). Für alle diese Aufgaben kann man Konfigurationsdateien verwenden, die von Tv automatisch beim Start ausgeführt werden oder die von automatisch ausgeführten Dateien oder von Hand aufgerufen werden können.

Falls die Environmentvariable TVPATH nicht definiert ist, durchsucht Tv die Verzeichnisse in der folgenden Reihenfolge nach Konfigurationsdateien. Nur die erste gefundene *.tvrc* Datei wird ausgeführt.

1. */ikp/local/lib/tv/site.tvrc*
2. */ikp/local/lib/tv/.tvrc*
3. *<aktuelles Verzeichnis>/.tvrc*

Falls TVPATH gesetzt ist, werden nur die Verzeichnisse, die darin definiert sind, durchsucht. Zuerst wird nach *site.tvrc* gesucht und anschließend nach *.tvrc*. Um TVPATH in der Datei *.bashrc* zu setzen, fügt man dort die folgende Zeile ein:
export TVPATH="$TVPATH:/ikp/local/lib/tv:~/.:."

3.14 Fernbedienung

Da TV keine **Kontrollstrukturen** unterstützt, wie zum Beispiel Schleifen oder Verzweigungen, wurde eine Erweiterung programmiert, um von einer beliebigen Skriptsprache Befehle an TV zu schicken. Das Programm, das diese Aufgabe erfüllt heißt STV (das heißt send tv). Zum Beipiel lädt folgendes Skript, wenn es aus einer Shell gestartet wird, sieben Spektren.

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

STV sendet Kommandos nur an TV's, die dem gleichen Benutzer gehören, der STV gestartet hat, und wartet auf die Ausführung des Kommandos. Die Kommandos werden nur an das letzte gestartete TV geschickt, falls nicht eine Portnummer explizit angegeben wird. Diese wird unmittelbar nach dem Start von TV ausgegeben und in der Datei *~/.tv-port* gespeichert. Um Kommandos an ein TV mit der Portnummer 10000 zu schicken, kann zum Beispiel das folgende Skript benutzt werden.

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

TV Shell Kommandos wie "%rm -rf *", die von STV geschickt werden, werden ignoriert.

Kapitel 4

Analyse von Spektren

Inhaltsangabe

4.1 Einführung

Die Hauptaufgabe von Tv ist die Darstellung und Analyse von Spektren. In diesem Kapitel wird erklärt, wie man eine Anzahl von Spektren lädt und alle erforderlichen Schritte durchführt, um sie zu analysieren. Es wird eingegangen auf die Wahl des geeignetsten Fentertyps, die Eichung, Rekalibrierung, Normierung, arithmetische Operationen, Integration und Fit ebenso wie die Peaksuche und letztlich den Ausdruck der Spektren.

4.2 Graphikfenster

4.2.1 Graphikfenstertypen

Tv bietet verschiedene Typen von Graphikfenstern, die für verschiedene Aufgaben angepaßt sind.

Der einfachste Typ eines Graphikfensters ist der Typ **simple**, der aus einer einzigen Scheibe besteht. Man kann ihn zur Darstellung einer beliebigen Anzahl von Spektren verwenden.

Es gibt ein spezielles Fenster des Typs simple, das Fenster mit dem Namen **plot**. Da die Standardbreite des linken und rechten Rahmens nicht groß genug ist, um die Achsenbeschriftung auf dem Ausdruck zu erkennen, muß man sie vergrößern (siehe Abschnitt 4.6), oder ein neues Fenster mit dem Namen **plot** öffnen, in dem die Rahmenbreite schon angepaßt ist. Der entsprechende Befehl lautet:

`tv>` **window create simple plot**

Neben diesem einfachsten Fenstertyp gibt es drei weitere **paned** (*deutsch:* Scheibe) Fenster, nämlich xy–, x– and y–paned. Diese sind aus einer oder mehreren Scheiben zusammengesetzt, die übereinander angeordnet sind. Alle Scheiben teilen sich eine gemeinsame Statuszeile, jede hat eine eigene y–Achsenbeschriftung und es gibt insgesamt zwei x–Achsenbeschriftungen. Im Fenstertyp xy–paned sind sowohl die x–Skalen als auch die y–Skalen unabhängig, die Skala am oberen Ende des Fensters gehört zum obersten Spektrum und die am unteren Ende zum untersten. Bei mehr als zwei Scheiben gibt es also Spektren ohne x–Skala. In Fenstern der Typen x– oder y–paned gehören die x–Skalen zu allen dargestellten Spektren, wenn man dann in einem der Fenster x–view Marker mit der Taste [Spacebar] setzt und den Ausschnitt mit [e] expandiert, verändert sich der Viewport aller Spektren. In einem Fenster des Type y–paned (siehe Abbildung 4.2) sind auch alle y–Skalen gekoppelt. Erzeugt werden die Fenster mit den folgenden Kommandos, wobei <number> die Anzahl der Scheiben angibt.

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

Fenster des Typs paned sind gut geeignet zum Vergleich mehrerer Spektren. Zum Beispiel kann man ein Fenster des Typs x–paned zum Vergleich von vier Abständen eines Lebensdauerexperiments verwenden, das mit folgendem Kommando erzeugt wird:

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

Die übrigen Graphikfenstertypen setzen sich aus jeweils zwei Unterfenstern zusammen. Das **fit** Fenster ist ein Fenster vom Typ **x–paned**, in dessen unterer Scheibe das Residuum des Fits dargestellt wird. Das **cut** Fenster hat ein kleines Unterfenster in seiner oberen rechten Ecke, in dem die Positionen der cut Marker in der Projektion angezeigt werden (siehe Abbildung 2.5).

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

Abbildung 4.1 zeigt ein Beispiel eines Fensters des Typs fit, das mit dem folgenden Befehl erzeugt wurde:

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

In der unteren Scheibe ist das Residuum dargestellt.
Um das graphikfenster mit dem Namen "fit example" zu löschen, verwendet man folgendes Kommando:

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

4.2.2 Konfiguration des Graphikfensters

Wenn es mehr als ein Graphikfenster gibt, ist es nicht eindeutig, wo im Textfenster eingegeben Befehle ausgewertet werden. Auf das entsprechende Fenster muß ein Fo-

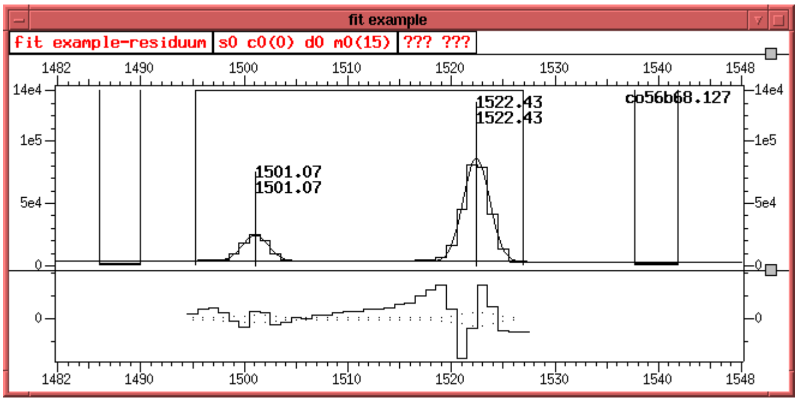

Abbildung 4.1: Ein Fit von zwei Peaks in einem ^{56}Co Spektrum in einem Fenster des Typs **fit**. In der unteren Scheibe wird das Residuum angezeigt.

kus gesetzt werden. Das betrifft Befehle zum Ausdrucken der Spektren, ändern der x–Skala oder anderer Eigenschaften des Fensters, wie z.B. die Breite des Rahmens oder die Länge der Skalenstriche. Alle Befehle zur Konfiguration der Graphikfenster sind aufgelistet in Abschnitt 6.9 im Kapitel 6.
Standardmäßig setzt man den Fokus auf ein Fenster, indem man den Mauszeiger hineinbewegt. Man kann den Tastaturfokus auch mit dem folgenden Kommando auf das Fenster mit dem Namen "lifetime" setzen, vorausgesetzt es existiert:

tv> **window setup keyboard–focus title lifetime**

und schaltet zurück zum Standardverhalten mit dem Hotkey L oder dem Befehl:

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

Drei Werte für die kurzen, mittellangen und langen Skalenstriche sowie ein weiterer für den Abstand zwischen Skalenstrichen und Achsenbeschriftung können eingestellt werden. Das Kommando um diese Werte zu ändern, die in der Einheit "Zeichengröße" angegeben werden, ist:

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

Eine andere Gruppe von Befehlen konfiguriert die y–Skala der Fenster. Zum Beispiel ändert der folgenden Befehl die y–Skala in die logarithmische Darstellung:

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

Diese drei Kommandos werden standardmäßig im Alias **log** zusammengefaßt.

Die graphische Darstellung läßt sich abschalten um die Ausgabe der Graphik zu verhindern und damit die Ausführung von Kommandodateien zu beschleunigen. Zum Ausschalten der Graphik führt das Kommando

tv> **graphic suspend**

und um sie wieder einzuschalten verwendet man den Befehl

tv> **graphic resume**

4.3 Laden und speichern von Spektren (I/O)

Tv unterstützt verschiedene Funktionen für die Ein– und Ausgabe von Spektren, die im folgenden dargestellt sind:

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

<fname> ist der Name der Spektrumdatei. **<fmt>** ist das Format der Spektrumdatei (s.u.). **<ind>...** sind die Buffernummern für das Spektrum. **<wc>** ist eine Wildcard, die gemäß Abschnitt 3.12.2 definiert ist. Die Kommandos im Menu `spectrum` werden im folgenden beschrieben.

read lädt das angegebene Spektrum in den ersten Buffer der Liste und kopiert es in die anderen aufgeführten Buffer. Tv stellt den gleichen Buffer immer in der gleichen Farbe dar. Wenn ein bestimmtes Spektrum immer in der gleiche Farbe dargestellt werden soll, kann man das durch Verwendung des read Kommandos erreichen.

get liest Spektren und ist das Standardkommando des `spectrum` Menus, d.h. es muß nicht eingegeben werden.

Der Unterschied zwischen **read** und **get** ist, daß **read** ein Spektrum in einen oder mehrere Buffer lädt, die explizit angegeben werden müssen, während **get** beliebig viele Spektren in die nächsten freien Buffer liest, die von Tv automatisch ausgewählt werden.

write speichert die Spektren in die angegebenen Dateien. Der Befehl bezieht sich auf alle (**all**), die angezeigten (**shown**) oder das aktive (**active**) Spektrum, oder auf die Spektren, die durch die **index** Liste definiert werden.

format setzt das Format für die Ein– oder Ausgabe der Spektren. **input** ist das Standardkommando im Menu `spectrum format`. Eine Liste der erlaubten Formate ist im Anhang C abgedruckt.

Tv führt I/O Operationen mit der Mfile–Bibliothek[1] durch. Falls Spektren geladen werden sollen, die nicht mit Funktionen der Mfile–Bibliothek gespeichert wurden, kann es notwendig sein, das Format explizit anzugeben.

Um zum Beispiel das Spektrum *prge0.0121* in die Buffer #0 bis #3 zu laden und sie zum Vergleich in einem Fenster vom Typ y–paned (siehe Abschnitt 4.2.1) darzustellen, erzeugt man zuerst ein entsprechendes Fenster:

tv> **window create ypaned ypwindow 4**

worauf Tv angibt welches Spektrum und welcher Cut in welcher Scheibe aktiv ist:

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

Nachdem das Fenster erzeugt ist, lädt man das Spektrum mit dem Kommando:

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

und Tv gibt an, in welchen Buffer das Spektrum geladen wurde und kopiert es in die anderen Buffer.

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

Das resultierende Fenster ist in Abbildung 4.2 dargestellt.
Als Beispiel für das get Kommando wird das Spektrum *ge0.0121* nach der obigen Operation geladen. Tv lädt es in den nächsten freien Buffer.

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

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

[1] Die Mfile–Bibliothek wurde von S. Esser als Diplomarbeit für das IKP programmiert und stellt Funktionen zum Speichern und Laden komprimierter Spektren zur Verfügung. Der Kompressionsfaktor ist etwa 1:5

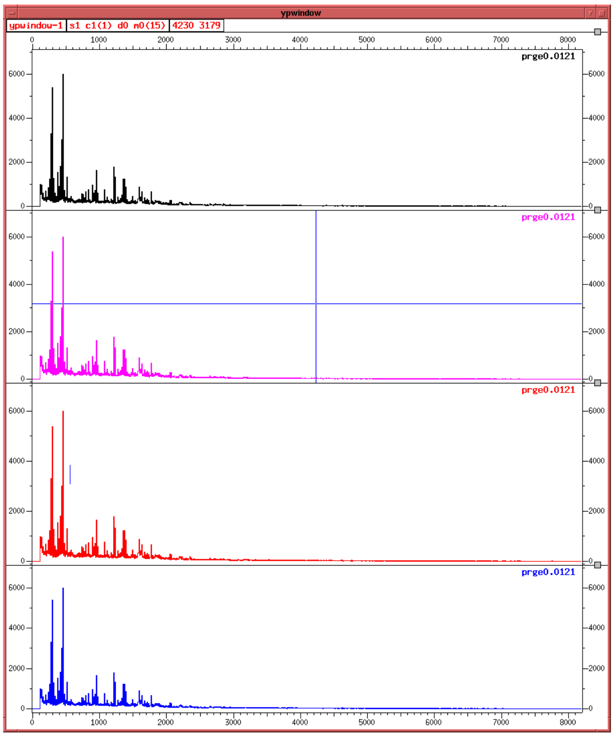

Abbildung 4.2: Ein Fenster des Typs y–paned, das das gleiche Spektrum in allen vier Scheiben anzeigt.

Damit wird das Spektrum mit der Auflösung 8k, das im Mfileformat (siehe Anhang C) linecompress Modus 2 gespeichert wurde, in den Buffer #4 geladen.

Um Tv zu zwingen ein bestimmtes Format zu verwenden, muß dieses entweder beim Laden des Spektrums angegeben werden, oder indem man das Standardformat mit dem format Kommando entsprechend setzt. Die Wahl eines falschen Formats

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

führt zu einem Fehler:

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

Es folgt ein Beispiel für das Format–Kommando:

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

Die maximale Anzahl von Buffern ist standardmäßig auf 16 gesetzt und kann beim Start von Tv in der Kommandozeile (siehe Abschnitt 3.2) geändert werden.

4.3.1 I/O mit mehreren Spektren

Um mit dem get Kommando mehrere Spektren zu laden, können bestimmte Muster für Dateinamen verwendet werden. Tv expandiert diese Muster (wie die Shell) und lädt alle passenden Dateien. Als Beispiel werden die Spektren *prge0.0121* . . . *prge5.0121* geladen.

tv> **spectrum prge[012345].***

Tv verteilt sie automatisch in die Buffer #0 bis #5, falls diese frei sind:

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

⋮

Tv hat die Spektren mit dem Format 8k.lc:2 in die Buffer #0 bis #5 geladen. Der zweite, allgemeinere Weg um mehrere Spektren zu laden, ist der ls–file–Mechanismus (siehe Abschnitt 4.3.2), der Spektren lädt (oder andere Operationen mit ihnen ausführt), die in einer speziellen Datei (der ls–Datei) aufgelistet sind. Die dritte Methode um mehrere Spektren zu laden ist die Verwendung von Wildcards (siehe Abschnitt 3.12.2).

4.3.2 ls–file–Mechanismus

Um eine Operation auf ein Gruppe von Spektren anzuwenden, schreibt man ihre Namen in eine ls–Datei. Eine ls–Datei kann man aus Tv heraus erzeugen, z.B. für die Spektren aus dem letzten Abschnitt, mit dem Befehl:

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

Vom ls Befehl hat der Mechanismus auch seinen Namen. Die Syntax eines ls–Datei Kommandos sieht folgendermaßen aus:

`tv>` **s ls <operation> <option>**

In Tabelle 4.1 sind alle erlaubten Befehle aufgeführt:

add	Addiert eine Anzahl Spektren zum angegebenen Buffer.
commandget	Lädt Spektren und führt für jedes ein Kommando aus.
get	Lädt eine Anzahl Spektren.
subtract	Subtrahiert eine Anzahl Spektren von einem angegebenen Buffer.

Tabelle 4.1: Erlaubte Operationen für das **ls–file** Kommando.

Falls eine Positionseichung für alle Spektren, die an der ls–file Operation beteiligt sind, geladen ist, werden die Operationen **add** und **subtract** auf die geeichten Spektren angewendet. Wenn die Spektren ungeeicht addiert oder subtrahiert werden sollen, muß die Eichung für mindestens ein Spektrum gelöscht werden.
Um auf Spektren gezielt zuzugreifen, die in einer ls–Datei stehen, dienen die Optionen, die in Tabelle 4.2 aufgeführt sind.

ls–file	Öffne eine ls–Datei.
+	Bearbeite das nächste Spektrum in der Liste.
++	Bearbeite alle Spektren bis zum Ende der Datei.
–	Bearbeite das vorhergehende Spektrum aus der Liste.
––	Bearbeite alle vorhergehenden Spektren.
line–index	Bearbeite das Spektrum an der Position line–index.

Tabelle 4.2: Optionen für die ls–file Operationen, um festzulegen, welches Spektrum bearbeitet wird.

Um eine ls–Datei in Tv zu verwenden, muß sie zuerst geöffnet werden:

`tv>` **spectrum ls–file open spc–file**

Um den Inhalt einer ls–Datei aufzulisten, kann man das folgende Kommando verwenden:

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

Der Zeiger wird auf den nächsten Eintrag gesetzt, nachdem ein Eintrag aufgelistet wurde. Wenn eine andere ls–Datei geöffnet ist als <filename>, dann wird die alte geschlossen und die neue geöffnet. Falls aber keine ls–Datei geöffnet ist, führt das Kommando list zu einem Fehler. Das Kommando list ohne Argumente zeigt den Index des nächsten Eintrags.

Um den Zeiger in der ls–Datei auf die Position <number> zu setzen, dient das Kommando:

tv> **spectrum ls–file position <number>**

Um alle Spektren, vom aktuellen Index bis zum Ende der Datei zu laden, verwendet man den Befehl:

tv> **spectrum ls–file get ++**

Geschlossen wird die ls–Datei mit:

tv> **spectrum ls–file close**

Als Beispiel werden die Spektren aus dem letzten Abschnitt geladen. Zuerst muß die ls–Datei *spc–file* geöffnet werden:

tv> **spectrum ls–file open spc–file**

Dann werden alle Spektren geladen:

tv> **spectrum ls–file get ++**

Natürlich können nicht mehr Spektren geladen werden, als Buffer vorhanden sind.

Die Syntax aller ls–file Kommandos ist in Abschnitt 6.8.10 aufgeführt. Eine Liste aller Hotkeys für die ls–file Operationen ist in Tabelle 4.3 angegeben.

Tastenkombination	Tv–Kommandos	Funktion
l +	tv> spectrum delete all; tv> spectrum ls-file get ++;	Lösche alle Spektren und lese entsprechend den Regeln für das ls–file Kommando (siehe Abschnitt 6.8.10)
l -	tv> spectrum delete all; tv> spectrum ls-file get --;	Lösche alle Spektren und lese entsprechend den Regeln für das ls–file Kommando (siehe Abschnitt 6.8.10)
l g	tv> edit; tv> spectrum ls-file get;	Fragt nach den zu ladenden Spektren und liest gemäß den Regelndes ls–file Kommandos (siehe Abschnitt 6.8.10)
l o	tv> edit; tv> spectrum ls-file open;	Fragt nach dem Namen einer ls–Datei und öffnet sie

Tabelle 4.3: Hotkeys für die **ls–file** Operationen.

4.4 Buffer Operationen

Tv stellt drei Kommandos zum Kopieren, Erzeugen und Löschen von Spektren in Buffern zur Verfügung:

tv> **spectrum copy <source index> {<destination index>...}**

tv> **spectrum create <name> <resolution> <index>**

tv> **spectrum delete {<index>...}**

copy kopiert den Buffer <source index> in die Buffer <destination index>.... **create** erzeugt ein **leeres Spektrum** mit Namen <name> und Auflösung <resolution> im Buffer <index>. **delete** löscht die Spektren in den Buffern <index> .... Das delete Kommando kann mit dem Hotkey k ausgeführt werden.

4.5 Funktionen der Maus

Die Benutzung der Maus wird im Abschnitt 3.5 beschrieben.

4.6 Ausdruck und Beschriftung

Labels (Beschriftungen) werden ausschließlich verwendet, um Punkte in einem Spektrum zu markieren, z.B. einen Peak. Sie haben keine Wirkung auf irgendeine Operation in Tv.

Labels bestehen aus einer Zeichenkette (**info string**) und einer **Position** auf einer Achse des Spektrums. Die Zeichenkette wird normalerweise automatisch von Tv erzeugt und enthält die Energie oder Kanalzahl. Man kann auch eigene Zeichenketten definieren. Die Position kann auf der x–Achse (Labels und vertikale Marker oder auf der y–Achse (horizontale Marker) liegen. Die Position der Zeichenkette auf der anderen Achse wird von Tv berechnet.

Man kann Label setzen für **peaks**, **fits**, **cuts** und **user** Marker, wobei jede beliebige Kombination von Energie, Kanalzahl und Info String ausgegeben werden kann. Mit dem folgenden Kommando, das eine Position als Standardargument nimmt, kann ein Label erzeugt werden:

`tv>` **label user enter** {**cursor** | <u>**value**</u> | **offset**}

Die Kombination der Daten, die an dem Label ausgedruckt werden sollen, wird mit dem folgenden Kommando festgelegt, wobei Paare von Argumentwerten wie Ein–Ausschalter wirken. Zum Beispiel schaltet **calibrated** den Ausdruck der Energie an, während **nocalibrated** ihn ausschaltet.

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

Es können mehrere Labellisten gleichzeitig verwaltet werden. Das Kommando zum Anlegen einer neuen Labelliste oder zum Aktivieren einer bestehenden ist das folgende:

`tv>` **label user create <title>**

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

Man kann Label zum Markieren eines Peaks im ausgedruckten Spektrum verwenden. Um einen Peak mit einem beliebigen Text zu markieren, dient der folgende Befehl:

`tv>` **label user enter <Position> <Farbindex> <"Text">**

Die Farben sind in der Tabelle 4.4 angegeben.

Farbindex	Farbe	Farbindex	Farbe
0	Gelb	7	Pink
1	Magenta	8	MediumOrchid
2	Rot	9	Seashell4
3	Blau	10	Firebrick
4	Weiß	11	LightSlateBlue
5	Wheat	12	Aquamarine
6	Cyan	13 ...	Gelb

Tabelle 4.4: Farbindizes für Label.

Falls die Marker im Ausdruck stören, können sie mit den folgenden Befehlen ein– und ausgeschaltet werden:

`tv>` **window hide labellist user**

`tv>` **window show labellist user**

Da das spezielle Fenster zum Ausdrucken von Spektren nur eine Scheibe hat, man aber auch einmal Fenster mit mehreren Scheiben drucken möchte, läßt sich die Rahmenbreite ändern. Als Beispiel wird ein Fenster mit drei Scheiben erzeugt und die Rahmenbreite vergrößert. Dazu wird nach dem Erzeugen des Fensters die Maus in die oberste Scheibe bewegt und folgender Befehl eingegeben.

tv> **window setup frame width 9 9 3 0**

Dann bewegt man den Mauszeiger in die mittlere Scheibe und gibt den Befehl ein:

tv> **window setup frame width 9 9 0 0**

Und zum Schluß wird die unterste Scheibe bearbeitet:

tv> **window setup frame width 9 9 0 3**

So kann man auch ein Fenster mit mehreren Scheiben und Beschriftungen an den y–Achsen ausdrucken.

4.7 Marker

Verschieden Kommandos von Tv, wie z.B. fit oder recalibration, arbeiten nicht auf dem ganzen Spektrum sondern nur auf bestimmten Bereichen (regions). Solche Bereiche werden mir Markern eingegrenzt. Es gibt eine Reihe Hotkeys für alle möglichen Regionen, die in Tabelle 4.5 zusammengestellt sind.

Marker beziehen sich immer auf ein Fenster und bleiben erhalten, selbst wenn keine Spektren in diesem dargestellt werden. Daher kann man mit einem Satz Marker viele Spektren bearbeiten, indem man die Liste der dargestellten Buffer ändert oder andere Spektren in die dargestellten Buffer lädt.

4.8 Eichung

Mit Tv lassen sich verschiedene Parameter eichen, nämlich **Efficiency**, **linker Tail**, **rechter Tail**, **Position** and **Breite**. Man kann den **startchannel** setzen, der festlegt welche Energie dem Kanal 0 entspricht, und die Einheit (**unit**) der x–Achse.

Um zum Beispiel die Position und die Breite eines Peaks in einem Spektrum in Buffer 0 in keV zu eichen, wobei die Energie für Kanal 0 17keV ist, muß man den Startkanal auf 17 setzen, die unit auf keV und die Eichung, wie im folgenden gezeigt, durchführen.

tv> **calibration startchannel enter 1000**

tv> **calibration unit enter keV**

tv> **calibration position enter 1204 1197 1579 1563**

tv> **calibration width enter 1213 4.26 1258 4.63**

Tv druckt die Parameter für die Positionseichung mit dem Kommando:

tv> **calibration position list 0**

und für die Breiteneichung mit dem Befehl:

tv> **calibration width list 0**

Man kann auch die Ergebnisse für den kompletten Satz (set) von Eichungen gleichzeitig ausdrucken:

tv> **calibration set list 0**

Man kann die Energieeichung in eine Datei, mit dem Standarddateinamen entsprechend der Definition der Wildcard *cal, speichern oder man muß einen Dateinamen angeben

tv> **calibration position write [<filename>]**

und aus der Datei *cal oder einer anderen Datei laden mit dem Hotkey E oder dem vollen Kommando:

tv> **calibration position read**

Man kann die Eichung auf Spektren in anderen Buffern anwenden:

tv> **cali position copy <src ind> {{<ind>...} | acti | all | show | visi}**

Tastenkombination	Tv–Kommandos	Funktion
$\boxed{b}$	tv> fit mark bg-region enter cursor;	Setze einen Untergrund–region Marker an der Cursor Position
$\boxed{c}$	tv> cut mark cut enter cursor;	Setze einen Cut Marker an der Cursor Position
$\boxed{G}\boxed{G}$	tv> cut marker gate enter cursor;	Definiere einen Gate Marker.
$\boxed{h}$	tv> window mark horizontal enter cursor;	Setze einen horizontalen Marker
$\boxed{o}$	tv> normalization mark enter cursor;	Setze einen Normierungs Marker an der Cursor Position
$\boxed{p}$	tv> fit mark peak enter cursor;	Setze einen Peak Marker an der Cursor Position
$\boxed{r}$	tv> fit mark region enter cursor;	Setze einen Fit–region Marker an der Cursor Position
$\boxed{R}\boxed{r}$	tv> recalibration marker enter cursor;	Setze einen Rekalibrierungs Marker an der Cursor Position
$\boxed{-}\boxed{b}$	tv> fit mark bg-region delete cursor;	Lösche den Untergrund Marker, der dem Cursor am nächsten ist
$\boxed{-}\boxed{c}\boxed{g}$	tv> cut mark gate delete cursor;	Lösche den Gate Marker, der dem Cursor am nächsten ist
$\boxed{-}\boxed{c}\boxed{b}$	tv> cut mark bg-gate delete cursor;	Lösche den Untergrund–Gate Marker, der dem Cursor am nächsten ist
$\boxed{-}\boxed{n}$	tv> normalization mark delete;	Lösche alle Normierungs Marker
$\boxed{-}\boxed{p}$	tv> fit mark peak delete cursor;	Lösche den Peak Marker, der dem Cursor am nächsten ist
$\boxed{-}\boxed{r}$	tv> fit mark region delete cursor;	Lösche den Fit–region Marker, der dem Cursor am nächsten ist
$\boxed{-}\boxed{A}$	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;	Lösche alle Marker. Es werden immer noch Marker dargestellt, die Teil des autonomen Fits sind
$\boxed{-}\boxed{B}$	tv> fit mark bg-region delete all;	Lösche alle Untergrund Marker
$\boxed{-}\boxed{C}$	tv> cut mark cut delete; tv> window hide mark cut;	Lösche alle Cut Marker
$\boxed{-}\boxed{F}$	tv> fit mark fit delete; tv> fit delete; tv> fit delete activ; tv> window hide mark fit;	Lösche den aktuellen Fit und alle seine Marker
$\boxed{-}\boxed{P}$	tv> fit mark peak delete 0 uncalib 8192 uncalib;	Lösche alle Peak Marker
$\boxed{-}\boxed{R}$	tv> recalibration marker delete cursor;	Lösche den Rekalibrierungs Marker, der dem Cursor am nächsten ist

Tabelle 4.5: Hotkey Definitionen für alle Marker, die Tv kennt.

Alle eingegebenen oder geladenen Eichungen werden automatisch in alle Buffer kopiert. Die Syntax aller Kommandos des Menus `calibration` ist in Abschnitt 6.2 zusammengestellt.

4.9 Rekalibrierung

Aufgrund von elektronischen Effekten während der Datenaufnahme können Peaks in manchen Spektren verschoben sein, wodurch Peaks, die eigentlich übereinanderliegen sollten, es nicht sind. Das Verschieben der Spektren, mit dem sie in die richtige Position gebracht werden, wird von Tv mit den Kommandos des Menus `recalibration` gemacht. Normalerweise ist das eine lineare Eichung.

Als Beispiel werden die beiden Spektren *prge[35].0121* geladen, die gegeneinander verschoben sind. Um die Rekalibrierung durchzuführen, muß mindestens ein Rekalibrierungs Bereich definiert werden. Die Marker für die Region werden mit der Tastenkombination R r gesetzt und dann wird die Rekalibrierung mit dem Kommando

`tv>` **recalibration create**

durchgeführt. Tv fragt dann nach einer Reihe weiterer Informationen, wie dem **Referenzspektrum** und einer Liste zu rekalibrierender Spektren. Standardmäßig wird der Typ der Rekalibrierung auf squared–distance gesetzt. Den Typ kann man mit dem folgenden Kommando auf center–of–mass setzen:

`tv>` **recalibration type center–of–mass**

Um das Ergebnis der Rekalibrierung zu sehen, muß die Darstellung des Fensters mit dem Hotkey CR oder dem folgenden Kommando aufgefrischt werden:

`tv>` **window redisplay**

Eine Liste der Shiftkoeffizienten erhält man mit:

`tv>` **recalibration list**

Eine Liste aller Kommandos des Menus `recalibration` ist in Abschnitt 6.7 zusammengestellt.

4.10 Normierung

Zum Vergleich von Spektren ist es oft sinnvoll sie zu normieren, um die Höhe ihrer Peaks auf vergleichbare Werte zu bringen.

Um eine Normierung für einen bestimmten Peak durchzuführen, müssen zunächst Marker mit dem Hotkey o gesetzt werden, einer auf der linken Seite des Peaks und einer auf der rechten, die den Normierungs–Bereich festlegen. Untergrund Marker für die Normierung werden folgendermaßen gesetzt:

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

Spektren werden in Tv mit dem folgenden Kommando normiert:

`tv>` **spectrum norm**

Um normierte Spektren zu identifizieren, wird im Spektrenstatus (siehe Abschnitt 6.8.18) ein Asterisk (*) mit ausgegeben.

Ein Beispiel rekalibrierter und normierter Spektren ist in Abbildung 4.3 dargestellt, wo der rechte Peak als Normierungsbereich markiert war.

Eine Liste aller Kommandos des Menus `normalization` ist in Abschnitt 6.6 zusammengestellt.

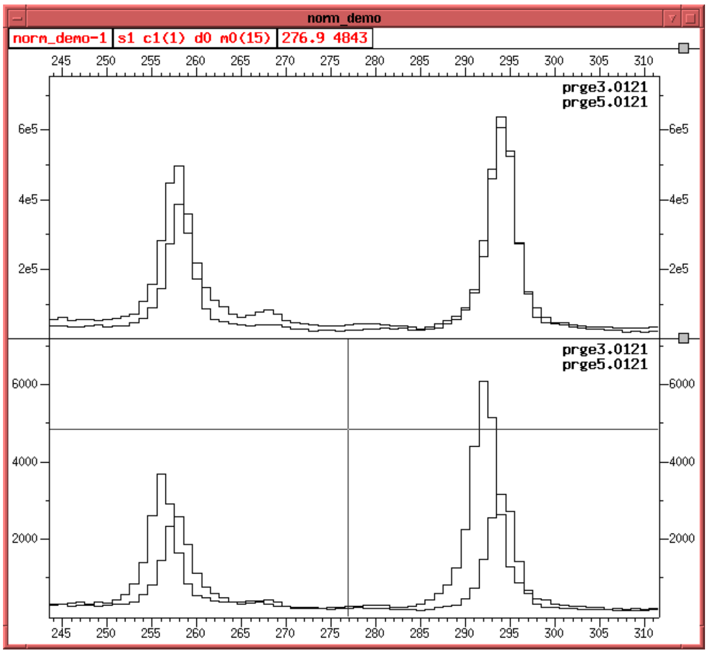

Abbildung 4.3: Ein Beispiel eines rekalibrierten und normierten Spektrums. In der unteren Scheibe sind die Originaldaten abgebildet.

4.11 Arithmetische Operationen

Tv stellt fünf Kommandos zur Verfügung, um arithmetische Operationen mit Spektren durchzuführen:

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 summiert die Inhalte aller Spektren die durch das zweite Argument angegeben werden und speichert sie in dem Buffer, der durch das erste Argument (<idx>) definiert wird. Als zweites Argument wird als Standardwert entweder ein ganzzahliger Wert als Bufferindex oder ein Textwert als Dateiname akzeptiert. Das Argument **all** summiert alle Spektren. Falls der Zielbuffer leer war, bekommt er den Namen des ersten addierten Spektrums.

subtract arbeitet analog zu add, subtrahiert aber die Spektren.

maximum ermittelt das Maximum für jeden einzelnen Kanal aus allen Spektren, die durch das letzte Argument gewählt werden und speichert es in den Buffer, der durch das erste Argument angegeben wird. Als Standardwert für das letzte Argument wird eine Liste von ganzzahligen Zahlen akzeptiert, die die zu vergleichenden Buffer angeben.

minimum arbeitet analog zu maximum, ermittelt aber das Minimum für jeden Kanal.

4.12 Fit und Integration

Die interessanten Ergebnisse der quantitativen Analyse von Spektren sind die Position, das Volumen und die Form der beobachteten Peaks. Die Analyse muß eine vernünftige Berücksichtigung des Untergrundes beinhalten. Da der Untergrund eine globale Struktur hat, wogegen die Peaks im Spektrum lokale Strukturen sind, kann man den Untergrund mit einem Polynom, gewöhnlich vom zweiten Grad, abschätzen. Andererseits haben Teilchenspektren breite Peaks, was eine physikalischere Beschreibung des Untergrunds nötig macht. Das wird durch einen Exponentialterm neben dem Polynom gewährleistet. Diese Untergrundfunktion kann auch im oberen Teil des Spektrums eines Germaniumdetektors nütlich sein, da der Exponentialterm die Efficiency in etwa reproduziert.

Tv kennt drei Methoden für die Analyse von Spektren, nämlich Integration, Peaksuche und Fit, wobei zwei verschiedene Funktionen für den Fit und zwei für die Abschätzung des Untergrundes zur Verfügung stehen. Die Methoden werden hier im weiteren beschrieben, während die Definitionen im Anhang D angegeben sind.

4.12.1 Integration

Der Hauptvorteil der Integration im Vergleich mit dem Fit liegt darin, daß keine Annahme über die Form des Peaks gemacht werden muß. Tv berechnet Volumen, Schwerpunkt, Breite (korrigiert um den Faktor $2 \cdot \sqrt{2 \cdot \ln 2}$ um die FWHM zu erhalten) und die Schiefheit (Asymmetrie). Dabei berücksichtigt Tv existierende Untergrundfits und berechnet untergrundkorrigierte Werte. Falls Untergrund Bereiche definiert aber nicht gefittet sind, berechnet Tv den mittleren Untergrund und verwendet diesen Wert für die Untergrundkorrektur. Die Ergebnisse der Integration hängen, in wachsendem Maß vom Volumen bis zur Schiefheit, von einer guten Beschreibung des Untergrunds ab. Integration ist passend für γ –Spektren mit schlecht beschreibbarer Peakform.

Mit Integration können nur gut getrennte Peaks ausgewertet werden. Sonst ist es notwendig Annahmen über die Form der Peaks zu machen und einen Fit für die Dekomposition überlappender Peaks zu verwenden. Weiterhin ist es aufwendig eine größere Anzahl Peaks zu integrieren.

Um eine Integration durchzuführen, muß ein Fit–Bereich definiert werden, mit dem Hotkey $\boxed{\text{r}}$ einmal für die linke Grenze und einmal für die rechte Grenze. Die Untergrund–Bereiche werden mit dem Hotkey $\boxed{\text{b}}$ definiert werden und der Untergrund Fit wird mit dem Hotkey $\boxed{\text{B}}$ oder folgendem Kommando durchgeführt:

`tv>` **fit background–create**

Die Integration wird durchgeführt und ein Kurzstatus ausgegeben mit dem Hotkey $\boxed{\text{I}}$. Eine Integration ohne Status wird durchgeführt mit

`tv>` **fit integration–create**

und die Ergebnisse werden ausgedruckt mit

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

wobei **full** detaillierte Informationen über den Untergrund und alle Parameter des Peaks ausgibt.

Laden und Speichern von Integrationen

Siehe Abschnitt 4.12.6 **(Laden und Speichern von Fits)**.

4.12.2 Peaksuche

Um eine Liste aller Peaks in einem Spektrum zu erhalten, bietet Tv die Funktion Peaksuche. Die verwendete Fitfunktion ist eine einfache Gaußfunktion mit konstantem Untergrund. Vor der Peaksuche muß eine Breiteneichung durchgeführt werden, um die Breite der Gaußfunktion zu erhalten. Um die zwei benötigten Breiten für die Breiteneichung zu erhalten, kann die Quickfit (siehe Abschnitt 2.3.3) Funktion verwendet werden und die Breiteneichung wird durchgeführt mit:

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

Die Amplituden der Gaußfunktion und der Untergrund werden gefittet und das Wahrscheinlichkeitsintegral mit einem vorgegebenen Wert verglichen, das standardmäßig 99,99% ist. Den vorgegebenen Wert kann man sich anschauen oder ihn ändern mit:

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

Wenn das Wahrscheinlichkeitsintegral den Grenzwert übersteigt, wird die Position mit einer Genauigkeit von 1/10 Kanalbreite ermittelt und der Peak zu einer Liste addiert. Die Peaksuche wird mit folgendem Kommando durchgeführt:

tv> **fit psearch create**

das die Anzahl der gefundenen Peaks als Ergebnis ausgibt. Eine Liste aller gefundenen Peaks mit ihren Positionen, Breiten und Volumina gibt der Befehl:

tv> **fit psearch list**

Falls nicht im ganzen Spektrum nach Peaks gesucht werden soll, kann mit Markern der Bereich begrenzt werden. Einen Marker setzt man mit:

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

Mit den folgenden Befehlen kann ein Paar Marker gespeichert und wieder geladen werden:

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

Abspeichern von Peaklisten

Die Ergebnisse der Peaksuche können abgespeichert werden. Entweder im Binärformat mit dem Befehl

tv> **fit psearch write**

in die Datei <Spektrumname>.fit oder im Klartext mit einem der Befehle

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

in eine Protokolldatei, die mit einem der Befehle

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

geöffnet wurde. Die Protokolldatei wird mit dem Befehl

tv> **fit result–file close**

oder beim Verlassen von Tv geschlossen.

Laden von Peaklisten

Peaklisten können nur aus Binärdateien wieder eingelesen werden. Das geschieht mit dem Befehl

tv> **fit psearch read**.

Mit dem Befehl

tv> **fit psearch open**

werden neben der Peakliste auch eventuell abgespeicherte Fits eingelesen und deren Binmarker im Spektrum angezeigt.

4.12.3 Der Fit

Ein **Fit** besteht aus **Markern** und **Parametern** und ist autonom, d.h. wenn das gefittete Spektrum gelöscht wird, bleiben die Fitparameter erhalten. Dadurch kann man ein anderes Spektrum laden und die gleichen Fitparameter und Marker wiederverwenden.

Um die besten Ergebnisse für den Fit zu erhalten, müssen **Fit– und Untergrund–Bereiche** sowie **Peak Marker** definiert werden. Um den Fit–Bereich zu definieren, d.h. die Region des Spektrums, die die zu fittenden Peaks enthält, verwendet man den Hotkey $\boxed{\text{r}}$ jeweils für die linke und rechte Grenze des Fit–Bereichs. Alle Peaks innerhalb dieses Bereichs werden mit $\boxed{\text{p}}$ markiert. Dann **führt man den Fit durch** mit $\boxed{\text{F}}$ oder dem Kommando:

tv> **fit region–create**

Durch die Definition einer beliebigen Anzahl von Untergrund–Bereichen mit dem Hotkey $\boxed{\text{b}}$ kann der Fit verbessert werden. Mit $\boxed{\text{B}}$ wird der spektrale Untergrund gefittet und beim anschließenden Fit der Peaks mit $\boxed{\text{F}}$ berücksichtigt. Tv unterstützt nur einen einzigen Fit–Bereich, aber eine beliebige Anzahl von Untergrund–Bereichen. In Abbildung 4.4 sind gefittete Spektren dargestellt. In der obersten Scheibe wurden keine Fitparameter geändert und wie man sieht, paßt der Fit den Peak nicht besonders gut an.

Die Fit Marker können im Speicher gesichert und daraus zurückgeladen werden:

tv> **fit store**

tv> **fit restore**

Um neue Fit Parameter und Marker zu definieren, müssen die alten zuerst gelöscht werden. Am einfachsten geht das mit der Tastenkombination $\boxed{\text{-}}\boxed{\text{F}}$. Versehentlich gelöschte Marker können zurückgeholt werden mit:

tv> **fit recover–backup**

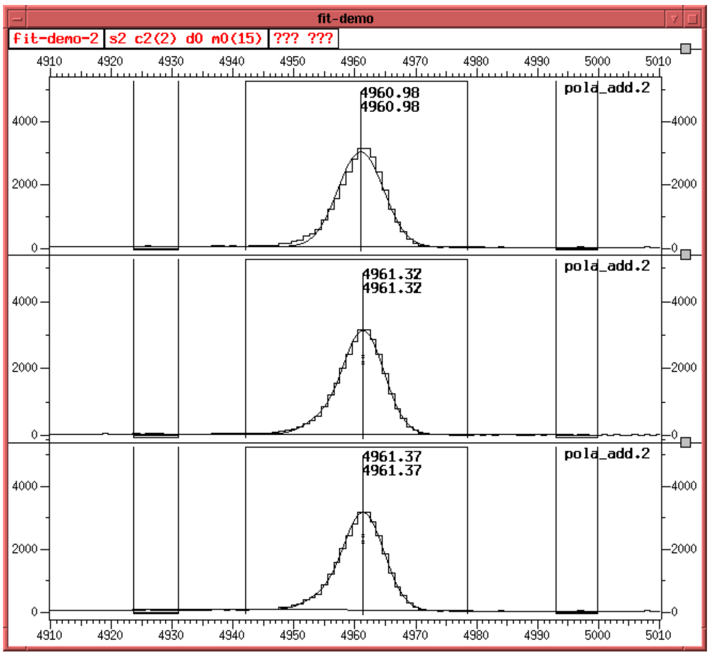

Abbildung 4.4: Graphik–Fenster mit einem geladenen und gefitteten ^{226}Ra Spektrum. **Oben:** Die Fit Parameter stehen auf ihren Standardwerten. **Mitte:** Der Parameter für den linken Tail wurde angepaßt. **Unten:** Die Parameter für den linken Tail und die Höhe der Stufe wurden angepßt.

Um den linken Tail zu berücksichtigen, muß man Tv erlauben ihn zu fitten. Das geschieht mit:

tv> **fit parameter tl free**

Das Ergebnis des Fits mit Berücksichtigung des Tails ist in der mittleren Scheibe von Abbildung 4.4 dargestellt und der Unterschied ist offensichtlich. Weiterhin kann man die Form des Untergrundes berücksichtigen. In der Elektronik kann es während der Datenaufnahme zu einem Energieverlust kommen, wodurch einige Ereignisse zu niedrigeren Energien hin verschoben werden und eine Stufe im Untergrund auftritt. Man kann sowohl Höhe wie auch Breite dieser Stufen berücksichtigen. Um die Höhe der Stufe zu fitten, verwendet man das Kommando:

`tv>` **fit parameter sh free**

In der unteren Scheibe in Abbildung 4.4 sieht man den Unterschied in der Form des Untergrunds. Die Stufe wird mir einem arctan angepaßt, aber man kann auch eine erf–Funktion verwenden (siehe Abschnitt D.1). Mit folgendem Kommando schaltet man dahin um:

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

Eine Liste aller Kommandos des Menus `fit parameter` ist in Abschnitt 6.4.12 aufgeführt. Tabelle 4.6 enthält alle Hotkeys, die etwas mit dem Fit Kommando zu tun haben.

4.12.4 Dekomposition

Ein Spektrum besteht nicht nur aus deutlich getrennten Peaks, sondern öfters überlappen Peaks zu sogenannten Multipletts. Für die Dekomposition (oder Trennung) von Multipletts muß ein Fit verwendet werden und damit ist eine Beschreibung der Peakform notwendig. Anderenfalls ist es nicht möglich, die korrekte Anzahl von Peaks zu finden und eine gute Beschreibung des Spektrums zu erhalten.

Als Beispiel, was passieren kann wenn man ein Multiplett falsch fittet, zeigt Abbildung 4.5 in der oberen Scheibe einen Quickfit. Wie man sieht, kann dieser die zwei Peaks nicht trennen. In der mittleren Scheibe wurde ein normaler Fit durchgeführt, bei dem zwei Peakmarker gesetzt waren und in der unteren Scheibe wird die Dekomposition angezeigt. Die Dekomposition kann mit den folgenden Kommandos angezeigt oder versteckt werden:

`tv>` **window show fit decomposition**

`tv>` **window show fit decomposition**

oder den Hotkeys $\boxed{\text{m}}\boxed{\text{d}}$ oder $\boxed{\text{u}}\boxed{\text{d}}$.

Tv verwendet eine modifizierte Gaußfunktion mit linkem und rechtem Tail und einer Stufe. Parameter der Gaußfunktion sind Position, Volumen und Breite, sowie linker und rechter Tail und Höhe und Breite der Stufe (siehe Abschnitt D.1). Für jeden gefitteten Peak gibt es diese sieben Parameter. Um die Anzahl der freien Parameter einzuschränken, kann man einzelne mit einem festen Wert belegen oder Korrelationen zwischen ihnen herstellen, z.B. alle Breiten gleichsetzen. Die Parameter für die Stufe und den Tail können meistens vernachlässigt werden, die für den Untergrund können gleichzeitig mit den Peaks gefittet werden, oder im Voraus in separaten Untergrund–Bereichen. Zur Wahl der Peak– und Untergrund–Funktion oder ob bestimmte Parameter angepaßt werden sollen oder nicht, werden die folgenden Kommandos verwendet (siehe Abschnitt 6.4.7 für die Kommandos des Menus `fit function` und Abschnitt 6.4.12 für Befehle des Menus `fit parameter`).

Um eine bestimmte Peak–Funktion zu aktivieren (standardgemäß ist continuous–exp–tail/arctan–step aktiv), verwendet man:

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

Man kann einen Parameter immer mit einem Wert belegen und diesen festhalten, d.h. Tv verbieten ihn anzupassen, oder den Parameter frei setzen und ihn von Tv fitten lassen.

Mit dem folgenden Befehl setzt man den Grad des Untergrundpolynoms (Standardwert ist 2) und hält diesen fest:

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

`tv>` **fit parameter background hold**

Tastenkombination	Tv–Kommandos	Funktion
b	tv> fit mark bg-region enter cursor;	Setze einen Untergrund–Region Marker an der Cursor Position
p	tv> fit mark peak enter cursor;	Setze einen Peak Marker an der Cursor Position
r	tv> fit mark region enter cursor;	Setze einen Fit–Region Marker an der Cursor Position
- b	tv> fit mark bg-region delete cursor;	Lösche den Untergrund Marker, der dem Cursor am nächsten ist
- p	tv> fit mark peak delete cursor;	Lösche den Peak Marker, der dem Cursor am nächsten ist
- r	tv> fit mark region delete cursor;	Lösche den Fit–Region Marker, der dem Cursoram nächsten ist
- 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;	Lösche alle Marker. Es werden immer noch Marker dargestellt, die Teil des autonomen Fits sind
- B	tv> fit mark bg-region delete all;	Lösche alle Untergrund Marker
- F	tv> fit mark fit delete; tv> fit delete; tv> fit delete activ; tv> window hide mark fit;	Lösche den aktuellen Fit und alle seine Marker
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;	Erzeuge einen Quickfit.
- P	tv> fit mark peak delete 0 uncalib 8192 uncalib;	Lösche alle Peak Marker
H P	tv> fit parameter peak hold;	Die Peak Position wird nicht gefitted
H W	tv> fit parameter width hold;	Die Peak Breite wird nicht gefitted
P b h	tv> fit param backgr hold; tv> fit param exponent hold; tv> fit param factor hold;	Die Untergrund–Parameter werden nicht gefitted
P b f	tv> fit param backgr free; tv> fit param exponent free; tv> fit param factor free;	Die Untergrund–Parameter werden gefitted
P w e	tv> fit parameter width equal;	Alle Breiten sind gleich
w e	tv> fit parameter width equal;	Alle Breiten sind gleich
w f	tv> fit parameter width free;	Die Breite wird gefitted

Tabelle 4.6: Hotkey Definitionen für Kommandos, die etwas mit dem Fit zu tun haben.

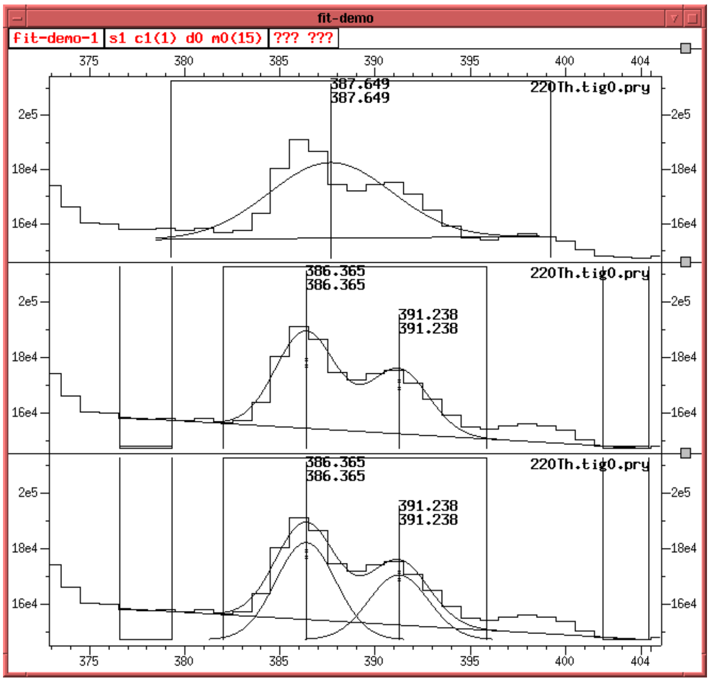

Abbildung 4.5: Graphik Fenster mit einem geladenen und gefitteten ^{220}Th Spektrum. **Oben:** Ein Quickfit wurde verwendet. Die Peaks wurden nicht getrennt. **Mitte:** Ein normaler Fit wurde durchgeführt, wobei zwei Peaks markiert waren. **Unten:** Die Fit Funktion und die Dekomposition sind abgebildet.

und aktiviert den Exponentialterm, indem man den Parameter FAC (siehe Abschnitt D.2) auf einen Wert ungleich 0 setzt:

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

`tv>` **fit parameter factor–background hold**

Die Skalierung des Exponentialterms wird folgendermaßen gesetzt:

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

`tv>` **fit parameter exponent–background hold**

Alle Parameter, die mit Kommandos aus dem Menu `fit parameter` beeinflußt werden können, sind in Tabelle 4.7 zusammengestellt.

Parameter	Bedeutung
background	Grad der Untergrund–Funktion.
exponent–background	Skalierung des Exponentialterms (siehe Abschnitt D.2.
factor–background	Faktor des Exponentialterms (siehe Abschnitt D.2).
position	Position.
sh	Höhe der Stufe.
sw	Breite der Stufe.
tl	Linker Tail.
tr	Rechter Tail.
volume	Volumen.
width	Breite.

Tabelle 4.7: Fit Parameter und ihre Bedeutung.

Es ist wichtig, die optimale Funktion zu wählen, um die Parameter zu optimieren. Die richtig Fit–Funktion gibt den Erwartungswert für den Inhalt jedes gefitteten Kanals. Da die gemessenen Werte einer Wahrscheinlichkeitsverteilung folgen, ist ihre Wahrscheinlichkeit, verglichen mit der erwarteten Funktion, für jeden Kanal bekannt. Für einen bestimmten Satz von Parametern kann eine Gesamtwahrscheinlichkeit, als Produkt der Wahrscheinlichkeiten der einzelnen Kanäle, ausgerechnet werden. Für den optimalen Satz von Parametern sollte sich die größte Wahrscheinlichkeit ergeben. Dieser Satz wird bestimmt, indem man das Maximum des Produktes, in Abhängigkeit von den Parametern, sucht (Maximum Likelihood).

Standardmäßig wird angenommen, daß die Kanalinhalte einer Gaußverteilung folgen. In diesem Fall erhält man die größte Wahrscheinlichkeit durch χ^2–Minimierung (siehe Abschnitt D.3). Unglücklicherweise ist das nicht anwendbar, wenn die Erwartungswerte für die Inhalte einzelner Kanäle in der Größenordnung von eins liegen. In dem Fall werden die Werte durch eine Poissonverteilung beschrieben. Die Fit–Funktion schätzt offensichtlich die Inhalte des Spektrums falsch ab. Die Fehlabschätzung hängt nicht vom Integral der gefitteten Daten ab, sondern von den Amplituden, d.h. die Größe des Fit–Bereichs spielt keine Rolle.

Neben der Minimierung des χ^2 kann Tv auch die Poissonverteilung maximieren (siehe Abschnitt D.3). Die Resultate, die man durch diese Methode erhält, sind so lange korrekt, wie die Spektren nur inkrementiert oder addiert wurden. Für normierte und subtrahierte Spektren ist die Varianz der Daten, die der Poissonverteilung folgen, nicht definiert und daher ist die Methode für diese Spektren nicht anwendbar. Auf Daten, die gemäß der Gaußverteilung verteilt sind, ist diese Methode hingegen anwendbar. Als Schluß aus dem gerade Gesagten kann man ziehen, daß keine der Verteilungsfunktionen korrekte Ergebnisse liefert, falls Spektren mit geringer Statistik subtrahiert wurden.

Zwischen den Meßfunktionen wird folgendermaßen umgeschaltet:

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

4.12.5 Festhalten von Parametern

Manchmal ist es nützlich Parameter nicht zu fitten, sondern konstant zu halten oder mit anderen Parametern zu korrelieren. Das kann z.B. der Fall sein, wenn eine Energie in einem Doppelpeak bekannt ist und die Position der anderen gefittet werden soll, oder um die Breiten mehrerer Peaks gleich zu machen. Dazu werden zunächst alle Schritte durchgeführt als gäbe es mehrere getrennte Peaks. Das heißt, es wird ein Fitbereich markiert, sowie Hintergrund- und Peakmarker. Anschließend wird der Fit ausgeführt.

Der Kurzstatus des Fits (`tv>` **fit status short**) sieht für zwei Peaks z.B. folgendermaßen aus:

```
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)
```

Daraus kann man ablesen, daß die Positionen nicht festgehalten werden und die Breiten korreliert sind, genaugenommen die Breite des Peaks 1 gleich der Breite des Peaks 0 gesetzt wird. Ist die Position des Peak 0 bekannt (z.B. 717.3) muß diese gesetzt und festgehalten und die zweite Position gefittet werden. Die folgenden Befehle setzen und halten die Peakposition:

`tv>` **fit parameter position0 = 717.3**

`tv>` **fit parameter position0 hold**

[2] Danke Constantin Chitu (Ayak).

Die entsprechenden Hotkeykombinationen sind P p = 0 bzw. P p h 0. Der Kurzstatus nach dem Fit sieht dann folgendermaßen aus:

```
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)
```

Im obigen Beispiel sind die Breiten der Peaks korreliert. D.h., die Breite des Peaks 1 wird gleich der Breite des Peaks 0 gesetzt. Korrelationen zwischen Parametern stellt man mit dem `equal` Kommando her. Der Befehl zum Gleichsetzen der beiden Breiten lautet voll ausgeschrieben:

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

Sollen alle Breiten individuell gefittet werden, wird der Befehl

`tv>` **fit parameter width free**

verwendet, für den es die Hotkeykombination w f gibt.

Alle Hotkeys zum Festhalten, Loslassen und Ändern von Parametern sind in der Tabelle 4.8 zusammengestellt.

4.12.6 Speichern und Laden von Fits

Auch die Ergebnisse eines Fits und einer Integration können in eine Datei gespeichert werden. Wie bei der Peaksuche kann auch hier zwischen der Binärform und Klartext gewählt werden. Das Laden kann wieder nur aus der Binärdatei durchgeführt werden.
Zum Speichern in die Binärdatei <Spektrumname>.fit wird der Befehl

`tv>` **fit write**

verwendet (Hotkey S f). Zum Lesen von Daten aus dieser Datei stehen verschiedene Befehle zur Verfügung:

`tv>` **fit read <u>bin</u>** {**<u>position</u>** | **first** | **last** | **next** | **previous** | **<u>index</u>**}
`tv>` **fit read fit** {**first** | **last** | **next** | **previous**}
`tv>` **fit read integration** {**first** | **last** | **next** | **previous**}
`tv>` **fit read record** {**first** | **last** | **next** | **previous**}

Mit dem ersten Befehl wird die Datei <Spektrumname>.fit nach gespeicherten Fits durchsucht und diese geladen. Dabei kann ein Fit für eine bestimmte **Position**, die über den Cursor oder eine Kanalzahl angegeben wird, geladen werden. Für das erste Laden wird in der Regel die Option **first** (Hotkey R f f) verwendet. Die Optionen **first** bis **last** laden die einzelnen Einträge aus der Datei "energetisch" sortiert. Weiterhin besteht die Möglichkeit, den Fit Nummer <**index**> zu laden, wenn man weiß, wo der zu ladende Fit in der Datei steht. Neben den Markern und Parametern für den gewünschten Fit werden auch die Positionen aller weiteren gespeicherten Fits gelesen und im Spektrum als sogenannte **bins** angezeigt. Mit Hilfe dieser bins können weitere Fits über das Kommando

`tv>` **fit read bin position cursor**

sehr einfach geladen werden. Für diese Operation steht auch der Hotkey R b c zur Verfügung.
Der zweite Befehl dient auch zum Laden von Fits, allerdings besteht hier nicht die einfache Möglichkeit mit den bins.
Mit dem dritten Befehl können Integrationen geladen werden.
Zum Laden von Fits und Integrationen dient der letzte Befehl. Mit diesem werden die einzelnen Einträge in der Reihenfolge gelesen, wie sie in der Datei abgespeichert sind.

P p h 0	tv> fit para position0 hold;	Hält die Position des Peaks 0 fest. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P p f 0	tv> fit para position0 free;	Die Position des Peaks 0 wird nicht mehr festgehalten. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P p n 0	tv> fit para position0 none;	Der Peak 0 wird aus der Peakliste entfernt. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P p = 0	tv> edit; tv> fit para position0 = ?	Die Position des Peaks 0 muß noch eingegeben werden. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P b h	tv> fit param backgr hold; tv> fit param exponent hold; tv> fit param factor hold;	Die Untergrund–Parameter werden nicht gefittet
P b f	tv> fit param backgr free; tv> fit param exponent free; tv> fit param factor free;	Die Untergrund–Parameter werden gefittet
P w e	tv> fit parameter width equal;	Die Breiten aller Peaks werden gleichgesetzt
H P	tv> fit parameter position hold;	Die Peak Position wird nicht gefittet
H W	tv> fit parameter width hold;	Die Peak Breite wird nicht gefittet
w e	tv> fit parameter width equal;	Die Breiten aller Peaks werden gleichgesetzt
w f	tv> fit parameter width free;	Die Breiten aller Peaks werden gefittet

Tabelle 4.8: Hotkeys zum Festhalten und Loslassen von Parametern.

Alle Hotkeys zum Speichern und Laden von Fits sind in der Tabelle 4.9 zusammengestellt.

R b c	tv> fit read bin position cursor;	Liest den Fit an der Cursorposition.
R b f	tv> fit read bin first;	Liest den ersten Fit aus der Fitdatei.
R b n	tv> fit read bin next;	Liest den nächsten Fit aus der Fitdatei.
R b p	tv> fit read bin position cursor;	Liest den Fit an der Cursorposition.
R f f	tv> fit read fit first;	Liest den ersten Fit aus der Fitdatei.
R f n	tv> fit read fit next;	Liest den ersten Fit aus der Fitdatei.
R i f	tv> fit read integration first;	Liest die erste Integration aus der Fitdatei.
R i n	tv> fit read integration next;	Liest die erste Integration aus der Fitdatei.
S f	tv> fit write;	Schreibt einen Fit oder eine Integration in die Fitdatei.

Tabelle 4.9: Hotkeys zum Speichern und Laden von Fits.

Um die Ergebnisse eines Fits oder einer Integration im Klartext in einer Protokolldatei zu speichern, muß diese zuerst geöffnet werden. Das geschieht mit einem der Kommandos:

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

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

und die Ergebnisse werden mit dem Befehl:

`tv>` **fit print**

gespeichert. Nachdem alle Resultate geschrieben wurden, muß die Datei mit:

`tv>` **fit result–file close**

geschlossen werden. Beim Verlassen von Tv wird die Datei automatisch geschlossen.

Kapitel 5

Analyse von Matrizen

Inhaltsangabe

5.1 Einführung

Die Analyse von Koinzidenzereignissen ist in der γ –Spektroskopie unerläßlich, da in der Regel nur mit dieser Methode neue physikalische Ergebnisse erzielt werden können. Tv trägt dieser Tatsache durch eine komfortable Matrixanalyse besonders Rechnung. In diesem Kapitel werden alle notwendigen Operationen erklärt, um eine Matrix zu öffnen und die Cutbuffer, Projektionen, Spektren und Verzeichnisse zuzuordnen. Weiterhin wird dargestellt, wie in einer einzelnen Matrix geschnitten wird oder mehreren gleichzeitig.

5.2 Setup

Um ein Schnittspektrum (Cut) zu erzeugen, benötigt man eine Matrix, ihre x– und y–Projektion, einen Satz Marker und einen Buffer für das Schnittspektrum.

Tv kann eine beliebige Anzahl Matrizen, Projektionen, Cuts und Spektren verwalten. Standardmäßig stellt Tv 16 Buffer (siehe Abschnitt 3.2) zum Abspeichern von Matrizen, Schnitten, Spektren und Verzeichnissen zur Verfügung. Ihre Namen setzen sich aus dem ersten Buchstaben des Buffertyps und der Buffernummer zusammen, z.B. c0,. . . ,c15 für die Cutbuffer.

5.2.1 Zuordnungen

Tv benötigt Beziehungen zwischen Matrix, Spektrum, Cut und Verzeichnis Buffern, die als **Zuordnungen (attachments)** bezeichnet werden.

Zum Beispiel erzeugt man durch das Setzen sogenannter Gates in der Matrix einen Schnitt und das Schnittspektrum wird in den zugeordneten Spektrumbuffer gespeichert. Beim Abspeichern des Cuts wird das Schnittspektrum in das zugeordnete Verzeichnis geschrieben. Standardmäßig macht Tv die in Tabelle 5.1 aufgeführten Zuordnungen.

c<n>	m0	Matrix m0 wird allen Cuts zugeordnet.
c<n>	s<n>	Spectrum s<n> wird Cut <n> zugeordnet.
c<n>	d0	Directory d0 wird allen Cuts zugeordnet.
s15	m0	Spectrumbuffer 15 enthält die Projection für Matrix 0.

Tabelle 5.1: Tv's Standardzuordnungen.

5.2.2 Ändern der Zuordnungen

Alle Änderungen werden für den aktiven Cut durchgeführt, der standardmäßig c0 ist. Einen Cut aktiviert man mit dem Kommando:

`tv>` **cut activate <index>**

Die Zuordnungen werden geändert mit:

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

Die einer Matrix zugeordnete Projektion ändert man mit

`tv>` **cut matrix attach–projection <s>**

wobei <s> der Spektrumbuffer ist der die Projektion enthält. Das wird benötigt, wenn man in mehreren Matrizen gleichzeitig schneiden möchte (siehe Abschnitt 3.10).

5.2.3 Öffnen einer Matrix

In den vorhergehenden Abschnitten wurden die Zuordnungen zwischen allen notwendigen Buffern definiert. Sie können mit den zugehörigen Daten in der folgenden Weise gefüllt werden. Um eine Matrix zu öffnen, gibt man ihren Index und den Dateinamen ein:

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

Die Projektion <filename> wird in Buffer <n> geladen mit:

`tv>` **spectrum read <filename> <n>**

Definiere den Pfadnamen des Schnittverzeichnisses mit:

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

5.2.4 Beispiel Setup

Sei z.B. Cut 0 aktiviert und Zuordnungen zu folgenden Buffern sollen gemacht werden: Matrix 1, Projektion 2, Spektrum 3 und Verzeichnis 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**

Dann wechselt man in ein passendes Verzeichnis und öffnet die Matrix sowie das Verzeichnis für die Schnitte und lädt die Projektion.

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**
Beispiele für den Vergleich von Matrizen, wo mehrere Umgebungen geöffnet werden, sind in Abschnitt 5.6 gegeben.

5.3 Schneller Setup

Da die Dateistruktur für die Matrixanalyse immer die gleiche ist, bietet Tv eine Möglichkeit den Setup viel leichter und schneller durchzuführen. Sie wird **Schnittumgebung** (cut environment) genannt und verwendet Wildcards, wie in Abschnitt 3.12.2 beschrieben, d.h. die Dateien müssen sich an eine Namenskonvention halten, um mit dem **cut environment** Kommando geladen zu werden.

5.3.1 Definition einer Umgebung

Das **cut environment** definiert eine Dateistruktur, die die Analyse von Matrizen vereinfacht, d.h. man muß nur einen Pfadnamen angeben, um auf eine Matrix zuzugreifen, ihre Projektion zu laden und das Verzeichnis für die Schnitte zu öffnen.

Um zum Beispiel die Umgebung (Environment) aus dem Verzeichnis */matrix1/220Th* zu laden, muß es die Dateien enthalten, die in Tabelle 5.2 angegeben sind.

220Th.mtx	Die Matrix.
220Th.pr[xy]	Die Projektionen.

Tabelle 5.2: Dateien, die für das **cut environment** Kommando existieren müssen.

5.3.2 Laden einer Umgebung

Falls alle Dateien existieren, die im vorigen Abschnitt genannt wurden, kann die Umgebung geladen werden, mit der gearbeitet werden soll. Entweder geschieht das durch Drücken des Hotkey $\boxed{\text{E}}$ und der anschließenden Eingabe des Verzeichnisses oder mit dem Kommando:
tv> **cut environment /matrix1/220Th**
wobei */matrix1/220Th* das Verzeichnis ist, in dem sich die Matrix und die Cuts befinden. Tv lädt dann die Umgebung in die Buffer, die standardmäßig auf den Werten stehen, die in Tabelle 5.3 definiert sind. Mit den Befehlen aus Abschnitt 5.2 können andere Buffer für die Umgebung definiert werden und damit Schnitte in mehreren Matrizen gleichzeitig durchgeführt werden.

Buffertyp	Standardwert
Cut	0
Matrix	0
Projektion	15
Spektrum	0

Tabelle 5.3: Standardwerte der Buffer für das **cut environment** Kommando.

Die Kommandos, die durch das Kommando cut environment ausgeführt werden, sind:
tv> **cut matrix open /matrix1/220Th/220Th.mtx**

tv> **spectrum read /matrix1/220Th/220Th.pry**
tv> **cut directory open /matrix1/220Th/220Th.cutdir**
Das Verzeichnis *220Th.cutdir* wird nach gespeicherten Schnitten durchsucht. Falls bereits Schnitte gespeichert wurden, werden diese durch Label in der Projektion angezeigt.

5.4 Erzeugen eines Cuts

Um einen Cut zu erzeugen, muß man einen Satz von Gates definieren. Das kann mit dem Hotkey c geschehen, oder dem Kommando:
tv> **cut marker cut enter cursor**
Die ersten beiden Marker definieren das positive Gate, alle weiteren definieren Untergrundgates.
Eine andere Methode ist, die Marker über ihre Kanalnummer oder Energie zu definieren. Zwischen Kanalnummer und Energie kann man mit folgenden Kommandos hin– und herschalten:
tv> **cut scale marker default**
tv> **cut scale marker calibrated**
tv> **cut scale marker uncalibrated**
und ein Marker wird gesetzt mit:
tv> **cut marker cut enter value**
Da man mit dem Hotkey c nur ein positives Gate setzen kann, bietet Tv eine Möglichkeit, weitere positive Gates mit der Tastenkombination G G oder dem folgenden Kommando zu definieren:
tv> **cut marker gate enter cursor**
Eine weitere Möglichkeit Untergrund Gates zu definieren ist das Kommando:
tv> **cut marker bg–gate enter value**
Um die Erzeugung des Cuts durchzuführen, drückt man den Hotkey C oder gibt das folgenden Kommando ein:
tv> **cut create cut**
Die Untergrund Gates werden gegen die positiven Gates gewichtet. Je mehr Untergrund markiert wird, desto besser, da dadurch der statistische Fehler reduziert wird. Der Untergrund wird über die Gatebreite gewichtet, kann aber alternativ auch über das Volumen gewichtet werden. Die Kommandos, die diesen Parameter ändern, sind:
tv> **cut weight gatewidth**
tv> **cut weight fit**
Es wird offensichtlich, wie wichtig die Wahl richtiger Untergrund Gates ist, wenn man sich die Abbildungen 5.1, 5.2 und 5.3 ansieht. Positive Gates sind durch einen horizontalen Strich oben gekennzeichnet und Untergrundgates durch einen horizontalen Strich unten. Wenn zuviel Untergrund abgezogen wird, wie im letzten Beispiel, dann sind im Schnittspektrum negative Zählraten zu erkennen.
Falls im aktiven Cut die Schnittmarker aus einem anderen Schnitt verwendet werden sollen, kann das mit folgendem Kommando initiiert werden:
tv> **cut use–marker <index>**
wodurch die Gates aus dem Cut <index> in den aktiven Cut kopiert werden.

Wenn ein guter Satz Gates gefunden wurde, die man sich merken möchte während man nach einem besseren Satz sucht, kann man sie mit den **store** Kommandos speichern:
tv> **cut marker gate store**
tv> **cut marker bg–gate store**
und zurückladen mit den entsprechenden **restore** Kommandos.

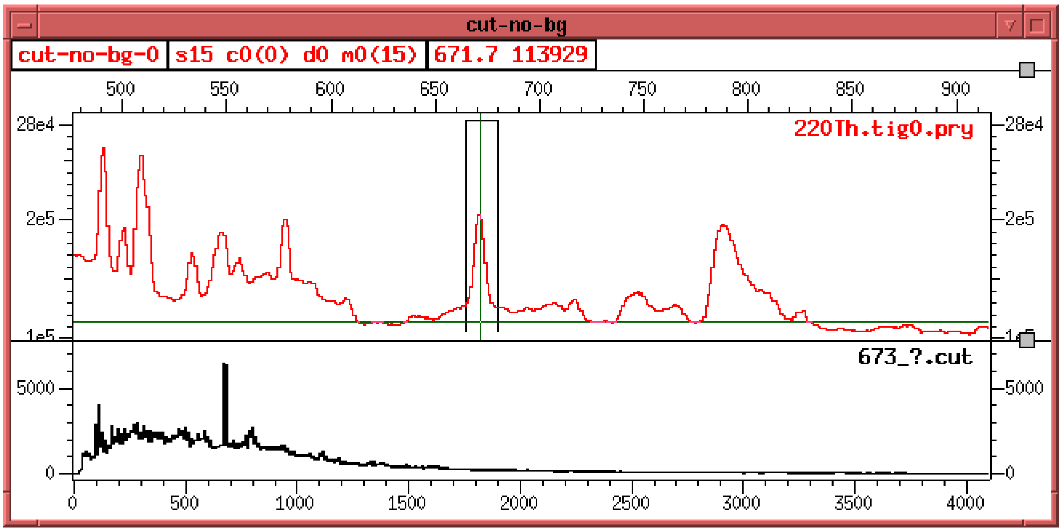

Abbildung 5.1: Schnitt ohne Untergrund Gates.

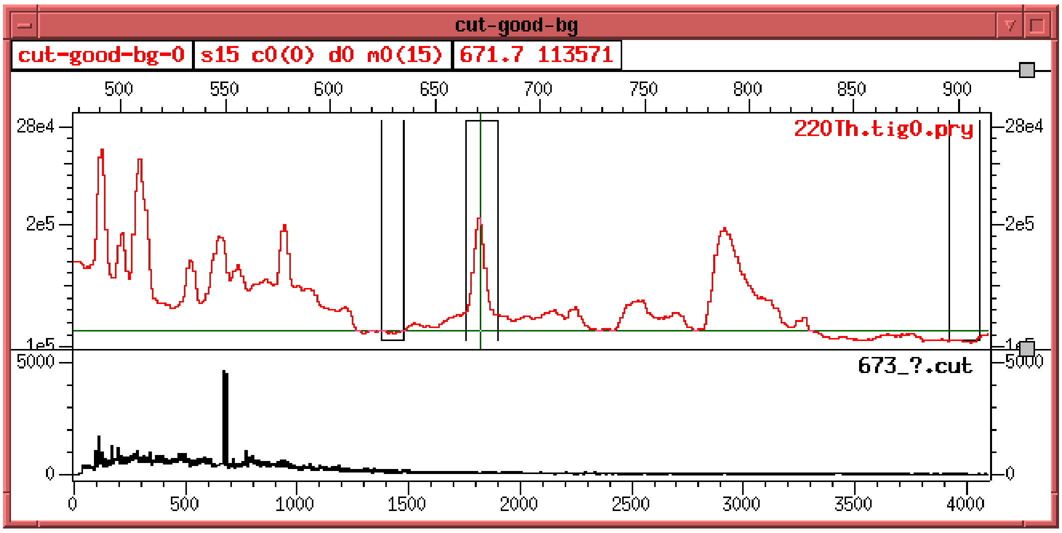

Abbildung 5.2: Schnitt mit guten Untergrund Gates.

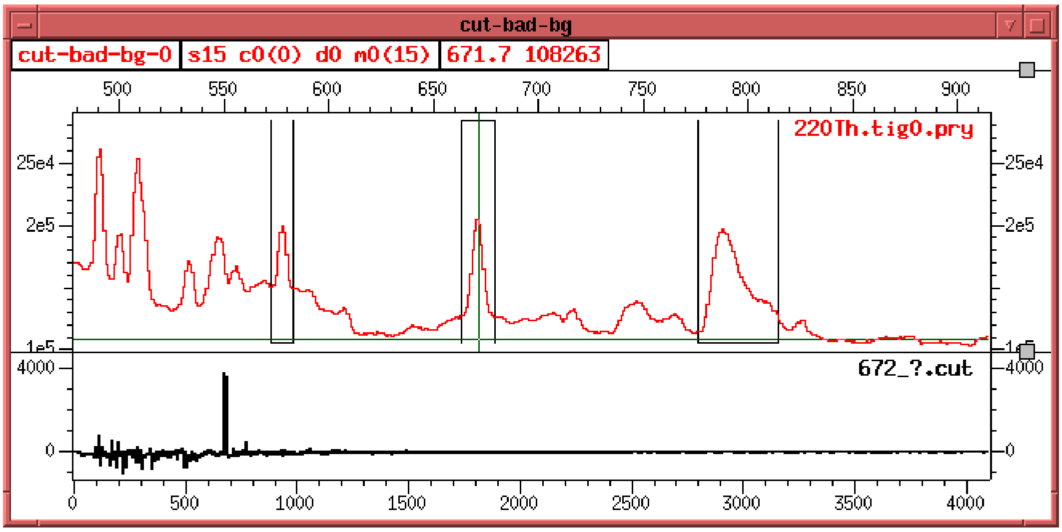

Abbildung 5.3: Schnitt mit schlechten Untergrund Gates.

Um einen neuen Cut durchzuführen, müssen zuerst die Gates des vorhergehenden Cuts gelöscht werden. Auch hier muß das getrennt für die positiven und Untergrund Gates geschehen. Beide Arten gleichzeitig können mit der Tastenkombination [-][C] gelöscht werden.

5.5 Speichern und Laden von Cuts

Cuts können abgespeichert und wieder geladen werden. Die Cuts werden im Schnittverzeichnis abgespeichert wo für jeden Cut ein Verzeichnis angelegt wird. Als Name wird die Energie oder der Kanal genommen, wo der Schnitt durchgeführt wird. Daran wird mit einem Unterstrich ein Index gehängt, damit an einer Position mehrere verschieden Cuts abgespeichert werden können. Wenn zum Beispiel ein Schnitt in der ^{220}Th Matrix im Kanal 1105 gemacht wurde, wird dieser mit dem Befehl

`tv>` **cut write cut**

oder dem Hotkey [S][c] im Verzeichnis *220Th/220Th.cutdir/1105_0* abgespeichert. In diesem Verzeichnis stehen die Marker in der Datei *1105_0.gate* und das Schnittspektrum in der Datei *1105_0.cut*.

Zum Laden stehen verschiedene Befehle zur Verfügung. Einerseits können nur Marker geladen werden und andererseits Marker und Schnittspektrum gleichzeitig. Im ersten Fall wird mit den geladenen Markern ein Schnitt erzeugt.

Die Befehle zum Laden der Schnittmarker bzw. Schnittmarker und Schnittspektrum für die Position an der sich der Cursor befindet, lauten

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

`tv>` **tv> cut read cut cursor**

Alle Hotkeys für das Speichern und Laden von Schnittmarkern und Schnittspektren sind in Tabelle 5.4 zusammengestellt.

5.6 Vergleich von Matrizen

Tv bietet die in Tabelle 5.5 aufgeführten Skripten zum Vergleich von Spektren, Matrizen oder Schnitten. Jedes dieser Skripte wird in den folgenden Abschnitten

R c c	tv> cut read cut cursor;	Liest das Schnittspektrum und den Satz Cut–Marker an der Cursorposition.
R c f	tv> cut read cut first;	Liest das erste Schnittspektrum und den ersten Satz Cut–Marker.
R c n	tv> cut read cut next;	Liest das nächste Schnittspektrum und den nächsten Satz Cut–Marker.
R h c	tv> cut read head cursor;	Liest den Satz Cut–Marker an der Cursorposition und führt den Cut aus.
R h f	tv> cut read head first;	Liest den ersten Satz Cut–Marker und führt den Cut aus.
R h n	tv> cut read head next;	Liest den nächsten Satz Cut–Marker und führt den Cut aus.
S c	tv> cut write cut;	Schreibt das Schnittspektrum und die Cut–Marker in ein Verzeichnis.

Tabelle 5.4: Hotkeys zum Speichern und Laden von Schnitten.

beschrieben. Um z.B. zwei Spektren zu vergleichen, gibt man folgendes Kommando ein:

`tv>` **@cmp2spc**

Diese Dateien sind normalerweise im Verzeichnis */ikp/local/lib/tv* gespeichert und auf manchen Systemen im Verzeichnis */usr/local/lib/tv*. Um das tatsächliche Verzeichnis zu finden, dient das Kommando:

`tv>` **which cmp2spc**

cmp2cut	→	Vergleiche zwei Schnitte.
cmp2mat	→	Vergleiche zwei Matrizen.
cmp2spc	→	Vergleiche zwei Spektren.
cmp3mat	→	Vergleiche drei Matrizen.
cmp4mat	→	Vergleiche vier Matrizen.

Tabelle 5.5: Beispiel Kommandodateien.

5.6.1 cmp2spc

cmp2spc definiert Zuordnungen für die Cuts 0 und 1. Matrix und Verzeichnis 0 sowie die Projektion 5 und die Spektren 0 und 1 werden dort zugeordnet. Ein Fenster vom Typ simple mit dem Namen **normalized spectra comparison** wird erzeugt. Ebenfalls werden die folgenden Hotkeys erzeugt (siehe Tabelle 5.6).

5.6.2 cmp2mat

cmp2mat definiert Zuordnungen für die Cuts 0 und 1. Matrizen, Verzeichnisse und Projektionen 0 und 1 werden zugeordnet, sowie die Spektren 4 und 5. Zwei Fenster vom Typ simple mit den Namen **projections**, das die Spektren 0 und 1 zeigt, und **comparison**, das die Spektren 4 und 5 zeigt, werden erzeugt. Weiterhin

Hotkey	Tv–Kommandos	Funktion
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;	

Tabelle 5.6: Hotkeys, die von **cmp2spc** definiert werden.

wird ein Hotkey definiert (siehe Tabelle 5.7), der die Cuts für die beiden Matrizen erzeugt.

Hotkey	Tv–Kommandos	Funktion
C	cut activate 0; cut create cut; cut activate 1; cut create cut;	

Tabelle 5.7: Hotkey, der von **cmp2mat** definiert wird.

Zum Schluß öffnet cmp2mat die Umgebungen für die beiden Matrizen, die als Kommandozeilenargumente übergeben werden. Zum Beispiel:

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

Die Skripte **cmp3mat** und **cmp4mat** erfüllen die gleichen Aufgaben, nur für mehr Matrizen, weswegen sie auch mit mehr Argumenten aufgerufen werden.

5.6.3 cmp2cut

cmp2cut erzeugt Zuordnungen für die Cuts 0 und 1. Matrix und Projektion 0 sowie die Projektion 5 und die Spektren 0 und 1 werden zugeordnet. Die Fenster vom Typ cut mit den Namen CUT0 für Spektrum 0 und CUT1 für Spektrum 1 werden erzeugt.

Hotkey	Tv–Kommandos	Funktion
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	

Tabelle 5.8: Hotkeys, die von **cmp2cut** definiert werden.

Kapitel 6

Liste aller Kommandos

Inhaltsangabe

6.1 Einführung

In diesem Kapitel sind alle Kommandos in alphabetischer Reihenfolge aufgeführt, wobei zu jedem eine kurze Beschreibung seiner Funktion gegeben ist. Unterstrichene Worte sind die Standardargumente im entsprechenden Menu und müssen nicht eingegeben werden. Das Argument für das value Kommando kann eine Energie (calibrated) oder ein Kanal (uncalibrated) sein. Die momentane Interpretation des eingegebenen Wertes wird angegeben und kann durch Angabe des entsprechenden Arguments (cal oder uncal) nach dem Zahlenwert überschrieben werden.

6.2 Calibration

Tabelle 6.1 zeigt die im calibration Menu erlaubten Kommandos und ihre zugehörigen Argumente. In den Unterabschnitten werden die einzelnen Kommandos des calibration Menus im einzelnen beschrieben.

Kommando	Argumente
copy	<Quellindex> {{<index>...} | active | all | shown | visible}
delete	{<index>...} | active | all | shown | visible
enter	<Kanal$_0$ > <Energie$_0$ > <Kanal$_1$ > <Energie$_1$ >
list	{<index>...}
read	<Dateiname>
write	<Dateiname>

Tabelle 6.1: Kommandos des **calibration** Menus.

6.2.1 Calibration efficiency

tv> **calibration efficiency** {**copy** | **delete** | **list** | **read** | **write**}
Führt eine Operation im Zusammenhang mit der Efficiency Eichung durch.

6.2.2 Calibration lefttail

tv> **calibration lefttail** {**copy** | **delete** | **enter** | **list** | **read** | **write**}
Legt fest, wie der linke Tail die Eichung beeinflußt.

6.2.3 Calibration position

tv> **calibration position** {**copy** | **delete** | **enter** | **list** | **read** | **write**}
Führt eine Operation im Zusammenhang mit der Eichung der Position durch.

6.2.4 Calibration righttail

tv> **calibration righttail** {**copy** | **delete** | **enter** | **list** | **read** | **write**}
Legt fest, wie der rechte Tail die Eichung beeinflußt.

6.2.5 Calibration set

tv> **calibration set** {**copy** | **delete** | **list** | **read** | **write**}
Führt eine Operation im Zusammenhang mit allen Parametern der Eichung durch.

6.2.6 Calibration startchannel

tv> **calibration startchannel** {**copy** | **delete** | **enter**}
Setzt den Kanal, bei der die Eichung beginnen soll.

6.2.7 Calibration unit

tv> **calibration unit** {**copy** | **delete** | **enter** <**unit**>}
Setzt die Einheit der x–Achse, z.B. keV.

6.2.8 Calibration width

`tv>` **calibration width** {**copy** | **delete** | **enter** | **list** | **read** | **write**}
Führt eine Operation im Zusammenhang mit der Eichung der Breite durch.

6.3 Cut

6.3.1 Cut activate

`tv>` **cut activate <index>**
Aktiviert den Cut <index> für die folgenden Cut Kommandos.

6.3.2 Cut attach

`tv>` **cut attach directory <index>**
Ordnet das Verzeichnis <index> dem aktiven Cut zu.

`tv>` **cut attach matrix <index>**
Ordnet die Matrix <index> dem aktiven Cut zu.

`tv>` **cut attach spectrum <index>**
Ordnet das Spektrum <index> dem aktiven Cut zu.

6.3.3 Cut create

`tv>` **cut create cut**
Erzeugt Head und Cutspektrum.

`tv>` **cut create head**
Erzeugt den Head, d.h. berechnet die Wichtung der Gates.

`tv>` **cut create spectrum**
Erzeugt ein Cutspektrum und kopiert es in den Buffer, der dem aktiven Cut zugeordnet ist. Vorher muß der Head erzeugt werden.

6.3.4 Cut directory

`tv>` **cut directory close <index>**
Schließt das Cutverzeichnis.

`tv>` **cut directory format <format>**
Definiert das Format, in dem die Spektren abgelegt werden sollen (siehe Kapitel C.2).

`tv>` **cut directory open <index> <Dateiname>**
Öffnet das Verzeichnis <Dateiname> für den Cut <index>.

`tv>` **cut directory status**
Gibt die Zuordnung zwischen Verzeichnissen und Buffern aus.

6.3.5 Cut environment

`tv>` **cut environment <pathname>**
Öffnet das Cut environment (siehe Kapitel 5.3).

6.3.6 Cut list

`tv>` **cut list all**
Zeigt alle Spektren im zugeordneten Verzeichnis an.

`tv>` **cut list range** {**cursor** | **<u>value</u>** | **offset**}
Setzt Marker für den Bereich, für den die abgespeicherten Cuts angezeigt werden sollen.

6.3.7 Cut marker

Kommando	Funktion
delete	Löscht Marker.
enter {cursor | <u>value</u> | offset}	Setzt neuen Marker.
list	Zeigt alle Marker an.
read <filename>	Liest eine Liste von Markern aus einer Datei.
restore	Lädt Marker aus dem Speicher zurück.
store	Speichert Marker im Speicher.
write <Dateiname>	Speichert eine Liste von Markern in der Datei.

Tabelle 6.2: Kommandos im **cut marker** Menu.

`tv>` **cut marker bg–gate** {**delete** | **enter** | **list** | **read** | **restore** | **store** | **write**}
Wendet eine Operation entsprechend der Tabelle 6.2 auf Untergrund–Gates an.

`tv>` **cut marker cut** {**delete** | **enter**}
Wendet eine Operation entsprechend der Tabelle 6.2 auf Cuts an.

`tv>` **cut marker gate** {**delete** | **enter** | **list** | **read** | **restore** | **store** | **write**}
Wendet eine Operation entsprechend der Tabelle 6.2 auf Gates an.

6.3.8 Cut matrix

`tv>` **cut matrix attach-projection** <**spectrum**>
Ordnet das Spektrum im Buffer <Spektrum> der aktiven Matrix als Projektion zu.

`tv>` **cut matrix close** <**index**>
Schließt die Matrix im Buffer <index>.

`tv>` **cut matrix format** <**format**>
Erwartet als Eingabe das Format der zu ladenden Matrix (siehe Kapitel C.2).

`tv>` **cut matrix open** <**index**> <**Dateiname**>
Öffnet die Matrix <Dateiname> im Buffer <index>.

`tv>` **cut matrix status**
Zeigt eine Liste aller geladenen Matrizen.

6.3.9 Cut read

`tv>` **cut read cut** {**cursor** | **<u>entry</u>** | **<u>position</u>** | **first** | **last** | **next** | **previous**}
Liest die Cut–Gate–Marker und das Cutspektrum entsprechend der Tabelle 6.3.

Kommando	Funktion
cursor	Liest den Cut für die Cursorposition.
entry	Erwartet als Eingabe einen Verzeichnisnamen.
position {cursor | value}	Liest den Cut für die Position.
first	Liest den ersten Cut im Cutverzeichnis.
last	Liest den letzten Cut im Cutverzeichnis.
next	Liest den nächsten Cut im Cutverzeichnis.
previous	Liest den vorigen Cut im Cutverzeichnis.

Tabelle 6.3: Kommandos in den **cut read** Menus.

`tv>` **cut read head** {**cursor** | **entry** | **position** | **first** | **last** | **next** | **previous**}
Liest die Cut–Gate–Marker entsprechend der Tabelle 6.3.

6.3.10 Cut rm

`tv>` **cut rm cut** <**Dateiname**>
Löscht den Cut <Dateiname> aus dem zugeordneten Cutverzeichnis.

`tv>` **cut rm fit** <**Dateiname**>
Löscht den Fit <Dateiname> aus dem zugeordneten Cutverzeichnis.

`tv>` **cut rm spectrum** <**Dateiname**>
Löscht das Spektrum <Dateiname> aus dem zugeordneten Cutverzeichnis.

6.3.11 Cut status

`tv>` **cut status**
Druckt Statusinformationen über alle Cuts und und ihre Zuordnungen.

6.3.12 Cut scale

`tv>` **cut scale i/o** {**calibrated** | **default** | **uncalibrated**}
Legt fest, ob die gelesenen bzw. geschriebenen Werte geeicht oder ungeeicht interpretiert werden.

`tv>` **cut scale marker** {**calibrated** | **default** | **uncalibrated**}
Legt fest, wie Marker bei ihrer Eingabe interpretiert werden.

6.3.13 Cut use–marker

`tv>` **cut use–marker** <**index**>
Die Cutmarker des Cuts <index> werden für den aktiven Cut verwendet.

6.3.14 Cut write

`tv>` **cut write cut**
Schreibt die Cut–Gate–Marker und das Cut-Spektrum für den aktiven Cut in das entsprechende Verzeichnis.

`tv>` **cut write head**
Schreibt die Cut–Gate–Marker für den aktiven Cut in das entsprechende Verzeichnis.

6.3.15 Cut weight

`tv>` **cut weight gatewidth**
Die Breite der Gates wird für die Berechnung der Wichtung verwendet.

`tv>` **cut weight fit**
Die Untergrundvolumina werden für die Berechnung der Wichtung verwendet.

6.4 Fit

6.4.1 Fit activate

`tv>` **fit activate** {**index <index>** | **next** | **previous** | **status**}
Aktiviert das Spektrum im Buffer <index>, im nächsten oder vorhergehenden Buffer oder gibt Statusinformationen über geladene Buffer, aktive Fits und Cuts aus.

6.4.2 Fit background–create

`tv>` **fit background–create**
Berechnet den Untergrundfit (siehe Kapitel D.2).

6.4.3 Fit bg–function

`tv>` **fit bg–function definition**
Druckt die Definition der Untergrundfunktion (siehe Kapitel D.2).

6.4.4 Fit region–create

`tv>` **fit region–create**
Berechnet einen Fit für den Bereich, der von den Bereichsmarkern begrenzt wird.

6.4.5 Fit integration–create

`tv>` **fit integration–create**
Integriert den Bereich, der von den Bereichsmarkern begrenzt wird.

6.4.6 Fit delete

`tv>` **fit delete**
Löscht den Fit und alle seine Marker.

6.4.7 Fit function

`tv>` **fit function background definition**
Druckt die Definition der Untergrundfunktion.

`tv>` **fit function peak**

`tv>` **fit function peak activate** {**continuous** | **additive**}
Erwartet die zu benutzende Fitfunktion als Eingabe (siehe D).

`tv>` **fit function peak definition** {**continuous** | **additive**}
Druckt die Definition der Fitfunktion.

tv> **fit function peak exponent** {**tl** | **tr**}
Erwartet den Exponenten für den Term des linken bzw. rechten Tails der Fitfunktion.

6.4.8 Fit list

tv> **fit list**
Gibt alle Fits und die sie betreffenden Informationen aus.

6.4.9 Fit marker

Kommando	Funktion
delete	Löscht Marker.
enter {cursor | value | offset}	Setzt einen neuen Marker.
list	Gibt eine Liste aller Marker aus.
read <filename>	Liest eine Liste von Markern aus einer Datei.
restore	Liest Marker aus dem Speicher zurück.
number <multiple | single>	
store	Schreibt Marker in den Speicher.
write <Dateiname>	Speichert eine Liste von Markern in einer Datei.

Tabelle 6.4: Kommandos im **fit marker** Menu.

tv> **fit marker bin** {**delete** | **enter** | **list** | **read** | **restore** | **store** | **write**}
Führt eine Operation mit den bin Markern durch.

tv> **fit marker bg–region** {**delete** | **enter** | **list** | **read** | **restore** | **store** | **write**}
Führt eine Operation mit den Untergrund Markern durch.

tv> **fit marker fit delete**
Löscht alle Marker für den aktiven Fit.

tv> **fit marker peak** {**delete** | **enter**}
Führt eine Operation mit den Peak Markern durch.

tv> **fit marker region** {**delete** | **enter** | **list** | **read** | **restore** | **number** | **store** | **write**}
Führt eine Operation mit den Fit Bereich Markern durch.

6.4.10 Fit measure

tv> **fit measure activate** {**dy–**χ^2 | **y–**χ^2 | **poisson**}
Aktiviert die ausgewählte Verteilungsfunktion (siehe Kapitel D.3).

tv> **fit measure definition** {**dy–**χ^2 | **y–**χ^2 | **poisson**}
Gibt die Definition der Verteilungsfunktion aus (siehe Kapitel D.3).

6.4.11 Fit open

tv> **fit open**
Öffnet die Datei <specfile>.fit und liest all Fits und Integrationen.

Kommando	Argument	Funktion
degree	<degree>	Setzt den Grad des Untergrundpolynoms (siehe Kapitel D.2). Der Standardwert ist 2.
number	{<value>}	Setzt den Faktor oder die Skalierung des exponentiellen Terms der Untergrundfunktion (siehe Kapitel D.2). Der Standardwert ist 0 für jede der beiden Größen.
free		Setzt diesen Parameter zum Fitten frei.
hold		Hält diesen Parameter fest.
=	<expression>	Setzt den Parameter auf den Wert <expression>.
none		Beim fit der Position wird der Peak aus der Fitliste entfernt, der Fit des Tails wird unterdrückt.
spaced	<parameter> <index>	?????
calibrated		?????
equal	<parameter> <index>	Korreliert den Wert des Parameters mit dem gleichen Parameter des Peaks <index>.

Tabelle 6.5: Kommandos im **fit parameter** Menu.

6.4.12 Fit parameter.

Siehe zu diesem Punkt auch Kapitel 4.12.5

`tv>` **fit parameter background.** {**degree** | **free** | **hold**}
Definiert, wie der Untergrund gefittet wird (siehe Tabelle 6.5).

`tv>` **fit parameter exponent–background.** {**number** | **free** | **hold**}
Definiert, wie der exponentielle Term der Untergrundfunktion gefittet wird.

`tv>` **fit parameter factor–background.** {**number** | **free** | **hold**}
Definiert, wie der Faktor in der Untergrundfunktion gefittet wird.

`tv>` **fit parameter position.** {**=** | **free** | **hold** | **none** | **spaced**}
Definiert, wie die Position gefittet wird.

`tv>` **fit parameter sh.** {**=** | **calibrated** | **free** | **hold** | **equal** | **none**}
Definiert, wie die Höhe der Stufe gefittet wird.

`tv>` **fit parameter sw.** {**=** | **calibrated** | **free** | **hold** | **equal** | **none**}
Definiert, wie die Breite der Stufe gefittet wird.

`tv>` **fit parameter tl.** {**=** | **calibrated** | **free** | **hold** | **equal** | **none**}
Definiert, wie der linke Tail gefittet wird.

`tv>` **fit parameter tr.** {**=** | **calibrated** | **free** | **hold** | **equal** | **none**}
Definiert, wie der rechte Tail gefittet wird.

`tv>` **fit parameter volume.** {**=** | **free** | **hold**}
Definiert, wie die Fläche gefittet wird.

`tv>` **fit parameter width.** {**=** | **calibrated** | **free** | **hold** | **equal**}
Definiert, wie die Breite gefittet wird.

6.4.13 Fit peaklist

`tv>` **fit peaklist list**
Gibt die Peakliste des aktiven Spektrums aus.

`tv>` **fit peaklist print**
Speichert die Peakliste in der Protokolldatei (siehe **fit result-file**).

`tv>` **fit peaklist read**
Liest die Peakliste aus der Fitdatei <Spektrumname>.fit.

6.4.14 Fit print

`tv>` **fit print**
Schreibt die Ergebnisse des Fits in die Protokolldatei (siehe **fit result-file**).

6.4.15 Fit psearch

`tv>` **fit psearch list**
Gibt die Position, den Fehler und das Volumen aller Peaks aus der letzten Peaksuche aus.

`tv>` **fit psearch create**
Führe eine Peaksuche durch, falls keine Marker gesetzt sind für das ganze Spektrum.

`tv>` **fit psearch marker** {**delete** | **enter** {**cursor** | **value** | **offset**} | **list** | **read** <**Dateiname**> | **restore** | **store** | **write** <**Dateiname**>}
Verwalte Marker für die Peaksuche.

`tv>` **fit psearch open**
Öffnet die Datei <Dateiname>.fit und liest die Liste der darin gespeicherten Fits und Peaks.

`tv>` **fit psearch print [<index>]**
Gibt die Anzahl der bei der letzten Peaksuche gefundenen Peaks für das Spektrum im Buffer <index>.

`tv>` **fit psearch probability [<limit>]**
Drucke oder setzte die Wahrscheinlichkeitsgrenze für die Peaksuche.

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

`tv>` **fit psearch scale [calibrated | default | uncalibrated]**
Gibt an oder legt fest, ob die Peaksuche auf der geeichten oder ungeeichten Skala durchgeführt wird.

`tv>` **fit psearch status**
Gibt die Anzahl der Peaks für alle Buffer aus, die bei der letzten Peaksuche gefunden wurden.

`tv>` **fit psearch write**
Speichert die Ergebnisse der letzten Peaksuche in die Datei <specname>.fit.

Kommando	Funktion
position {cursor | value | offset}	Liest den Fit für die Position.
first	Liest den ersten Fit.
last	Liest den letzten Fit.
next	Liest den nächsten Fit.
previous	Liest den vorigen Fit.
index	Liest den Fit Nummer <Index>.

Tabelle 6.6: Kommandos in den **fit read** Menus.

6.4.16 Fit read

tv> **fit read bin {position | first | last | next | previous | index}**
Liest einen Fit aus der Datei <Spektrumname>.fit gemäß Tabelle 6.6.

tv> **fit read fit {first | last | next | previous}**
Liest einen Fit aus der Datei <Spektrumname>.fit gemäß Tabelle 6.6.

tv> **fit read integration {first | last | next | previous}**
Liest eine Integration aus der Datei <Spektrumname>.fit gemäß Tabelle 6.6.

tv> **fit read record {first | last | next | previous | index}**
Liest einen Eintrag aus der Datei <Spektrumname>.fit gemäß Tabelle 6.6.

6.4.17 Fit recover–backup

tv> **fit recover–backup {<index> | all | shown | active}**
Liest den vorletzten Fit aus dem Speicher, für alle Buffer, die durch das letzte Argument definiert werden. Der Standardwert ist das aktive Spektrum.

6.4.18 Fit restore

tv> **fit restore**
Liest den vorletzten Fit für das aktive Spektrum aus dem Speicher (siehe **fit recover–backup**)

6.4.19 Fit result–file

tv> **fit result–file**

tv> **fit result–file append–open <Dateiname>**
Öffnet die Protokolldatei <Dateiname>, um mit **fit print** die Ergebnisse eines Fits anzuhängen.

tv> **fit result–file close**
Schließt die Protokolldatei.

tv> **fit result–file format**

tv> **fit result–file format default <arg>**
Setzt das Ausgabeformat für <arg> auf den Standardwert.

tv> **fit result–file format enter <arg>**
Erwartet als Eingabe das Ausgabeformat für <arg>.

tv> **fit result–file format read <arg> <Dateiname>**
Liest das Ausgabeformat für <arg> aus der Datei <Dateiname>.

Argument	Funktion
bg–polynom	Format um das Untergrund–Polynom in einer Datei zu speichern.
bg–exponential	Format um den Exponentialanteil des Untergrunds in einer Datei zu speichern.
fit	Format um die Ergebnisse eines Fits in einer Datei zu speichern.
integration	Format um die Ergebnisse einer Integration in einer Datei zu speichern.
mark	Format um Marker in einer Datei zu speichern.
peak–fit	Format um die Ergebnisse eines Peak–Fits in einer Datei zu speichern.
peak–integration	Format um die Ergebnisse einer Peak–Integration in einer Datei zu speichern.
peak–search	Format um die Ergebnisse einer Peaksuche in einer Datei zu speichern.

Tabelle 6.7: Argumente für das **result–file format** Kommando.

`tv>` **fit result–file format status**
Gibt das Ausgabeformat für alle Werte an.

`tv>` **fit result–file format write <arg> <Dateiname>**
Speichert das Ausgabeformat für <arg> in die Datei <Dateiname>.

`tv>` **fit result–file open <Dateiname>**
Öffnet die Datei <Dateiname>, um mit **fit print** oder **fit peaklist print** die Ergebnisse eines Fits darin zu speichern.

6.4.20 Fit status

`tv>` **fit status full**
Gibt detaillierte Statusinformationen über den momentanen Fit aus.

`tv>` **fit status short**
Gibt eine kurze Statusinformation über den momentanen Fit aus.

6.4.21 Fit scale

`tv>` **fit scale input {calibrated | default | uncalibrated}**
Legt fest, ob gelesene Eingabewerte als geeichte oder ungeeichte Werte interpretiert werden.

`tv>` **fit scale marker {calibrated | default | uncalibrated}**
Legt fest, ob Fit Marker Werte als geeichte oder ungeeichte Werte festgelegt werden.

`tv>` **fit scale output {calibrated | default | uncalibrated}**
Legt fest, ob Ausgabewerte als geeichte oder ungeeichte Werte interpretiert werden.

6.4.22 Fit store

`tv>` **fit store**
Speichert den momentanen Fit im Speicher (siehe **fit restore**).

6.4.23 Fit use–fitdata

`tv>` **fit use–fitdata <index> | [active]**
Verwende die Marker aus dem Buffer <index> für den aktiven Buffer.

6.4.24 Fit write

`tv>` **fit write**
Speichert den Fit mit allen Parametern und verwendeten Markern in der Datei <Spektrumname>.fit ab.

6.5 Label

Argument	Funktion
library	?????
calibrated	Die geeichte Energie wird an den Peak geschrieben.
nocalibrated	Die geeichte Energie wird nicht an den Peak geschrieben.
uncalibrated	Die ungeeichte Energie wird an den Peak geschrieben.
nouncalibrated	Die ungeeichte Energie wird nicht an den Peak geschrieben.
string	Eine freidefinierbare Information wird an den Peak geschrieben.
nostring	Die freidefinierbare Information wird nicht an den Peak geschrieben.

Tabelle 6.8: Argumente für das **parameter** Kommando.

6.5.1 Label cut

`tv>` **label cut parameter <arg>**
Legt fest, welche Informationen an die Cut label geschrieben werden, dabei ist <arg> eine beliebige Kombination der Einträge aus Tabelle 6.8.

`tv>` **label cut status**
Gibt aus, welche Informationen an Cut label geschrieben werden.

`tv>` **label cut write <Dateiname>**
Speichert die aktuelle Cut label Liste in die Datei <Dateiname>.

6.5.2 Label fit

`tv>` **label fit parameter <arg>**
Legt fest, welche Informationen an die Fit label geschrieben werden, dabei ist <arg> eine beliebige Kombination der Einträge aus Tabelle 6.8.

`tv>` **label fit status**
Gibt aus, welche Informationen an Fit label geschrieben werden.

`tv>` **label fit write <Dateiname>**
Speichert die aktuelle Fit label Liste in die Datei <Dateiname>.

6.5.3 Label peaklist

`tv>` **label peaklist parameter <arg>**
Legt fest, welche Informationen an die Peak label geschrieben werden, dabei ist <arg> eine beliebige Kombination der Einträge aus Tabelle 6.8.

`tv>` **label peaklist status**
Gibt aus, welche Informationen an Fit label geschrieben werden.

`tv>` **label peaklist write <filename>**
Speichert die aktuelle Peak label Liste in die Datei <Dateiname>.

6.5.4 Label user

tv> **label user activate <title>**
Aktiviert die User Liste <title>.

tv> **label user create <titel> [calibrated | uncalibrated]**
Erzeugt die Label Liste <titel>, wobei die Werte als Energien (cal) oder Kanäle (uncal) interpretiert werden.

tv> **label user delete {all | cursor | list <title>}**
Löscht einzelne Labels aus der aktiven Liste oder die ganze Liste.

tv> **label user enter {cursor | <u>value</u> | offset} [<Farbidx> ["Text"]]**
Setzt ein neues Label an der Cursorposition oder an der angegebenen Position. Die angegebenen Position wird als Energie oder Kanal interpretiert, je nachdem wie die Liste erzeugt wurde (siehe **label user create** und **label user scale**). Die Farbe des Markers kann mit dm Argument <Farbidx> gewählt werden. Eine beliebige Zeichenkette kann an den Marker geschrieben werden, falls die Option mit dem Kommando **label user parameter string** erlaubt wurde.

tv> **label user parameter <arg>**
Legt fest, welche Informationen an ein User Label geschrieben werden, dabei ist <arg> eine beliebige Kombination der Einträge aus Tabelle 6.8.

tv> **label user read**

> tv> **label user read merge <Dateiname> [<title>]**
> Liest eine Label Liste aus der Datei <Dateiname> und addiert sie zur Liste <title> oder zur aktiven Liste, wenn <title> nicht angegeben ist.
>
> tv> **label user read replace <Dateiname> [<title>]**
> Liest eine Label Liste aus der Datei <Dateiname> und speichert sie in der Label Liste <title>, oder der aktiven Liste, falls <title> nicht angegeben ist.

tv> **label user scale <title> [calibrated | uncalibrated]**
Legt fest, wie die eingegebenen Werte für die User Label Liste <title> interpretiert werden.

tv> **label user status {active | existing}**
Gibt an welche Label Liste aktiv ist bzw. welche Listen existieren.

tv> **label user write <Dateiname> <title>**
Speichert die Label Liste <title> in die Datei <Dateiname>.

6.6 Normalization

Die eigentliche Normierung wird mit dem Kommando (siehe Kapitel 6.8.14) durchgeführt:
tv> **spectrum norm {{<u><index>...</u>} | active | all | shown | visible}**

6.6.1 Normalization marker

tv> **normalization marker delete**
Löscht einen Satz von Normierungs Markern.

`tv>` **normalization marker enter** {**cursor** | **value** | **offset**}
Die ersten beiden Marker begrenzen den Bereich, mit dem die Normierungsfaktoren berechnet werden, alle weiteren Marker definieren Untergrundbereiche für die Normierung.

`tv>` **normalization marker list**
Gibt alle definierten Normierungs Marker aus.

`tv>` **normalization marker read <Dateiname>**
Liest einen Satz von Normierungs Markern aus der Datei <Dateiname>.

`tv>` **normalization marker restore**
Lädt einen zuvor im Speicher abgelegten Satz von Normierungs Markern.

`tv>` **normalization marker store**
Legt einen Satz von Normierungs Markern im Speicher ab.

`tv>` **normalization marker write <Dateiname>**
Speichert einen Satz von Normierungs Markern in der Datei <Dateiname>.

6.6.2 Normalization off

`tv>` **normalization off**
Schaltet die Normierung aus, so daß das Kommando **spectrum norm** keine Wirkung hat.

6.6.3 Normalization scale

`tv>` **normalization scale [calibrated | default | uncalibrated]**
Legt fest, ob die Werte für Normierungs Marker als Energien oder Kanäle interpretiert werden.

6.6.4 Normalization status

`tv>` **normalization status**
Gibt an, wie Spektren normiert werden.

6.6.5 Normalization type

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

`tv>` **normalization type maximum** {{**<index>**...} | **active** | **all** | **shown** | **visible**}
Normiert die durch das letzte Argument definierten Spektren auf dasjenige unter ihnen, welches den größten Kanalinhalt hat.

`tv>` **normalization type reference <reference>** {{**<index>**...} | **active** | **all** | **shown** | **visible**}
Normiert die durch das letzte Argument definierten Spektren auf das Spektrum in Buffer <reference>.

`tv>` **normalization type unity** {{**<index>**...} | **active** | **all** | **shown** | **visible**}
Normiert die durch das letzte Argument definierten Spektren auf das Spektrum mit dem größten Gesamtvolumen.

6.7 Recalibration

6.7.1 Recalibration append

`tv>` **recalibration append <Dateiname>** {**<index>**...}
Hängt die Rekalibrierungsdaten der Buffer <index>... an das Ende der Datei <Dateiname> an.

6.7.2 Recalibration create

`tv>` **recalibration create <reference>** {{**<index>**...} | **active** | **all** | **shown** | **visible**}
Rekalibriert (Shiftet) die Spektren, die durch das letzte Argument definiert werden auf das Spektrum im Buffer <reference>. Alle rekalibrierten Spektren werden im Spektrenstatus (siehe **spectrum status**) mit dem Zeichen < markiert.

6.7.3 Recalibration delete

`tv>` **recalibration delete** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Löscht die Rekalibrierung für die Spektren, die durch das letzte Argument definiert werden.

6.7.4 Recalibration marker

`tv>` **recalibration marker delete** {**all** | **range** {**cursor** | **value** | **offset**}
Löscht Rekalibrierungs Bereichs Marker.

`tv>` **recalibration marker** {**enter** {**cursor** | **value** | **offset**} | **list** | **read <Dateiname>** | **restore** | **store** | **write <filename>**}
Führt eine Operation mit den Rekalibrierungs Bereichs Markern durch.

6.7.5 Recalibration list

`tv>` **recalibration list** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Gibt die Rekalibrierungs Faktoren für die Spektren an, die durch das letzte Argument definiert werden.

6.7.6 Recalibration read

`tv>` **recalibration read <Dateiname>** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Liest Rekalibrierungs Faktoren aus der Datei <Dateiname> für die Spektren, die durch das letzte Argument definiert werden.

6.7.7 Recalibration type

`tv>` **recalibration type** {**center–of–mass** | **squared–distance** | **pair** | **polynom**}
Legt fest, welche Methode für die Rekalibrierung verwendet wird.

6.7.8 Recalibration write

tv> **recalibration write <Dateiname>** {{<u>**<index>...**</u>} | **active** | **all** | **shown** | **visible**}
Speichert die Rekalibrierungs Faktoren für die Spektren, die durch das letzte Argument definiert werden, in der Datei <Dateiname> ab.

6.8 Spectrum

6.8.1 Spectrum activate

tv> **spectrum activate** {**<index>** | **next** | **previous** | **status** Aktiviert das Spektrum, welches durch das letzte Argument definiert wird.

6.8.2 Spectrum add

tv> **spectrum add <index>** {<u>**<filename>['<format>]**</u> | {<u>**<index>...**</u>} | **active** | **all** | **shown** | **visible**}
Addiert die Spektren, die durch das letzte Argument definiert werden, zu dem Spektrum im Buffer <index>. Dieses Kommando hat entweder einen Text oder einen Ganzzahlen Standardwert.

6.8.3 Spectrum analysator

tv> **spectrum analysator <hostname>**
Analysatorspektren werden vom host <hostname> gelesen, der standardmäßig bibo ist.

6.8.4 Spectrum copy

tv> **spectrum copy <source>** {**<destination> ...**}
Kopiert das Spektrum aus dem Buffer <source> in die Buffer <destination> ....

6.8.5 Spectrum create

tv> **spectrum create <name> <resolution> <index>**
Erzeugt ein leeres Spektrum mit dem Namen <name> und der Auflösung <resolution> im Buffer <index>.

6.8.6 Spectrum delete

tv> **spectrum delete** {**<index> ...**}
Löscht die Spektren in den Buffern <index> ....

6.8.7 Spectrum enter

tv> **spectrum enter <index> <channel> <value>**
Setzt den Kanal <channel> im Spektrum im Buffer <index> auf den Wert <value>.

6.8.8 Spectrum format

`tv>` **spectrum format input <format>**
Setzt das Eingabeformat für Spektren auf <format>.

`tv>` **spectrum format output <format>**
Setzt das Ausgabeformet für Spektren auf <format>.

6.8.9 Spectrum get

`tv>` **spectrum get <Dateiname>**
Lädt Spektren aus den Dateien <Dateiname> in die nächsten freien Buffer.

6.8.10 Spectrum ls–file

Um die Dateien aus einer ls–Datei zu bearbeiten, gibt es folgende Kommandos:

Kommando	Funktion
ls–filename	Name der zu bearbeitenden ls–Datei.
+	Verwende das nächste Spektrum in der Liste.
++	Verwende alle Spektren von der aktuellen Position bis zum Ende der ls–Datei.
-	Verwende vorhergehendes Spektrum.
--	Verwende alle vorhergehenden Spektren bis zum Anfang der ls–Datei.
line–index	Verwende das Spektrum an der Position line–index.

Tabelle 6.9: Arguments für die **ls–file** Kommandos.

`tv>` **spectrum ls–file add <index> <arg>**
Addiere die durch <arg> beschriebenen Spektren zum Buffer <index>.

`tv>` **spectrum ls–file close**
Schließe die momentan geöffnete ls–Datei.

`tv>` **spectrum ls–file commandget <Kommandodatei> <arg>**
Wende die Kommandos aus der Datei <Kommandodatei> auf die durch <arg> definierten Spektren an.

`tv>` **spectrum ls–file get <arg>**
Lade die durch <arg> definierten Spektren.

`tv>` **spectrum ls–file list**
Gibt die momentane Position des Zeigers in der ls–Datei an.

`tv>` **spectrum ls–file open <ls–Dateiname>**
Öffnet die ls–Datei <ls–Dateiname>.

`tv>` **spectrum ls–file position <index>**
Positioniert den Zeiger auf <index> in der geöffneten ls–Datei.

`tv>` **spectrum ls–file subtract <index> <arg>**
Subtrahiert die durch <arg> definierten Spektren vom Spektrum im Buffer <index>.

6.8.11 Spectrum maximum

`tv>` **spectrum maximum <index>** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Kopiert das Spektrum aus der Auswahl, die durch das letzte Argument gegeben ist, mit dem maximalen Kanalinhalt in den Buffer <index>.

6.8.12 Spectrum minimum

`tv>` **spectrum minimum <index>** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Kopiert das Spektrum aus der Auswahl, die durch das letzte Argument gegeben ist, mit dem minimalen Kanalinhalt in den Buffer <index>.

6.8.13 Spectrum multiply

`tv>` **spectrum multiply <factor>** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Multipliziert die Spektren, die durch das letzte Argument definiert werden mit dem Faktor <factor>.

6.8.14 Spectrum norm

`tv>` **spectrum norm** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Normiert die Spektren, die durch das Argument definiert werden (siehe auch Abschnitt 6.6). Alle normierten Spektren werden im Spektrenstatus mit dem Zeichen * markiert.

6.8.15 Spectrum offset

`tv>` **spectrum offset <offset>** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Addiert den Offset <offset> zu jedem Kanal der Spektren, die durch das letzte Argument definiert werden.

6.8.16 Spectrum read

`tv>` **spectrum read <Dateiname>** {{**<index>...**} | **active** | **all** | **shown** | **visible**}
Liest das Spektrum <Dateiname> in die Buffer, die durch das letzte Argument definiert werden.

6.8.17 Spectrum resolution

`tv>` **spectrum resolution <index> <resolution>**
Ändert die Auflösung des Spektrums im Buffer <index> auf <resolution>.

6.8.18 Spectrum status

`tv>` **spectrum status**
Gibt Statusinformationen über alle geladenen Spektren aus. Die letzte Spalte dieser Informationen gibt Auskunft darüber, welche Operationen mit dem Spektrum durchgeführt wurden. Die Bedeutung der dazu verwendeten Zeichen ist in Tabelle 6.10 angegeben.

Zeichen	Bedeutung
*	Das Spektrum wurde normiert oder multipliziert.
<	Das Spektrum wurde recalibriert.
+	Das Spektrum wurde addiert, subtrahiert oder die Offset Operation wurde angewandt.
&	Das Spektrum wurde für eine Maximierungsoperation verwendet.

Tabelle 6.10: Bedeutung der Zeichen, die im Spektrum Statusreport verwendet werden, um die auf ein Spektrum angewendeten Operationen anzuzeigen.

6.8.19 Spectrum subtract

tv> **spectrum subtract <index>** {**<Dateiname>['<format>]** | {**<index>**...} | **active** | **all** | **shown** | **visible**}
Subtrahiert Buffer, die durch das letzte Argument definiert werden, vom Buffer <index>. Dieses Kommando hat entweder einen Text oder einen Ganzzahl Standardwert.

6.8.20 Spectrum update

tv> **spectrum update** {**<index>** | **all** | **shown** | **active**}
!!!!!!!!!!!!!!! VORSICHT !!!!!!!!!!!!!!!
Lädt alle angegebenen Spektren neu, ohne nachzufragen.

6.8.21 Spectrum write

tv> **spectrum write <Dateiname>** {{**<index>**...} | **active** | **all** | **shown** | **visible**}
Speichert die Spektren in den Buffern, die durch das letzte Argument definiert werden, in die Datei <filename>.

6.9 Window

6.9.1 Window create

Siehe Kapitel 4.2.1 für eine Beschreibung der Fenstertypen, die mit diesem Kommando erzeugt werden.

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 <Fenstername>**
Zerstört das Fenster <Fenstername>.

6.9.3 Window hide

Versteckt das Objekt im aktiven Fenster, das durch das zusätzlich Argument gegeben ist.

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**
Gibt eine Liste aller Graphikfenster aus.

6.9.5 Window marker

tv> **window marker horizontal** {**delete** | **enter** {**cursor** | **value**} | **list** | **read** <**filename**> | **restore** | **store** | **write** <**filename**>}
Verwaltet horizontale Marker im aktiven Fenster.

tv> **window marker vertical** {**delete** | **enter** {**cursor** | **value**} | **list** | **read** <**filename**> | **restore** | **store** | **write** <**filename**>}
Verwaltet vertikale Marker im aktiven Fenster.

6.9.6 Window plot

tv> **window plot create**

tv> **window plot create unix–plot [Dateiname]**
Erzeugt einen Ausdruck im unix–plot Format und speichert die Ausgabe in der Datei <filename>. Falls kein Dateiname angegeben ist, wird der Dateiname entsprechen der wilcard verwendet.

tv> **window plot create xfig [filename]**
Erzeugt einen Ausdruck im xfig Format (Protokoll 3.2).

tv> **window plot format**

tv> **window plot format cm–format** <**x**> <**y**>
Definiert die Größe des Ausdrucks in der Einheit cm.

tv> **window plot format px–format** <**x**> <**y**>
Definiert die Größe des Ausdrucks in der Einheit pixel.

tv> **window plot format pts/inch** <**x**> <**y**>
Definiert die Größe des Ausdrucks in der Einheit inch.

tv> **window plot format fontscale–factor** <**factor**>
Definiert den Skalierungsfaktor für Zeichensätze.

6.9.7 Window raise

tv> **window raise** <**Fenstername**>
Bringt das Fenster <Fenstername> in den Vordergrund.

6.9.8 Window redisplay

tv> **window redisplay**
Erneuert den Inhalt des aktiven Fensters.

6.9.9 Window scaling

tv> **window scaling function–y**

tv> **window scaling function–y linear**
Bereitet die lineare Skalierung der y–Achse vor.

tv> **window scaling function–y logarithmic**
Bereitet die logarithmische Skalierung der y–Achse vor.

tv> **window scaling function–y normalization {on | off}**
Dieses Kommando aktiviert die eingestellte Skalierung der y–Achse.

tv> **window scaling function–y squared**
Bereitet die quadratische Skalierung der y–Achse vor.

tv> **window scaling modification–y**

tv> **window scaling modification–y no–modification**
Nimmt alle Veränderungen der y–Achse zurück.

tv> **window scaling modification–y efficiency**
Streckt die y–Achse entsprechend der Efficiency Eichung?????

tv> **window scaling modification–y multiplied**
Multipliziert den Inhalt eines jeden Kanals mit der Kanalnummer.

tv> **window scaling scale–x [calibrated | default | uncalibrated]**
Legt fest, ob eingegebene Werte als Energien oder Kanäle interpretiert werden.

6.9.10 Window setup

tv> **window setup cursor**

tv> **window setup cursor cross–hair**
Der Graphikcursor wird zum Fadenkreuz.

tv> **window setup cursor vertical**
Keine Funktion.

tv> **window setup cursor subwindow**

tv> **window setup cursor subwindow cross–hair**
Der Graphikcursor des Unterfensters wird zum Fadenkreuz.

tv> **window setup cursor subwindow doppler <factor>**
Der Graphikcursor des Unterfensters wird zum Raster (grid), wobei die Anzahl der Elemente (number) 1 ist und der Offset von der Cursorposition abhängt. Der Standardwert für <factor> ist 0.

tv> **window setup cursor subwindow grid <number> <offset>**
Der Graphikcursor des Unterfensters wird zum Raster (grid) mit <number> Linien links und rechts des vertikalen Cursors. Der Abstand zwischen den Linien ist <offset> Skaleneinheiten. Die Standardwerte für <number> und <offset> sind 0.

tv> **window setup frame**

tv> **window setup frame distances <tic> <label>**
Definiert den minimalen Abstand zwischen den Skalierungslinien und den Achsenbeschriftungen. Standardwerte sind 0,3 Zeichen (chars) für Skalierungslinien und 3 chars für Achsenbeschriftungen.

tv> **window setup frame lengths <short> <medium> <long> <dist>**
Legt die Längen für kurze, mittlere und lange Skalierungslinien und den Abstand zwischen Skalierungslinie und Achsenbeschriftung fest. Standardwerte sind 0,2, 0,4 und 0,6 chars für die Skalierungslinien und 0,75 chars für den Abstand.

tv> **window setup frame widths <left> <right> <bottom> <top>**
Definiert die Breiten des linken, rechten, unteren und oberen Rahmens. Standardwerte sind 3, 3, 2, 2 chars in der Reihenfolge der Argumente.

tv> **window setup histogram–compression**

tv> **window setup histogram–compression each**
Faßt eine Gruppe von Kanälen zu einem zusammen, wobei der Mittelwert aller dieser Kanäle genommen wird.

tv> **window setup histogram–compression first–found**
Faßt eine Gruppe von Kanälen zusammen, wobei nur der erste dieser Kanäle genommen wird.

tv> **window setup histogram–compression min–max**
Faßt eine Gruppe von Kanälen zusammen, wobei nur der Kanal mit dem minimalen oder maximalen Inhalt genommen wird, abhängig vom Vorzeichen.

tv> **window setup keyboard–focus {title <subwindow> | cursor | none}**
Legt das Graphikfenster fest, für das die Eingabe im Textfenster ausgewertet wird. Es kann ein beliebiges Unterfenster sein, das Fenster in dem sich der Cursor befindet oder keins.

6.9.11 Window show

tv> **window show cut**

tv> **window show cut spectrum <index>**
Stellt das Spektrum dar, das dem Cut <index> zugeordnet ist.

tv> **window show cut projection**
Stellt die Projektion dar, die dem aktiven Cut zugeordnet ist.

tv> **window show cut gate–marker**
Stellt die Gate Marker für den aktiven Cut dar.

tv> **window show fit**

tv> **window show fit bin–list**
Stellt die bin Liste zum aktiven Fit dar.

tv> **window show fit decomposition**
Stellt die Dekomposition des Fits dar.

tv> **window show fit function/marker**
Stellt die Fit Fuktionen und Marker dar.

tv> **window show labellist**

tv> **window show labellist cut**
Stellt die Beschriftungen für den aktiven Cut dar.

tv> **window show labellist peaklist**
Stellt die Beschriftungen für alle Peaks in der aktiven Peakliste dar.

tv> **window show labellist user**
Stellt alle Beschriftungen aus der User Labelliste dar.

tv> **window show marker**

tv> **window show marker cut**
Stellt die Marker für den aktiven Cut dar.

tv> **window show marker fit**
Stellt die Marker für den aktiven Fit dar.

tv> **window show marker normalization**
Stellt Normierungs Marker dar.

tv> **window show marker recalibration**
Stellt Rekalibrierungs Marker dar.

tv> **window show marker psearch**
Stellt Peaksearch Marker dar.

tv> **window show peaklist**

tv> **window show spectrum**

tv> **window show spectrum index <index>**
Stellt das Spektrum im Buffer <index> dar.

tv> **window show spectrum next**
Stellt das Spektrum im nächsten Buffer dar.

tv> **window show spectrum previous**
Stellt das Spektrum im vorhergehenden Buffer dar.

tv> **window show spectrum + {<index>...}**
Stellt zusätzlich zum dargestellten Spektrum die Spektren in den Buffern <index>...dar.

tv> **window show spectrum - {<index>...}**
Versteckt die Spektren in den Buffern <index>....

tv> **window show spectrum names**
Zeigt die Namen der dargestellten Spektren in der oberen rechten Ecke des Graphikfensters an.

tv> **window show status**
Gibt die Nummer des aktiven Spektrums und Cuts aus und eine Liste aller Spektren, die im aktiven Fenster dargestellt sind.

6.9.12 Window status

tv> **window status**
Gibt eine Liste aller Graphikfenster aus.

6.9.13 Window view

tv> **window view center**

tv> **window view center both <x–value> <y>**
Zentriert die dargestellten Spektren im aktiven Fenster um die Energie oder den Kanal <x–value> und die Zahl der Ereignisse <y>.

`tv>` **window view center x <x–value>**
Zentriert die dargestellten Spektren im aktiven Fenster um die Energie oder den Kanal <x–value>.

`tv>` **window view center y <y>**
Zentriert die dargestellten Spektren im aktiven Fenster um die Ereignisse <y>.

`tv>` **window view expand**
Expandiert die dargestellte Region zwischen den Markern, die mit einem der Befehle **window mark vertical enter** oder **window view full** gesetzt wurden.

`tv>` **window view full**

`tv>` **window view full both**
Bereitet das Fenster vor, um Spektren in voller x– und y–Größe darzustellen.

`tv>` **window view full x**
Bereitet das Fenster vor, um Spektren in voller x–Größe darzustellen.

`tv>` **window view full y**
Bereitet das Fenster vor, um Spektren in voller y–Größe darzustellen.

`tv>` **window view position <width> <position>**
Zentriert die dargestellten Spektren im aktiven Fenster um die Energie oder den Kanal <position>. Die Anzahl der dargestellten Kanäle wird durch den Wert <width> bestimmt.

`tv>` **window view shift <x–percent> <y–percent>**
Verschiebt den Fensterausschnitt um <x–percent> entlang der x–Achse und <y–percent> entlang der y–Achse.

`tv>` **window view stretch <-log$_2$(x–fac)> <cursor | x–value> <-log$_2$(y–fac)> <cursor | y–value>**
Streckt den Fensterausschnitt um den Faktor <-log$_2$(x–fac)> in x–Richtung, wobei das Zentrum durch <x–value> oder die Cursorposition gegeben ist und um den Faktor <-log$_2$(y–fac)> in y–Richtung, wobei das Zentrum durch <y–value> oder der Cursorposition gegeben ist.

`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 [<Kommando> [<alias> [<arg–list>]]]**
Definiert den Aliasnamen <alias>. Führe <alias> aus, wenn <Kommando> eingegeben wird. <arg–list> wird ausgedruckt, wenn ein Fragezeichen als Argument zu <Kommando> eingegeben wird. Falls alias ohne Argumente aufgerufen wird, druckt es eine Liste aller definierten Aliasnamen aus.

6.10.2 ReadMe

tv> **ReadMe**
Gibt eine Kopierrecht Information und eine Liste aller erlaubten Kommandozeilenargumente aus.

6.10.3 cd

tv> **cd <Pfadname>**
Wechselt das Arbeitsverzeichnis nach <Pfadname>. Falls kein Pfadname angegeben wird, wird eine Eingabe verlangt und der Pfad des momentanen Arbeitsverzeichnisses wird als Standardwert genommen.

6.10.4 command–file

tv> **command–file close**
Schließt eine Kommandodatei.

tv> **command–file open <Dateiname>**
Öffnet die Kommandodatei <Dateiname>.

tv> **command–file resume**
Schaltet die Graphikanzeige an und führt die Kommandos aus der geöffneten Kommandodatei aus.

tv> **command–file suspend**
Schaltet die Graphikanzeige aus und führt die Kommandos aus der geöffneten Kommandodatei aus.

6.10.5 edit–lock

tv> **edit–lock**
Einige Kommandos fordern die Eingabe von alphanumerischen Zeichen, die im Cursor–Modus nicht möglich ist. edit–lock schaltet den Cursor–Modus bis zum nächsten RETURN aus.

6.10.6 graphic

tv> **graphic resume**

tv> **graphic resume–forced**

tv> **graphic suspend**

Man kann die graphische Ausgabe abschalten, um die Abarbeitung von Kommandodateien zu beschleunigen, zum Beispiel das Shiften von Projektionen. Die Kommandos suspend und resume werden immer paarweise verwendet, daß heißt wenn man die Graphik mit suspend n mal ausgeschaltet hat, muß man sie mit resume auch n mal wieder einschalten. Mit resume–forced können beliebig viele suspend Kommandos auf einmal aufgehoben werden.

6.10.7 input–mode

tv> **input–mode cursor**
Schaltet den Eingabe–Modus auf Cursor–Modus um (siehe Kapitel 3.4.2).

tv> **input–mode edit**
Schaltet den Eingabe–Modus auf Edit–Modus um (siehe Kapitel 3.4.1).

6.10.8 keyalias

tv> **keyalias [<Tastenkombination> [<alias>]]**
Ohne Argumente wird eine Tabelle aller definerter Hotkeys (siehe Kapitel A) ausgegeben. Wenn eine Tastenkombination als Argument angegeben wird, werden nur die entsprechenden Kommandos für diese ausgegeben und falls ein Kommando als drittes Argument angegeben ist, wird ein neuer Eintrag in der Hotkey Tabelle hinzugefügt, bzw. ein alter Eintrag ersetzt.

6.10.9 keyunalias

tv> **keyunalias <Tastenkombination>**
Die <Tastenkombination> wird aus der Tabelle der Hotkeys entfernt.

6.10.10 keytable

tv> **keytable activate <title>**
Deaktiviert die momentan benutzte Hotkey Tabelle und aktiviert die Tabelle mit dem Namen <title>.

tv> **keytable ascii [<char–value> [<char–value>]]**
Gibt eine Liste von Zeichen in den fünf verschiedenen Formaten dezimal, hexadezimal, oktal, einem symbolischen Namen und der entsprechenden Taste. Die Control Taste wird durch das Zeichen ^ dargestellt.

tv> **keytable create <title>**
Erzeugt eine leere Hotkey Tabelle.

tv> **keytable delete <title>**
Löscht die Hotkey Tabelle mit dem Namen <title>.

tv> **keytable list**
Gibt die aktive Hotkey Tabelle aus.

tv> **keytable read**

> tv> **keytable read merge <Dateiname> [<title>]**
> Addiert die Hotkey Tabelle , die aus der Datei <Dateiname> gelesen wird, zur Tabelle mit dem Namen <title> hinzu, oder zur aktiven Tabelle, falls kein Name angegeben ist.
>
> tv> **keytable read replace <Dateiname> [<title>]**
> Ersetzt die Hotkey Tabelle, die durch <title> definiert wird, oder die aktive Tabelle, falls <title> nicht angegeben ist, durch die Tabelle aus der Datei <Dateiname>.

tv> **keytable status**
Gibt eine Liste aller Hotkey Tabellen und der Dateinamen, in denen sie gespeichert sind, aus.

tv> **keytable write <Dateiname>**
Speichert die aktive Hotkey Tabelle in die Datei <Dateiname>.

6.10.11 precision

tv> **precision <digits>**
Legt die Präzision, daß heißt die Anzahl der Nachkommastellen für Marker fest. <digits> muß ganzzahlig sein.

6.10.12 protocol

tv> **protocol command {close | open <Dateiname>}**
Alle eingegebenen Kommandos werden in der Protokolldatei <Dateiname> aufgezeichnet.

tv> **protocol session {close | open <Dateiname>}**
Sowohl alle eingegebenen Kommandos als auch alle Ausgaben von Tv werden in der Protokolldatei <Dateiname> aufgezeichnet.

tv> **protocol standard–output {close | open}**
Alle Ausgaben auf die Standardausgabe werden unterbunden oder erlaubt.

6.10.13 which

tv> **which [config–file | command–file]**
Ohne weitere Argumente gibt das **which** Kommando den Suchpfad für Konfigurations– und Kommandodateien aus. Falls ein Dateiname als Argument angegeben ist, gibt **which** den absoluten Pfad der von Tv verwendeten Datei.

6.10.14 wildcard

tv> **wildcard delete <wildcard>**
Löscht die <wildcard> aus der Liste.

tv> **wildcard enter <wildcard> <expression>**
Fügt der Liste die <wildcard> hinzu bzw. ersetzt sie, falls sie schon existiert.

tv> **wildcard status**
Druckt eine Liste aller definierten wildcards aus.

Siehe Kapitel 3.12.2 für weitere Erklärungen des wildcard Konzepts.

6.10.15 exit

6.11 Default aliases

6.11.1 quit

tv> **quit**
Führe das Kommando

tv> **exit**

durch.

6.11.2 lin

tv> **lin**
Ändere die Skalierung der y–Achse auf lineare Einheiten:

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**
Ändere die Skalierung der y–Achse auf logarithmische Einheiten:

tv> **window scaling function–y logarithmic;**

tv> **window scaling function–y normalization on;**

tv> **window scaling function–y normalization off;**

tv> **ysqr**
Ändere die Skalierung der y–Achse auf quadratische Einheiten:

tv> **window scaling function–y squared;**

tv> **window scaling function–y normalization on;**

tv> **window scaling function–y normalization off;**

Anhang A

Die Standard Hotkey Tabelle

In der Standarddistribution von Tv hat die Datei *.tvkeys* folgenden Inhalt.

Hotkey	Tv–Kommandos	Funktion
Strg c	tv> input–mode edit;	Umschalten von Cursor–Modus in Edit–Modus, sonst Tv beenden
Strg l	tv> window redisplay;	Fensterinhalt erneuern
Strg o	tv> window redisplay;	Fensterinhalt erneuern
Esc	tv> input-mode edit;	Umschalten zwischen Edit– und Cursor–Modus
Spacebar	tv> window mark vertical enter cursor;	Setze vertikale Marker zum Definieren des zu erweiternden x–Ausschnitts
- 1	tv> window view full y; tv> window view stretch -1 cursor 0;	Verdoppele die Zahl der dargestellten Kanäle
- 2	tv> window view full y; tv> window view stretch -2 cursor 0;	Vervierfache die Zahl der dargestellten Kanäle
- 3	tv> window view full y; tv> window view stretch -3 cursor 0;	Verachtfache die Zahl der dargestellten Kanäle
- 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;	Lösche alle Marker, außer Fitmarkern.

Weiter auf der nächsten Seite

Hotkey	Tv–Kommandos	Funktion
	tv> label user delete all; tv> normalization mark delete; tv> recalibration marker delete all;	
- C	tv> cut mark cut delete; tv> window hide mark cut;	Lösche Cut Marker
- D	tv> window hide fit decomposition;	Verstecke die Dekomposition des Fits
- F	tv> fit mark fit delete; tv> fit delete; tv> fit delete activ; tv> window hide mark fit;	Lösche den aktuellen Fit und alle seine Marker
- P	tv> fit mark peak delete 0 uncalib 8192 uncalib;	Lösche alle Peak Marker
- p	tv> fit mark peak delete cursor;	Delete peak marker closest to cursor
- B	tv> fit mark bg-region delete all;	Lösche alle Untergrund Marker
- b	tv> fit mark bg-region delete cursor;	Lösche den Untergrund Marker, der dem Cursor am nächsten ist
- c g	tv> cut mark gate delete cursor;	Lösche den Gate Marker, der dem Cursor am nächsten ist
- c b	tv> cut mark bg–gate delete cursor;	Lösche den Untergrund–Gate Marker, der dem Cursor am nächsten ist
- l	tv> label user delete cursor;	Lösche User Label
- n	tv> normalization mark delete;	Lösche die Normalisierungs Marker
- r	tv> fit mark region delete cursor;	Lösche den Fit–Region Marker, der dem Cursor am nächsten ist
- R	tv> recalibration marker delete cursor;	Lösche den Rekalibrierungs Marker, der dem Cursor am nächsten ist
- w n	tv> window scaling function-y normalization off;	
,	tv> window view full y; tv> window view shift -70.0 0.0;	Verschiebe den Viewport nach links um 70% der sichtbaren Kanäle
.	tv> window view full y; tv> window view shift 70.0 0.0;	Verschiebe den Viewport nach rechts um 70% der sichtbaren Kanäle
0	tv> window view full y; tv> window view stretch -1 cursor 0;	Verdoppele die Anzahl der dargestellten Kanäle
1	tv> window view full y; tv> window view stretch 1 cursor 0;	Halbiere die Anzahl der dargestellten Kanäle
2	tv> window view full y; tv> window view stretch 2 cursor 0;	Zeige ein Viertel der dargestellten Kanäle
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;	

Weiter auf der nächsten Seite

Hotkey	Tv–Kommandos	Funktion
	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> 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;	Drucke eine Liste aller Hotkeys im Text–Fenster aus
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;	Lese eine Eichung für alle geladenen Spektren
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;	Definiere einen Gate Marker
H P	tv> fit parameter position hold;	Die Peak Position wird nicht gefittet
H W	tv> fit parameter width hold;	Die Peak Breite wird nicht gefittet
I	tv> fit integration–create active; tv> fit status short;	Integriere das aktivierte Spektrum und drucke einen Kurzstatus
L	tv> window setup keyboard–focus cursor;	Der Tastaturfokus wird auf das Graphikfenster gesetzt, in dem sich der Cursor befindet
M	tv> edit; tv> window create cut Cut; tv> cut environment	Öffnet ein Cut–Fenster mit dem Namen “Cut” und fragt nach einem zu ladenden Environment

Weiter auf der nächsten Seite

Hotkey	Tv–Kommandos	Funktion
N n tv> window redisplay;	tv> window show spectrum next;	Zeige das nächste Spektrum in der Bufferliste an
N p tv> window redisplay;	tv> window show spectrum previous;	Zeige das vorherige Spektrum in der Bufferliste an
P b h	tv> fit param backgr hold; tv> fit param exponent hold; tv> fit param factor hold;	Die Untergrund–Parameter werden nicht gefittet
P b f	tv> fit param backgr free; tv> fit param exponent free; tv> fit param factor free;	Die Untergrund–Parameter werden gefittet
P p h 0	tv> fit para position0 hold;	Hält die Position des Peaks 0 fest. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P p f 0	tv> fit para position0 free;	Die Position des Peaks 0 wird nicht mehr festgehalten. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P p n 0	tv> fit para position0 none;	Der Peak 0 wird aus der Peakliste entfernt. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P p = 0	tv> edit; tv> fit para position0 = ?	Die Position des Peaks 0 muß noch eingegeben werden. Diese Hotkey Kombination gibt es auch für die Peaks 1 bis 4.
P w e	tv> fit parameter width equal;	Die Breiten aller Peaks werden gleichgesetzt
P u	tv> edit; tv> window plot create unix-plot	Drucke einen Unix–Plot in eine wählbare Datei
P x	tv> edit; tv> window plot create xfig	Drucke einen xfig v3.2 Plot in eine wählbare Datei
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;	Erzeuge einen Schnellfit.
R b c	tv> fit read bin position cursor;	Liest den Fit an der Cursorposition.
R b f	tv> fit read bin first;	Liest den ersten Fit aus der Fitdatei.

Weiter auf der nächsten Seite

Hotkey	Tv–Kommandos	Funktion
R b n	tv> fit read bin next;	Liest den nächsten Fit aus der Fitdatei.
R b p	tv> fit read bin position cursor;	Liest den Fit an der Cursorposition.
R c c	tv> cut read cut cursor;	Liest das Schnittspektrum und den Satz Cut–Marker an der Cursorposition.
R c f	tv> cut read cut first;	Liest das erste Schnittspektrum und den ersten Satz Cut–Marker.
R c n	tv> cut read cut next;	Liest das nächste Schnittspektrum und den nächsten Satz Cut–Marker.
R f f	tv> fit read fit first;	Liest den ersten Fit aus der Fitdatei.
R f n	tv> fit read fit next;	Liest den nächsten Fit aus der Fitdatei.
R h c	tv> cut read head cursor;	Liest den Satz Cut–Marker an der Cursorposition und führt den Cut aus.
R h f	tv> cut read head first;	Liest den ersten Satz Cut–Marker und führt den Cut aus.
R h n	tv> cut read head next;	Liest den nächsten Satz Cut–Marker und führt den Cut aus.
R i f	tv> fit read integration first;	Liest die erste Integration aus der Fitdatei.
R i n	tv> fit read integration next;	Liest die nächste Integration aus der Fitdatei.
R r	tv> recalibration marker enter cursor;	Setze einen Rekalibrierungs Marker an der Cursor Position
R R	tv> edit; tv> recalibration create;	Fragt nach einem Referenz–Buffer und erzeugt eine Rekalibrierung
S f	tv> fit write;	Schreibt einen Fit oder eine Integration in die Fitdatei.
S c	tv> cut write cut;	Schreibt das Schnittspektrum und die Cut–Marker in ein Verzeichnis.
U	tv> window setup keyboard-focus none;	
a	tv> edit; tv> spectrum activate;	Fragt nach einer Buffernummer und aktiviert das darin befindliche Spektrum
b	tv> fit mark bg-region enter cursor;	Setze einen Untergrund–Region Marker an der Cursor Position
c	tv> cut mark cut enter cursor;	Setze einen Cut Marker an der Cursor Position
e	tv> window view expand;	Expandiere den Viewport auf den x–Bereich zwischen den Markern, die mit Spacebar gesetzt wurden
f	tv> window view full both;	Expandiere den sichbaren Teil

Weiter auf der nächsten Seite

Hotkey	Tv–Kommandos	Funktion
	tv> window view expand;	auf die volle Größe
g	tv> edit; tv> spectrum get	Fragt nach einem Spektrum–namen und liest das Spektrum
h	tv> window mark horizontal enter cursor;	Set a horizontal marker
i	tv> edit; tv> window view full y; tv> window view position 100;	Fragt nach einer Energie und zeigt 100 Kanäle an, wobei die eingegebene Position das Zentrum ist
k	tv> edit; tv> spectrum delete;	Fragt nach einer Buffernummer und löscht das darin befindliche Spektrum
l +	tv> spectrum delete all; tv> spectrum ls-file get ++;	Löscht alle Spektren und liest entsprechend den ls–file Regeln (siehe 6.8.10)
l -	tv> spectrum delete all; tv> spectrum ls-file get --;	Löscht alle Spektren und liest entsprechend den ls–file Regeln (siehe 6.8.10)
l g	tv> edit; tv> spectrum ls-file get;	Fragt nach zu ladenden Spektren und liest entsprechend den ls–file Regeln (siehe 6.8.10)
l o	tv> edit; tv> spectrum ls-file open;	Fragt nach einem ls–file Namen und öffnet die Datei
m P	tv> window show peaklist; tv> window show labellist peaklist; tv> fit open active;	Zeigt alle Peak Marker
m b	tv> window show fit bin-list; tv> fit open active;	Zeigt alle bin Marker
m d	tv> window show fit decomposition; tv> fit open active;	Zeigt die Dekomposition des Fits für aktivierte Spektrum an
m f	tv> window show fit function/marker;	Zeige Fitfunktion und Marker an
m n	tv> window show spectrum next;	Zeige das nächste Spektrum in der Bufferliste an
m p	tv> window show spectrum previous;	Zeige das vorhergehende Spektrum an
m s	tv> edit; tv> window show spectrum index;	Fragt nach einer Liste von Buffernummern und zeigt die zugehörigen Spektren an
n	tv> edit; tv> window show spectrum index;	Fragt nach einer Liste von Buffernummern und zeigt die zugehörigen Spektren an
o	tv> normalization mark enter cursor;	Setze einen Normalizierungs–Marker an der Cursor Position
p	tv> fit mark peak enter cursor;	Setze einen Peak Marker an der Cursor Position
r	tv> fit mark region enter cursor;	Setze einen Fit–Region Marker an der Cursor Position
s	tv> fit status full;	Drucke alle Statusinformationen über den Fit
u P	tv> window hide peaklist; window hide labellist peaklist;	Verstecke alle Peak Marker

Weiter auf der nächsten Seite

Hotkey	Tv–Kommandos	Funktion
u b	tv> window hide fit bin–list;	Verstecke alle bin Marker
u c	tv> window hide cut–gate–marker;	Verstecke alle Cut Gate Marker
u d	tv> window hide fit decomposition;	Verstecke die Dekomposition des Fits
u f	tv> window hide fit function/marker;	Verstecke Fit–Funktion und Marker
u m c	tv> window hide marker cut;	Verstecke alle Cut Marker
u m f	tv> window hide marker fit;	Verstecke alleFit–Region Marker
u m n	tv> window hide marker normalization;	Verstecke alle Normalisierungs–Marker
u m r	tv> window hide marker recalibration;	Verstecke alle Rekalibrierungs–Marker
u m s	tv> window hide marker psearch;	Verstecke alle Peaksearch–Region Marker
w e	tv> fit parameter width equal;	Die Breiten aller Peaks werden gleichgesetzt
w f	tv> fit parameter width free;	Fitte auch die Breite
w i	tv> window scaling function-y linear; tv> window scaling function-y normalization on; tv> window scaling function-y normalization off;	Schaltet die y–Achse auf lineare Skalierung
w l	tv> window scaling function-y logarithmic; tv> window scaling function-y normalization on; tv> window scaling function-y normalization off;	Schaltet die y–Achse auf logarithmische Skalierung
w n	tv> window scaling function-y normalization on;	
w s	tv> window scaling function-y squared; tv> window scaling function-y normalization on; tv> window scaling function-y normalization off;	Schaltet die y–Achse auf quadratische Skalierung
x	tv> window view full x; tv> window view expand;	Expandiere den Viewport auf die volle Weite in x–Richtung
y	tv> window view full y; tv> window view expand;	Expandiere den Viewport auf die volle Höhe in y–Richtung
$\vert$	tv> window view full y; tv> window view center x cursor;	Zentriere die Spektren um die Cursorposition

Anhang B

Xresources

B.1 Konfiguration von Tv

Durch die Verwendung von Xresources können Voreinstellungen für Tv schnell und einfach gesetzt werden. Eine detaillierte Beschreibung der Handhabung von Xresources wird in der zugehörigen Manualseite **X(1)** gegeben. Die Konfiguration wird aus folgenden Dateien gelesen

1. app–defaults/Xtv
2. ~/.Xdefaults
3. ~/.Xresources

Nachdem eine dieser Dateien geändert wurde, muß sie neu eingelesen werden, damit die Änderungen wirksam werden.

B.2 Resources

B.2.1 Beispiel Xtv–Datei

```
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 Hierarchie

```
<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
```

Anhang C

Dateiformate

C.1 Definition von mfile

Mfile ist eine Bibliothek, die von Stefan Esser als Diplomarbeit am IKP angefertigt wurde, für das systemunabhängige Lesen und Schreiben von n–dimensionalen Spektren ($n < 4$). Es stellt ein Dateiformat namens lc (line compressed) für komprimierte Spektren zur Verfügung. Der Kompressionsalgorithmus kodiert die Differenzen von benachbarten Kanälen in 4 Bits.

C.2 Dateiformate

Tabelle C.1 auf Seite 107 zählt alle in der Datei *mfile.h* definierten Dateiformate auf.

Elementformat	Kompressionsformat
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 Spectren Format
txt	ASCII Spectrum, Integer or Double
trixi	trixi save_matrix Format
le2t	triagonal LE2
le4t	triagonal LE4
he2t	triagonal HE2
he4t	triagonal HE4

Tabelle C.1: Dateiformate aus der Datei *mfile.h*.

Die Reihenfolge der Bytes in einem Datenwort wird durch seine Endianess beschrieben. Bei high (big) Endianess ist das most significant byte als erstes, während bei little Endianess das least significant byte als erstes geschrieben. Das Format eines Spektrums wird in der folgenden Weise angegeben:

<levels>.<lines>.<columns>.<elementformat>:[<version>]

wie zum Beispiel

1k.2k.4k.he4

was einen Kubus beschreibt, mit 1024 Ebenen, 2048 Zeilen und 4096 Spalten mit 4–byte Elementen (Gesamtgröße = 35 GB) kodiert im high Endian Format.

Erlaubte Abkürzungen bei der Angabe der Ebenen, Zeilen und Spalten sind:

k	=	kilo
m	=	mega

Anhang D

Die Fit– Untergrund– und Verteilungsfunktionen

Inhaltsangabe

D.1 Fitfunktionen

Tv verwendet zwei alternative Parameterisierungen für Peaks, die beide vergleichbare Resultate liefern. Die zweite hier aufgeführte reduziert die Korrelation zwischen Stufe und Tail, da der erf Stufe den asymptotischen Wert, verglichen mit dem arctan Stufe, schnell erreicht. Weiterhin kann die Stufen Breite beim erf Stufe auf 1,0 (in Einheiten von σ) festgehalten werden, was die Anzahl der freien Parameter um eins reduziert. Außerdem ist die Funktion mit dem erf Stufe analytisch integrierbar, während die Funktion mit erf Stufe numerisch integriert werden muß.

In beiden Funktionen wird das Volumen und nicht die Amplitude der Peaks gefittet. Das Volumen ist die interessierende Größe und so ist es nicht nptwendig die resultierende Funktion zu integrieren und den Fehler des erhaltenen Volumens abzuschätzen.

`tv>` **fit function peak definition continuous-exp-tail/arctan-step**

$$F(x) = BG(x) + \sum_{i=0}^{Peaknummer} PEAK_i(x))$$

BG(x): Untergrundfunktion siehe Kapitel D.2 auf Seite 111

Peakfunktion 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 des i–ten Peaks (Parameter)

V$_i$**:** Volumen des i–ten Peak (Parameter)

NORM$_i$**:** Numerisches INTEGRAL(GAUSM$_i$)

Modifizierte Gaussfunktion des i–ten Peaks

$$GAUSMi(dx) = \begin{cases} \exp(\frac{SL_i}{S_i^2} \cdot (dx + \frac{SL_i}{2})) & dx < SL_i \\ \exp(\frac{-dx^2}{2 \cdot S_i^2}) & SL_i < dx < SR_i \\ \exp(\frac{-SR_i}{S_i^2} \cdot (dx - \frac{SR_i}{2})) & SR_i < dx \end{cases}$$

dx: x - P_i

S_i: σ des Gaussanteils des i–ten Peaks (Parameter)

SL_i: $TL_i \cdot S_i^{EL_i}$

SR_i: $TR_i \cdot S_i^{ER_i}$

TL_i: Linker tail des i–ten Peaks (Parameter)

TR_i: Rechter tail des i–ten Peaks (Parameter)

EL_i: Exponent des σ–Gewichts von TL_i [0..2]

ER_i: Exponent des σ–Gewichts von TR_i [0..2]

Stufen Funktion des i–ten Peaks

$$STEP_i(dx) = SH_i \cdot (\frac{p_i}{2} + \arctan(\frac{SW_i \cdot dx}{S_i \cdot \sqrt{2}}))$$

SH_i: Stufen Höhe des i–ten Peaks (Parameter)

SW_i: Stufen Breite des i–ten Peaks (Parameter)

tv> fit function peak definition additive-tail/erf-step

$$F(x) = BG(x) + \sum_{i=0}^{Peaknummer} PEAK_i(x)$$

BG(x): Untergrundfunktion siehe Kapitel D.2 auf Seite 111

Peakfunktion des i-ten Peaks

$$PEAK_i(x) = \frac{V_i}{NORM_i} \cdot (((1 + TAIL_i(x - P_i)) \cdot GAUSS_i(x - P_i)) + STEP_i(x - P_i))$$

P_i: Position des i–ten Peaks (Parameter)

V_i: Volumen des i–ten Peaks (Parameter)

$NORM_i$: $S_i \cdot (\sqrt{(}2 \cdot \pi) + TL_i + TR_i)$

Gauss Funktion des i–ten Peaks

$$GAUSS_i(dx) = \exp(\frac{-dx^2}{2 \cdot S_i^2})$$

S_i: σ des Gaussanteils des i–ten Peaks (Parameter)

Zusätzlicher Tailfaktor des i–ten Peaks

$$TAIL_i(dx) = \begin{cases} \frac{TL_i \cdot (\frac{|dx|}{S_i})^E L_i}{FACFAC(EL_i)} & dx < 0 \\ \frac{TR_i \cdot (\frac{|dx|}{S_i})^E R_i}{FACFAC(ER_i)} & dx \geq 0 \end{cases}$$

TL_i: Linker Tail des i–ten Peaks (Parameter)

TR_i: Rechter Tail des i–ten Peaks (Parameter)

EL_i: Exponent des linken Tails [2..16]

ER_i: Exponent des rechten Tails [2..16]

zur Vereinfachungs des Integrals

$$FACFAC(n) = \begin{cases} (n-1)!! & n \in \{3, 5, 7, \ldots\} \\ (n-1)!! \cdot \sqrt{\frac{\pi}{2}} & n \in \{2, 4, 6, \ldots\} \end{cases}$$

Stufen Funktion des i–ten Peaks

$$STEP_i(dx) = \frac{1}{2} \cdot SH_i \cdot (1 - erf(\frac{dx}{SW_i \cdot S_i \cdot \sqrt{2}}))$$

SH_i: Stufen Höhe des i–ten Peaks (Parameter)

SW_i: Stufen Breite des i–ten Peaks (Parameter)

erf: $erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt$

D.2 Die Untergrundfunktionen

Man kann sich die Definition der Untergrundfunktionen anschauen, mit dem Kommmando:

`tv>` **fit function background definition polynom**
$BG(x) = \sum_{n=0}^{N} B_n \cdot (x - x_0)^n$

$\mathbf{B}_n$: Untergrund Koeffizient (Parameter)

$\mathbf{x}_0$: Offset für die numerische Optimierung während des Fits

`tv>` **fit function background definition exponential**
$BG(x) = \sum_{n=0}^{N} B_n \cdot (x - x_0)^n + FAC \cdot exp(-(x - x_0) \cdot EXP)$

$\mathbf{B}_n$: Untergrund Koeffizient (Parameter)

$\mathbf{x}_0$: Offset für die numerische Optimierung während des Fits

FAC: Faktor des exponentiellen Terms

EXP: Skalierung des exponentiellen Terms

D.3 Verteilungsfunktionen

Man kann sich die Definitionen der Verteilungsfunktionen mit folgenden Kommandos anschauen:

`tv>` **fit measure definition dy–chi–square**

Minimierung des χ^2 gewichtet mit den Fehlern der Daten (Maximierung der Gaussfunktion).

$$chi(f)^2 = \sum_{i=0}^{numdata} \left(\frac{f(X_i) - Y_i}{dY_i} \right)^2$$

f: Fitfunktion

$\mathbf{X}_i$ Kanal

$\mathbf{Y}_i$ Wert des Spektrums im Kanal X_i

$\mathbf{dY}_i$ Fehler von Y_i

`tv>` **fit measure definition y–chi–square**

Minimierung des χ^2 gewichtet mit den Daten (Maximierung der Gaussverteilung).

$$chi(f)^2 = \sum_{i=0}^{numdata} \frac{(f(X_i) - Y_i)^2}{|Y_i|}$$

f: Fitfunktion

$\mathbf{X}_i$ Kanal

$\mathbf{Y}_i$ Wert des Spektrums im Kanal X_i

`tv>` **fit measure definition poisson**

Maximierung der Poissonverteilung.

$$P(f) = \prod_{i=0}^{numdata} \frac{f(X_i)^{Y_i}}{Y_i!} \cdot \exp(-f(X_i))$$

f: Fitfunktion

$\mathbf{X}_i$ Kanal

$\mathbf{Y}_i$ Wert des Spektrums im Kanal X_i

Die tatsächlich maximierte und dargestellte Funktion sieht folgendermaßen aus:

$$
\begin{aligned}
G(f) &= \ln(P(f)) - C \\
&= \sum_{i=0}^{numdata} Y_i \cdot (\ln(f(X_i)) - f(X_i))
\end{aligned}
$$

C: $\sum_{i=0}^{numdata} - \ln(Y_i!)$

Anhang 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