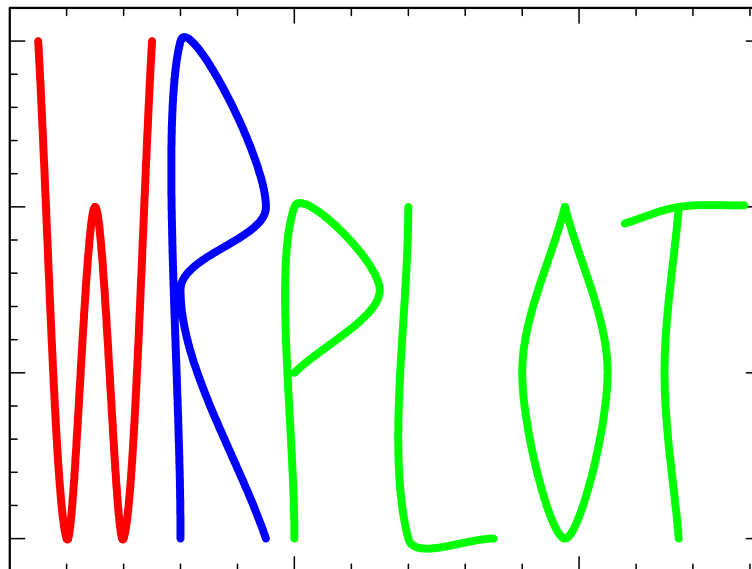


Filename of this manual: ~wrh/wrplot.dir/manwrplot.dir/wr\_info.ps  
Last update: 28. November 2014



## **Bedienungsanleitung**

WOLF-RAINER HAMANN

unter früherer Mitarbeit von Lars Koesterke und vielen anderen

# Inhaltsverzeichnis

<b>1</b>	<b>Programm installation and execution</b>	<b>3</b>
<b>2</b>	<b>Der Aufbau eines WRPLOT-Files</b>	<b>5</b>
<b>3</b>	<b>Die KASDEF-Befehle</b>	<b>7</b>
3.1	Vorbemerkungen . . . . .	7
3.2	Settings . . . . .	8
3.3	Instructions . . . . .	10
3.3.1	Ausgabe im Kommandofenster . . . . .	10
3.3.2	Variable . . . . .	10
3.3.3	IF-Konstruktionen . . . . .	12
3.3.4	DO-Loops . . . . .	12
3.3.5	GOTO . . . . .	12
3.3.6	Schreiben in ein File . . . . .	13
3.4	Plotcommands . . . . .	13
3.4.1	Wechsel von Strichstärke, Farbe oder Font . . . . .	13
3.4.2	Zeichenaktionen . . . . .	14
3.4.3	Schraffierte Flächen . . . . .	16
3.4.4	Ausgabe von Text . . . . .	16
3.4.5	Sonderzeichen . . . . .	19
3.4.6	LaTeX-Formelsatz . . . . .	20
3.4.7	Text-Attribute . . . . .	20
3.4.8	Linienidentifikation . . . . .	20
3.4.9	Einbinden von PostScript-Files . . . . .	22
3.4.10	Objekte aus LaTeX-Quellcode . . . . .	23
<b>4</b>	<b>Die Koordinaten-Box</b>	<b>25</b>
<b>5</b>	<b>Die Plotdaten</b>	<b>26</b>
5.1	Datensätze . . . . .	26
5.2	Kopfzeile und Plot-Attribute . . . . .	27
5.3	Datentabellen . . . . .	28
5.4	COMMAND-Zeilen . . . . .	29
5.5	COMMAND-Funktionen . . . . .	35
<b>6</b>	<b>Die Plot-Symbole</b>	<b>36</b>
<b>7</b>	<b>Erzeugen von Plot-Files mit FORTRAN-Programmen</b>	<b>39</b>
<b>8</b>	<b>Einbinden von Plots in Tex-Dokumente</b>	<b>40</b>
<b>A</b>	<b>Musterblatt Fonts</b>	<b>41</b>
<b>B</b>	<b>Musterblatt Farben</b>	<b>42</b>
	<b>Index</b>	<b>43</b>

# 1 Programm installation and execution

At the Workstation-Cluster `astro.physik.uni-potsdam.de`, the WRplot system is presently installed in the user directory of W.-R. Hamann: `/home/corona/wrh/wrplot.dir` for user under TrueUnix (OSF, Dec-Alpha machines), and under `/home/corona/wrh/linux-wrplot.dir` for use from Linux systems.

As the installation is not local to your machine and not in a standard path, each user must define a path variable, and load a couple of alias, by inserting the following two lines in his `.login` or `.cshrc` script. The following lines make sure that you get the correct version, depending on your host system:

```
if ( 'uname' == 'OSF1' ) then
    setenv PATH_WRPLOT /home/corona/wrh/wrplot.dir
else
    setenv PATH_WRPLOT /home/corona/wrh/linux-wrplot.dir/wrplot.dir
endif
source $PATH_WRPLOT/proc.dir/wrplot-aliases
```

**Note: Each WRplot user needs permission to use that program.** The WRplot manager must register the user in the file `$PATH_WRPLOT/admin.dir/wruser.dat`.

In order to install WRplot on your own computer, download via anonymous ftp:

```
ftp ftp.astro.physik.uni-potsdam.de
cd pub/wrhamann
get wrplot.README
get wrplot.tgz
```

and follow the README instructions

After having done so, the following commands are available:

## **manwrplot**

displays this manual with ghostview at your screen.

## **wrplot**

enters an interactive mode where the source file can be specified later.

## **wrplot filename**

opens the WRplot-window on your screen and shows the first plot contained in the source file *filename*. Entering the *return* key in that (active) window opens the next plot in that file or closes the window if the end is reached. Note that the WRplot-window is designed for a quick inspection of data; it has no nice (PostScript) fonts, and cannot display encapsulated PS files (but indicates the BoundingBox of EPSF files in the place they would appear).

## **wrplot filename +**

creates a PostScript file *wrplot.ps.1* with the plot of the source file. If *wrplot.ps.1* already exists, the trailing number is incremented. If the file contains more than one plot, each plot is written into an own *wrplot.ps*-file with incrementing trailing numbers. **psappend** combines all files *\*.ps* found in the present work directory into one

PS file *total.ps* with subsequent pages. The **dps** command removes all *wrplot.ps.\** files in the present work directory.

**wrps *filename* [*.plot*]**

means “WRplot-to-PS”. This command is designed for WRplot files containing only one plot. The extension of the WRplot-file must be *.plot* and can be omitted in the call. A PostScript file is generated which has the name of the source file, the extension being replaced by *.ps*.

**wrmult *filename nn* [+ ]**

is especially usefull for viewing quickly a WRplot file which contains a number of plots, each of about a “standard” size of 20cm × 15cm in landscape orientation. The first *nn* plots (*nn* = 2, 4, 6, 8, 10 or 12) of that file are scaled down and mounted onto *one* page. The trailing “+” option creates a *wrplot.ps.\** file.

**ttw *filename* [f77 as132 ]**

is a WRplot-based ascii-to-ps converter. The option “f77” supports the classical FORTRAN format by marking the columns 6 and 73. Option “as132” allows for long lines with 132 characters by producing landscape format. The output is written into the file *ttw.ps*. The command **dttw** removes all *ttw* debris.

**psappend**

concatenates all ps-files in the current directory into one file *total.ps* which then may contain many pages. This works only for sufficiently simple PostScript files, such as generated by WRplot.

**new *filename***

is equivalent to *wrps filename*; *psappend*

**newall**

performs *wrps* for all WRplot-Files in the current directory, and finally a *psappend*.

**ps2pdf\_portrait *filename* [.ps ]**

converts *filename.ps* into *filename.pdf* while enforcing portrait orientation, i.e. preventing any automatic rotation which might happen otherwise by the standard *ps2pdf*. Media format is A4.

**ps2pdf\_landscape *filename* [.ps ]**

same as above, but for landscape orientation.

**ps2pdf\_screen *filename* [.ps ]**

same as above, but for landscape orientation and media format ArchA, which is 4:3 and thus fitting to a XGA full-screen.

## 2 Der Aufbau eines WRPLOT-Files

Vorbemerkungen:

Ein WRPLOT-File kann in beliebiger Weise mehrere PLOTS und MULTILOTS enthalten.

Alle WRPLOT-Befehle sind stets in GROSSBUCHSTABEN zu schreiben; Kleinbuchstaben haben nur in Strings und Kommentaren etwas zu suchen; auch in Variablenamen können sie verwendet werden, wobei Groß/Kleinschreibung signifikant ist.

Kommentarzeilen beginnen mit einem \* und können überall stehen. Leerzeilen sind vermutlich auch fast überall möglich.

Parameter werden in der Regel nach Belieben getrennt durch: ein Trennzeichen (Komma, Gleichheitszeichen oder Doppelpunkt), ein oder mehrere Blanks, oder ein Trennzeichen und zusätzliche Blanks. TAB-Character wirken wie Blanks.

Zeilen können 132 Character lang sein. KASDEF-Zeilen können Fortsetzungszeilen haben (siehe Kap. 3).

Achtung, EMACS-Benutzer! WRplot ist ein FORTRAN-Programm. Auf einem formatierten File liest FORTRAN zeilenweise, wobei jede Zeile mit einem *carriage-return*-Zeichen (CR, Enter-Taste) abgeschlossen wird. Manche Editoren, insbesondere EMACS in seiner Standard-Einstellung, haben jedoch die Macke, die *letzte* Zeile des Files nicht mit einem CR abzuschließen, sondern direkt mit *end-of-file* (EOF). Diese letzte Zeile kann dann von FORTRAN aus nicht gelesen werden. Abhilfe bringt beim EMACS bzw. XEMACS der Eintrag einer Zeile

(setq require-final-newline t) ; always add a newline at EOF  
in das betreffende Konfigurationsfile .emacs bzw. .xemacs/custom.el. Andernfalls muss man halt immer darauf achten, am Ende der letzten Zeile ein CR (*enter*) einzugeben.

### MULTILOT START

Optional; alle nachfolgenden Plots bis zu dem Kommando MULTILOT END werden zu *einem* Plot zusammengefaßt.

### PAPERFORMAT <Format: string> [Keyword]

Optional; wenn nicht spezifiziert, gilt als Default A4Q.

Zulässige Formate sind: A4Q (Default), A4H, A3Q, A3H usw. bis A0; "H" bedeutet Hochkant (*Portrait*), "Q" = Quer (*Landscape*). Seit Programmversion 2.9.2002 gibt es auch das selbsterfundene Format SCREEN; dieses ist ein Landscape-Format im Achsenverhältnis 4:3 wie ein Bildschirm, wobei die kurze Achse A4 entspricht, aber die lange Achse etwas kürzer ist (28.03cm×21.02cm). Noch besser ist das Format SCREEN2; es hat auch ein Achsenverhaeltnis 4:3 und entspricht zudem dem Format ARCH A, das in pdf bekannt ist. Damit kann man mithilfe der Prozedur ps2pdf\_screen pdf-Files erzeugen, die sich im Fullscreen-Modus betrachten lassen.

Die optionalen Keywords BBNOROTATE (default) und BBROTATE legen fest, ob die Bounding-Box mitrotiert wird. Bei BBROTATE können z.B. Landscape-Plots in Latex hochkant eingebunden werden.

Das optionale Keyword EPS veranlasst, dass das erzeugte PostScript-File in seiner ersten Zeile als EPS-File gekennzeichnet wird, ohne allerdings am Ende das showpage zu unterdrücken (könnte man bei Bedarf auch leicht machen). Anlass dieses Features war die berichtete Schwierigkeit, WRplot-generierte PS-Files in "word"-Dokumente einzubinden.

**FORMATEFACTOR** <Skalenfaktor: real>

(Default=1., optional), skaliert den folgenden Plot bzw. alle folgenden Plots eines Multiplots um den angegebenen *Skalenfaktor*.

**PLOT :** <Plotname>

Beginn eines Plots. *Plotname* ist der Namen des Plots im interaktiven Betrieb (darf auch leer sein).

**KASDEF-Befehle** (siehe Kap. 3)

KASDEF-Befehle können auch noch später zwischen den Datensätzen eingestreut sein, werden aber *vor* dem Kasten und den Datensätzen ausgeführt. Durch KASDEF PAUSE (siehe unten) wird diese Ausführung unterbrochen und am Ende des Plots (nach Fertigstellung der Datensätze) fortgeführt. Mit KASDEF LATEBOX wird das Zeichnen des Kastens ans Ende verschoben.

**HEADER :** <String>

Obere Beschriftung des Plots. Danach folgen zwingend fünf weitere Zeilen zur Spezifikation der Achsen (siehe Kap. 4).

**N=...** [*Parameter*]

Mit dieser Zeile beginnt die Beschreibung eines Datensatzes (siehe Kapitel 5).

**END** oder ENDE: Ende des Plots

**MULTILOT END** oder MULTILOT ENDE steht hinter dem letzten Plot eines Multiplots, der mit MULTILOT START begonnen wurde.

## 3 Die KASDEF-Befehle

### 3.1 Vorbemerkungen

Die Bezeichnung KASDEF ist historisch; KASDEF-Befehle beginnen mit dem Wort KASDEF; moderner ist stattdessen die Verwendung eines Backslash (\). In Fehlermeldungen wie auch in dem nachfolgenden Text dieses Manuals wird z.T. weiterhin die alte Bezeichnung mit KASDEF verwendet.

KASDEF-Kommandos können **Fortsetzungszeilen** haben; diese beginnen mit `\ > <Fortsetzung ...>` und wirken, als würde die vorangegegangene KASDEF-Zeile mit dem auf den > folgenden Character fortgesetzt.

Bei (fast) allen KASDEF-Befehlen, bei denen eine Position in *units* angegeben wird, können an deren Stelle auch die vordefinierten Konstanten **XMIN**, **XMAX**, **YMIN**, **YMAX** sowie **PXMIN** usw. angegeben werden; XMIN usw. sind die Koordinaten der Box, PXMIN usw. sind die Koordinaten des Blattrandes. Vom Gebrauch dieser veralteten Möglichkeit wird aber abgeraten. Stattdessen können bei den KASDEF-Befehlen die Parameter durch *Variable* gesetzt werden:

KASDEF-Kommandos können **Variable** enthalten, die bei der Ausführung durch den aktuellen Inhalt der Variablen ersetzt werden. Variable (siehe Kap. 3.3.2) werden bei ihrer Benutzung mit einem vorgestellten \$ gekennzeichnet (wie in Unix) und sind grundsätzlich *Strings*, mit denen man aber auch rechnen kann, falls sie aus Zahlen bestehen. Das Kommando `\PREDEFINE-VARIABLES` (siehe Seite 10) erzeugt eine ganze Reihe von vordefinierten Variablen, wie z.B. \$XMIN. Auch bei der Ausgabe von Text wird jedes \$ als Beginn einer Variablen interpretiert und zu ersetzen versucht. Dollars zur Ausgabe müssen daher mit einem Backslash maskiert werden (\\$). *Ausnahme*: Solange in einem Plot keinerlei Variablen definiert worden sind, bleibt auch die Ersetzung abgeschaltet.

Außerdem ist es in vielen Fällen möglich (ausprobieren!), die wiederholte Angabe gleichbleibender Parameter durch das Codewort SAME zu vereinfachen. Beispiel:

```
\SYM 3. 4. 0. 0. 1. 4
```

```
\SYM SAME SAME 0. 0. 1. 8
```

zeichnet ein Quadrat und einen Kreis an dieselbe Stelle.

Numerische Kasdef-Parameter können meist an Ort und Stelle logarithmiert oder de-logarithmiert werden. Man schreibe z.B. LOG35000. (ohne Blank) für  $\log(35000)$  oder DEX5. für  $10^5$ . Arithmetische Ausdrücke an Stelle von Parametern sind noch nicht möglich, aber geplant.

#### Drei Arten von KASDEF-Kommandos

Der Ablauf von WRplot zerfällt in drei Phasen:

- Einlesen der Input-Files
- Manipulation der Daten (nur im *Interactive-Mode*)
- Ausgabe des Plots

WRplot kennt nun drei Arten von KASDEF-Kommandos, die sich vor allem hinsichtlich des Zeitpunkts unterscheiden, an dem sie ausgeführt werden.

- *Settings* werden beim Einlesen des Input-Files sofort zur Kenntnis genommen. Durch *Settings* werden bestimmte Voreinstellungen gesetzt, die für den Plot gelten und später nicht verändert werden können. Zu den *Settings* gehört auch das `\INCLUDE`-Kommando, durch das der Inhalt eines externen Files eingefügt wird.
- *Instructions* sind typischerweise derartige KASDEF-Kommandos, die *keine* Zeichen-Aktion durchführen, sondern z.B. Operationen mit Variablen. Instructions, die in einem INSTRUCTION EXPORT-Block stehen, werden schon in der Einlese-Phase ausgeführt. Die übrigen KASDEF-Kommandos werden in einem Puffer gespeichert; ihre Ausführung erfolgt erst zusammen mit den *Plotcommands* in der Plot-Phase.
- *Plotcommands* veranlassen die Ausgabe von Schrift oder Graphik, oder beziehen sich auf derartige Aktionen (Beispiel: Definition der aktuellen Farbe durch `\COLOR`). Da *Plotcommands* nicht in der Einlese-Phase ausgeführt werden können dürfen sie nicht in einem INSTRUCTION EXPORT-Block vorkommen. Weil dies in manchen älteren Skripts missachtet wurde, gibt es den *compatibility mode*: Wird in einem INSTRUCTION EXPORT-Block ein dort eigentlich unzulässiges Plotcommand entdeckt, werden alle Kommandos der INSTRUCTION EXPORT-Blöcke in der Plotphase ein zweites Mal ausgeführt, mit entsprechenden nicht-fatalen Fehlermeldungen. Man sollte dies aber vermeiden, weil die doppelte Ausführung unbeabsichtigte Folgen haben kann.

*Settings* und *Instructions* dürfen nur in der Präambel stehen, d.h. vor dem Block, der die Datenbox beschreibt (beginnend mit `HEADER ...`). *Plotcommands* dürfen dagegen auch noch weiter unten zwischen den Datenblöcken eingestreut sein.

## 3.2 Settings

Folgende Befehle nehmen eine Voreinstellung vor und werden vorab ausgeführt. Diese KASDEFs **müssen** vor der `HEADER`-Zeile stehen!

### `\INTERACTIVE`

enters the interactive mode; the same mode can be reached by the interactive menus, when calling `wrplot` without filename. Clearly, it works only with the X11 window (i.e. not by the `wrps` call nor with trailing "+" option).

### `\PENDEF <integer>`

Definiert den Default-Wert für die Strichstärke der Datensätze.

### `\DEFAULTCOLOUR <integer>`

Definiert den Default-Wert für die Farbe der Datensätze. Der Default ist schwarz.

### `\OFS <xoffset:cm> <yoffset:cm>`

Koordinaten der linken unteren Ecke des Plotkastens relativ zur Papierecke Default (4., 3.). Wird benötigt, um bei mehreren Plots auf einer Seite (MULTILOT) die einzelnen Plots zu verschieben.

### `\SKL <Skalenfaktor>`

Skaliert den folgenden Plot um den *Skalenfaktor*. Wird benötigt, um bei mehreren Plots auf einer Seite (MULTILOT) die einzelnen Plots zu skalieren. (Default = 1.)



### **\INBOX**

Bewirkt, daß Datensätze nur innerhalb des eigentlichen Plot-kastens geplottet und am Rand korrekt abgeschnitten werden. Die Option wirkt nicht auf KASDEFs und auf Datensatz-Darstellungen mit diskreten Symbolen.

### **\NOBOX**

Der Plotkasten inklusive Tickmarken und deren Beschriftung wird unterdrückt. Dies gilt nicht für den Header und die Beschriftungen der Achsen.

### **\LATEBOX**

Der Plotkasten inklusive Tickmarken und deren Beschriftung wird erst nach den Datensätzen gezeichnet.

### **\LETTERSIZ** *<size: real>*

Größe der Charakter für die Beschriftung der Box (Marken, Achsen, Header).  
Default: 0.4

### **\TICKSIZE** *<size: real>*

Größe der Markierungen (Ticks) an der Box; per Default ist deren Größe harmonisch auf LETTERSIZE abgestimmt.

### **\SET\_NDATMAX** *n*

changes the maximum number of data pairs in a dataset. The Default is NDATMAX = 100000. The Maximum Value allowed is 10 times the default. Increasing NDATMAX leads to a corresponding decrease of the maximum number of datasets, NSETMAX.

### **\SET\_NSETMAX** *n*

changes the maximum number of datasets. The Default is NSETMAX = 100. The Maximum Value allowed is 10 times the default. Increasing NSETMAX leads to a corresponding decrease of the maximum number of data pairs in a dataset, NDATMAX.

### **\INCLUDE** *<filename> [keyword]*

übernimmt an dieser Stelle alle KASDEF-Befehle aus dem angegebenen File. Anderweitige Zeilen werden ignoriert. Bei Angabe eines *keywords* beginnt die Suche in der Zeile nach dem angegebenen Keyword und endet bei einer END-Zeile.

### **\INSTRUCTION EXPORT**

bis

### **\INSTRUCTION NO-EXPORT**

Die in einem solchen Block stehenden *Instructions* werden bereits in der Einlese-Phase ausgeführt. Dies ist insbesondere nötig, wenn hier Variablen gesetzt werden, die beim nachfolgenden Einlesen der Datensätze benutzt werden sollen.

## 3.3 Instructions

### 3.3.1 Ausgabe im Kommandofenster

**\ECHO** *string*

Ausgabe des Strings im Kommandofenster

### 3.3.2 Variable

Variablen können in WRPLOT in einer ähnlichen Syntax wie in UNIX zugewiesen und benutzt werden. Bei der Verwendung muß vor der Variable ein \$ stehen, bei der Zuweisung nicht. Klein- und Großschreibung sowohl des Variablennamens, als auch des Inhalts sind signifikant. Die Länge, sowohl des Namens als auch des Inhalts, ist 132 Character. Das Ende des Variablennamens wird durch ein Blank oder ein anderes Trennzeichen aus der Menge `,=:+-*/^{}()"$` markiert; diese Zeichen können also nicht Bestandteil eines Variablennamens sein. Diese Regel wird aus Gründen der Aufwärtskompatibilität dahingehend abgemildert, dass (bis auf Weiteres?) die Rechenzeichen `+*/` in Variablennamen verwendet werden dürfen, solange diese Variablen nicht in `\CALC` aufgerufen werden. Um einen Variablennamen von direkt dahinter stehendem Text abzutrennen, kann man den Namen in Gänsefüßchen setzen (z.B.: `"$Mv"mag`); letztere werden dann nur als Trennzeichen interpretiert und verschwinden beim Einsetzen des Variableninhalts.

Eckige Klammern in Variablennamen sind zulässig und können verwendet werden, um *indizierte Variable* zu emulieren. In eckigen Klammern auftretende Variable (mit Dollar!) werden zu diesem Zweck von WRplot zuerst durch ihren Wert ersetzt, dann erst kommen die restlichen mit dem Dollar beginnenden Variablennamen an die Reihe. Beispiel:

```
\CALC x2[$i] = $x[$i]**2
```

In den eckigen Klammern dürfen keine Blanks verbleiben. Mehrfach indizierte Variable sind nicht vorgesehen.

**\VAR** *A=...*

setzt eine Variable auf den angegebenen Inhalt.

**\PREDEFINE-VARIABLES**

setzt eine ganze Reihe von Variablen auf einen vorgegebenen Inhalt. Darunter sind:

- die Min- und Max-Werte der aktuellen Koordinatenbox (in den gewählten Units), sowie die jeweilige Mitte (XMIN, XMAX, XMID, YMIN, YMAX, YMID)
- die Skalenfaktoren zum Umrechnen von Units in cm (XSCALE, YSCALE)
- die Koordinaten (in Units) der Papierränder sowie der Mitte des Blattes (PXMIN, PXMAX, PXCENTER, PYMIN, PYMAX, PYCENTER)
- Filename, Datum und Uhrzeit (FILENAME, DATE, TIME). Achtung, nur diese drei letztgenannten Variablen stehen sofort zur Verfügung; alle anderen werden erst definiert, nachdem die Koordinaten-Box spezifiziert wurde.

**\GETTIME**

schreibt in die Variable TIME die aktuelle Uhrzeit im Format hh:mm:ss

### **\CALC** *A = expression*

Ausrechnen arithmetischer Ausdrücke. Die Syntax ist sehr flexibel und robust und entspricht etwa den arithmetischen Ausdrücken in FORTRAN. Erlaubt sind die Rechenoperationen  $+ - * /$ , beliebig geschachtelte Klammern, Exponentiation mit  $**$  oder  $^$  und Funktionsaufrufe. Konstanten können im Integer, F- oder auch E-Format auftreten. Variablen müssen natürlich mit einem vorangestellten Dollar aufgerufen werden. Blanks sind nicht signifikant.

Folgende Funktionen sind bekannt: sin, cos, atan, dex, log, exp, ln, sqrt, abs, rand, int, nint. Die Funktionsnamen können alternativ auch in Großbuchstaben geschrieben werden (aber nicht gemischt). *Achtung:* Im Unterschied zu FORTRAN, wo bei den Winkelfunktionen (sin, cos) das Argument in Bogenmaß ist, ist es in WRplot in Winkelgrad. Analog ist auch das Ergebnis von atan in Grad. Bei rand (Zufalls-generator) ist das Argument dummy.

**\CALC.DEBUG** oder **\CALC.DEBUG ON** schaltet einen Debug-Output ein (nach Standard-Output), an dem man das schrittweise Auswerten der arithmetischen Ausdrücke durch **\CALC** mit den eingesetzten Variablen im Detail verfolgen kann.

**\CALC.DEBUG OFF** schaltet den Debug-Output wieder aus (Default).

### **\EXPR** *A = \$var1 OPERATOR \$var2*

Dieses Kommando wird nur noch für das Aneinanderfügen von Strings empfohlen (z.B. **\EXPR A = \$var1 // \$var2**). Als Operator sind zwar auch die Grundrechenarten erlaubt, wenn die Argumente Zahlen sind, jedoch gibt es zum arithmetischen Rechnen jetzt das Kommando **CALC** (siehe oben).

**\VAR-LIST** Gibt eine Liste der Variablen auf dem Bildschirm aus.

### **\FORMAT** *formatstring \$var*

ermöglicht das Formatieren einer Variablen, die eine Gleitkomma-Zahl enthält, z.B. um nach arithmetischen Operationen auf eine gewünschte Stellenzahl zu runden. *formatstring* ist das Format nach FORTRAN-Syntax, inklusive der Klammern, also z.B. (F5.2).

### **\FORMATI** *formatstring \$var*

ermöglicht das Formatieren einer Integer-Variablen, um mit einer bestimmten Stellenzahl rechtsbündig zu schreiben. *formatstring* ist das Format nach FORTRAN-Syntax, inklusive der Klammern, also z.B. (I2).

### **\CUTVAR** *var n m*

ermöglicht das cutten von Strings. Der Inhalt der Variablen *var* wird auf den Substring (*n:m*) reduziert. Dabei kann das erste Argument leer sein, das zweite Argument leer sein oder fehlen (mit offensichtlichen Defaults). Optional können *n* oder *m* auch negativ sein; in diesem Fall zählt die Position vom Ende des Strings her. Beispiel: **\CUTVAR \$var 1:-5** schneidet von dem in der Variablen *var* enthaltenen String die letzten 5 Zeichen ab. Übrigens darf der \$ vor dem Variablennamen hier auch fehlen.

### 3.3.3 IF-Konstruktionen

Wie in FORTRAN können Konstruktionen aus beliebig geschachtelten IF, ELSEIF, ELSE und ENDIF gebildet werden. Im Unterschied zur FORTRAN-Syntax darf der logische Ausdruck jedoch nur aus einem einfachen Vergleich bestehen; es dürfen keine Klammern um den logischen Ausdruck stehen, und es erübrigt sich das THEN. Die Vergleichsoperatoren .EQ. und .NE. vergleichen die Operanden als Character-Strings; die übrigen Operatoren (.LT., .LE., .GT., .GE.) werten die Operanden als Zahlen aus und vergleichen diese.

Der Befehl ELSEIF muss als ein Wort geschrieben werden.

```
\IF $k .LT. $l
  \ECHO branch 0
\ELSE m .EQ. M
  \IF a .EQ. b
    \ECHO branch1
  \ELSEIF a .EQ. a
    \ECHO branch 2
  \ELSE
    \ECHO branch 3
  \ENDIF
  \ECHO branch 4
\ENDIF
```

### 3.3.4 DO-Loops

Auf der Ebene der KASDEF-Befehle können beliebig geschachtelte Do-Loops programmiert werden:

```
\DO labelname I=start, end, increment
  \...
\LABEL labelname
```

Das LABEL-Statement markiert also das Ende der Loop wie ein CONTINUE im klassischen FORTRAN. Geschachtelte Loops dürfen auf demselben Label enden. LABEL-Statements ohne korrespondierendes DO stören nicht. Es findet keine Kontrolle auf doppelte Vergabe von *labelname* statt; eine DO-Loop endet beim nächsten passenden LABEL. LABEL-Statements – auch solche am Loop-Ende – können durch GOTOs angesprungen werden.

### 3.3.5 GOTO

**\GOTO *labelname***

veranlasst einen unbedingten Sprung zu den Zeile `\LABEL labelname`. Es findet keine Kontrolle auf doppelte Vergabe von *labelname* statt; GOTO durchsucht die KASDEF-Zeilen vom Beginn.

### 3.3.6 Schreiben in ein File

Man kann mit KASDEF-Kommandos ein File öffnen und darauf schreiben. Die Kommandos heissen:

`\OPEN filename`

`\WRITE ....`

`\CLOSE`

Wenn man kein OPEN gibt, heisst das File `fort.50`. Wenn man das CLOSE vergisst, kann das File unvollständig sein oder sich bei mehrmaliger Ausführung unkontrolliert verlängern.

Eine interessante Anwendung besteht darin, sich mit WRITE eine Funktion zu tabellieren, die man gleich in dem selben Plot als Datensatz wieder einliest. Dazu muss man nur dieses File mit Hilfe von `\INSTRUCTION EXPORT` vorneweg erzeugen.

Analog kann man aus dem aktuell geöffneten File (bzw. `fort.50`) auch lesen, wobei die eingelesene Zeile in die Variable *varname* gespeichert wird:

`\READ varname`

Ist man am Ende des Files angelangt, bekommt *varname* den Inhalt E-O-F

## 3.4 Plotcommands

Dies sind Kommandos, die die Ausgabe von Schrift oder Graphik veranlassen oder sich auf solche Ausgaben beziehen.

**`\PAUSE`**

verschiebt die Ausführung der restlichen KASDEF-Befehle an das Ende des Plots, nachdem die Datensätze geplottet wurden.

### 3.4.1 Wechsel von Strichstärke, Farbe oder Font

**`\PEN <integer>`**

Definiert die Strichstärke für nachfolgende Zeichenoperationen (Default bei Plotbeginn: `PEN=1`).

**`\FONT <font>`**

mit *font* = WRPLOT (default), TIMES, HELVETICA (kurz HELVET), COURIER oder ZAPF. Definiert den Schriftfont ab dieser Stelle. Zum Aussehen der Fonts siehe das Musterblatt am Ende dieser Anleitung.

**`\COLOR=i`**

definiert die Zeichenfarbe *i* (*i* = 0 ... 9) für nachfolgende Zeichenoperationen (Default bei Plotbeginn: `COLOR=1`).

**`\DEFINECOLOR=i r g b`**

Wenn einem die voreingestellten Farben nicht gefallen oder ausreichen, kann man sie mit diesem Befehl umdefinieren. Die neue Definition gilt ab dieser Stelle (also für die nachfolgenden KASDEF-Optionen und den Rest des Plots). Die Werte *r*, *g* und *b* sind die Intensitäten der Farben rot, grün und blau; 0 steht für Null-Intensität (schwarz), 1 für volle Intensität der betreffenden Farbe bei additiver Farbmischung.

Die Farben 0 bis 9 sind vor-definiert gemäß folgender Tabelle (siehe auch Abbildung auf der letzten Seite dieser Anleitung). Für Beamer-Präsentation gibt es in `wrplot.dir/screen.kasdefs` einen Satz von vordefinierten Farben mit dunklem Hintergrund (siehe `wrplot.dir/colortest-screen.ps`).

i	Farbe	0	1	2	3	4	5	6	7	8	9
i	Farbe	weiss	schwarz	rot	grün	dunkelblau	gelb	pink	hellblau	hellgrün	blau
r	rot	1.	0.	1.	0.	0.	1.	1.	0.	.5	0.
g	grün	1.	0.	0.	1.	0.	1.	0.	1.	1.	.5
b	blau	1.	0.	0.	0.	1.	0.	1.	1.	0.	1.

### **\RESETCOLOR**

Mit diesem Befehl kann man, als Gipfel des Luxus, alle voreingestellten Farbwerte wieder restaurieren.

### **\LIMITCOLOR = nr ng nb**

Dieses Kommando begrenzt die Zahl der Farben, die man mit `DEFINECOLOR` verbrauchen kann, indem der RGB-Farbwürfel auf nr, ng und nb Stufen vergrößert wird. Diese Option dient der Anpassung an die Grafikkarte und wirkt natürlich nur auf das X-Fenster, nicht auf generierte PS-Files.

### **\NCOLORSTEP = nstep**

Dieses Kommando wirkt nur bei Verwendung von `SYMBOL=40` (Falschfarbendarstellung) und begrenzt die Zahl der Farbstufen, die *pro Farbtabelle* vergeben werden kann. Diese Option dient der Anpassung an die Grafikkarte und wirkt natürlich nur auf das X-Fenster, nicht auf generierte PS-Files.

## **3.4.2 Zeichenaktionen**

### **\LIN <x1:cm> <y1:cm> <x2:cm> <y2:cm>**

Zeichnet eine Linie von  $x_1$ ,  $y_1$  nach  $x_2$ ,  $y_2$ .

### **\LINREL <x:units> <y:units> < $\Delta x$ :cm> < $\Delta y$ :cm> [< $x_{\text{offset}}$ :cm> < $y_{\text{offset}}$ :cm> SYMBOL=*i* SIZE=*x*]**

Zeichnet eine Linie vom Ausgangspunkt  $(x, y)$  der Länge  $(\Delta x, \Delta y)$ . Das Parameter-Paar  $x_{\text{offset}}$   $y_{\text{offset}}$  ist optional. Dahinter können auch noch per Keyword `SYMBOL` (Default=5) und `SIZE` (Default=0.5) angegeben werden, um die entsprechenden Plotsymbole (insbesondere gestrichelte Linien) zu zeichnen.

### **\LINUN <x1:units> <y1:units> <x2:units> <y2:units> < $x_{\text{offset}}$ :cm> < $y_{\text{offset}}$ :cm> [<SIZE=Size> <SYMBOL=Symbol>]**

Zeichnet eine Linie von  $x_1$ ,  $y_1$  nach  $x_2$ ,  $y_2$ , wobei die Linie um  $x_{\text{offset}}$ ,  $y_{\text{offset}}$  verschoben wird. Sind die Keywords `SIZE` (Default=0.5) und `SYMBOL` (default=5) angegeben, so können beliebige Symbole, insbesondere gestrichelte Linien gezeichnet werden.

### **\ARR <x:units> <y:units> <Länge:cm> <Winkel:Grad> <Größe der Pfeilspitze:cm> < $x_{\text{offset}}$ :cm> < $y_{\text{offset}}$ :cm> <Modus:0,1> [<FILLED> <BAR>]**

Zeichnet einen Pfeil (Koordinaten-Punkt ist der Anfang des Pfeiles). Bei Modus=1 ist  $y_{\text{offset}}$  abgeschaltet und der Offset erfolgt in Pfeilrichtung. Die Verschiebung ist dann  $x_{\text{offset}}$  in cm. Ist das Keyword 'FILLED' gesetzt, dann wird der Pfeil ausgefüllt. Das Keyword 'BAR' bewirkt, daß ein Fuß an den Pfeilanfang gezeichnet wird. ACHTUNG : Dieser Befehl wurde mittlerweile stark verbessert, aber der Autor ist zu faul, dies im Moment genauer zu beschreiben!

MODUS=4 ist besonders nützlich, wenn man einen Pfeil von Punkt 1 nach Punkt 2 zeichnen will. In diesem Modus bedeuten die ersten vier Parameter die Koordinaten von Anfangs- und Endpunkt (in Units). Ferner kann man mit SHRINK=x.x einen Faktor angeben, um den der Pfeil verkürzt wird, und somit einen eleganten Zwischenraum zu den zu verbindenden Punkten zu lassen.

**\ARC** <x:units> <y:units> <  $x_{\text{offset}}$  :cm> <  $y_{\text{offset}}$  :cm> <Radius:cm> <  $\alpha$  > <  $\Delta\alpha$  > [FILLED]

Zeichnet einen Kreisbogen mit den angegebenen Koordinaten und Radius, beginnend beim Winkel  $\alpha$  mit Bogenlänge  $\Delta\alpha$ . Mit der Option FILLED wird der Kreis bzw. Sektor ausgefüllt.

**\ELLI** <x:units> <y:units> < $x_{\text{offset}}$  :cm> < $y_{\text{offset}}$  :cm> <a:units> <b:units> <  $\phi$  > <  $\alpha$  > <  $\Delta\alpha$  > [FILLED] [SYMBOL=i] [SIZE=x]

Zeichnet einen Ausschnitt einer Ellipse mit den Halbachsen a und b, wobei die große Achse um den Winkel  $\phi$  gedreht ist. Der Ausschnitt beginnt beim Startwinkel  $\alpha$  und hat die Bogenlänge  $\Delta\alpha$  - diese Winkel beziehen sich auf den Kreis mit der großen Halbachse vor der Stauchung zur Ellipse. Mit der Option FILLED wird die Ellipse bzw. der Sektor ausgefüllt. Mit SYMBOL=i und SIZE=x können verschiedene Zeichensymbole (z.B. gestrichelt) gewählt werden.

**\RECT** <x1:units> <y1:units> <x2:units> <y2:units> < $x_{\text{offset}}$  :cm> < $y_{\text{offset}}$  :cm> [FILLED]

Zeichnet ein Rechteck mit den engegebenen Eckpunkten, ggfs. alle um den Offset verschoben.

**\RECTLUN** <x:units> <y:units> < $x_{\text{offset}}$  :cm> < $y_{\text{offset}}$  :cm> <Breite> <Höhe> [FILLED] [ROTATE= $\alpha$ ]

Zeichnet ein Rechteck. Die Koordinaten beziehen sich je nach Prefix von  $x_{\text{offset}}$  bzw.  $y_{\text{offset}}$  auf den linken, rechten, oberen, unteren Rand oder die Mitten der Seiten. Breite und Höhe sind defaultmäßig in cm, jedoch bei nachgestelltem U in den jeweiligen x- bzw. y-Units - alles wie üblich. ROTATE dreht das Rechteck um  $\alpha$ , wobei der Drehpunkt der durch die Prefixe bezeichnete Referenzpunkt ist.

x[units] und y[units] sind bei Verwendung von SAME mit anderen, vergleichbaren Kommandos gemeinsam.  $x_{\text{offset}}$  bzw.  $y_{\text{offset}}$  werden mit SAME samt ihres Prefix von der LUN-Umgebung uebernommen, d.h. es gilt die aktuelle Schreibposition, in die z.B. NEXTLUN die nächste Zeile plazieren würde. Damit eignet sich RECTLUN besonders, um in Textfolien z.B. einen Merksatz einzukasteln. Umgekehrt beeinflussen bei RECTLUN angegebene  $x_{\text{offset}}$  bzw.  $y_{\text{offset}}$  nicht die aktuelle Schreibposition.

**\SYM** <x:units> <y:units> < $x_{\text{offset}}$  :cm> < $y_{\text{offset}}$  :cm> <Size:real> <Symbol:integer> [<CFILL=i>]

Zeichnet das Symbol in der Größe *Size* an den Ort  $x + x_{\text{offset}}$ ,  $y + y_{\text{offset}}$ . *Symbol* ist die Nummer des Plot-Symbols (Abb. 2 – nur diskrete Plotsymbole!). CFILL=i bewirkt ein Füllen des Symbols mit der Farbe i und eine Umrandung in der aktuellen Zeichenfarbe.

**\BAR** *<x:units> <y:units> <x<sub>offset</sub>:cm> <y<sub>offset</sub>:cm>*

Zeichnet einen Fehlerbalken an den spezifizierten Ort.

**\BARLEN** *<Δ<sub>y,up</sub>> <Δ<sub>y,down</sub>> <Δ<sub>x,up</sub>> <Δ<sub>x,down</sub>>*

Gibt die Länge des Fehlerbalkens in jeder Richtung an. Default: 0.,0.,0.,0.

**\BARPAR** *<Space:cm> <edge:cm>* Spezifiziert den Platz, der für das Symbol ausgespart wird, und gibt die Länge der Kanten an. Default: 0.,0.

### 3.4.3 Schraffierte Flächen

Bei allen flächenhaften Objekten, die mit dem Keyword FILLED farbig ausgefüllt werden können, ist alternativ auch das Füllen mit einer Schraffur möglich. Dazu gibt man statt FILLED das Keyword HATCHED an. – Das Schraffieren ist nur im PS-File wirksam. Im X11-Fenster von WRplot erscheinen die Flächen vollständig ausgefüllt.

Die Schraffur lässt sich detailliert beeinflussen:

HTYPE = *t* spezifiziert das Muster der Schraffur, wobei für *t* folgende Werte möglich sind: D für diagonal (default), V für vertikal, H für horizontal, K für kariert (horizontal und vertikal), C für *cross-hatched*. Bei Angabe von HTYPE ist das Keyword HATCHED redundant.

Ferner kann das Keyword HATCHED bzw. HTYPE=*t* noch von weiteren Parametern in dem gleichen Kommando begleitet werden:

HW = *x.x* setzt die Linienstärke der Schraffur auf die Stärke *x.x* (in typografischen Punkten = 1/72 Inch, d.h. wie beim PEN-Kommando, aber mit der Möglichkeit von Dezimalbrüchen).

HSEP = *x.x* setzt in analoder Weise den Abstand zwischen den Linien der Schraffur.

### 3.4.4 Ausgabe von Text

**\LAB** *<x:cm> <y:cm> <Size:real> <Text:string>* Schreibt den String in der Größe *Size* am Ort *x*, *y*.

**\LUN** *<x:units> <y:units> <x<sub>offset</sub>:cm> <y<sub>offset</sub>:cm> <Size:real> <Text:string>* wie LAB, nur Ort =  $x + x_{\text{offset}}$ ,  $y + y_{\text{offset}}$ . Bei *x<sub>offset</sub>* kann ein L (default), R oder M vorangestellt werden, dann bezieht sich die angegebene Position auf das linke Ende, das rechte Ende oder die Mitte des Strings. Entsprechend kann bei *y<sub>offset</sub>* ein D (default), U oder M vorangestellt werden, dass ist die angegebene Position die Grundlinie ("Down") oder die obere ("Up") Begrenzung der Charakter-Box bzw. deren Mitte. Diese Optionen erleichtern sehr das Positionieren von Beschriftung!

**\LUNA** *<x:units> <y:units> <x<sub>offset</sub>:cm> <y<sub>offset</sub>:cm> <Size:real> <angel:real> <Text:string>* wie LUN, nur durch die zusätzliche Winkelangabe wird die Schrift gedreht. Wie bei \LUN legen die Präfixe fest, auf welchen Punk (Ecke, Mitte) des Strings sich die Koordinaten beziehen. Dieser Punkt ist dann auch der Drehpunkt.



**\LINUNLAB**  $\langle x_1:units \rangle \langle y_1:units \rangle \langle x_2:units \rangle \langle y_2:units \rangle \langle x_{offset}:cm \rangle \langle y_{offset}:cm \rangle$   
 $\langle S_{offcenter}:cm \rangle \langle Size \rangle \langle String \rangle$  Zeichnet eine Linie von  $(x_1, y_1)$  nach  $(x_2, y_2)$  und setzt in dabei um  $S_{offcenter}$  versetzt in die Mitte den *String* ein.

**\LUNINC**  $\langle x:units \rangle \langle y:units \rangle \langle x_{offset}:cm \rangle \langle y_{offset}:cm \rangle \langle Size:real \rangle \langle File:string \rangle$   
 $\langle Prefix:string \rangle \langle Suchstring:string \rangle$  Sucht aus dem FILE die Zeile, die mit dem SUCHSTRING beginnt heraus und gibt diese, erweitert um das PREFIX, an der angegebenen Stelle aus.

**\NEWSIZELUN**  $\langle Size:real \rangle$  Verändert die Schriftgröße der LUN-Befehle auf den angegebenen Wert.

**\NEXTLUN**  $\langle text \rangle$   
ermöglicht das leichte Schreiben von mehreren Textzeilen. Die Parameter vom vorangegangenen LUN-Kommando werden beibehalten, jedoch wird der Y-Offset um einen Lineskip verringert.

**\SAMELUN**  $\langle text \rangle$   
arbeitet wie NEXTLUN, schreibt aber in die gleiche Zeile. Die Parameter vom vorangegangenen LUN-Kommando werden beibehalten.

**\LINESKIP**  $\langle lineskip \rangle$   
setzt den von NEXTLUN benutzten Lineskip (real, in cm) auf den angegebenen Wert (Default:  $2 * SIZE$ ). Bei nachgestelltem S bzw. U (z.B.  $0.5S$ ) wird der angegebene Wert in Einheiten der Symbolsize (S) bzw. in Units (U) interpretiert.

**\TEXTLINESKIP**  $\langle lineskip \rangle$   
setzt den für Fließtext benutzten Lineskip (real, in cm) auf den angegebenen Wert. Bei nachgestelltem S bzw. U (z.B.  $0.5S$ ) wird der angegebene Wert in Einheiten der Symbolsize (S) bzw. in Units (U) interpretiert. Default ist der normale LINESKIP.

**\MEDSKIP**  
erzeugt einen Zeilenvorschub um die Hälfte des in LINESKIP definierten Wertes.

**\TABLUN**  $\langle tab \rangle$   
rückt der linken Anfangspunkt der folgenden mit NEXTLUN geschriebenen Zeilen um  $tab$  (real, in cm) ein oder aus.  $tab$  kann auch einen Prefix tragen, d.h. durch Voranstellen von L, R oder M wird der aktuelle Prefix entsprechend modifiziert.

**\TEXT** [ $\langle x:units \rangle \langle y:units \rangle \langle \Delta x : cm \rangle \langle \Delta y : cm \rangle \langle Size:real \rangle$ ]  
Die nachfolgenden Zeilen sind Fließtext bis zu der Zeile \ENDTEXT. Fehlen die Parameter, wird der Textblock wie NEXTLUN automatisch positioniert. *Silbentrennung* erfolgt innerhalb von Fließtext vorerst nur an Trennstellen, die durch “\” (wie bei Tex) von Hand vorgegeben sind. Man beachte die Möglichkeit, einen abweichenden Zeilenabstand für Fließtext zu setzen (siehe \TEXTLINESKIP).

**\RIGHTMARGIN**  $\langle x:cm \rangle$  Bestimmt die Lage des rechten Randes für Fließtext.

**\BLOCK** oder **BLOCK ON**  
Schaltet Fließtext auf Blocksatz (Anfangsstellung: ausgeschaltet)

**\BLOCK OFF** Schaltet Blocksatz wieder aus

**\ITEMIZE** *size string*

wirkt auf alle nachfolgenden Zeilen (mit LUN bzw. NEXTLUN geschrieben) und Fließtext-Absätze (mit TEXT geschrieben). Der Text wird eingerückt, und mit dem *string* als Item-Symbol markiert (Beispiel für den *string*: &2\o macht einen roten Kreis). *size* gibt die Schriftgröße für das Item-Symbol (auch SAME möglich).

**\ITEMIZE OFF** verläßt die Itemize-Umgebung

**\ITEMSEP** *x.x*

erzeugt vor jedem item einen zusätzlichen Zeilenabstand von *x.x* cm. Bei einem nachgestellten S ist *x.x* in Einheiten der jeweiligen Schriftgröße.

**\BGRLUN** [*<OFF>*] [*<ON>*] [*<RESET>*] [*<COLOR=...>*] *<L=...; R=...; U=...; D=...>*

Wenn diese Option eingeschaltet ist, wird jede Schrift, die mit \LUN oder \LUNA ausgegeben wird, mit farbigem (default: weißem) Hintergrund hinterlegt. Um wieviel die Box größer ist als die Schrift wird in Einheiten von *SIZE* angegeben. Durch das Keyword *RESET* wird auf die Defaultgröße (0.5\**SIZE* an jeder Seite) zurückgesetzt. Die Farbnummer kann optional *negativ* angegeben werden: dann wird nur die Begrenzung der Fläche gezeichnet ohne Ausfüllen. Das Ausfüllen kann auch mit einer Schraffur erfolgen (Keyword und Parameter siehe Abschnitt 3.4.3).

Seit Programmversion vom 2.9.2002 wirkt BGRLUN in analoger Weise auch auf mit EPSF eingebundene Objekte. Damit kann man also z.B. in Latex geschriebene Formeln hinterlegen, oder eingebundene Grafiken. Die Hinterlegung gehört *nicht* zum eingebundenen Objekt, d.h. dessen BoundingBox und Positionierung wird davon nicht verändert.

**\TABON** *<Anordnung>* [*<Prefix>*]

startet die Tabellenumgebung. Die Anordnung der Spalten erfolgt ähnlich wie in Tex. Jeder Spalte wird ein Buchstabe (L, M oder R) zugeordnet. Je nach Buchstabe wird die Spalte linksbündig (L), zentriert (M) oder rechtsbündig (R) ausgegeben. Das Prefix wird jedem String vorangeschrieben und ermöglicht so die einfache Kontrolle der Farbe oder des Textstyles in der Tabelle.

Die Tabelle insgesamt beginnt mit der aktuellen Schreibposition (wo auch NEXTLUN hinschreiben würde). In X-Richtung wird der aktuelle Prefix angewendet, also bei beispielsweise bei "M" zentriert. Unter der Tabelle kann mit NEXTLUN usw. einfach weitergeschrieben werden. Die aktuellen Prefixe sind erhalten geblieben.

**\TAB** *<Strings>*

ist eine Zeile in der Tabellenumgebung. Die Zahl der Strings sollte der Zahl der Buchstaben der Anordnung im Befehl TABON entsprechen. Abweichungen werden derzeit restriktiv gehandhabt.

**\TABOFF**

schaltet die Tabellenumgebung ab.

**\HLINE**

zeichnet eine horizontale Linie vom aktuellen Zeilenanfang bis RIGHTMARGIN;

steht HLINE innerhalb der TAB-Umgebung, geht die Linie über die Breite der Tabelle.

**\DATE** [*<x:units> <y:units> <x<sub>offset</sub>:cm> <y<sub>offset</sub>:cm> <Size:real> <angel:real>*] zeichnet hochkant an der rechten Seite des Plots das Datum, falls keine Koordinaten angegeben sind.

**\FILE** [*<x:units> <y:units> <x<sub>offset</sub>:cm> <y<sub>offset</sub>:cm> <Size:real> <angel:real>*] zeichnet hochkant an der rechten Seite des Plots den File-Namen, falls keine Koordinaten angegeben sind.

**\{Zähler\|Nenner\}**

Brüche mit automatischer Zentrierung von Zähler und Nenner

### 3.4.5 Sonderzeichen

Für das Aussehen der Sonderzeichen und Text-Attribute mit den verschiedenen Fonts siehe das Musterblatt im Anhang (Kap. A). Die Codierung von Sonderzeichen geschieht durch Kombinationen mit vorangestelltem \ oder & (siehe Abb. 1).

Code	Zeichen		Code	Zeichen		Code	Zeichen
\S	◊		\A	Å		\M	Ḣ
\*	*		\>	→		\<	←
\8	∞		\+	±		\o	•
&a	ä		&o	ö		&u	ü
&A	Ä		&O	Ö		&U	Ü
&s	ß		&i	í		\~ n	ñ
\\	\		&&	&		&#	#
&'	"		&'	“		\'e	é
\'e	è						

Abbildung 1: Kodierung von Sonderzeichen

**Griechische Buchstaben:** Die Buchstaben im Text zwischen den Symbolen #, also # ... Text ... #, werden in griechische Buchstaben übersetzt.

**\GLATEX** oder **\GLATEX ON**

schaltet für sämtliche Textausgabe die German-Latex-Syntax ein, d.h. die Umlaute und ß werden durch vorangestelltes Gänsefüßchen (") codiert. Die Anführungszeichen selbst erhält man wie in Latex durch doppeltes Apostroph (Single-Quote) bzw. doppelte Backquotes, oder - deutsch unten/oben - durch Gänsefüßchen-Backquote bzw. Gänsefüßchen-Apostroph.

**\GLATEX OFF** schaltet die German-Latex-Syntax wieder aus.

**Zwischenräume** können im String eingefügt werden durch:

- `\,` : “thinspace” (0.3 SIZE)
- `\!` : “negative thinspace” (-0.3 SIZE)
- `\^` : small upward displacement (only PS fonts)
- `\v` : small downward displacement (only PS fonts)

### 3.4.6 LaTeX-Formelsatz

LaTeX-Code kann man (ab WRplot-Version 1.1.2003) direkt in auszugebende Strings aller Art einzufügen (also z.B. bei `\LAB`, `\LUN`, `\LUNA`, `\NEXTLUN` ebenso wie bei den Achsen-Beschriftungen). Der LaTeX-Code beginnt mit der Zeichenfolge `\(` und endet mit `\)`. Diese Zeichen haben die gleiche Wirkung wie in LaTeX selbst, d.h. sie führen direkt in den Math-Mode. Der von LaTeX erzeugte PS-Output wird mit aktueller Schriftgröße in den String eingefügt (der Skalierungsfaktor beträgt  $3.6 \cdot \text{SIZE}_{\text{LUN}}$ ), jedoch kann dieser Faktor durch die Setzung von `\SCALE_LATEX` manipuliert werden (siehe unten). Die aktuelle Schreibfarbe wird nach LaTeX übernommen (aber nicht zurückgegeben). Hinter dem LaTeX-Einschub gelten die vorherigen Textattribute fort. Dollars im LaTeX-Code werden von WRplot als WRplot-Variable expandiert.

Beispiel:

```
\NEXTLUN Die Formel \((c = \sqrt{a^2+b^2})\) kennt jedes Kind.
```

#### `\SCALE_LATEX = x.x`

Dieser Faktor wirkt auf alle LaTeX-Elemente. Default ist 1.0; mit etwa 1.25 sehen Formeln innerhalb von SansSerif-Schrift (`\FONT=HELVET`) harmonisch aus, während sie mit der Default-Skalierung (die genau wie in LaTeX ist) etwas pillerig bleiben.

Beachte: Wenn man bei separaten LaTeX-Objekten SCALE von Hand setzt (z.B. `\LATEX SCALE=3`), wird der `SCALE_LATEX`-Faktor zusätzlich angewendet. Wenn man dagegen die Abmessung des LaTeX-Objekts fest vorschreibt (z.B. `EPSFXSIZE=10`), bleibt `SCALE_LATEX` natürlich wirkungslos.

### 3.4.7 Text-Attribute

Die Darstellung der Schrift kann durch bestimmte Präfixe beeinflusst werden, die an beliebigen Stellen in einem Textstring auftreten dürfen (siehe Tabelle 3.4.7).

### 3.4.8 Linienidentifikation

#### `\IDLENG <real>`

bestimmt die Länge des Identifikationsstrichs (Default=2cm). Angabe auch in Units möglich durch nachgestelltes U (z.B. `\IDLENG 0.1U`). Ist der angegebene Wert *negativ*, zeigt der Identifikationsstrich nach unten und die Beschriftung steht unterhalb (seit Programmversion vom 30.12.2008).

`\IDENT <x:units> <Text:string>` Schreibt einen Identifikationsstrich und den String senkrecht an Ort x. Dabei wird automatisch darauf geachtet, daß die einzelnen Beschriftungen nicht ineinander laufen. Der Default-Wert der Schriftgröße ist 0.4 und die Default-höhe ist 8cm.

Tabelle 1: Präfixe in Textstrings

Präfix	Bedeutung
&T	tiefgestellt (Subscript)
&H	hochgestellt (Superscript)
&M	Rückkehr zur mittigen Stellung (hebt auch Eng/Weitstellung und Neigung auf)
&E	eng gestellte Schrift (30% komprimiert)
&B	weit gestellte Schrift (30% gestreckt)
&I	<i>Italics</i> Font
&N	Rückkehr zum normalen Font (hebt auch Eng/Weitstellung und Neigung auf)
&R	rechtsgeneigte Schrift
&L	linksgeneigte Schrift
&G	Rückkehr zur geraden (d.h. nicht geneigten) Schrift
&F	fette ("boldface") Postscript-Fonts
&f	Aufhebung von &F
&W	fett durch doppeltes Zeichnen ein/aus (nur bei WRplot-Fonts)
&i	Umschalten auf Farbe $i = 0 \dots 9$
&PW	lokales Umschalten des Fonts auf WRplot (nur am Stringanfang erlaubt)
&PT	lokales Umschalten des Fonts auf Times (nur am Stringanfang erlaubt)
&PH	lokales Umschalten des Fonts auf Helvetica (nur am Stringanfang erlaubt)

**\IDY** <*y:cm*>

definiert die Höhe der Identifikations-Beschriftung über der X-Achse (Default: 8cm). Angabe auch in Units möglich durch nachgestelltes U (z.B. \IDY 1.5U)

**\IDSIZE** <*Size:real*>

legt die Schriftgröße der Identifikation fest. (Default=0.4)

**\IDXOFF** <*Offset:units*>

verschiebt die Identifikation (nur von IDENT und IDMULT) um Offset in x-Richtung.

**\IDXFAC** <*xfac*>

multipliziert die Identifikations-Wellenlängen (nur von IDENT und IDMULT) mit *xfac*, z.B. für Radialgeschwindigkeit oder Rotverschiebung. Das Zusammenspiel mit IDXOFF ist:  $xfac * \lambda + xoff$

**\IDSPACE** <*real*>

bestimmt den Freiraum zwischen den einzelnen Identifikationen.

**\IDRESET** Initialisiert neu, d.h. bisher geschriebenen Ident-Marken wird nicht ausgewichen.

**\IDHOR** Die folgenden Identifikationen werden senkrecht geschrieben.

**\IDVER** Die folgenden Identifikationen werden waagerecht geschrieben.

**\IDBACKWARD** Die folgenden Identifikationen werden nach links (bzw. unten) ausgerückt.

**\IDFORWARD** Die folgenden Identifikationen werden nach rechts (bzw. oben) ausgerückt.

**\ID\_START** Die Identifikationbeschriftung startet bei der angegebenen Wellenlänge.

**\ID\_CONNECT** Die Identifikationbeschriftungen werden zusammengehalten.

**\ID\_NOCONNECT** Schaltet die vorhergehende Option wieder zurück (Default).

**\IDMULT**  $\langle \lambda_1:\text{\AA} \rangle, \langle \lambda_2:\text{\AA} \rangle, \langle \lambda_3:\text{\AA} \rangle, \dots, \langle \text{Text:string} \rangle$  Linienidentifikation für mehrere Blendkomponenten (Kamm). Der letzte Parameter ist der Textstring. Enthält dieser Blanks, so muß der String durch Anführungsstriche verbunden werden.

**\IDMLENG**  $\langle a:\text{real} \rangle \langle b:\text{real} \rangle$  oder *RESET* Bestimmt die Aufteilung des Identifikationsstriches. Bei Angabe von RESET wird die Aufteilung auf den Default-Wert gesetzt. Der untere sowie der mittlere Teil haben dann die Länge von jeweils 43%. Dem oberen Teil fällt der Rest zu. Die explizite Angabe von a und b beeinflusst den mittleren und unteren Teil. Beide Zahlen müssen im Bereich [0, 1) liegen und deren Summe darf 1.0 nicht überschreiten.

**\IDMCOMB** [ $\langle \text{TRUE}, \text{FALSE} \rangle$ ] Wenn diese Option eingeschaltet ist, dann wird beim COMMAND IDMULT keine Identifikation in den Plot geschrieben.

**\IDLOG** Die Wellenlängen, die in den einschlägigen Ident-Kommandos angegeben werden, werden vor der Ausführung logarithmiert. Damit kann man also die Identifikationen über einer  $\log \lambda$ -Skala plotten.

**\IDNOLOG** macht die IDLOG-Option wieder rückgängig (Default).

**\ID\_INBOX** unterdrückt Linienidentifikationen, die außerhalb des  $x$ -Bereiches der Box liegen. Es wirkt nur auf **\IDENT** und **\IDMULT**; bei **\IDMULT** wird die gesamte Identifikation unterdrückt, sobald eine der Multiplett-Komponenten außerhalb der Box liegt.

Veraltete Kommandos sind:

**\LAM**  $\langle x:\text{units} \rangle \langle y:\text{cm} \rangle \langle \Delta y \rangle$  Zeichnet eine senkrechte Linie (z.B. zur Linienidentifikation) der Länge  $\Delta y$  an den Ort  $x, y$ .

**\TRA**  $\langle x:\text{units} \rangle \langle y:\text{cm} \rangle \langle \text{size:real} \rangle \langle \text{Text:string} \rangle$  Schreibt den String senkrecht (z.B. zur Linienidentifikation) in der Größe Size an den Ort  $x, y$ .

**\CON**  $\langle x1:\text{units} \rangle \langle x2:\text{units} \rangle \langle y:\text{cm} \rangle$  Zeichnet eine waagerechte Linie von  $x1$  bis  $x2$  in der Höhe  $y$ . (Anwendung: Verbindung Multiplettkomponenten)

### 3.4.9 Einbinden von PostScript-Files

Andere PostScript-Files können in des ps-output von WRPLOT eingebunden werden. Im WRPLOT-Fenster wird die Bounding Box des eingebundenen Bildes als gefülltes Rechteck der aktuellen Farbe dargestellt. Die Positionierung und Skalierung des eingebundenen Bildes orientiert sich – wie in Latex – an dessen Bounding Box. Das Kommando ist:

`\EPSF file.ps [options]`

Bei Abwesenheit jeglicher Positionierungsangaben wird das einzubindende Bild nicht skaliert oder verschoben (diese Variante funktioniert deshalb auch bei abwesender oder kaputter Bounding Box).

`[COORD  $x_{\text{units}}$   $y_{\text{units}}$   $\Delta x_{\text{cm}}$   $\Delta y_{\text{cm}}$ ]`

Positionierung nach Koordinaten. Bei den  $\Delta x$ ,  $\Delta y$  kann – wie bei LUN – durch vorangestelltes L (Default) oder R bzw. U oder D (Default) spezifiziert werden, ob sich die angegebene Position auf den linken (L), rechten (R), oberen (U) oder unteren (D) Rand der Bounding Box des einzubindenden Bildes beziehen soll.

NEXTLUN

alternativ zu COORD; positioniert das Bild an die Stelle, an die die nächste Textzeile (mit NEXTLUN) geschrieben werden würde.

`EPSFXSIZE =  $xsize$`  oder `EPSFysize =  $ysize$`

bewirkt eine Skalierung auf die angegebene Breite bzw. Höhe  $xsize$  (Default: in cm; bei angehängtem U: in Units).

`SCALE =  $x.x$`

bewirkt eine Skalierung um den angegebenen Faktor.

`ROTATE =  $x.x$`

bewirkt eine Rotation um den angegebenen Winkel im mathematisch positiven Sinn. Drehpunkt ist der Bezugspunkt der Koordinaten.

LATEX

ermöglicht das Einbinden von epsf-Files, die mit latex und

`dvips -o -E filename`

(alias: **dvieps**) erzeugt worden sind. Die Option versucht, das störende *showpage* aus dem PS-File zu eliminieren.

NOSHOW

reduziert das EPSF auf reine Platzhalter-Funktion, unterdrückt aber die Darstellung des eingebundenen Files. Man bekommt also neue Variable EPS\_X1 usw., die NEXTLUN-Position wird aktualisiert, und ggfs. BGRLUN ausgeführt.

Die Koordinaten der Box werden automatisch in Variablen geschrieben, und zwar in EPS\_X1, EPS\_XM, EPS\_X2 für die X-Koordinate und EPS\_Y1, EPS\_YM, EPS\_Y2 für die Y-Koordinate. Diese Angaben sind in units. Die gleichen Variablen, nur mit einem angehängten CM, also z.B. EPS\_X1CM, geben die Positionen in cm an.

### 3.4.10 Objekte aus LaTeX-Quellcode

`\LATEX [options]`

`...latex-script ...`

`...latex-script ...`

`\ENDLATEX`

entspricht ungefähr dem, als würde man mit LaTeX aus dem ...latex-script... ein EPS-File erzeugen und dieses dann mit `\EPSF` in WRplot einbinden. Der zwischen `\LATEX` und `\ENDLATEX` eingeschlossene LaTeX-Code wird in ein Dokument (documentstyle, 12pt) eingefügt, das einsprechende EPS-File erzeugt und in den WRplot eingebunden. Im

X11-Fenster erscheint natürlich nur die BoundingBox. Der String kann WRplot-Variable enthalten, aber sonst keine Dollars. Die aktuelle Farbsetzung in WRplot (`\COLOR`) wird nach Latex übernommen, aber eine ggfs. in LaTeX veränderte Farbe nicht nach WRplot zurückgegeben.

Die Optionen hinter `\LATEX` entsprechen denen von `\EPSF` (Positionierung, Skalierung, Rotation). Im Unterschied zu `\EPSF` gelten jedoch folgende Defaults:

- Positionierung wie `NEXTLUN` (also nächste Zeile), falls kein `COORD=...` angegeben ist
- Skalierung auf die aktuelle Schriftgröße (`SIZELUN`), falls keine andere Skalierung (`SCALE`, `EPSFXSIZE`, `EPSFYSIZE`) angegeben ist.

In den meisten Fällen wird man also ganz ohne Optionen auskommen. Wie bei `\EPSF` kann mit `\BGRLUN` hinterlegt oder umrandet werden.

*Zum Einfügen von LaTeX-Code in beliebige Text-Strings siehe auch Kapitel 3.4.6.*



## 4 Die Koordinaten-Box

Der Plot-Header beschreibt die Achsen und Beschriftungen eines Diagramms. Für reine Textfolien, die kein solches Diagramm enthalten, kann man diesen Teil sparen durch:

\DEFAULTS

Die Default-Werte erzeugen eine Diagramm-Box mit dem Maßstab 1 cm pro Einheit auf beiden Achsen, setzen den Ursprung in die linke untere Ecke des Blattes (wie OFS 0 0) und unterdrücken die Ausgabe der Box (wie NOBOX).

Wenn man jedoch eine Box benötigt: Der Plotheader beginnt mit dem Keyword HEADER und umfaßt sechs Zeilen, deren Reihenfolge fest vorgeschrieben ist (Kommentarzeilen können aber eingestreut werden).

Beispiel:

HEADER :Text des Headers

X-ACHSE:Beschriftung der X-Achse

Y-ACHSE:Beschriftung der Y-Achse

.	MASSTAB	MIN.	MAX.	TEILUNGEN	BESCHRIFT.	DARUNTER
X:	1.	15.	16.	1.	5.	0. [options (x)]
Y:	0.	0.	20.	1.	5.	0. [options (y)]

*Autoscale:* Steht in der Zeile X: irgendwo das Wort AUTO, dann wird die Skala für beide Achsen automatisch so generiert, dass sie die Wertebereiche der Datensätze abdecken. Die numerischen Parameter der X:-Zeile werden dann ignoriert, ebenso wie die numerischen Parameter in der nachfolgenden, mit Y: beginnenden Zeile. Steht dagegen in der Zeile mit X: das Wort AUTOX, erstreckt sich die automatische Skalierung nur auf die X-Achse. Umgekehrt bewirkt in der Y:-Zeile das Wort AUTO (oder AUTOY) eine automatische Skalierung nur für die Y-Achse.

Erklärung der Parameter:

**HEADER :** Titel des Plots (darf leer sein)

**X-ACHSE:** Beschriftung der X-Achse (darf leer sein)

**Y-ACHSE:** Beschriftung der Y-Achse (darf leer sein)

Standardmässig stehen die Beschriftungen linksbündig (bzw. bei der Y-Achse unten).

Wird vor die Strings der drei vorstehenden Befehle \CENTER\ geschrieben, dann wird der String zentriert. Beginnen die Strings dagegen mit \RIGHT\ schließen sie rechtsbündig mit der Box ab.

**MASSSTAB:** in Zentimeter pro benutzte Einheit.  $\Delta x(y) * \text{Maßstab} = \text{Plotbreite(Plathöhe)}$ .

Wird Maßstab = 0 gewählt, so wird ein Plot von 15cm Höhe  $\times$  20cm Breite erzeugt. Steht hinter dem Wert direkt die Angabe CM, dann wird der Maßstab so ausgerechnet, daß der Plotkasten diese Breite (Höhe) in cm hat. Steht anstelle des Massstabs das Wort AUTO, wird automatisch skaliert (siehe oben).

**MINIMUM:** Beginn des x(y)-Bereichs

**MAXIMUM:** Ende des x(y)-Bereichs

**TEILUNGEN:** Abstand der Markierungen (negativ, falls die Achse gespiegelt ist)

**BESCHRIFTUNGEN:** Abstand der Beschriftungen (negativ, falls die Achse gespiegelt ist)

**DARUNTER:** Einer der zu beschriftenden Werte

**options** Optionale Parameter, insbesondere nützlich wenn man als Boxen direkt aneinanderrücken will. Die Optionen gelten immer für die jeweilige Achse. Es gibt:  
NOTICK unterdrückt die Ticks an der unbeschrifteten Seite der Koordinatenbox;  
NOTICK-BOTH unterdrückt die Tickmarken an beiden x- bzw. y-Achsen;  
NOLAB unterdrückt die Beschriftung (Zahlen) an der jeweiligen Achse;  
NOMIN unterdrückt das am weitesten links bzw. unten stehende Label;  
NOMAX unterdrückt das am weitesten rechts bzw. oben stehende Label;  
MININD zieht das am weitesten links bzw. unten stehende Label etwas ein;  
MAXIND zieht das am weitesten rechts bzw. oben stehende Label etwas ein;  
MAXNOIND unterdrückt das standardmäßige Einziehen des Labels, wenn es nach rechts bzw. oben über die Box hinausragt;  
ALT positioniert die Labels und Beschriftung der betreffenden Achse an die *rechten* bzw. *oberen* Seite der Koordinatenbox.

Die Box wird mit der aktuell (d.h. am Ende der KASDEF-Kommandos) eingestellten Strichstärke, Zeichenfarbe und Fontwahl ausgeführt. Die Schriftgröße kann mit \LETTERSIZE, die Größe der Ticks optional unabhängig mit \TICKSIZE beeinflusst werden.

## 5 Die Plotdaten

### 5.1 Datensätze

Ein Datensatz ist eine Tabelle aus Wertepaaren. Entsprechend seinem Auftreten im WRplot-Quellfile bekommt jeder Datensatz eine laufende Nummer. Optional kann man ihm einen symbolischen Namen geben (siehe COMMAND SETNAME).

Jeder Datensatz wird durch einen Block definiert, der mit einer Zeile  
N= ...

beginnt, und mit einer der Zeilen

FINISH

COMMAND INCLUDE

COMMAND INFITS

abgeschlossen wird. Diese Abschluss-Kommandos werden in der Subsektion 5.3 beschrieben.

Dazwischen stehen Daten und COMMANDs. Fast alle COMMANDs können sich auch auf einen anderen (davorstehenden) Datensatz beziehen, wenn die COMMAND-Zeile ein nachgestelltes DATASET=*dataset* aufweist.

## 5.2 Kopfzeile und Plot-Attribute

**N=... SYMBOL=... SIZE=... PEN=... XYTABLE COLOR=...**

Mit dieser Kopfzeile beginnt jeder Datensatz-Block. Der Beginn der Zeile mit "N=" ist zwingend vorgeschrieben. Alle weiteren Parameter sowie deren Reihenfolge sind optional. Folgende Parameter gibt es:

**N=...**

gibt die Anzahl der folgenden Datenpunkte an. Ist  $N = ?$  gegeben, dann werden alle Daten eingelesen, bis ein FINISH folgt.

**SYMBOL= $n$**

(auch: PLOTSYMBOL= $n$ ) gibt an, durch welche Symbole oder Strichart die Daten dargestellt werden sollen. Die verschiedenen Möglichkeiten werden in Kapitel 6 sowie Abb. 2 beschrieben. Bei SYMBOL=0 wird der Datensatz nicht geplottet, steht aber zum Rechnen oder für Errorbars zur Verfügung.

**SIZE =  $x.x$**

(auch SYMBOLSIZE) gibt die Größe der Symbole an (Default: 0.3). Hat der Wert ein negatives Vorzeichen, wird die Fläche, die von der durch die Tabelle beschriebenen Kurve eingeschlossen wird, mit der Zeichenfarbe gefüllt. Durch zusätzliche Angabe entsprechender Parameter (siehe Abschnitt 3.4.3) kann das Füllen auch mit einer Schraffur erfolgen.

**PEN =  $n$**

gibt die Strichstärke an (Default: 1 bzw. PENDEF)

**COLOR =  $n$**

gibt die Zeichenfarbe an ( $n = 0 \dots 9$ ); default COLOR=1 bzw. durch DEFAULT-COLOR gesetzt. Bei SYMBOL=40 (Falschfarbendarstellung) bezeichnet COLOR die Farbtabelle. Es stehen bisher fünf Farbtabellen bereit 1=Greyscale, 2=Blackbody, 3=blue-black-red, 4=blue-white-red, 5=rainbow). Zur Veranschaulichung dient der Beispielplot `~lars/wrplot/colortable.plot` (siehe Abb. im Anhang). Um die Zahl der verwendeten Farben an die Grafikkarte anzupassen, gibt es das KASDEF-Kommando `\NCOLORSTEP` (siehe dort).

**XYTABLE**

Ist dieser Parameter in der N= -Zeile angegeben, wird eine normale mehrspaltige Tabelle erwartet. Default ist dagegen das platzsparende WRPLOT-Tabellenformat. Es beginnt mit einer Zeile von x-Werten, gefolgt von einer Zeile mit den dazugehörigen y-Werten. Die Anzahl der Daten in einer Zeile ist dabei beliebig. Es muß nur darauf geachtet werden, daß nach einer Zeile mit x-Daten eine Zeile mit gleichvielen y-Daten folgt.

**SELECT <  $n1$  > <  $n2$  >**

spezifiziert die Spalten der XYTABLE, die als X- bzw. Y-Werte betrachtet werden sollen (Default: SELECT=1:2). Achtung: Es wird nicht überprüft, ob die angegebenen Spalten auch existieren – falls nicht, passiert irgendein Unsinn. Speziell vorgesehen ist die Angabe einer Spalte "-1" (z.B. SELECT=-1,1): In diesem Fall wird die laufende Nummer des Tabelleneintrags als Wert genommen.

## **FLEX-XYTABLE**

Die Datenzeilen können beliebig viele Paare von Daten (z.B.: x1 y1 x2 y2 x3 y3) enthalten (SELECT ist dann natürlich nicht möglich).

## **5.3 Datentabellen**

Die Datentabellen können direkt im WRplot-File stehen, oder von einem externen File eingelesen werden. Kombiniert man beides, kommen erst die an Ort und Stelle angegebenen Daten, dann das externe File.

**FINISH** Beendet einen Datensatz, der mit  $N = ?$  begonnen wurde, falls dieser nicht durch COMMAND INCLUDE oder COMMAND INFITS abgeschlossen wird. Zwischen der  $N$ -Kopfzeile und dem *FINISH* kann die Datentabelle stehen.

**COMMAND APPEND *dataset*** fügt in den aktuellen Datensatz die Daten aus einem schon bestehenden Datensatz *dataset* ein.

**COMMAND INCLUDE:** *<File Name>* Die Daten dieses Datensatzes werden aus dem File *<File Name>* gelesen. Dabei ist zu beachten, daß die "N=-Zeile" aus dem rufenden Plot-File gilt. In dem Include-File können wiederum Include Commands stehen. Es gibt drei Möglichkeiten, wie der Include-Datensatz aussehen kann:

**ROH-DATEN-SATZ:** In dem File stehen nur die Daten-Punkte.

**NORMALES PLOT-FILE:** Das File ist ein normales WRPLOT Daten-File mit einem Datensatz. Es wird dann nach der ersten "N=-Zeile" gesucht.

**NORMALES PLOT-FILE:** Das File ist ein normales WRPLOT Daten-File mit mehreren Datensätzen. In dem rufenden Plot-File wird nach COMMAND INCLUDE durch das Keyword INCKEY= *String* angegeben, welcher der Datensätze eingespielt werden soll. Durch das Keyword DATASET= *Nummer* wird nicht der erste folgende Datensatz eingelesen, sondern der angegebene.

In allen Fällen braucht man sich nicht um das ordnungsgemäße Ende des Include-Files zu kümmern, da sowohl das File Ende als auch die Worte END und FINISH in dem Include-File erkannt werden.

**COMMAND INFITS:** *<Filename.fits>* Liest ein (einfaches) Fits-File ein. Das Kommando hat verschiedene Optionen, die nicht alle dokumentiert sind. Die nützlichste Anwendung ist das Auslesen von einer oder mehreren Zeilen aus einem zweidimensionalen Image (z.B. Spektrum). Die X-Werte werden durchnummeriert. Beispiel:

```
COMMAND INFITS filename.fits NHDU=2 ROW=7
```

liest die 7. Zeile aus dem Fits-File in den aktuellen Datensatz.

```
COMMAND INFITS filename.fits NHDU=2 ROWS= 7 15
```

Liest und addiert die Zeilen 7 – 15 aus dem Fits-File in den aktuellen Datensatz.

## 5.4 COMMAND-Zeilen

Bestimmte COMMANDs werden sofort zur Kenntnis genommen und ausgeführt bzw. setzen Parameter: APPEND, X-RANGE, Y-RANGE, SKIP, SETNAME, INCLUDE, INFITS. Diese COMMANDs können daher auch nicht durch IF-Konstruktionen übersprungen werden. Die übrigen COMMANDs werden zunächst gespeichert und erst auf dem fertigen Datensatz der Reihe nach ausgeführt.

### COMMAND SETNAME *name*

ordnet dem aktuellen Datensatz einen symbolischen Namen zu. Der Datensatz kann später alternativ mit diesem Namen oder seiner laufenden Nummer angesprochen werden.

**COMMAND X-RANGE:**  $\langle x_1 :units \rangle \langle x_2 :units \rangle$  Alle Punkte deren x-Koordinate nicht im angegebenen Intervall liegen werden beim Einlesen ignoriert.

**COMMAND Y-RANGE:**  $\langle y_1 :units \rangle \langle y_2 :units \rangle$  Alle Punkte deren y-Koordinate nicht im angegebenen Intervall liegen werden beim Einlesen ignoriert.

**COMMAND X-CUT:**  $\langle x_1 :units \rangle \langle x_2 :units \rangle$  Alle Punkte deren x-Koordinate nicht im angegebenen Intervall liegen werden ausgeschnitten.

**COMMAND Y-CUT:**  $\langle y_1 :units \rangle \langle y_2 :units \rangle$  Alle Punkte deren y-Koordinate nicht im angegebenen Intervall liegen werden ausgeschnitten.

**COMMAND X-OMIT:**  $\langle x_1 :units \rangle \langle x_2 :units \rangle$

Negation des Kommandos X-CUT. Punkte im Bereich zwischen  $x_1$  und  $x_2$  werden gelöscht.

**COMMAND Y-OMIT:**  $\langle y_1 :units \rangle \langle y_2 :units \rangle$

Negation des Kommandos Y-CUT. Punkte im Bereich zwischen  $y_1$  und  $y_2$  werden gelöscht.

**COMMAND XY-SWAP** Die X- und Y-Werte des angegebenen Datensatzes werden vertauscht.

### COMMAND WAVECAL *i*

Die Y-Werte des aktuellen Datensatzes werden als X-Werte in den Datensatz *i* übernommen. Dies setzt voraus, dass zuvor beide Datensätze identische X-Werte haben. Dieses Kommando ist nützlich zum Jonglieren mit Datensätzen.

Speziell für die Wellenlängen-Kalibration kann der aktuelle Datensatz auch eine abweichende X-Tabelle haben. z.B. nur wenige Stützstellen mit den vermessenen Wellenlängen der Kalibrations-Linien. Diese müssen monoton geordnet sein. WAVECAL interpoliert bzw. extrapoliert dann in dieser Tabelle für alle X-Werte des zu kalibrierenden Datensatzes *i*. Interpolation benutzt Splines, Extrapolation ist linear aus den beiden ersten bzw. letzten Punkten der Tabelle.

**COMMAND SKIP** Der Datensatz wird nicht geplottet und nach der Abarbeitung wieder entfernt. COMMAND SKIP *i* entfernt die letzten *i* Datensätze.

### COMMAND EXPAND $< N > [< \lambda_1, \lambda_2 >]$

In den Datensatz werden  $N$  Punkte eingefügt. Sind die Parameter  $\lambda_1$  und  $\lambda_2$  gesetzt, dann werden die Punkte nur in diesen Bereich eingefügt. Die Interpolation ist defaultmäßig linear; optional liefert COMMAND EXPAND-SPLINE eine stückweise kubische Interpolation.

### COMMAND INSERT $< \lambda_1 >$

In den Datensatz wird an der Stelle  $\lambda_1$  ein Punkt eingefuegt.

### COMMAND HLYMAN *parameter*

(auch HLYMANA) – Korrektur für interstellare Absorption durch die Lyman-Linien bis  $L_{20}$ . Die Y-Werte müssen lineare Flüsse, die X-Werte Wellenlängen in Å enthalten. Das Kommando *korrigiert* die Beobachtung um die Absorption; um die Absorption einem Modell aufzuprägen, muss EBV bzw. COLDENS als negativer Wert eingegeben werden. By default, this HLYMAN uses the formula from Groenewegen & Lamers (1989) which accounts only for the damping (i.e. Lorentzian) profiles. However, if one of the parameters T and/or VTURB is specified, HLYMAN applies Voigt profiles with the according Doppler velocity.

The parameters consist of pairs “keyword = value”; allowed keywords are:

EBV = x.x (color excess  $E_{B-V}$  in magnitudes)

COLDENS = x.x (column density of H-atoms per  $\text{cm}^2$ ,  $N_H$ )

VRAD = x.x (radial velocity in km/s)

T = x.x (temperature in Kelvin) accounts additionally for Doppler broadening with the thermal velocity

VTURB = x.x (turbulence velocity in km/s causing additional Doppler broadening)

One of the parameters (EBV, COLDENS) is mandatory.

Example:

COMMAND HLYMANA EBV=0.2 VRAD=255.

For compatibility with older syntax, one can use numerical parameters without keyword (but this is not recommended):

- only one parameter means the value of EBV

- two numerical parameters mean EBV, VRAD

### COMMAND ISMLINE *parameter*

correction or simulation of an interstellar line; the command is similar to HLYMAN, but now for an arbitrary line which is fully specified by the command.

The following parameters are mandatory:

COLDENS = x.x (column density of the absorbing atoms per  $\text{cm}^2$ )

L0 = x.x (central wavelength of the ISM line in Å)

GAMM = x.x (damping constant for this transition in  $\text{s}^{-1}$ )

F = x.x (oscillator strength)

If the following optional parameters are also specified, the synthetic profile will also include a Doppler core and produce a Voigt profile:

T = x.x (temperature in Kelvin) accounts additionally for Doppler broadening with the thermal velocity

A = x.x (atomic weight in Atomic Mass Units)

VTURB = x.x (turbulence velocity in km/s causing additional Doppler broadening)

Further optional parameter:

VRAD = x.x (radial velocity in km/s)

### **COMMAND REDDENING** <E(B-V): real> [law] [parameter]

Korrektur für interstellare Rötung. Die Y-Werte müssen logarithmierte Flüsse, die X-Werte logarithmierte Wellenlängen in Å enthalten. Die X-Werte müssen monoton geordnet sein. Sie werden automatisch auf mindestens 1000 Punkte expandiert.

Standardmäßig wird das Rötungsgesetz nach Seaton (1979: MNRAS 187, 73) verwendet. In fact, Seaton has determined the reddening only for  $< 3700 \text{ Å}$ , but adopts for  $3500 - 10000 \text{ Å}$  results from Nandy et al. (1975: A&A 44, 195).

Langwellig ihres jeweiligen Gültigkeitsbereiches werden alle Rötungsgesetze ergänzt gemäß einer Tabelle aus Moneti A., Stolovy S., Blommaert J.A.D.L., Figer D.F., & Najarro F. (2001: A&A 366, 106). Ab  $24 \mu\text{m}$  wird eine Extrapolation  $\propto \lambda^{-2}$  angeschlossen.

Im Röntgengebiet ( $1.24 - 413 \text{ Å}$ ) erfolgt die Absorption gemäß Table 2 in Morrison R. & McCammon D., 1983, ApJ 270, 119. Die in diesem Gebiet sichtbaren Kanten werden durch K-Schalen-Absorption hervorgerufen. Die Zusammensetzung ist als solar angenommen.

Zwischen der Kante von He I bei  $504 \text{ Å}$  und dem Beginn der Tabelle wird die Formel der Tabelle extrapoliert. Zwischen der Lyman-Kante und der He I-Kante, also im Bereich  $911 - 504 \text{ Å}$ , wird nur die Opazität von H I berücksichtigt, und zwar wie im PoWR-Code einschließlich dem Gaunt-Faktor.

Durch Angabe eines Keywords [law] können andere Rötungsgesetze gewählt werden. Verdrahtet sind:

**CARDELLI** : nach Cardelli J.A., Clayton G.C., Mathis J.S.: 1989, APJ 345, 245, from  $1000 - 33333 \text{ Å}$ .

**FITZPATRICK** : nach Fitzpatrick E.L.: 1999, PASP 111, 63; valid approximately till  $5 \mu\text{m}$ .

Note: The laws CARDELLI and FITZPATRICK have an optional parameter  $R_V$  which is the Extinction in the V band per  $E_{B-V}$ . The default value is 3.1 for both laws.

**LMC** : nach Howarth I., 1983, MNRAS 203, 301

**SMC** : Same law as for the LMC (Howarth 1983). However, in contrast to the LMC law, the SMC law has the optional  $R_V$  parameter. The SMC default is  $R_V = 2.7$  (Bouchet et al. 1985: A&A 149, 330); for  $R_V = 3.2$  the SMC law becomes identical to the LMC law.

**SMC\_BAR** : reddening law for the bar of the SMC (Gordon et al., 2003, ApJ 594, 279). This law is valid for  $\lambda < 3000 \text{ \AA}$ . Longwards, it is augmented by the FITZPATRICK law, assuming  $R_V=2.74$  (which consistent with the UV law). Longward of  $5\mu$  the reddening follows Nandy et al. (see above).

**COMMAND CONVOL:** *<Keyword> <C>* Faltung des Datensatzes mit der durch *Keyword* angegebenen Funktion.

Keyword *GAUSS*: Gauss-Funktion mit der Halbwertsbreite (FWHM) *C*

Keyword *KASTEN*: Box-Profil mit der Breite  $\pm C$  – also eine Glättung über  $2C$  !!!

Keyword *ROT[ATION]*. *C* gibt dabei die der Rotationsgeschwindigkeit entsprechende Verschiebung an, also auch hier umfasst die Faltung den Bereich  $\pm C$ .

**COMMAND BINNING** *w*

Voraussetzung ist ein monoton geordneter Datensatz. Beginnend mit dem ersten Index werden Datenpunkte, deren *x*-Werte sich um  $\leq w$  unterscheiden, zu einem einzigen Punkt zusammengefasst, indem sowohl von den *x*-Werten als auch von den *y*-Werten das arithmetische Mittel gebildet wird.

**COMMAND X+** *< A : units >* (Wahlweise auch COMMAND XADD oder COMMAND XOFFSET) Addiert die Konstante *A* zu den *x*-Koordinaten des gesamten Datensatzes (Achtung: eine Trennung zwischen dem + und der Konstanten ist zwingend!)

**COMMAND X-** *< A : units >* (Wahlweise auch COMMAND XSUB) Subtrahiert *A* von den *x*-Koordinaten des gesamten Datensatzes

**COMMAND Y+** *< A : units >* (Wahlweise auch COMMAND YADD oder COMMAND YOFFSET) Addiert *A* zu den *y*-Koordinaten des gesamten Datensatzes

**COMMAND Y-** *< A : units >* (Wahlweise auch COMMAND YSUB) Subtrahiert *A* von den *y*-Koordinaten des gesamten Datensatzes

**COMMAND X\*** *< A : units >* (Wahlweise auch COMMAND XMULT) Multipliziert die *x*-Koordinaten des gesamten Datensatzes mit *A*

**COMMAND X/** *< A : units >* (Wahlweise auch COMMAND XDIV) Dividiert die *x*-Koordinaten des gesamten Datensatzes durch *A*

**COMMAND Y\*** *< A : units >* (Wahlweise auch COMMAND YMULT) Multipliziert die *y*-Koordinaten des gesamten Datensatzes mit *A*

**COMMAND Y/** *< A : units >* (Wahlweise auch COMMAND YDIV) Dividiert die *y*-Koordinaten des gesamten Datensatzes durch *A*

**Hinweis:** Bei den obigen acht Befehlen kann als Argument statt der Konstanten *A* auch **X** bzw. **Y** angegeben werden.

**COMMAND XINV** oder **COMMAND 1/X** bildet den Kehrwert der *x*-Koordinaten des gesamten Datensatzes.

**COMMAND YINV** oder **COMMAND 1/Y** bildet den Kehrwert der *y*-Koordinaten des gesamten Datensatzes.



**COMMAND XDEX** oder **COMMAND 10\*\*X** delogarithmiert die x-Koordinaten des gesamten Datensatzes.

**COMMAND YDEX** oder **COMMAND 10\*\*Y** delogarithmiert die y-Koordinaten des gesamten Datensatzes.

**COMMAND XLOG** oder **COMMAND LOG(X)** logarithmiert die x-Koordinaten des gesamten Datensatzes.

**COMMAND YLOG** oder **COMMAND LOG(Y)** logarithmiert die y-Koordinaten des gesamten Datensatzes.

**COMMAND XMIN = x.x**

setzt diejenigen x-Werte des Datensatzes, die kleiner als x.x sind, auf den angegebenen Minimalwert. analog sind die COMMANDs

**COMMAND XMAX = x.x**

**COMMAND YMIN = x.x**

**COMMAND YMAX = x.x**

**COMMAND Y-NORM** *< X : units > < Y : units >* addiert die Y-Werte des Datensatzes derart, daß die Kurve durch den angegebenen Punkt (X,Y) geht. In dem Datensatz werden die ersten beide Datenpunkte ermittelt, welche ein Intervall um X bilden. Wahlweise kann durch Angabe von Y-NORM\* auch eine Normierung durch Multiplikation erreicht werden.

**COMMAND SIGN(X)** ersetzt die X-Werte durch -1., 0., oder +1. entsprechend der Signum-Funktion.

**COMMAND SIGN(Y)** ersetzt die Y-Werte durch -1., 0., oder +1. entsprechend der Signum-Funktion.

**COMMAND X-INC**

löscht alle Punkte die in der X-Koordinate nicht aufsteigen aus einem Datensatz. Dies ist hilfreich, wenn z.B. fremde Spektren (z.B. IUE-Daten) importiert werden, die nicht dem strengen deutschen Reinheitsgebot für Spektren entsprechen, die aber mit Funktionen bearbeitet werden sollen, welche eine monotone Anordnung verlangen (z.B. CONVOL).

**COMMAND ARI** *< Dataset<sub>1</sub> > < Operator > < Dataset<sub>2</sub> > [KEEP\_X1]*

führt die durch *Operator* (+, -, \* oder /) angegebene Operation mit den Y-Werten der angegebenen Datensätze aus. Das Ergebnis wird in *Dataset<sub>1</sub>* abgespeichert. Bei identischen X-Tabellen werden die Elemente indexweise zugeordnet. Bei verschiedenen X-Tabellen werden die Funktionswerte wechselseitig im jeweils anderen Datensatz interpoliert (nur im Überlappungsbereich). Dazu müssen beide x-Tabellen monoton sein, sonst erfolgt eine Fehlermeldung.

Der optionale Parameter *KEEP\_X1* bewirkt, dass die Ergebnis-Funktion nur die Stützstellen aus *Dataset<sub>1</sub>* verwendet.

**COMMAND STRIP y1 y2**

wirkt nur bei SYMBOL=40 (Falschfarben-Darstellung) und definiert den waagerechten Streifen, in dem die Daten dargestellt werden (y1, y2 in Units).

**COMMAND XERROR < n > < type >**

ordnet dem aktuellen Datensatz den Datensatz *n* zu, der den Fehlerbalken in *x*-Richtung beschreibt. Der Fehlerdatensatz muss zu dem Zeitpunkt der Zuordnung bereits existieren, also *vor* dem Datensatz angelegt werden, dem die Fehlerbalken zugeordnet werden. Der obligatorische Parameter *type* spezifiziert, ob die Fehler-Daten als relativ zum Datenpunkt (PLUSMINUS) oder absolut (MINMAX) zu verstehen sind. Der Fehler-Datensatz muss das SYMBOL=0 tragen. Er muss entweder genau *ein* Datenpaar besitzen (gleicher Fehlerbalken für alle Werte), oder genau so viele Datenpaare wie der Hauptdatensatz (individuelle Fehlerbalken). Die Attribute des Fehlerbalkens werden vom Fehler-Datensatz genommen (COLOR, PEN, SIZE). SIZE skaliert die Größe der Querstriche (Füße).

**COMMAND YERROR < n > < type >**

wie XERROR, jedoch in *y*-Richtung. Datenpunkte können also unabhängig mit Fehlerbalken in *x* und/oder in *y* verziert werden.

**COMMAND MERGE dataset1 dataset2**

vereinigt zwei geordnete Datensätze. Dabei wird im Überlappbereich der Mittelwert gebildet, wobei an nicht gemeinsamen *x*-Werten jeweils die andere Funktion an dieser Stelle interpoliert wird. MERGE ist insbesondere dafür gedacht, überlappende Spektren wie z.B. die einzelnen Echelle-Ordnungen zu vereinigen.

**COMMAND SORT [INV] [JOIN]**

sorts the dataset such that the *x*-values are in monotonic order. With the optional parameter INV given, the dataset is sorted in a falling sequence. With the optional parameter JOIN given, the dataset becomes *strictly* monotonic; entries with identical *x*-values are joined into one entry, with the *y*-value set to their arithmetic mean. – Note that this command is useful in preparing datasets for such operations for which a monotonic order is prerequisite.

**COMMAND WRITE [FILE=filename]**

schreibt des aktuellen Datensatz im aktuellen Zustand als XYTABLE in ein File mit dem angegebenen Namen. Existiert die Datei schon, wird sie überschrieben. Ist kein Filename angegeben, heißt das File WRPLOT\_DATASET*n*.DAT mit *n* = laufende Nummer des Datensatzes.

**COMMAND kasdef-kommando, ...**

Darüberhinaus können als COMMANDs auch viele Kommandos ausgeführt werden, die unter den KASDEF-Befehlen beschrieben sind. Ausgenommen sind natürlich KASDEF-Befehle, die eine Zeichaktion auslösen. Erlaubt sind aber alle Befehle, die mit Variablen hantieren (z.B. COMMAND CALC ...), sowie IF...ELSE...ENDIF-Konstruktionen, ECHO, GOTO, LABEL und DO-Loops.

## 5.5 COMMAND-Funktionen

### COMMAND CF-... ..

sind sogenannte *Command-Functions*. Sie schreiben ihr Ergebnis in Variable, die dann von nachfolgenden COMMANDs oder von „KASDEF-Kommandos“ weiter verarbeitet werden können. Bisher existieren:

#### COMMAND CF-YEXT < $\lambda$ > <VAR>

liefert den interpolierten Y-Wert des Datensatzes an der Stelle  $\lambda$  und schreibt ihn in die Variable mit dem Namen VAR. Die Tabelle muss monoton sein. Wenn das PLOTSYMBOL = 6 oder 7 ist, wird mit Splines interpoliert, sonst linear.

#### COMMAND CF-YMAX <XVAR> <YVAR>

schreibt das Wertepaar mit dem größten vorkommenden Y-Wert in die Variablen mit Namen XVAR und YVAR

#### COMMAND CF-YMIN <XVAR> <YVAR>

schreibt das Wertepaar mit dem kleinsten vorkommenden Y-Wert in die Variablen mit Namen XVAR und YVAR

#### COMMAND CF-YSUM <VAR>

schreibt die Summe aller Y-Werte der Tabelle in die Variable VAR.

#### COMMAND CF-INT <VAR> [FROM $x_1$ TO $x_2$ ]

schreibt das Integral der tabellierten Funktion in die Variable VAR. Dabei spielt keine Rolle, ob die Funktion aufsteigend oder absteigend tabelliert ist – sie muss nur monoton sein. Optional können Integrationsgrenzen angegeben werden (wenn, dann beide!). Defaultmäßig wird über den ganzen tabellierten Bereich integriert.

#### COMMAND CF-LINREG $x_1$ $y_1$ $x_2$ $y_2$ $a$ $b$ $a_{\text{err}}$ $b_{\text{err}}$

fittet mittels linearer Regression die Funktion  $y = a + bx$  durch die Punkte des Datensatzes. (Hinweis: Durch vorheriges Logarithmieren des Datensatzes kann man auch mit Potenzgesetzen fitten.)

Man kann optional bis zu acht Parameternamen (keine Werte, nur Namen!) angeben, denen dann durch die Command-Function Werte zugewiesen werden. Ihre Bedeutung ist:

$x_1$  = kleinster im Datensatz vorkommender  $x$ -Wert

$y_1$  = Wert der linearen Regression an dieser Stelle

$x_2$  = größter im Datensatz vorkommender  $x$ -Wert

$y_2$  = Wert der linearen Regression an dieser Stelle

Diese ersten vier Variablen sind vor allem dafür gedacht, um mittels

\LINUN \$x1 \$y1 \$x2 \$y2 0 0

die Regressionsgerade durch die Punktwolke zu zeichnen.

Weitere optionale Variablennamen werden der Reihe nach gefüllt mit:

$a$  = Koeffizient  $a$  der Regressionsgeradengleichung

$b$  = Koeffizient  $b$  der Regressionsgeradengleichung

$a_{\text{err}}$  = formale Genauigkeit ( $1 \sigma$ ) von  $a$

$b_{\text{err}}$  = formale Genauigkeit ( $1 \sigma$ ) von  $b$

Wenn dem Datensatz, durch den die Regressionsgerade gelegt wird, Fehlerbalken in y zugeordnet sind, werden die Punkte entsprechend gewichtet. Man beachte, dass in diesem Fall die formalen Fehler der Geraden-Koeffizienten nach einer völlig anderen (den Numerical Recipes entnommenen) Methode erfolgt als bei Datenpunkten ohne Fehlerbalken. Insbesondere kann der formale Fehler der Regressionsgeraden-Koeffizienten sogar kleiner ausfallen, wenn die Datenpunkte mit Fehlerbalken verziert werden.

Alle Ausgabeparameter werden auch ins Kommandofenster geschrieben.

**COMMAND CF-NDATA <VAR>**

schreibt die aktuelle Anzahl der Wertepaare des Datensatzes in die Variable VAR.

## 6 Die Plot-Symbole

**0:** Datensatz wird überhaupt nicht geplottet.

**1:** Kreuze

**2:** Sterne

**3:** Dreiecke, bei negativem SIZE wird das Symbol automatisch ausgefüllt.

**4:** Quadrate, bei negativem SIZE wird das Symbol automatisch ausgefüllt.

**5:** Polygonzug (SIZE überflüssig)

**6:** Kubische Spline Interpolation Funktioniert auch bei nicht injektiven Abbildungen. SIZE gibt die Feinheit (Punktabstand) der Interpolationskurve an.

**7:** Kubische Spline-Interpolation gestrichelt (Strichlänge nicht genau konstant!) (SIZE = Strichlänge in mm) Funktioniert auch bei nicht injektiven Abbildungen.

**8:** Kreise, bei negativem SIZE wird das Symbol automatisch ausgefüllt. Der Durchmesser des Kreises beträgt dabei  $\frac{2}{3} * \text{Size}$  und paßt sich der Größenangabe für Dreiecke, Quadrate etc. an.

**9:** Polygonzug gestrichelt (SIZE gibt die Strichlänge an)

**10:** Polygonzug strich-punktiert (SIZE gibt die Strichlänge an)

**11:** Dreiecke, welche auf der Spitze stehen. Bei negativem SIZE wird das Symbol automatisch ausgefüllt.

**12:** Quadrate, die auf der Spitze stehen.

**13:** Bomben, vor Mißbrauch wird gewarnt.

**14:** Dreizackige Sterne

**15:** Rauten (Diamonds, "Salmis", "Buoys" = Bojen) aus zwei gleichseitigen Dreiecken

**16:** Senkrechte zentrierte Striche der Länge SIZE

+	1 : Kreuz
*	2 : Stern
▲	3 : Dreieck
■	4 : Q uadrat
—	5 : Linear
—	6 : Kubische Splines
- - -	7 : Kubische Splines (gestrichelt)
●	8 : Kreis
- - -	9 : Linear (gestrichelt)
- · -	10 : Linear strich-punktiert
▼	11 : Dreieck auf der Spitze
◆	12 : Q uadrat auf der Spitze
💣	13 : <b>Bombe</b>
✱	14 : Daimler-Stern
◇	15 : Salmi
	16 : Strich
◊	17 : Salmi (liegend)
★	18 : fünfzackiger Stern
〰	19 : Spline mit Sinus
⋯	20 : Linear (gepunktet)
+ + + + +	21 : Perlschnur mit Kreuzen
* * * * *	22 : Perlschnur mit Sternen
▲ ▲ ▲ ▲ ▲	23 : Perlschnur mit Dreiecken
■ ■ ■ ■ ■	24 : Perlschnur mit Q uadraten
▼ ▼ ▼ ▼ ▼	25 : Perlschnur mit Dreiecken auf der Spitze
⊕	26 : aufrechtes Kreuz, füllbar
⊗	27 : Andreaskreuz, füllbar
○ ○ ○ ○ ○	28 : Perlschnur mit Kreisen
▼ ▼ ▼ ▼ ▼	31 : Perlschnur mit Dreiecken auf der Spitze
■ ■ ■ ■ ■	32 : Perlschnur mit Q uadraten auf der Spitze
	35 : Histogramm
	36 : Histogramm (gestrichelt)

Abbildung 2: Die verfügbaren Plotsymbole

- 17:** Rauten wie Symbol 15, aber liegend
- 18:** fünfzackige Sterne, füllbar
- 19:** Kubische Spline-Interpolation, aber zusätzlich mit einem Sinus verziert; Amplitude und Wellenlänge sind fest gekoppelt und können gemeinsam mit SIZE verändert werden
- 20:** Polygonzug gestrichelt (wie SYMBOL=9), jedoch mit doppelt so großen Lücken
- 21:** Perlenschnur, mit Kreuzen (Symbol 1). SIZE gibt sowohl den Abstand als auch die Größe der Symbole an.
- 22:** Perlenschnur mit Sternen (Symbol 2).
- 23:** Perlenschnur mit Dreiecken (Symbol 3). Negatives SIZE füllt die Dreiecke aus.
- 24:** Perlenschnur mit Quadraten (Symbol 4). Negatives SIZE füllt die Quadrate aus.
- 25:** Perlenschnur mit auf der Spitze stehenden Dreiecken (Symbol 11). Negatives SIZE füllt die Dreiecke aus.
- 26:** Kreuz, füllbar
- 27:** Andreaskreuz, füllbar
- 28:** Perlenschnur mit Kreisen (Symbol 8). Negatives SIZE füllt die Kreise aus.
- 31:** Perlenschnur mit Dreiecken auf der Spitze (Symbol 11). Negatives SIZE füllt die Dreiecke aus.
- 32:** Perlenschnur mit Quadraten auf der Spitze (Symbol 12). Negatives SIZE füllt die Quadrate aus.
- 35:** Histogramm (voller Strich); negatives SIZE füllt die Fläche unter der Kurve aus.
- 36:** Histogramm (gestrichelt)
- 40:** ist eine Besonderheit, denn hier werden die y-Werte des Datensatzes in Falschfarben codiert und in einem wagerechten Streifen geplottet, dessen Lage durch das COMMAND STRIP (siehe dort) festgelegt werden muss. COLOR bezeichnet bei diesem Symbol die Farbtabelle (siehe COLOR).

Die Cuts der Farbtabelle werden automatisch auf Minimum und Maximum des jeweiligen Datensatzes skaliert. Man kann die Cuts jedoch auch von Hand setzen, dann sind sie *global* für alle Datensätze. Dies geschieht durch die Definition der Variablen MIN\_COL bzw. MAX\_COL, z.B.

```
\INSTRUCTION EXPORT
\VAR MIN_COL = 123.4
\VAR MAX_COL = 5000
\INSTRUCTION NO-EXPORT
```

**Ausfüllen beliebiger Flächen:** Beliebige Flächen können mit der aktuellen Zeichenfarbe (bzw. Rasterung entsprechend deren Grauwert bei Schwarzweiß-Ausgabe) ausgefüllt werden. Die Randkurve wird dazu als Polygon (SYMBOL=5) oder Interpolationskurve (SYMBOL=6) in einem Datensatz angegeben. *Negatives* SIZE bewirkt dann das Ausfüllen. Achtung: Wenn die Fläche über den Kasten hinausragt, kann man sie anscheinend nicht mit KASDEF INBOX clippen, ohne daß Unsinn geschieht.

## 7 Erzeugen von Plot-Files mit FORTRAN-Programmen

Standard-Plotfiles können in bekannter Weise durch Aufruf der beliebten FORTRAN-Subroutines PLOTANF bzw. PLOTCON erzeugt werden. Möchte man Zeichen-Attribute (PEN=..., COLOR=..., SIZE=...) angeben, muss man die Routinen PLOTANFS und PLOTCONS verwenden. Die Routinen sind in der WRplot-Installationsdirectory in der Library (archive) *lib\_plotcalls* abgelegt. Die Parameterliste entnehme man den Programmentexten.

Die Verwendung von KASDEF-Optionen aus FORTRAN-Programmen heraus ist ebenfalls möglich:

a) Man beginnt den Plot von Hand, indem man die PLOT-Zeile schreibt:

```
WRITE (KANAL,*) 'PLOT: Text ...'
```

Der Text erscheint lediglich zur Orientierung beim interaktiven Ablauf des WRPLOT-Programms.

b) Dann schreibt man die gewünschten KASDEF-Optionen als Strings, z.B.:

```
WRITE (KANAL,*) '\ LUN ... ..'
```

usw.

c) Schließlich wird der gewohnte Teil des Plotfiles begonnen durch Aufruf von PLOTANF. Die von PLOTANF jetzt überflüssigerweise erneut geschriebene PLOT-Zeile wird von WRPLOT überlesen und ist in diesem Fall also *Dummy*.

## 8 Einbinden von Plots in Tex-Dokumente

WRPLOT erzeugt PS-Files, die sich problemlos in Tex-Dokumente einbinden lassen. Im Latex-Skript schreibt man:

a) Am Anfang:

```
\usepackage{epsf}
```

b) In der Figure-Umgebung (hinter `\begin{figure}`)

```
\epsffile{ps-filename}
```

Schon erscheint die Figure in der Originalgröße 1:1, linksbündig, an der richtigen Stelle.

Will man die Figure stattdessen zentrieren, schreibt man:

```
\centering
```

```
\mbox{\epsffile{ps-filename}}
```

Hat die Graphik im vorliegenden ps-file noch nicht die gewünschte Größe, kann man sie noch skalieren:

Z.B. auf genau volle Satzbreite:

```
\epsfxsize=\textwidth
```

```
\epsffile{ps-filename}
```

oder auf 70% der Satzbreite, zentriert:

```
\centering
```

```
\epsfxsize=0.7 \textwidth
```

```
\mbox{\epsffile{ps-filename}}
```

oder auf eine Höhe von 8cm, zentriert:

```
\centering
```

```
\epsfysize=8.0cm
```

```
\mbox{\epsffile{ps-filename}}
```

Achtung: Im zweispaltigen Stil (A&A Latex!) ist `\textwidth` immer die volle Satzbreite; bei einspaltigen Figuren muß man stattdessen `\columnwidth` angeben.



## A Musterblatt Fonts

<p><b>WRPLOT Font</b></p> <p>!"\$%&amp;'()*+,-./0123456789:;&lt;=&gt;?@          ABCDEFGHIJKLMNOPQRSTUVWXYZ          []Δ `abcdefghijklmnopqrstuvwxyz          äöüíÄÖÜß&amp;#;" ° Å * Ñ ∞ → ← ± \ °</p>	<p><b>TIMES Fonts</b></p> <p>!"\$%&amp;'()*+,-./0123456789:;&lt;=&gt;?@          ABCDEFGHIJKLMNOPQ RSTUV WX YZ          []^_ `abcdefghijklmnopqrstuvwxyz          äöüíÄÖÜß&amp;#;" ° Å * Ñ ∞ → ← ± \ •</p>
<p><b>H ELV ETIC A Fonts</b></p> <p>!"\$%&amp;'()*+,-./0123456789:;&lt;=&gt;?@          ABCDEFGHIJKLMNOPQRSTUVWXYZ          []^_ `abcdefghijklmnopqrstuvwxyz          äöüíÄÖÜß&amp;#;" ° Å * Ñ ∞ → ← ± \ •</p>	<p><b>COURIER Fonts</b></p> <p>!"\$%&amp;'()*+,-./0123456789:;&lt;=&gt;?@          ABCDEFGHIJKLMNOPQRSTUVWXYZ          []^_ `abcdefghijklmnopqrstuvwxyz          äöüíÄÖÜß&amp;#;" ° Å * Ñ ∞ → ← ± \ •</p>
<p><b>PS Symbol Font (# switch)</b></p> <p>∇ !∇∇%∇() *+,-./0123456789:;&lt;=&gt;?≡∇          ABXΔEΦΓHIΘKΛMNOPΘPΣTYζΩΞΨZ          []⊥_ αβχδεφγηιφκλμνοπθρστυωξψζ</p>	<p><b>PS Italic Font (&amp; I switch)</b></p> <p>" !"\$%&amp;'()*+,-./0 1 2 3 4 5 6 7 8 9 :;&lt; = &gt; ? @ "</p> <p><i>ABCDEF G H IJK L M NO PQ RST U VW XYZ</i>          []^_ `abcd efghijk lmnopq rstuv wxyz</p>
<p><b>WRPLOT Symbol Font (# switch)</b></p> <p>" !"\$%&amp;'()*+,-./0123456789:;&lt;=&gt;?@"          ABXΔEΦΓHIΨKΛMNOPΘPΣT~≈ΩΞΤΖ          []Δ αβχδεφγηιψκλμνοπρστ≈ωξυζ</p>	<p><i>ZAPF Fonts (Zapf-Chancery medium italic)</i></p> <p>!"\$%&amp;'()*+,-./0123456789:;&lt;=&gt;?@  <i>ABCDEF GHIJ KLMNOPQRSTU VWXYZ</i>          []^_ `abcdefghijklmnopqrstuvwxyz          äöüíÄÖÜß&amp;#;" ° Å * Ñ ∞ → ← ± \ •</p>
<p><b>Text-Attribute (&amp; -Switches)</b></p> <p>&amp;H: <sup>hoch</sup> &amp;M: mittig &amp;T: <sub>tief</sub> &amp;H...&amp;T (direkt nacheinander): <sup>hoch</sup><sub>tief</sub>          &amp;E: eng &amp;B: <b>breit</b> &amp;N: normal &amp;F: <b>fett</b> &amp;f: nicht fett          &amp;R: <i>rechtsgeneigt</i> &amp;L: <i>linksgeneigt</i> &amp;G: gerade          Brüche: \ { Zähler \   Nenner \ } → <math>\frac{\text{Zähler}}{\text{Nenner}}</math></p>	

Abbildung 3: Die unter PostScript verfügbaren Fonts

## B Musterblatt Farben

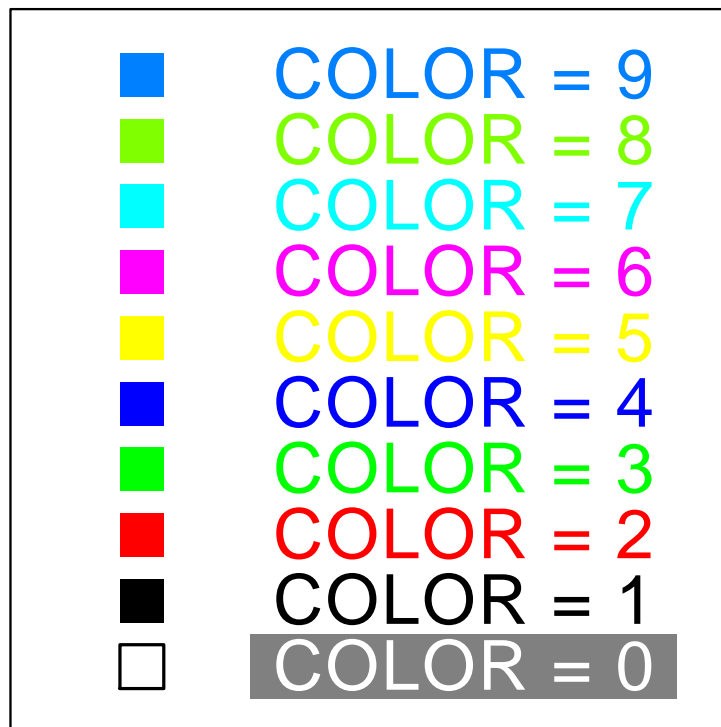


Abbildung 4: Die vordefinierten Farben 0 ... 9; 0 ist weiss

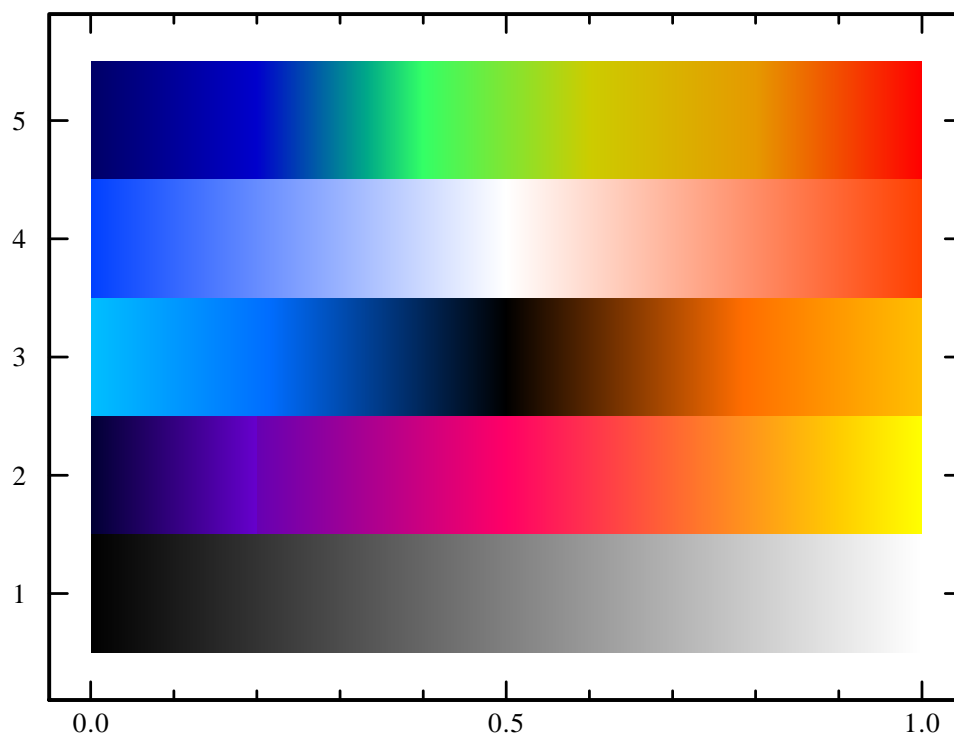


Abbildung 5: Die definierten Farbtabelle (COLOR=1...5) für die Falschfarben-Darstellung (SYMBOL=40).

# Index

- Arc of a circle, 15
- Arithmetic (Datasets), 33
- Arithmetic (Variables), 10
- ARR, arrow, 14
- ASCII-to-PostScript, 4
- AUTO-Skalierung, 25
  
- BGRLUN, 18
- BINNING, 32
- Box mit Koordinaten, 25
- Brüche, 19
  
- CALC (Arithmetik), 10
- Colors, 13
- Convolution, 32
- CUT (strings), 11
  
- Datensätze, 26
- Default pen, 8
- DEFAULT-Koordinaten, 25
- DEFAULTCOLOR, 8
- DO-Loops, 12
  
- ECHO, 10
- Ellipse zeichnen, 15
- EPSF-Files einbinden, 22
- Error bars (data), 34
- Error bars (KASDEF), 16
- EXPAND dataset, 29
- EXPR (concatenate strings), 11
  
- Faltung, 32
- FILLING of areas – datasets, 27
- FILLING of areas – KASDEF, 16
- FITS files, 28
- Fließtext schreiben, 17
- Fonts, 13
- FORMAT - formatting real numbers, 11
- FORMATFACTOR, 6
- FORMATI - formatting integer numbers, 11
- Fortsetzungszeilen, 7
  
- GETTIME, 10
- GOTO, 12
- Greek letters, 19
  
- HATCHED areas, 16
- Hinterlegen, 18
- HLINE, 19
  
- HLYMAN, 30
  
- Identification marks, 20
- IF...ELSE...ENDIF (COMMAND), 34
- IF...ELSE...ENDIF (KASDEF), 12
- INBOX, 8
- INCLUDE (COMMAND), 28
- INCLUDE (KASDEF), 9
- INFITS, 28
- Installation, 3
- INSTRUCTION EXPORT, 9
- Instructions, 10
- Interactive mode, 8
- ISMLINE, 30
- ITEMIZE, 18
  
- Kreis(bogen) zeichnen, 15
  
- LATEBOX, 9
- LaTeX, 20, 23
- LETTERSIZE, 9
- Linien zeichnen, 14
  
- Max. no. of datapairs, 9
- Max. no. of datasets, 9
- MULTIPLY, 5
  
- NOBOX, 9
  
- OFS, box offset, 8
  
- PAPERFORMAT, 5
- PAUSE, 13
- PEN, 13
- PENDEF, 8
- Permission, 3
- Pfeil zeichnen, 14
- Plotsymbols (Datasets), 36
- PREDEFINE-VARIABLES, 10
  
- READ (KASDEF), 13
- Rechteck zeichnen, 15
- Reddening/dereddening, 31
  
- SKL (scaling), 8
- Sonderzeichen, 19
- Strichstärke, 13
- Symbole zeichnen, 15
  
- Tabellen, 18

Text (String) schreiben, 16

Text einrücken, 17

TICKSIZE, 9

ttw, 4

Umlaute, 19

VAR-LIST – list of variables, 11

Variable, 10

WAVECAL, 29

WRITE (KASDEF), 13

wrmult, 4

wrps, 4