GRAZIL

Beschreibung der Version 6.1 des Plotpakets

J. Langendorf O. Paetsch

28. Juni 1994

Zusammenfassung

GRAZIL ist ein interaktives Programmpaket zur grafischen Darstellung von zwei-dimensionalen Kurvenverläufen. Dem Benutzer stehen zahlreiche Kommandos und ein grafisches User-Interface zum Gestalten des Layouts der Zeichnung zur Verfügung. Die Eingabedaten müssen dem GRAZIL-Eingabe-Format genügen. Somit wird eine hohe Flexibilität und eine große Bandbreite der Einsatzmöglichkeiten erreicht. GRAZIL wurde mit der grafischen Grundsoftware GKS entwickelt. Dadurch kann ein breites Rechner- und Ausgabegerätespektrum genutzt werden.

Inhaltsverzeichnis

1 Einführung

Das Softwarepaket GRAZIL (GRAphical Zib Language) [LAN] hat seinen historischen Ursprung in einem einfachem Postprozessor zur grafischen Darstellung von Trajektorien gewöhnlicher Differentialgleichungen. Die darzustellenden Daten hatten folgende Struktur: eine im voraus nicht festlegbare Anzahl monoton steigender X-Werte (Stützstellen), sowie zu jedem X-Wert n zugehörige Y-Werte (Lösungsvektor). Das aus diesen Anforderungen heraus entwickelte Programm war stark an dieser Struktur orientiert und wenig flexibel, um Daten anderer Struktur zu visualisieren.
Es bot es sich an, ein Plotprogramm mit einem *offenen Lesemodul* zu implementieren und die Steuerung über Kommandos zu realisieren. Die ersten Versionen wurden in FORTRAN unter dem IBM-Betriebssystem MVS erstellt und dann später nach UNIX und VMS portiert. Zur grafischen Ausgabe wird die 2D-Grafik-Bibliothek GKS verwandt. In den folgenden Kapiteln werden daher häufig Bezeichnungen aus der GKS-Welt verwendet. In Anhang C werden diese Begriffe näher erläutert.
Das Softwarepaket GRAZIL ist nach seinem Erscheinen im ZIB vielfach genutzt worden. In Diskussionen mit den Benutzern sind neue Ideen und Anforderungen an das Paket entstanden. Deren Verwirklichung erforderte eine Erweiterung der Benutzerkommandos. Desweiteren wurde eine grafische Benutzungsoberfläche entwickelt, so daß der Anwender nicht mehr die in Kap. 7 beschriebenen Kommandos und deren vielfältige Parameter kennen und eingeben muß, sondern GRAZIL über Mausklicks steuern kann.
In der vorliegenden Version von GRAZIL können folgende grafische Elemente verwendet werden:

- verschiedene Darstellungsarten (auch gleichzeitig verwendbar):
  - Linienplots mit und ohne Markersymbole
  - Balkendiagramme
  - Fehlerbalken
- Achsen mit Beschriftung (bis zu vier unabhängige pro Plot)
- Legenden und
- Überschriften.

Es können nur numerisch vorliegende Daten mit GRAZIL gezeichnet werden, Plots durch Angabe von Funktionen, wie in *Gnuplot* [GNU], sind nicht möglich. Die Daten können nur in zwei Raumdimensionen dargestellt werden; es sind also nur zweistellige Relationen $(x_i, y_i)_{i \in I}$ darstellbar.

2 Einfache Beispiele

Bevor wir die Möglichkeiten von GRAZIL systematisch darstellen, starten wir zunächst mit zwei einfachen Anwendungsbeispielen. Ausgangspunkt für ein er-

stes Beispiel ist eine kurze Tabelle mit Temperaturen und Regenmengen je Monat:

Monat	Regenmenge		Temperatur
	monatl.	akkum.	
1	42.	42.	0.0
2	30.	72.	1.0
3	38.	110.	3.8
4	43.	153.	8.5
5	49.	202.	13.8
6	54.	256.	17.0
7	71.	327.	19.0
8	59.	386.	18.0
9	49.	435.	14.5
10	40.	475.	9.5
11	38.	513.	4.0
12	48.	561.	1.0

Um diese Tabelle mit GRAZIL darzustellen, muß sie in die Form

```
&name rain_m, rain_accu, temp[1]
&def (x = 1, y = 3) * 12[2]

&data
      1  42.   42.   0.0
      2  30.   72.   1.0
      3  38.  110.   3.8
      4  43.  153.   8.5
      5  49.  202.  13.8
      6  54.  256.  17.0
      7  71.  327.  19.0
      8  59.  386.  18.0
      9  49.  435.  14.5
     10  40.  475.   9.5
     11  38.  513.   4.0
     12  48.  561.   1.0
&end
```

gebracht und in einer Datei, beispielsweise `rain+temp` abgelegt werden. Wesentlich zum Verständnis von GRAZIL ist die Tatsache, daß (x, y)-Wertepaare, die als *eine* Kurve erscheinen sollen, einen Namen mit dem Kommando `&name` erhalten. Dieser Name[3] dient sowohl zur Auswahl, welche Kurven im Plot gezeichnet werden sollen, als auch zum Setzen von Darstellungsparametern soweit sie sich auf die auswählbaren Kurven beziehen, also z.B. Farbe, Linienart und Linienstärke einer Kurve. Da GRAZIL sowohl mit Kommandos als auch über

[1] Vereinbarung von drei Namen für die drei darzustellenden Kurven.
[2] Definition der Struktur des Datensatzes (für jeden Monat drei Werte).
[3] Nachfolgend auch als Variablenname bezeichnet.

eine grafische Benutzungsschnittstelle (GUI) gesteuert werden kann, werden im folgenden beide Möglichkeiten jeweils gegenüber gestellt.

In einem ersten Beispiel soll die Temperaturkurve unter Ausnutzung der Voreinstellungen von GRAZIL dargestellt werden.

Steuerung über die Kommandoschnittstelle

Aufruf von GRAZIL

```
%grazil -c rain+temp
```

Auswahl des Ausgabegerätes, z.B.[a]

```
grazil>&wkt 3100
```

Auswahl der darzustellenden Kurve

```
grazil>&vars = temp
```

Darstellen des Bildes

```
grazil>&run
```

Beenden von GRAZIL

```
grazil>&stop
```

[a] falls an einer SUN-Workstation mit 15" Farbmonitor gearbeitet wird.

Steuerung über GUI

```
%grazil rain+temp
```

Das GRAZIL-Grundfenster erscheint. Dort den Menüknopf Settings drücken und Menü Workstation ... wählen. Es erscheint ein neues Fenster. Dort im Menü Set Workstation Type beispielsweise SUN X Terminal 15" Colour auswählen und den apply- Knopf drücken. Anschließend im GRAZIL-Grundfenster im Menüknopf Settings das Menü Vars ... wählen und im jetzt erscheinenden Fenster die Variable TEMP aus der Liste der selectable Vars anklicken. Drücken des apply- Knopfes läßt den Plot erscheinen. Der quit- Knopf beendet GRAZIL.

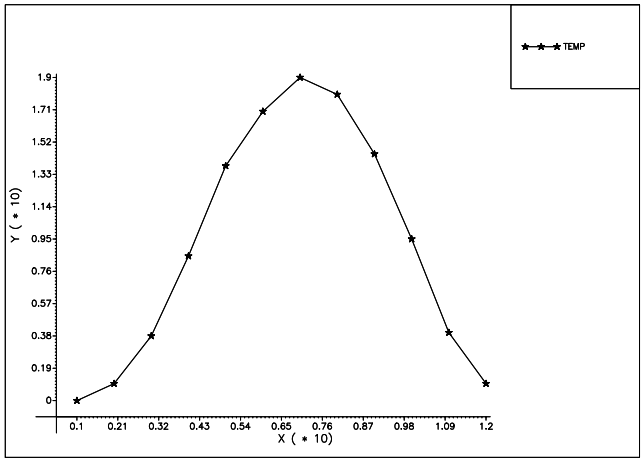

Fig. 1 Darstellung der Temperaturkurve

Ein Problem ist, das die Defaults in GRAZIL für Papierausgabe in der Größe von A4-Format optimiert sind. Dies hat zur Folge, das die Schriften für kleine Bilder, wie z.B. in dieser Beschreibung, zu klein sind. Dieses Problem kann durch Einlesen eines Kommandofiles, wie in Anhang F aufgeführt, behoben werden.

Sollen Temperatur und Regenmenge zu Vergleichszwecken in *einem* Plot gezeichnet werden, muß wegen des unterschiedlichen Wertebereichs je eine Achse pro Kurve verwendet werden. Es werden die zwei Kurven übereinandergelegt (GRAZIL: in zwei Layer gezeichnet). Daneben bietet es sich an, die Regenmenge als Treppendiagramm zu zeichnen. Um die Gegenüberstellung der beiden Schnittstellen nicht zu unübersichtlich werden zu lassen, werden für das GUI Fenster und Menüs, in denen Einstellungen vorzunehmen sind, aufgeführt.
Die Wirkungsweise und die Parameter aller hier aufgeführten Kommandos können im Kap. 7 nachgeschlagen werden.

Steuerung über die Kommandoschnittstelle

```
%grazil -c rain+temp

grazil>&wkt 3100
grazil>&comfile 'doku.layout'
grazil>&layer 1
grazil>&vars = temp
grazil>&axis x = = 1.
grazil>&layer 2
grazil>&vars = rain_m
grazil>&axis x del
grazil>&axis y lin max
grazil>&bounds y 3 30. =
grazil>&render rain_m
           -curve 2 1 0 0 3
grazil>&prof -curve
           rain_m = = = = = = 4
grazil>&grid +y
grazil>&run
grazil>&stop
```

Steuerung über GUI

```
%grazil rain+temp
```

Das GRAZIL-Grundfenster erscheint.

File → load

Settings → Workstation
→ SUN X Terminal

Settings → Vars
→ Layer1 → temp
→ Layer2 → rain_m

Properties → Axis → Layer2
→ Y-axis → Intersection
→ X-axis → Mode

Properties → Render Attributes
→ rain_m
→ Render Curve
→ Interpol.
→ Stepmode

Properties → Grid
→ Vertical→ on

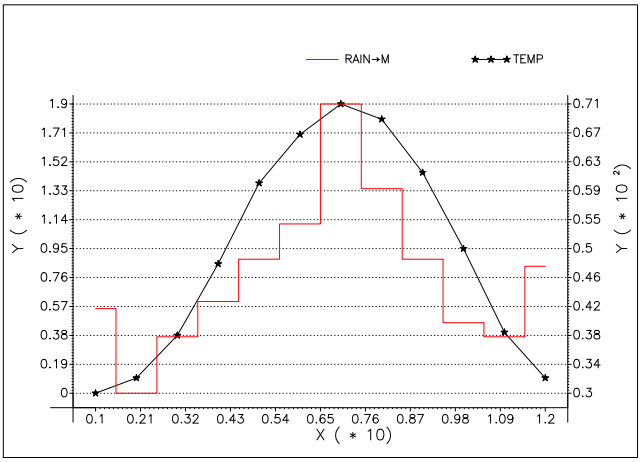

Fig. 2 Die Kurven *temp* und *rain_m*, dargestellt in zwei Layern.

Dieses Beispiel macht die Vorteile der Verwendung des GUI offenkundig. Aus historischen Gründen wird in der Kommandoschnittstelle mit stellungsgebundenen Parametern gearbeitet, so daß der Anwender bei den einzelnen Kommandos alle Parameter, einschließlich ihrer Reihenfolge kennen muß (die Gleichheitszeichen = stehen für nicht spezifizierte Parameter). Bei Steuerung von GRAZIL durch das GUI sind diese Kenntnisse nicht mehr notwendig.

3 Aufbau von GRAZIL

3.1 Struktur

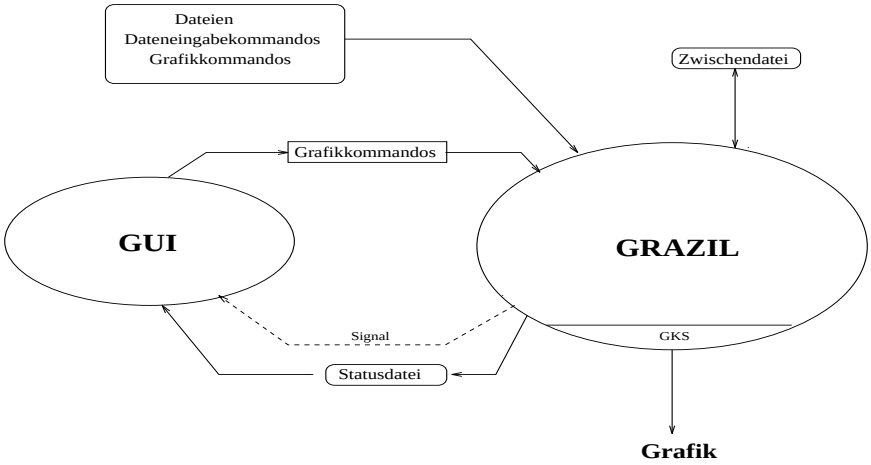

Fig. 3 Daten- und Kontrollfluß.

GRAZIL macht bei der Eingabe keinen Unterschied zwischen Daten und Kommandos, beides kann zusammen in der eingelesenen Datei vorhanden sein. Eingelesene *Daten* werden in einer Zwischendatei abgelegt. Jedes Kommando, auch das Daten einleitende Kommando `&data`, verändert den internen Zustand von GRAZIL. Erst bei Eingabe des Kommandos `&run` wird überprüft, ob die gewünschten Grafikattribute (z.B. Farbe) auf dem gewählten Ausgabegerät möglich sind. Ist beispielsweise eine Farbe nicht darstellbar, wird, nach vorheriger Warnung, in der voreingestellten Farbe (Schwarz-Weiß) ausgegeben. Die Grafikausgabe erfolgt über GKS. Die Kommunikation und Synchronisation der beiden getrennten Hauptprogramme, GRAZIL und GUI, erfolgt über eine Statusdatei und die Unix-Mechanismen Pipe und Signal. GRAZIL schreibt seinen internen Zustand in die Statusdatei und informiert das GUI über ein Signal, wenn diese Datei neu geschrieben worden ist. Das GUI liest diese Datei und zeigt die Zustände in seinen Fenstern an. Eingaben im GUI werden in GRAZIL-Kommandos gewandelt und über eine Pipe an GRAZIL als Eingabe übergeben.

4 Aufruf und Dateireferenzen des Programmpakets GRAZIL

Eine typische Sitzung mit GRAZIL sieht folgendermaßen aus: Der Benutzer ruft GRAZIL unter Angabe einer Datei, die Daten gemäß dem GRAZIL-Eingabeformat enthält, auf. GRAZIL liest diese Datei ein und startet die grafische Benutzeroberfläche. Wird GRAZIL mit der Option `-c` aufgerufen, geht es in den Kommandomodus und meldet sich mit dem Prompt grazil>. Der Benutzer spezifiziert die gewünschten Darstellungsoptionen und gibt das Bild aus, z.B. zunächst in einem Fenster auf einem Monitor, und dann, nach eventuellen Korrekturen, auf einem Drucker. Wenn der Benutzer umfangreiche Darstellungsoptionen gesetzt hat, wird er diese vor dem Verlassen von GRAZIL in einer Datei sichern, um sie später wieder verwenden zu können.

4.1 Systemvoraussetzungen

Die aktuelle Version von GRAZIL läuft unter SUN OS4.1.x und SUN OS5.1(Solaris 2.1). GRAZIL ist in FORTRAN implementiert und benutzt für die grafische Ausgabe GKS Level 0a. Das GRAZIL-GUI ist ein C-Programm und verwendet das Athena Widget Set (Xaw) und die XToolkits (Xt).
Es ist geplant, GRAZIL in der *eLib*[1] des ZIB zur Verfügung zu stellen.

4.2 Aufruf von GRAZIL

Aufruf[2]: `grazil [-c | -i] [ <indatei_1> [ <indatei_2> [<indatei_3> ] ] ]`

Bei Angabe von `-c` startet GRAZIL nur mit der Kommandoschnittstelle, also ohne GUI. Wird keine Option oder `-i` angegeben, erscheint die grafische Benutzungsoberfläche. Es können bis zu drei Dateien angegeben werden. Sie müssen Kommandos und Daten im GRAZIL-Format enthalten. Die Dateien werden in der angegebenen Reihenfolge abgearbeitet.

4.3 Dateireferenzen von GRAZIL

Folgende Environment-Variablen und Dateien werden von GRAZIL verwendet:

- GKSERR, zur Angabe der Datei mit den GKS-Fehlermeldungen[3]:
  `%setenv GKSERR "/zib/etc/gks/gkserr"`
- GKSFNT, zur Angabe der Datei mit den GKS-Textfonts[3]:
  `%setenv GKSFNT "/zib/etc/gks/gksfnt"`
- GKSWDT, zur Angabe der Datei mit der GKS-Workstation Description Table[3]:
  `%setenv GKSWDT "/zib/etc/gks/gkswdt"`

[1] Elektronische Bibliothek mit Zugriff auf Software und Publikationen

[2] Die am ZIB allgemein verfügbare GRAZIL-Version für SUN-Workstations befindet sich auf dem SUN-Server SERV04 und ist ansprechbar über den Pfad /usr/local/zib_bin/grazil.

- Für das Kommando &reset:
  Bei jedem Aufruf des Kommandos &run wird eine Datei mit dem Namen `tmp2d.run_<n>` (n = 0 ... 9) im aktuellen Dateiverzeichnis angelegt. In diese Datei werden die aktuellen GRAZIL-Einstellungen gesichert (Autosave). Das erlaubt, durch Aufruf des Kommandos &reset, den vorherigen Zustand wieder herzustellen. Beim Beenden von GRAZIL werden diese Dateien gelöscht.

- `.grazil.zugr`:
  Wenn eine Datei dieses Namens im aktuellen Dateiverzeichnis existiert, wird sie vor allen anderen Eingabedateien gelesen.

- `GRAZIL-gks_error`:
  Auf diese Datei im aktuellen Dateiverzeichnis werden die GKS-Meldungen geschrieben. Die Datei wird eingerichtet, wenn sie nicht vorhanden ist.

- `indatei_1.gksm`:
  Auf eine Datei dieses Namens im aktuellen Dateiverzeichnis wird der Metafile-Output geschrieben[4].

- `indatei_1.cgm`:
  Auf eine Datei dieses Namens im aktuellen Dateiverzeichnis wird der CGM-Output geschrieben[4].

- `indatei_1.hpgl`:
  Auf eine Datei dieses Namens im aktuellen Dateiverzeichnis wird Plotoutput für HP7475-Plotter ausgegeben[4].

- `indatei_1.ps`:
  Auf eine Datei dieses Namens im aktuellen Dateiverzeichnis wird der Output für PostScript ausgegeben[4].

- `indatei_1.eps`:
  Auf eine Datei dieses Namens im aktuellen Dateiverzeichnis wird der Output für Encapsulated PostScript ausgegeben[4].

Mit Hilfe der folgenden Environment-Variablen können Voreinstellungen für das GUI definiert werden. Sie werden nur einmal beim Start gelesen und sind optional. Die Environment-Variablen GRAZIL_RUN_ON_APPLY und GRAZIL_INPUT_MODE können auch jederzeit in einer GRAZIL-Sitzung durch Auswahl der entsprechenden Menu-Items im Settings-Menu des Kontrollfensters verändert werden.

[3]Die Namen der Files sind von der GKS-Implementation und Instellation abhngig. Bei der im ZIB vorhandenen GKS-Implementierung müssen die Enviroment-Variablen GKSERR, GKSFNT und GKSWDT gesetzt werden.

[4]Die Datei wird eingerichtet, wenn sie nicht vorhanden ist. Wurden die Daten nicht über eine Datei (indatei_1 oder mit dem Kommando &comfile) eingelesen, lautet der Präfix der Ausgabedatei fort.xx.Falls im Workstationkommando eine Ausgabedatei angegeben wird, hat diese Angabe Vorrang.

– `GRAZIL_RUN_ON_APPLY`
Diese Variable steuert das Verhalten des GUI beim Betätigen des Apply-Knopfes in allen, über das Properties- und Settings- Menü zu öffnenden Fenstern.

`%setenv GRAZIL_RUN_ON_APPLY 0`

Beim Anklicken von Apply werden die aktuellen Einstellungen nur an GRAZIL übertragen, aber noch nicht ausgewertet. Erst nach Betätigen des Run- Knopfes im GRAZIL-Kontrollfenster werden die neuen Einstellungen von GRAZIL ausgewertet und angezeigt.

`%setenv GRAZIL_RUN_ON_APPLY 1`

Beim Anklicken von Apply werden die Einstellungen an GRAZIL gesendet, sogleich ausgewertet und angezeigt.

Default: GRAZIL_RUN_ON_APPLY 1

– `GRAZIL_INPUT_MODE`
Diese Variable steuert das Übernahmeverhalten bei Dialogeingabe.

`%setenv GRAZIL_INPUT_MODE 0`

Dialogeingabe wird nur nach Abschluß mit Return übernommen.

`%setenv GRAZIL_INPUT_MODE 1`

Dialogeingabe wird mit dem Apply- Knopf übernommen.

Default: GRAZIL_INPUT_MODE 1
Die Variable GRAZIL_INPUT_MODE wirkt nicht in den File- Fenstern. In diesen ist das Verhalten stets so, als wäre die Variable auf 0 gesetzt.

– `GRAZIL_LOAD_PATH`
Mit dieser Variablen kann der Anfangssuchpfad, d.h. der Startpunkt im Dateibaum, für das File-Load-Fenster eingestellt werden.

Default: das aktuelle Dateiverzeichnis.

– `GRAZIL_SAVE_PATH`
Mit dieser Variablen kann der Anfangssuchpfad, d.h. der Startpunkt im Dateibaum, für das File-Save-Fenster eingestellt werden.

Default: das aktuelle Dateiverzeichnis.

– `GRAZIL_LOAD_DISSCR` und `GRAZIL_SAVE_DISSCR`
Mit diesen Variablen kann ein Suchmuster für das jeweilige File-Menü angegeben werden. Es gelten die von UNIX her bekannten Metazeichen '?' und '∗' (Wildcards).

Default: '∗', d.h. alle Dateien im eingestellten Dateiverzeichnis werden angezeigt.

Resourcen für die Fenster und Widgets des GUI können mit den üblichen X-Resource-Files gesetzt werden.

Beispiele:

ggui∗background: LightSkyBlue
: Für das gesamte GUI wird die Farbe LightSkyBlue als Hintergrundfarbe gesetzt.

ggui∗ <name>.background: <Farbe>
: Für das Widget <name> wird die Farbe <Farbe> gesetzt.

ggui∗ <name>.<resource>: <Wert>
: Eine Resource des Widgets <name> wird für das GUI auf den Wert <Wert> gesetzt.

Für <name> kann der Name eines Widgets im GUI angegeben werden. Dann wird die Resource genau für dieses Widget gesetzt. Auch der Name einer Widgetklasse kann angegeben werden, so daß für alle Widgets eines Types eine Resource gesetzt werden kann. Folgende Klassen werden im GUI verwendet:

Command	(für alle Knöpfe)
MenuButtons	(für alle Menuknöpfe)
SimpleMenu	(für alle Menueinträge)
Toggle	(für alle Schalter)
Dialog	(für alle Eingabebereiche)
List	(für alle Listen)
Label	(für alle festen Texte).

Folgende Resourcen sind im GUI sinnvoll zu nutzen:

background	Hintergrundfarbe
foreground	Vordergrundfarbe
shapeStyle	Stil von Knöpfen; möglich sind: rectangle, oval, ellipse und roundedRectangle.
cornerRoundPercent	Rundungsangabe für den ShapeStyle roundedRectangle.

Der im GUI verwendete Textfont sollte nicht verändert werden! Ein anderer Textfont würde die Größe der einzelnen Widgets verändern, was zu Konflikten beim Aufbau der Fenster des GUI führen kann.

5 Datenschnittstelle

5.1 Syntax

In einer GRAZIL-Datendatei werden zunächst alle Kurvennamen[1] deklariert, denen dann die eigentlichen Wertepaare zugewiesen werden sollen. Durch Angabe dieser Namen können in der Folge die darzustellenden Kurven ausgewählt werden. Vor Angabe der Daten ist deren Format zu definieren. Das Paar (Definition, Daten) kann mehrfach in einer Datei vorkommen. Die Reihenfolge Namensvereinbarung, Definition, Daten ist zwingend vorgeschrieben.
In den nachfolgenden Syntaxbeschreibungen bedeutet:

`< ... >`	nicht terminales Symbol, d.h. metasprachliche Variable, die noch weiter definiert werden muß
`[ ... ]`	optional
`... *`	kann mehrfach angegeben werden
`|`	exklusives `"oder"`

5.2 Namensvereinbarung

Jeder Name steht intern für eine Menge von (x, y)-Wertepaaren, die zusammen eine Kurve bilden.

Kommandosyntax

```
&name [<name1>, <name2>, ..., <name-n>] | [< n >]
```

*name*1 *name*2 ... *name* − *n*	:: Es werden die *n* angegebenen Namen als Kurvennamen vereinbart. -oder-
n	:: Es werden *n* Kurvennamen (*V1* - *Vn*) vereinbart.

Die Namensvereinbarung muß genau einmal vor der ersten Datendefinition vorgenommen werden. Nach einer erneuten Namensvereinbarung ist eine evtl. vorher gegebene Definition aufgehoben, einschließlich der dazugehörigen Daten.

5.3 Definition der Daten

Nach dem Schlüsselwort `&def` ist die Anzahl der X- und Y-Werte anzugeben. Dabei wird der Achsenbezeichner (`x` oder `y`), gefolgt von einem Gleichheitszeichen und der Anzahl der einer Achse zuzuordnenden Datenwerte, angegeben. Die Reihenfolge der Achsenbezeichner ist frei bzw. wird von der Struktur der Daten vorgegeben.
Die einfachste Definition, `&def x=1,y=1`, bedeutet, daß zwei Datenwerte auftreten, von denen der erste als X-Wert und der zweite als Y-Wert angesehen wird.

[1] Nachfolgend auch als Variablenname bezeichnet.

Bei der Datendefinition muß gewährleistet sein, daß allen vereinbarten Namen auch Werte eindeutig zugeordnet werden können. Dazu muß die Länge eines X- oder Y-Vektors mit der Anzahl der deklarierten Namen in einem ganzzahligen Verhältnis stehen. Beispielsweise bedeutet `&def x=1, y=6`, daß zu je einem X-Wert ein Y-Tupel mit sechs Werten in den Daten vorhanden ist. Wenn sechs Kurvennamen deklariert wurden, dann werden jedem Namen neben den X-Werten die Werte je einer Y-Komponente zugewiesen.

richtig:
```
&name a1 a2 a3 a4 a5 a6
&def x=1, y=6
-oder-
&def x=6, y=1
```

Im ersten Fall werden für alle 6 Kurven *a1 ... a6* dieselben X-Werte verwendet, nämlich der 1., 8., 15., ... Datenwert, im zweiten Fall jeweils dieselben Y-Werte (7., 14., 21., ... Datenwert).

falsch:
```
&name a1 a2 a3 a4 a5 a6
&def x=1, y=5.
```

Neben den 2-dimensionalen Datenfeldern sind auch 1-dimensionale Datenfelder möglich. Letztere können z.B. zur Definition symmetrischer Fehlerbalken dienen.

Beispiel

```
&name a1
&def x=6
```

Für die Anzahl der Datenwerte kann auch ein arithmetischer Ausdruck der Form X = n ∗ m, angegeben werden. Das bedeutet, daß insgesamt n ∗ m Datenwerte der X-Achse zugeordnet werden sollen. Lautet die Definition beispielsweise `&def x = 3 * 4` und ist die Anzahl der Namen gleich 3, dann werden die ersten 12 Datenwerte folgendermaßen zugeordnet: Für jede der drei Kurven folgen zunächst deren jeweils erster X-Wert (die ersten drei Datenwerte), dann deren zweiter (die nächsten drei Datenwerte), usw. Sind dagegen 4 Namen deklariert, dann werden die ersten drei Datenwerte als drei aufeinanderfolgende X-Werte der ersten Kurve, die nächsten drei als X-Werte für die zweite Kurve usw. interpretiert. Anders ausgedrückt: Ist x_i^n der i-te Wert der Kurve a_n, so werden nach den Definitionen

```
&name a1 a2 a3
&def x = 3 * 4, y = ...
```

die ersten 12 in der `&data`-Anweisung erscheinenden Datenwerte als

$$x_1^1\, x_1^2\, x_1^3 \quad x_2^1\, x_2^2\, x_2^3 \quad x_3^1\, x_3^2\, x_3^3 \quad x_4^1\, x_4^2\, x_4^3$$

interpretiert. Bei

```
&name a1 a2 a3 a4
```

```
&def x = 3 * 4, y = ...
```

hingegen stellen die ersten 12 Datenwerte

$x_1^1\, x_2^1\, x_3^1 \quad x_1^2\, x_2^2\, x_3^2 \quad x_1^3\, x_2^3\, x_3^3 \quad x_1^4\, x_2^4\, x_3^4$

dar.
In der Datendefinition sind auch Klammerausdrücke einfacher Schachtelungstiefe mit Multiplikationsfaktoren zugelassen.

Beispiel

```
&name druck
&def (x = 1, y = 1) * 5
&data 1 11 2 12 3 13 4 14 5 15
```

Interpretation der Datenwerte: Es sind 5 *(x,y)*-Datenpaare angegeben, nämlich (1,11), (2,12), (3,13), (4,14), (5,15).
Wenn innerhalb der gleichen Datei die Datendefinition eines Namens geändert werden soll, so darf dabei nicht die Dimension geändert werden.
In einer Definition kann vor Nennung der Achse ein Datenbezeichner in der Form
Datenbezeichner = *Achsenbezeichner* = *n*
angegeben werden. Der Wert des Datenbezeichners wird als Voreinstellung für den an der X-Achse anzubringenden Text[2] verwendet. Er kann dann auch im Datenblock nach der &data-Anweisung angegeben werden. Beispielsweise:

```
&def MONAT = X = 12, ...
&data MONAT = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ...
                  ( hier müssen genau 12 Datenwerte stehen!)
```

Sind in den Daten irgendwelche Datenbezeichner in der oben genannten Form vorhanden, so müssen sie vorher mit &def definiert sein.
Wenn in &def und &data kein Datenbezeichner angegeben ist, wird der Achsenbezeichner zur Beschriftung der Achsen verwendet. Auch die Achsenbezeichner können in der &def-Anweisung weggelassen werden, es muß dann allerdings ein Datenbezeichner vorhanden sein. In diesem Fall wird der erste Datenbezeichner gleichzeitig als Achsenbezeichner X und der zweite als Achsenbezeichner Y angesehen.

[2] Achsentexte werden bei dieser Art der Einstellung in Großbuchstaben ausgegeben.

Beispiel

```
&def time = 1, value = 1
```

entspricht `&def x = 1, y = 1` und der Beschriftung der X- bzw Y-Achse mit "time" bzw. "value".
Soll sich eine Datendefinition nur auf eine Teilmenge der vereinbarten Namen beziehen, wird die Liste der Namen nach `&def` in der Form $name_1$ $name_2$ ... $name_n$ angegeben.
Sind in den Daten Werte vorhanden, die nicht einer der Achsen zugeordnet werden sollen und demnach in der Grafik auch nicht dargestellt werden sollen, dann müssen sie als Dummywerte durch ein vorgestelltes %= gekennzeichnet werden.

Beispiel Wenn die Temperatur- und Regentabelle aus Kap. 2 eingelesen wird, die akkumulierte Regenmenge in der dritten Spalte allerdings ignoriert werden soll, dann muß folgende Definition verwendet werden:

```
&name rain_m, rain_accu, temp
&def rain_m, temp : (x = 1, y = 1, % = 1, y = 1) * 12
```

Bei Dummywerten können auch Datenbezeichner mit angegeben werden, z.B. `DUMMY = % = 5`. In einer Definition darf jeder Achsenbezeichner auch mehrfach vorkommen.

Nachfolgend die komplette Syntax für die Datendefinition:

```
Datadefinition          ::= &DEF [ <Namelist> ] <Datastructure>

Namelist                ::= <Varnamelist> :
Varnamelist             ::= <Varname> , <Varnamelist> | <Varname>
Varname                 ::= <Characterstring> |
                            [<Characterstring>]<Wildcard>
Wildcard                ::= * | ?
Characterstring         ::= <Character> |
                            <Character><Characterstring>
Character               ::= a|b|c...|z|A|B|C...|Z|0|1|...|9

Datastructure           ::= <Axisdefinitionlist> |
                            <Compound Axisdefinition> |
                            <Axisdefinitionlist> ,
                                     <Compound Axisdefinition> |
                            <Compound Axisdefinition> ,
                                     <Axisdefinitionlist> |
                            <Axisdefinitionlist> ,
                                     <Compound Axisdefinition> ,
                                     <Axisdefinitionlist>

Compound Axisdefinition ::= ( <Axisdefinitionlist> ) *
```

```
                                        <Number of Items> |
                              ( <Axisdefinitionlist> ) *
                                        <Number of Names>

Axisdefinitionlist      ::= <Axisdefinition> |
                                    <Axisdefinition> ,
                                    <Axisdefinitionlist>

Axisdefinition          ::= [Dummydefinition ,] <Axisstructure>
                            [, Dummydefinition]

Axisstructure           ::= <Axiskey> = <Axisexpression> |
                            <Axis-Id> = <Axiskey> = <Axisexpression> |

                            <Axis-Id> = <Axisexpression>[3]
Axis-Id[4]               ::= <Characterstring>

Axiskey                 ::= X | Y | Z
Axisexpression          ::= <Number of Items> |
                            <Number of Items> * <Number of Names> |
                            <Number of Names> * <Number of Items>

Dummydefinition         ::= % = <Number of Items>

Number of Items         ::= 1|2|3 ...


Number of Names         ::= N[5]
```

5.4 Daten

Die Daten sind nach dem Schlüsselwort `&data` entsprechend der vorangegangenen Datendefinition anzugeben. Ein Datenblock reicht immer von dem `&data`-Kommando bis zum nächsten mit `&` beginnenden Schlüsselwort. Die einzelnen Datenwerte können mit Datenbezeichnern eingeleitet werden, wenn diese im vorangehenden `&def`-Kommando vereinbart wurden. Wenn Datenbezeichner verwendet werden, kann immer nur der gesamte der Achse zugeordnete Wertetupel mit einem Achsenbezeichner eingeleitet werden.
In einem Datenblock müssen sich immer genau so viele Datenwerte befinden, wie in dem letzten gültigen `&def`-Kommando deklariert wurden. Wenn im `&def`-Kommando für X oder Y nur ein Wert angekündigt wurde, dann werden die

[3] In dieser Form sind nur 2 `Axisstructures` erlaubt, und es gilt die Reihenfolge: X = erste `Axisstructure`, Y = zweite `Axisstructure`.

[4] Bezeichner für die Daten, der als Voreinstellung für den entsprechenden Achsentext verwendet wird.

[5] Es wird die Anzahl der für diese Definition gültigen Variablennamen ergänzt. Wenn in der Definition eine `Namelist` angegeben wurde, so wird die Anzahl der dort angegebenen Namen genommen, andernfalls gilt die Anzahl der beim letzten `&name`-Kommando vereinbarten Variablen.

weiteren Datenwerte für X oder Y in den folgenden Datenblöcken erwartet. Werden für X und Y in der Definition mehr als ein Wert angegeben, so können weitere Daten für dieselben Namen nur nach erneuter Definition angegeben werden.

Kommandosyntax

```
&data <wert1>, <wert2>, ..., <wert-n>
```

$wert_1\ wert_2\ ...\ wert_n$ :: Datenwerte oder Datenbezeichner

Die Vereinbarungs- und Definitionskommandos können über mehrere Zeilen bis zum Beginn des nächsten, mit einem & beginnenden Kommandos (Schlüsselwort) reichen. Datenwerte können durch Leerzeichen, Komma oder beides getrennt werden.

Beispiele

Beispiel 1

Eine GRAZIL-Inputdatei könnte die nachfolgend angegebene Struktur haben. Es ist hierbei im voraus nicht festgelegt, wieviele Datenböcke eingelesen werden.

```
&NAME VAL1  VAL2  VAL3  VAL4  VAL5  VAL6  VAL7
&DEF
T = 1, C = N
&END

&DATA
T=   0.000000000000D+00
C=   0.1000000D-06  0.5000000D-07  0.0000000D+00  0.0000000D+00
     0.0000000D+00  0.0000000D+00  0.0000000D+00
&END
&DATA
T=   0.160626572133D-05
C=   0.1000000D-06  0.5000000D-07  0.1040122D-23  0.4817827D-18
     0.8733438D-22  10689D-22  0.4818797D-18
&END
&DATA
T=   0.162232837854D-03
C=   0.1000000D-06  0.5000000D-07  0.9189164D-18  0.4923452D-16
     0.6033600D-18  0.3420592D-19  0.4866985D-16
&END
     .
     .   (weitere 19 Data-Blöcke)
```

```
     .
&DATA
T=   0.800000000000D-01
C=   0.0000000D+00  0.4699259D-08  0.6251866D-07  0.7642479D-08
     0.1838245D-09  0.1838147D-09  0.1688180D-13
&END
```

Die Zuordnung der Daten zu den Kurven und den Achsen ist wie folgt

```
VAL1:           X              Y
    1. : 0.0000000D+00  0.1000002D-06
    2. : 0.1606269D-05  0.1000002D-06
    3. : 0.1622330D-03  0.1000002D-06
             . . .
   23. : 0.7999998D-01  0.0000000D+00

VAL2:           X              Y
    1. : 0.0000000D+00  0.5000038D-07
    2. : 0.1606269D-05  0.5000038D-07
    3. : 0.1622330D-03  0.5000038D-07
               . . .
   23. : 0.7999998D-01  0.4699309D-08
```

Den anderen Variablen werden die Daten in gleicher Weise zugeordnet.

Beispiel 2

Hier werden verschiedene Definitionen für jeweils nur einige der deklarierten Namen verwendet, wobei für eine Kurve mehrere Definitionen angegeben werden:

```
&NAME VARIABLE_1, VARIABLE_2, VARIABLE_3, VARIABLE_4, VARIABLE_5
&DEF VARIABLE_1,VARIABLE_2:(X=1,Y=1)*N
&DATA 1101.0 1201.0 2101.0 2201.0
&DATA 1102.0 1202.0 2102.0 2202.0
&DATA 1103.0 1203.0 2103.0 2203.0
&DATA 1104.0 1204.0 2104.0 2204.0
&DATA 1105.0 1205.0 2105.0 2205.0
&DEF VARIABLE_3:(X=3,Y=3)
&DATA 3101.0 3102.0 3103.0 3201.0 3202.0 3203.0
&DEF VARIABLE_1,VARIABLE_3,VARIABLE_5:(X=N,Y=N)*4
&DATA 1106.0 3104.0 5101.0 1206.0 3204.0 5201.0
      1107.0 3105.0 5102.0 1207.0 3205.0 5202.0
      1108.0 3106.0 5103.0 1208.0 3206.0 5203.0
      1109.0 3107.0 5104.0 1209.0 3207.0 5204.0
&END
&DEF X=1,Y=N
&DATA
      X = 0101.0, Y = 1210.0 2206.0 3208.0 4201.0 5205.0
```

```
 &END
 &DATA
       X = 0102.0, Y = 1211.0 2207.0 3209.0 4202.0 5206.0
 &END
 &DATA
       X = 0103.0, Y = 1212.0 2208.0 3210.0 4203.0 5207.0
 &END
 &DATA
       X = 0104.0, Y = 1213.0 2209.0 3211.0 4204.0 5208.0
 &END

 &DATA
       X = 0105.0, Y = 1214.0 2210.0 3212.0 4205.0 5209.0
 &END
```

Zuordnung der Daten zu den Kurven und den Achsen:

```
VARIABLE_1:    X              Y
   1. : 0.1101000D+04  0.1201000D+04
   2. : 0.1102000D+04  0.1202000D+04
            . . .
  14. : 0.1050000D+03  0.1214000D+04

VARIABLE_2:    X              Y
   1. : 0.2101000D+04  0.2201000D+04
   2. : 0.2102000D+04  0.2202000D+04
            . . .
  10. : 0.1050000D+03  0.2210000D+04

VARIABLE_3:    X              Y
   1. : 0.3101000D+04  0.3201000D+04
   2. : 0.3102000D+04  0.3202000D+04
            . . .
  12. : 0.1050000D+03  0.3212000D+04

VARIABLE_4:    X              Y
   1. : 0.1010000D+03  0.4201000D+04
   2. : 0.1020000D+03  0.4202000D+04
            . . .
   5. : 0.1050000D+03  0.4205000D+04

 VARIABLE_5:    X              Y
   1. : 0.5101000D+04  0.5201000D+04
   2. : 0.5102000D+04  0.5202000D+04
            . . .
   9. : 0.1050000D+03  0.5209000D+04
```

6 Grafische Benutzungsoberfläche (GRAZIL-GUI)

Die grafische Benutzeroberfläche wurde auf der Basis des weithin verfügbaren Athena Widget Sets implementiert, welches über die Xtoolkits benutzt wird. Das GUI lehnt sich in der Funktionalität der einzelnen Fenster an die Struktur der alten Kommandoschnittstelle an, wobei aussagekräftige Namen anstelle von numerischen Werten verwendet werden.
Das GUI und GRAZIL sind zwei separate Hauptprogramme. Die Synchronisation und Kommunikation zwischen diesen ist mittels einer Pipe, eines Signals, und einem Statusfile realisiert[1]. Daraus folgt, daß GRAZIL nur in einer UNIX-Umgebung mit diesem GUI zu bedienen ist.
GRAZIL-Fehlermeldungen werden zur Zeit noch nicht an das GUI durchgereicht, sondern erscheinen in dem Fenster, in dem GRAZIL gestartet wurde.
Soll GRAZIL mit dem GUI gestartet werden, muß die Option -i als erster Parameter beim Aufruf von GRAZIL angegeben werden. Zusätzlich können bis zu 3 Dateien angegeben werden. Ein nachträgliches Starten des GUI aus GRAZIL ist dann nicht mehr möglich. Siehe hierzu auch Kap. 4.2.

6.1 GRAZIL-Kontrollfenster

Ist die Option -i angegeben, meldet sich GRAZIL mit dem Hauptkontrollfenster des GUI.
Es besitzt zwei Aktions-Knöpfe, den Run- und den Quit-Knopf. Anklicken des Run-Knopfes bewirkt ein Neuzeichnen des Bildes. Ist noch kein Grafikausgabefenster eröffnet, wird dieses erzeugt. Betätigen des Quit-Knopfes beendet GRAZIL. Alle Einstellungen sind danach verloren.

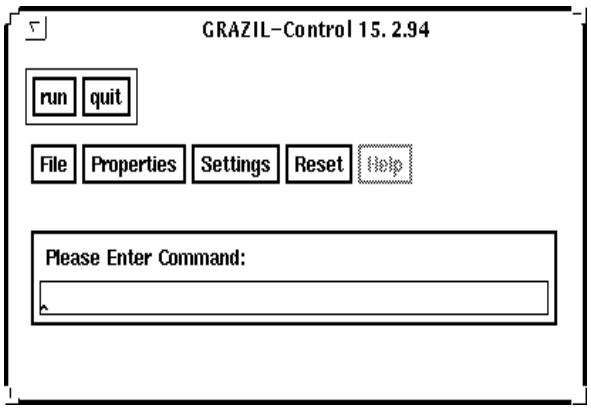

Fig. 4 Das Kontrollfenster von GRAZIL

Unterhalb dieser Aktionszeile befinden sich die Auswahlmenüs und ein Dialogeingabebereich zum Verändern der GRAZIL-Einstellungen. Sind noch keine Va-

[1] Sollte das GUI aufgrund eines internen Fehlers abnormal beendet werden, geht GRAZIL in den Kommandomodus über, wobei alle bisher gemachten Einstellungen erhalten bleiben. Über die Kommandoschnittstelle können dann weitere Kommandos (z. B. &save...) an GRAZIL gegeben werden.

riablen angegeben, so werden im Properties- und Settings-Menü des GRAZIL-Kontrollfensters die Auswahl-Items gesperrt, in deren Fenstern variablenbezogene Einstellungen vorgenommen werden. Beim Einlesen von Daten werden diese Menü-Items automatisch entsperrt. Dies betrift im Properties-Menü die Items *Area, Auxiliary-Line, Render Attributes*, und im Settings-Menü die Items *Alias Names* und *Vars*.
Im unteren Teil des GRAZIL-Kontrollfensters befindet sich ein Eingabebereich zur Eingabe von GRAZIL-Kommandos. Hier getätigte Eingaben werden erst nach Abschluß mit *Return* an GRAZIL übertragen und ausgeführt. Es können hier auch GRAZIL-Kommandos eingegeben werden, für die es derzeit keine Fenster im GUI gibt. Im einzelnen sind dies folgende Kommandos:

Help — Zur Ausgabe einer Liste der vorhandenen GRAZIL-Kommandos, bzw. einer Kurzbeschreibung von deren Syntax (siehe Kap. 7.15, S.48).

List — Zur Auflistung der im GRAZIL vorgenommenen Einstellungen (siehe Kap. 7.18, S.53).

Receive — Zum Empfangen und Darstellen von Daten aus einem gleichzeitig laufenden Anwendungsprogramm (siehe Kap. 7.24, S.57).

Rename — Zum Verändern der Namen von Variablen und Kommandos. Ein Verändern von Kommandonamen ist im GUI nicht erlaubt (siehe Kap. 7.25, S.57).

Ferner gibt es keine Fenster für die Kommandos der Datenschnittstelle &name, &def, und &data.
In der zweiten Menüzeile des GRAZIL-Kontrollfensters findet man die Menüs zum Öffnen weiterer Fenster.
Ähnliche Funktionen wurden in diesen Einstellungsfenstern immer am gleichen Ort plaziert. In der linken oberen Ecke befinden sich jeweils (eingerahmt) die Aktions-Knöpfe.
Die vier Aktions-Knöpfe der Einstellungsfenster sind:

done — schließt das Fenster und stellt es ohne Icon in den Hintergrund. Die im Fenster gemachten Einstellungen bleiben im GUI erhalten werden aber nicht an GRAZIL weitergereicht. Durch erneute Selection im jeweiligen Auswahlmenü wird das Fenster wieder geöffnet und die Einstellungen wieder sichtbar.

apply — sendet die im jeweiligen Fenster vorgenommenen Einstellungen an GRAZIL. In Abhängigkeit von der Environment-Variable GRAZIL_RUN_ON_APPLY werden diese neuen Einstellungen sofort oder erst nach Anklicken des Run-Knopfes im GRAZIL-Kontrollfenster von GRAZIL ausgeführt.

reset — setzt in diesem Fenster alle Einstellungen in den Zustand nach dem letzten Anklicken des Apply-Knopfes.

cancel — löscht das Fenster. Alle nach dem letzten Anklicken des Apply-Knopfes gemachten Einstellungen werden vergessen. Es gelten danach wieder die Einstellungen, die nach dem letzten Anklicken des Apply-Knopfes galten.

Im GRAZIL können bis zu 4 Darstellungsschichten (LAYER) gleichzeitig, auch übereinander, angezeigt werden. Bei Fenstern, die layerbezogene Einstellungen vornehmen, erscheint rechts oben, neben dem Feld der Aktions-Knöpfe, das Layer-Auswahlfeld. Hier wird eingestellt, für welche Layer die in diesem Fenster gemachten Einstellungen gelten sollen. Angezeigt werden immer die Einstellungen des aktiven Layers mit der kleinsten Nummer. Ist kein Layer aktiviert, wird Layer 1 als aktiv angenommen.

6.2 Dateiauswahl

Die Dateiauswahl wird über den Menü-Knopf Fileselection aktiviert. Folgende Menüpunkte sind vorhanden:

Load — dient dem Einlesen von GRAZIL-Daten und Einstellungen (z.B. aus einer früher angelegten Sicherungsdatei).

Save — dient dem Sichern von GRAZIL-Einstellungen. Daten können nicht gesichert werden. Dies ist insbesonders bei mit dem &receive-Kommando empfangenen Daten zu beachten.

Save GUI Status — dient dem Sichern von GUI-Einstellungen in Form von GRAZIL-Kommandos.

Die unter dem File-Menü zu öffnenden Fenster nehmen im GUI eine Sonderstellung ein. Sie sind exklusiv, d.h. ist ein Dateifenster geöffnet, sind alle anderen Funktionen im GUI gesperrt; ein Weiterarbeiten ist dann erst nach dem Beenden des Dateifensters möglich. Dialogeingabe (Eingabe eines Dateinamens) muß immer mit Return abgeschlossen werden!
Der Name des aktuellen Dateiverzeichnisses wird in der Kopfzeile angezeigt, darunter steht die Liste der Dateien und Dateiverzeichnisse (mit / gekennzeichnet). Umfaßt die Liste von Dateien mehr Namen, als dargestellt werden können, so kann durch Verschieben des Rollbalkens (mit der mittleren Maustaste) die Liste verschoben werden.
Eine Datei wird durch einmaliges Anklicken mit der linken Maustaste ausgewählt. Handelt es sich um ein Verzeichnis, so wird der Name des aktuellen Dateiverzeichnisses sowie dessen Inhalt angezeigt. Wird eine Datei ausgewählt, so erscheint der Dateiname mit dem gesamten Pfad in dem Eingabefeld *File*. Im Eingabefeld *Pattern* kann ein Muster mit Wildcards angegeben werden, so daß nur Dateien angezeigt werden, die diesem Muster genügen.
Am rechten Rand des Fensters befinden sich folgende drei Aktions-Knöpfe:

Cancel — dient dem Verlassen des Dateifensters und dem Entriegeln der anderen GUI-Funktionen; es wird keine Datei eingelesen oder geschrieben.

Apply — dient dem Einlesen oder Schreiben der ausgewählten Datei. Wenn keine Datei ausgewählt wurde, wird das Dateifenster verlassen und die Verriegelung der anderen GUI-Funktionen aufgehoben.

Info — dient dem Eröffnen eines weiteren Fensters, in dem Informationen über die zuletzt ausgewählte Datei angezeigt werden. Durch nochmaliges Anklicken des Knopfes wird das Fenster wieder geschlossen.

6.3 Grafikeinstellungen

Über dem Menü-Knopf *Properties* erscheint ein Auswahlmenü zum Öffnen weiterer Fenster. Diese dienen zum Anzeigen und Ändern der Layout-Einstellungen. Im einzelnen sind dies folgende Fenster:

Area — zum Füllen von Flächen unter und zwischen Kurven. Die hier eingestellten Flächenattribute gelten auch für die Darstellungsart Bar Chart. Die für Bar Chart erforderlichen Auswahlmenüs befinden sich zusätzlich im Bar Chart-Fenster. Siehe AREA-Kommando (Kap. 7.2, S. 28).

Auxiliary Line — zum Positionieren der Hilfslinien und Einstellen von Linienattributen. Siehe AUXLINE-Kommando (Kap. 7.4, S. 32).

Auxiliary Line Textattributes — zum Einstellen von Attributen der Hilfslinientexte. Siehe AUXLINE-Kommando (Kap. 7.4, S. 32).

Axis — zum Einstellen der Achseneinteilung, Position der Achsen in der Zeichnung und weiterer Attribute. Siehe AXISSPEC-Kommando (Kap. 7.5, S. 36).

Axis Text — zur Eingabe von Achsentexten und Einstellen der Attribute. Siehe ATEXT-Kommando (Kap. 7.3 S. 30).

Axis Ticks — zum Einstellen der Attribute der Texte an den Achsenticks. Siehe AXISSPEC-Kommando (Kap. 7.5, S. 36).

Bounds — zum Einstellen der Darstellungsgrenzen. Siehe BOUNDS-Kommando (Kap. 7.7, S. 40).

Curve Drawing Area — zum Einstellen der Kurvenzeichenflächen für die jeweiligen Layer. Es wird jeweils die linke untere Ecke und die rechte obere Ecke angegeben. Die einzelnen Werte können sowohl mit den Rollbalken als auch durch Eingeben in den dazugehörigen Dialogeingabebereichen verändert werden. Siehe PROFILE-Kommando (Kap. 7.22, S. 55).

[1] Die Positionsangaben erfolgen in normalisierten Koordinaten. Die linke untere Ecke hat die Koordinaten (0.0, 0.0), die rechte obere (1.0, 1.0).

Grid — zum Ein- und Ausschalten eines Gitters und Setzen der dazugehörigen Attribute. Siehe GRID-Kommando (Kap. 7.13, S. 45).

Headline — zur Eingabe von Überschriften, sowie deren Positionen und Attribute. Siehe HEADLINE-Kommando (Kap. 7.14, S. 47).

Legend — zur Auswahl der Legendenart und zum Einstellen der Attribute. Siehe LEGENDTYPE-Kommando (Kap. 7.17, S. 50).

Render Attributes — zur Auswahl der Darstellungsarten der einzelnen Variablen. Es sind folgende Modi implementiert:

- Curve — Die Variablen werden als Kurven interpretiert. Verschiedene Interpolationsarten können ausgewählt werden. Siehe RENDER-Kommando (Kap. 7.26, S. 58).
- Bar Chart — Die Variablen werden als Balkendiagramm interpretiert. Das Layout der Balken und die Art der Beschriftung wird hier eingestellt. Siehe RENDER-Kommando (Kap. 7.26, S. 58).
- Errorbar — Auswahl der Variablen, die die Werte für die Fehlerbalken liefern, und Auswahl der Art der Fehlerbalken. Siehe RENDER-Kommando (Kap. 7.26, S. 58).

Stamp — Setzen eines Datums- und Zeitstempels oder zweier kurzer Texte. Siehe STAMP-Kommando (Kap. 7.31, S. 69).

6.4 Settings

Hier findet man Fenster, die das Layout nicht beeinflussen:

Alias Names — zum Einführen von Alias-Namen für Kurvennamen. Diese werden dann in der Legende verwendet. Aliasnamen für GRAZIL-Kommandos können hier nicht eingeführt werden. Siehe ALIAS-Kommando (Kap. 7.1, S. 27).

Texttable — zur Eingabe von Texttabellen. Es können bis zu 10 verschiedene Tabellen eingegeben werden. Vorhandene Tabellen lassen sich verändern (Löschen oder Einfügen von Tabelleneinträgen, Edieren einzelner Tabelleneinträge). Siehe PROFILE-Kommando (Kap. 7.22, S. 56).

Vars — zur Auswahl der darzustellenden Variablen. Die auswählbaren Variablen werden in der Liste *Selectable Vars* angezeigt. Sind hier nicht alle Variablen darstellbar, erscheint in den Listen ein Rollbalken. Durch Betätigen mit der mittleren Maustaste kann die Liste verschoben werden. Das Auswählen von Variablen geschieht durch einmaliges Anklicken mit der linken Mausteaste.

Die ausgewählte Variable erscheint dann in der Liste *Requested Vars*. Analog können Variablen auch wieder aus der Liste der ausgewählten Variablen entfernt werden. Anklicken des Knopfes *Clear Request List* entfernt alle Variablen aus der Liste. Siehe VARIABLES-Kommando (Kap. 7.33, S. 71).

Workstation zur Auswahl des Ausgabegerätes, zum Setzen der Zeichnungsgröße, sowie dem Einstellen des Zeichnungshintergrundes und der Attribute des Zeichnungsrahmens.

Die Eingabe eines Rechnernamens im Eingabefeld *Display on Host* lenkt die Grafikausgabe auf diesen Rechner. Wird kein Rechnername angegeben, erscheint das GRAZIL-Ausgabefenster auf dem aktuellen Bildschirm. Siehe WORKSTATION-Kommando (Kap. 7.34, S. 72).

Send only bewirkt, daß beim Anklicken von Apply die aktuellen Einstellungen nur an GRAZIL übertragen, aber noch nicht ausgewertet werden. Erst nach Betätigen des Run-Knopfes im GRAZIL-Kontrollfenster werden die neuen Einstellungen von GRAZIL ausgewertet und angezeigt.

Send and Run bewirkt, daß beim Anklicken von Apply die Einstellungen an GRAZIL gesendet, und gleich ausgewertet sowie angezeigt werden.

Expect <RET> bewirk, daß Dialogeingabe nur nach Abschluß mit Return übernommen wird.

Expect apply bewirkt, daß Dialogeingabe mit Betätigen des Apply-Knopfes übernommen wird.

7 Grafikkommandos

Zur Steuerung der Grafik gibt es zahlreiche, in diesem Kapitel aufgeführte Kommandos. Die Kommandos können abgekürzt werden, wobei zur Eindeutigkeit mindestens die großgeschriebenen Buchstaben eines Kommandonamens eingegeben werden müssen. Im Kommandofile sind Kommentare erlaubt. Sie werden mit ';' eingeleitet und gelten dann bis zum Zeilenende. Es empfiehlt sich, Eingabetexte in Apostrophe einzuschließen, um Groß- und Kleinschreibung zu erhalten und Leer- sowie Sonderzeichen zu ermöglichen. Ohne Apostrophe werden Kleinbuchstaben in Großbuchstaben umgewandelt und Leer- oder Sonderzeichen (fehl-)interpretiert. Gleitkommazahlen können auch in Exponentendarstellung eingegeben werden, z.B. der Wert 0.012 in der Form 1.2E-2.
Auf den folgenden Seiten sind die verfügbaren Kommandos in alphabetischer Reihenfolge aufgeführt und erläutert. Hierbei werden gelegentlich Begriffe aus der GKS-Welt verwendet; diese sind in Anhang C kurz erläutert. Die Parameterleiste der Kommandos besteht aus bis zu 14 *stellungsgebundenen* Parametern P 1 ... P14. Es müssen daher alle Parameter bis zum letzten, der verändert

werden soll, angegeben werden. Die Eingabe eines '=' (Platzhalter) läßt den Parameterwert unverändert.
Die in den folgenden Kapiteln gezeigten Beispiele bauen jeweils aufeinander auf. Wir starten mit einem Bild für das weitgehend Defaults genutzt werden. Diese Voreinstellungen werden dann schrittweise durch spezifische Angaben ersetzt. Das verwendete Eingabefile rain+temp ist das in Kap. 2 eingeführte. Es ist in der GRAZIL-Installation mitenthalten (Verzeichnis, grazil/examples). Nehmen Sie sich als Erstbenutzer eine Stunde Zeit und vollziehen Sie die in der nachfolgenden Beschreibung angegebenen Beispiele nach. Es werden nur die unbedingt notwendigen Kommandos zur Ausgabe eines Bildes auf ein Sichtgerät angegeben.
In dem File *doku.layout* sind einige GRAZIL-Kommandos zusammengefaßt, mit denen Voreinstellungen für die in dieser Dokumentation abgedruckten Plots vorgenommen wurden. Diese Kommandos sind im Anhang F näher erläutert.

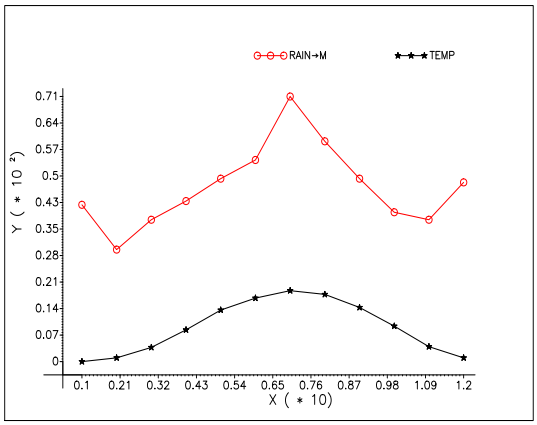

Fig. 5 Beispiel mit Defaulteinstellungen

Notwendige Kommandos

```
%grazil -c rain+temp

grazil>&wkt 3100
grazil>&vars = temp rain-m
grazil>&comfile 'doku.layout'
grazil>&run
```

7.1 ALIAS

Dieser Befehl hat zwei verschiedene Bedeutungen: Er führt einen weiteren Namen, z.B. eine Abkürzung, für ein vorhandenes Kommando ein, oder einen Alias-Namen für einen Variablennamen, welcher dann in der Legende erscheint und zur Auswahl verwendet werden kann.

Beispiel

Ziel : In der Legende sollen andere Namen für die Kurven erscheinen. Zwischenticks sollen nicht gezeichnet werden.

Kommando : &alias -n temp temperatur
&alias -n rain_m regenmenge
&axis * = = = 0

Settings → Alias
Properties → Axis
→ all-axis
→ Subticks

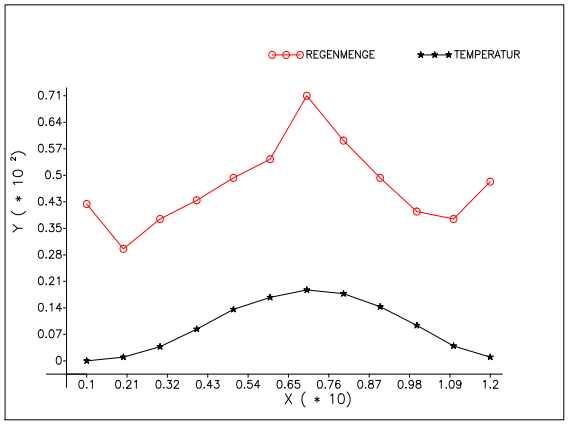

Fig. 6 Alias für Variablenname

Kommandosyntax

```
&ALias -c <alter Name> [ <neuer Name> [ <iwert> ] ]
```

alter Name :: Ein bereits definierter Kommandoname oder Alias. Wird nur *alter Name* eingegeben, so werden alle Aliaseinträge für diesen Kommandonamen gelöscht.

neuer Name :: Alias.

iwert :: legt die Mindestzahl der bei Abkürzung des Alias einzugebenden Buchstaben fest.

bzw. für die zweite Bedeutung

```
&ALias -n <alter Name> [ <neuer Name> ]
```

alter Name :: Name einer Variablen (keine Wildcards!). Wird nur *alter Name* eingegeben, so wird der Aliaseintrag für diesen Variablennamen gelöscht.

neuer Name :: Neuer, zusätzlicher Variablenname, der auch zur Kurvenauswahl verwendet werden kann. Dieser Alias-Name erscheint auch in der Kurvenlegende (s. LEGENDTYPE, S. 50). Wird für mehrere Variablennamen der gleiche Alias-Name angegeben, so wird in der Legende nur die erste ausgewählte Kurve mit diesem Namen gezeigt. Der Alias-Name kann max. 20 Zeichen lang sein.

7.2 AREA

Das Kommando füllt Flächen unter Kurven und/oder stellt die Attribute für Balkendarstellungen (siehe Option -VBARCHART im Kommando RENDER

auf Seite 58) ein.

Beispiel

Ziel : Die Fläche zwischen den beiden dargestellten Kurven soll gefüllt werden.

Kommando : &area temp +1 1 = 7

Properties → Area
→ selectable Vars
→ temp
→ Fillstyle
→ Boundary
→ Areacolour

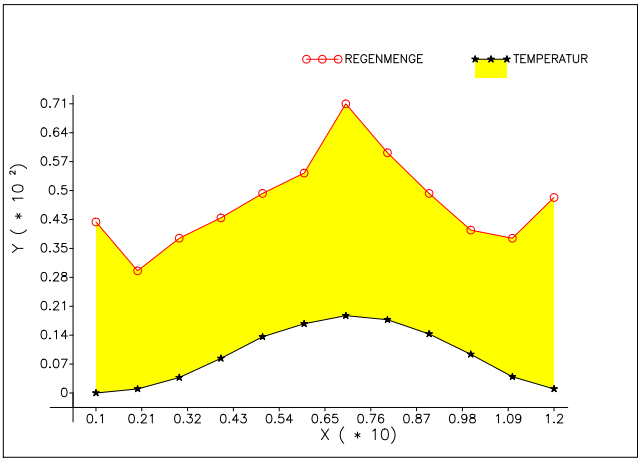

Fig. 7 gefüllte Fläche

Kommandosyntax

```
&ARea <name> <begrenzung> [ <style> [ <style_index>
            [ <colour> ] ] ]
```

name :: Variablenname einer mit dem Kommando &name definierten Variablen (Kurve); Wildcards sind analog dem &vars-Kommando (s.Kap. 7.33) sind erlaubt.

begrenzung :: *name* : Name einer weiteren mit dem &vars-Kommando ausgewählten Variablen, um das Füllgebiet zu begrenzen.

+1 : Zweite Begrenzung durch die nächste Variable (in Auswahlreihenfolge); kein Füllgebiet zwischen der ersten Kurve und der X-Achse.

0 : Zweite Begrenzung durch die X-Achse.

-1 : Zweite Begrenzung durch die vorige Variable (in Auswahlreihenfolge); das Füllgebiet der ersten ausgewählten Variablen wird von der X-Achse begrenzt.

-2 : Löschen der Variablen *name* aus der Liste der Area-Variablen.

= : keine Änderung

Default1 : +1

style :: Interior Style* des Füllgebiets.

0 : kein Füllen der Fläche.

1 : solid*

2 : pattern*†

3 : hatch*

= : keine Änderung

Default : 0

style-index :: Index für Interior Style Pattern oder Hatch

-1 : Die für das Ausgabegerät definierten Interior Styles werden abwechselnd genutzt.

≥ 0 : Der angegebene Interior Style wird genutzt.

= : keine Änderung

Default : -1

colour :: Farbe des Füllgebiets

-1 : Die in dem Ausgabegeät vorhandenen Farben werden abwechselnd genutzt.

≥ 0 : Die angegebene Farbe wird genutzt.

= : keine Änderung

Default : -1

7.3 AXISTEXT

Das Kommando dient zur Angabe einer neuen Achsenbeschriftung. Die maximale Länge des Textes beträgt 128 Zeichen. Ist er länger, so wird er abgeschnitten.
Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

*GKS-Terminus, s. Glossar, Anhang C

†Parameter hat z. Z. keine Funktion.

Beispiel

Ziel : Der X-Achsentext soll geändert werden.

Kommando : &atext x 'Monat'

Properties → Axis Text → Axis Text → Monat

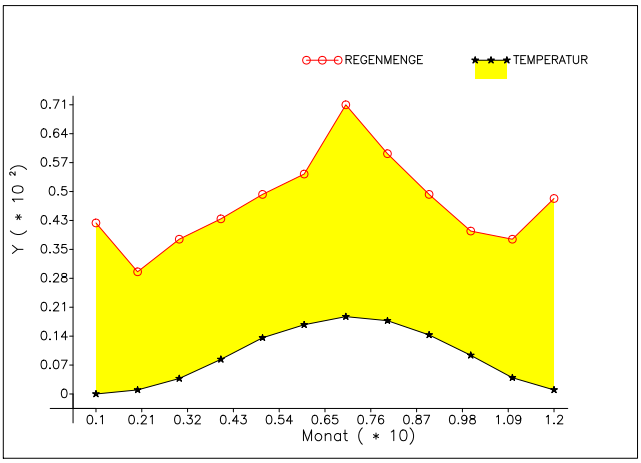

Fig. 8 Neuer Achsentext für die X-Achse

Kommandosyntax

```
&Atext <x|y|*> [ <'text'> ]
```

x | *y* |∗ :: Achsenbezeichner

∗ : beide Achsen.

text :: Text oder '∗'; die Eingabe eines '∗' bewirkt ein Umschalten zwischen dem mit ATEXT eingelesenen und dem Default-Text.

Default : Der in der Datendefinition angegebene Achsenbezeichner.

Mit dem Profile-Kommando können weitere Achsentext-Voreinstellungen verändert werden.

```
&PROFILE -ATEXT <x|y|*> <parameterliste>
```

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	1	...	n	Textfont*
P 2	:	0	...	2	Textprecision*
P 3	:	0	...	n	Textfarbe
P 4	:	>	0.0		Faktor der Character Height*
P 5	:	>	0.0		Character Expansion Factor*
P 6	:	>	0.0		Character Spacing*
P 7	:				Textwinkel*
P 8	:	0	...	3	Horizontales Text Alignment*
P 9	:	0	...	5	Vertikales Text Alignment*
P 10	:	0	...	3	Textpath*
P 11	:	-1	...	n	Rahmenfarbe
P 12	:	-1	...	n	Texthintergrundfarbe
P 13	:		T | F		Flag, ob der Exponent dem Achsentext zugefügt werden soll.

7.4 AUXLINE

Mit diesem Kommando werden eine oder mehrere gerade, horizontale oder vertikale Hilfslinien in die Zeichenfläche gezeichnet. Wenn eine Hilfslinie eine Achse ohne Achsenticks schneidet, so erscheint der Wert des Schnittpunktes als Linientext der Hilfslinie. Bei Angabe eines Linientextes wird dieser an die Hilfslinie geschrieben. Wird die Hilfslinienposition durch einen Kurvenextremwert definiert, so können pro Liniennummer mehrere Hilfslinien gezeichnet werden, falls es mehrere Kurvenpunkte gibt, deren X- bzw. Y-Koordinate gleich dem Extremum ist. Eine Hilfslinie wird in dem Layer gezeichnet, in dem sie positioniert wurde. Eine nachträgliche Änderung von Attributen einer Hilfslinie ist davon unabhängig, welcher Layer aktuell eingestellt ist.

Beispiel

Ziel : Die Fläche zwischen den Kurven soll nicht mehr gefüllt werden.

Eine horizontale Hilfslinie soll an das Temperaturmaximum gezeichnet werden.

Kommando :
```
&area temp = 0
&auxline 1 hor temp.maxy
```

Properties → Area → Fillstyle

Properties → Auxline → Direction → Position

*GKS-Terminus, s. Glossar, Anhang C

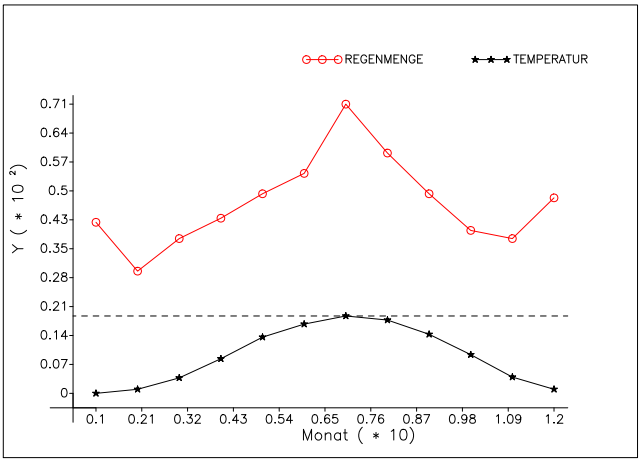

Fig. 9 Horizontale Hilfslinie am Maximum der Temperatur

Kommandosyntax

```
&AUXLine <liniennummer> <richtung> [ <position> [ <linetype>
        [ <colour> [ <linientext> ] ] ] ]
```

liniennummer :: 1 ... 10 : Nummer der zu zeichnenden Hilfslinie.
∗ : alle schon bestehenden Hilfslinien.

richtung :: HORIZONTAL : Die Linie verläuft waagerecht.
VERTICAL : Die Linie verläuft senkrecht.
DELETE : Die Linie wird gelöscht.
= : keine Änderung

position :: real-wert : Gibt die Position der Hilfslinie an: bei waagerechten Hilfslinien Y-Wert des Schnittpunktes mit der Y-Achse, bei senkrechten Hilfslinien X-Wert des Schnittpunktes mit der X-Achse. Liegt der angegebene Schnittpunkt außerhalb der Achsengrenzen, wird die Hilfslinie nicht gezeichnet.

<name>.MINX : Die Hilfslinie läuft durch den Datenpunkt der Kurve <name> mit dem kleinsten X-Wert. Waagerechte Hilfslinien können mehrfach erscheinen.

<name>.MAXX: Die Hilfslinie läuft durch den Datenpunkt der Kurve <name> mit dem größten X-Wert. Waagerechte Hilfslinien können mehrfach erscheinen.

	<name>.MINY	: Die Hilfslinie läuft durch den Datenpunkt der Kurve <name> mit dem kleinsten Y-Wert. Senkrechte Hilfslinien können mehrfach erscheinen.
	<name>.MAXY	: Die Hilfslinie läuft durch den Datenpunkt der Kurve <name> mit dem größten Y-Wert. Senkrechte Hilfslinien können mehrfach erscheinen.
		name : Name einer Variablen (ohne Wildcards).
	=	: keine Änderung
linetype	:: >0	: Die Linien werden mit dem angegebenen Linientyp gezeichnet. Bei Ausführung des Kommandos &run wird überprüft, ob dieser Linientyp für das gewählte Auggabegerät existiert. Ist das nicht der Fall, werden gestrichelte Linien (Linientyp 2) gezeichnet.
	=	: keine Änderung
	Default	: 2 (gestrichelte Linien)
colour	:: ≥0	: Die Linien werden in der angegebenen Farbe gezeichnet. Bei Ausführung des Kommandos &run wird überprüft, ob diese Farbe für das gewählte Ausgabegeät existiert. Wenn nicht, so wird mit Farbe 1 gezeichnet.
	=	: keine Änderung
	Default	: 1
linientext	::	: Der angegebene Text, maximal 20 Zeichen lang, wird an die Linie geschrieben.

Weitere Hilfslinien-Voreinstellungen können mit dem Profile-Kommando verändert werden:

```
&PROFILE -AUXLINE <liniennummer> <parameterliste>
```

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	1	...	n	Textfont*
P 2	:	0	...	2	Textprecision*
P 3	:	0	...	n	Textfarbe
P 4	:	>	0.0		Faktor der Character Height*
P 5	:	>	0.0		Character Expansion Factor*
P 6	:	>	0.0		Character Spacing*
P 7	:				Textwinkel*
P 8	:	0	...	3	Textpath*
P 9	:	-1	...	3	Position des Textes längs der Hilfslinie
			-1 | 0	:	Position wird automatisch eingestellt
			1	:	am unteren bzw. linken Ende
			2	:	in der Mitte
			3	:	am oberen bzw. rechten Ende
P 10	:	-1	...	3	Position des Textes quer zur Hilfslinie
			-1 | 0	:	Position wird automatisch eingestellt
			1	:	horizontal : oberhalb, vertikal : links
			2	:	auf Hilfslinienhöhe
			3	:	horizontal : unterhalb, vertikal : rechts
P 11	:	>	0.0		Linewidth der Hilfslinien
P 12	:	-1	...	n	Textrahmenfarbe
P 13	:	-1	...	n	Texthintergrundfarbe
P 14	:	0	...	3	Art der Positionierung mittels Extremwerten

0 : Die Extrema werden aus allen Stützpunkten der Variablen ermittelt; Mehrfachlinien werden nicht dargestellt.

1 : Die Extrema werden aus den Stützpunkten der Variablen ermittelt, die innerhalb der eingestellten Achsengrenzen liegen; Mehrfachlinien werden nicht dargestellt.

2 : Die Extrema werden aus allen Stützpunkten der Variablen ermittelt; Mehrfachlinien werden, falls erforderlich, dargestellt.

3 : Die Extrema werden aus den Stützpunkten der Variablen ermittelt, die innerhalb der eingestellten Achsengrenzen liegen; Mehrfachlinien werden, falls erforderlich, dargestellt.

*GKS-Terminus, s. Glossar, Anhang C

7.5 AXISSPEC

Mit diesem Kommando werden der Schnittpunkt der Achsen, ihre Einteilung und die Anzahl der Ticks festgelegt. Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

Beispiel

Ziel : An beiden Achsen sollen keine Subticks gezeichnet werden.
Die Y-Achse soll am rechten Rand der Zeichnung stehen.
Die x-Achse soll von 1 bis 12 beschriftet werden.

Kommando : &axis * = = = 0
&axis y = max
&axis x = = 11

Properties → Axis → Y-axis → Intersection
→ X-axis → Mainticks

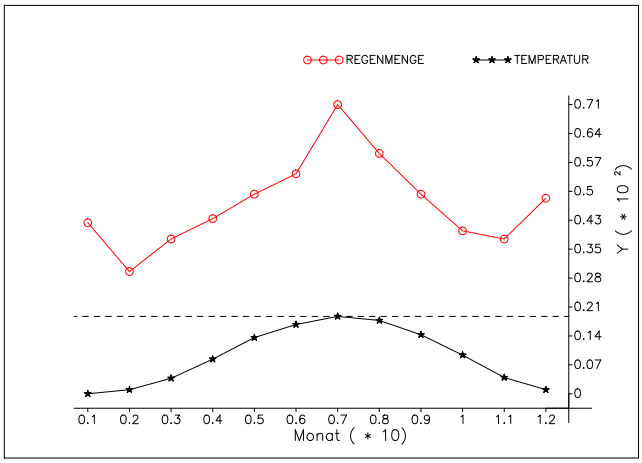

Fig. 10 Andere Einteilung der X-Achse
und die Y-achse am rechten Rand der Zeichenfläche.

Kommandosyntax

```
&AXisspec <x|y|*> <einteilung> [ <schnittpunkt>
          [ <hauptticks/delta> [ <zwischenticks>
          [ <beschriftungsmodus> [ <tickseite> ] ] ] ] ]
```

`x|y|*` :: Achsenbezeichner.

	*	: Beide Achsen.
einteilung	:: LIN	: Lineare Einteilung der Achse.
	LOG	: Logarithmische Einteilung der Achse.
	DEL	: Es wird keine Achse gezeichnet.

		=	: keine Änderung
schnittpunkt	::	real-wert	: Die bezeichnete Achse schneidet die andere Achse am angegebenen Punkt. Bei Ausführung des Kommandos &run wird überprüft, ob dieser Punkt innerhalb der Zeichenfläche liegt; Ist das nicht der Fall, wird MIN bxw. MAX entsprechend eingestellt.
		MIN	: Es wird der kleinste Wert verwendet, der mit dem &bounds Kommando eingestellt wurde, oder der durch &vars implizit eingestellt ist.
		MAX	: Es wird der größte Wert verwendet, der mit dem &bounds Kommando eingestellt wurde, oder der durch &vars implizit eingestellt ist.
		MID	: Es wird die Mitte der anderen Achse eingestellt.
		=	: keine Änderung
		Default	: min
hauptticks/delta	::	0	: Es werden keine Hauptticks und keine Tickbeschriftung gezeichnet.
		≥1	: Anzahl der Zwischenräume zwischen den beschrifteten Hauptticks.
		real-wert>0.0	: Delta der Hauptticks (= Abstand zwischen zwei Hauptticks in Weltkoordinaten; sinnvoll nur bei linearer Achse).
		=	: keine Änderung
		Default	: 10
		Wird mit dem Parameter *beschriftungsmodus*(s.u.) eine Nummer einer Texttabelle angegeben, so ist die Anzahl der beschrifteten Hauptticks gleich der Anzahl der Texteinträge in der Texttabelle.	
zwischenticks	::	0 | 1	: Es werden keine Zwischenticks gezeichnet.

		>1	: Anzahl der Zwischenticks.
		=	: keine Änderung
		Default	: 10
		Die Gesamtzahl aller Ticks einer Achse darf wegen der begrenzten Auflösung 100 nicht überschreiten.	
beschr.modus	::	-1	: Die Hauptticks werden mit den entsprechenden Variablenwerten beschriftet.
		0	: Keine Beschriftung der Hauptticks.
		1 ... 10	: Die Hauptticks werden mit den Einträgen der Tabelle n beschriftet. (S. Kommando &profile, S. 56).
		=	: keine Änderung
		Default	: -1
tickseite	::	Stellung der Achsenticks zum Zeichengebiet.	
		1	: Ticks und Beschriftung außen
		2	: Ticks und Beschriftung innen
		3	: Ticks innen, Beschriftung
		=	: keine Änderung
		Default	: 1

Weitere Voreinstellungen für die Achsen und Beschriftungen der Achsenticks können mit Hilfe des Profile-Kommandos vorgenommen werden:

```
&PROFILE -AXIS <x|y|*> <parameterliste>
```

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	0	...	n	Farbe der Achsen
P 2	:	>	0.0		Linewidth* der Achsen
P 3	:	0	| 1		mit | ohne Pfeil am Achsenende
P 4	:	-1	...	3	Position des Achsentextes längs der Achse -1|0: wird automatisch eingestellt 1 : am unteren bzw. linken Ende 2 : in der Mitte 3 : am oberen bzw. rechten Ende
P 5	:	-1	...	3	Position des Achsentextes quer zur Achse -1|0: wird automatisch eingestellt 1 : horizontal: oberhalb; vertikal: links 2 : auf Achsenlinienhöhe 3 : horizontal: unterhalb; vertikal: rechts
P 6	:		.....		Parameter hat z. Z. keine Funktion.
P 7	:	0.	...	1.	Schwellwert zur Achsentickumschaltung s. Anhang. B

`&PROFILE -AXTICK <x|y|*> <parameterliste>`

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	1	...	n	Textfont*
P 2	:	0	...	2	Textprecision*
P 3	:	0	...	n	Textfarbe
P 4	:	>	0.0		Faktor der Character Height*
P 5	:	>	0.0		Character Expansion Factor*
P 6	:	>	0.0		Character Spacing*
P 7	:				Textwinkel
P 8	:	0	...	3	Text Path*
P 9	:	0	...	3	Position der Tickbeschriftung 1 : horizontal: links; vertikal: oberhalb 2 : auf Achsentickhöhe 3 : horizontal: rechts; vertikal: unterhalb
P 10	:		≥ 0		Tick-Beschriftungslänge
P 11	:	real-wert			Tick-Beschriftungsfaktor
P 12	:	real-wert			Tick-Beschriftungsoffset
P 13	:	8	Char		Format-Anweisung zur Beschriftung der Achsenticks. Achtung: Es sind nur FORTRAN F-Formate möglich; für ganzzahlige Beschriftung (Fn.0) angeben.

*GKS-Terminus, s. Glossar, Anhang C

7.6 AXISTEXT

Dies ist ein Synonym für das Kommando ATEXT (s. S. 30.

7.7 BOUNDS

Mit diesem Kommando werden die Grenzen der Achsen im aktuellen Layer eingestellt. (siehe Kommando LAYER, S. 49).

Beispiel

Ziel : Es soll der Verlauf für die Monate März bis September dargestellt werden.

Kommando : &bounds x 3 3.0 9.0 Properties → Bounds → manual Bounds

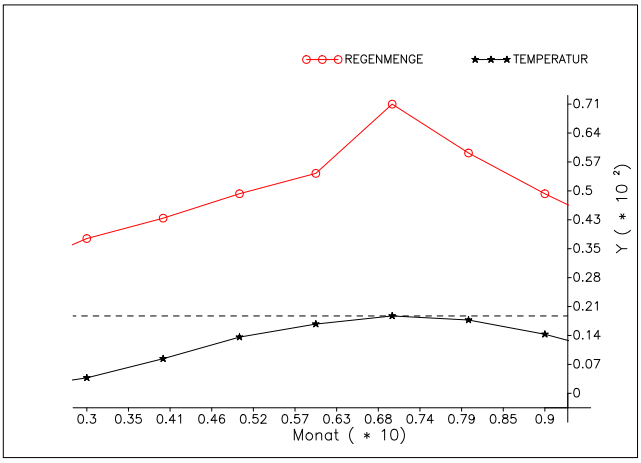

Fig. 11 Bildausschnitt für den Bereich [3.0, 9.0].

Kommandosyntax (Form 1)

```
&Bounds <x|y|*> <modus>
```

`x|y|*` :: Achsenbezeichner.

∗ : beide Achsen.

modus :: 0 : Die Grenzen werden für die ausgewählten Kurven/Variablen so eingestellt, daß die Kurven möglichst bildfüllend dargestellt werden (i.A. führt dies zu Verzerrungen, vgl. *modus* 4).

1 : Die Grenzen werden so eingestellt, daß alle im Datensatz vorhandenen Variablen auf die Zeichenfläche passen.

Default : 0 (für beide Achsen)

Kommandosyntax (Form 2)

`&Bounds <x|y|*> <modus> <untere Grenze> <obere Grenze>`

`x|y|*` :: Achsenbezeichner.

- $*$: beide Achsen.

modus ::

- 2 : Die Grenzen werden mit dem unteren und oberen Grenzwert fest eingestellt. Die Grenzwerte müssen innerhalb des Variablenbereiches liegen.
- 3 : Wie bei 2, aber ohne Überprüfung der Grenzwerte.
- 4 : Der Grenzwert für die angegebene Achse wird, beginnend mit der unteren Grenze, so eingestellt, daß sich ein verzerrungsfreies Bild ergibt. Bei diesem Modus entfällt die Angabe der oberen Grenze.

untere|obere Grenze :: Gleitkommazahl oder `"="`. `"="` steht für den zuletzt eingegebenen Wert (auch für den impliziten Wert bei *modus* = `0|1`).
Es muß gelten : untere Grenze < obere Grenze.
Bei *modus* 4 ist nur die untere Grenze anzugeben!

7.8 COMMANDFILE

Mit diesem Kommando wird die Kommando- und Dateneingabe auf eine Datei umgeschaltet. In UNIX-Systemen kann der Dateiname mit Pfad angegeben werden. Wird der Dateiname nicht in Apostrophe eingeschlossen, werden alle Kleinbuchstaben in Großbuchstaben übersetzt. Ein Verschachteln von Kommandodateien ist nicht möglich.

Kommandosyntax

`&COMfile <filename>?`

filename :: Dateiname; zur Vermeidung der Übersetzung in Großbuchstaben in Apostrophe einschließen.

7.9 CURVE

Das Kommando dient zur Einstellung der Attribute von Kurven.

Beispiel

Ziel : Die Kurve `rain_m` soll gestrichelt gezeichnet und die Datenpunkte mit Kreuzen markiert werden. Desweiteren soll wieder der ganze Kurvenverlauf gezeigt werden.

Kommando : &curve rain_m = 2 5
&bounds x 0

Properties → Bounds
→ selected Vars
Properties → Render Attributes
→ Render Curve
→ Linetype
→ Markertype

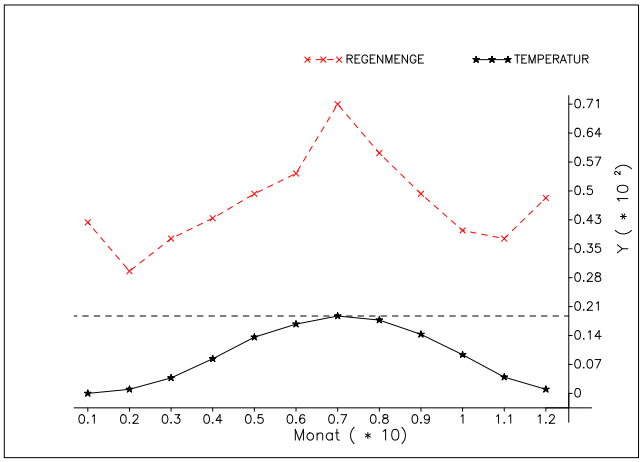

Fig. 12 Die Kurve *rain_m* mit anderem Marker und mit gestrichelter Linie

Kommandosyntax

```
&CURve <name> <colour> [ <linetype> [ <markertype>
       [ <markerfrequenz> [ <interpolation> ] ] ] ]
```

name ::char-string Name einer mit dem Kommando &name definierten Variablen (= Kurve). Wildcards sind analog dem &vars-Kommando erlaubt.

colour :: -1 : Die bei dem jeweiligen Ausgabegerät möglichen Farben werden zur Darstellung der Kurven genutzt, d.h. Kurven werden, soweit möglich, durch verschiedene Farben unterschieden.

≥0 : Die Kurve wird in der angegebenen Farbe gezeichnet. Bei Ausführung des Kommandos &run wird überprüft, ob diese Farbe für das gewählte Ausgabegerät existiert. Wenn nicht, so wird mit Farbe 1 gezeichnet.

= : keine Änderung

Default: -1

linetype :: -1 : Die im GKS genormten Linientypen werden zur Darstellung der Kurven genutzt, d.h. mehrere Kurven werden zur Unterscheidung mit verschiedenen Linienarten gezeichnet. Die punktierte Linie wird nicht verwendet, da sie für das Gitter reserviert ist.

0 : Es wird keine Linie gezeichnet.

≥0 : Die Kurve wird mit dem angegebenen Linientyp gezeichnet. Bei Ausführung des Kommandos &run wird überprüft, ob dieser Linientyp für das gewählte Ausgabegerät existiert. Wenn nicht, so wird mit Linientyp 1 gezeichnet.

= : keine Änderung

Default: -1

markertype :: -3 : Die ersten 26 Stützpunkte werden mit 'A' bis 'Z' als Markierung durchnumeriert.

-3x : Die Stützpunkte werden mit dem Markertyp x markiert; die ersten 26 Stützpunkte werden zusätzlich mit 'A' bis 'Z' durchnumeriert (die Numerierung steht oberhalb oder unterhalb der Stützpunkte).

-2 : Die Stützpunkte werden zur Markierung von 1 bis n durchnumeriert.

-2x : Die Stützpunkte werden mit dem Markertyp x markiert und zusätzlich die Stützpunkte durchnumeriert (die Numerierung steht oberhalb oder unterhalb der Stützpunkte).

-1 : Die im GKS genormten Markertypen werden zur Markierung der Kurven genutzt, d.h. mehrere Kurven werden durch verschiedene Markertypen unterschieden. Der Markertyp 1 (Punkt) wird nicht verwendet, da er auf Linien nicht zu sehen ist.

0 : Es wird kein Marker gezeichnet.

>0 : Die Kurve wird mit dem angegebenen Markertyp markiert. Bei Ausführung des Kommandos &run wird überprüft, ob dieser Markertyp für das gewählte Ausgabegerät existiert. Wenn nicht, so wird mit Markertyp 2 (+) markiert.

zeichen : Die Kurve wird mit dem angegebenen nichtnumerischen 'Zeichen' markiert.

= : keine Änderung

Default: -1

markerfreq. :: 0 : Es werden keine Marker gezeichnet.

1 : Jeder Stützpunkt wird markiert.

n : Jeder n-te Stützpunkt wird markiert. Es wird mit dem ersten darstellbarem Stützpunkt begonnen und auch der letzte darstellbare Stützpunkt wird markiert.

= : keine Änderung

Default: 1

interpolation :: 0 : Lineare Interpolation

1 : Stückweise kubische Hermitesche Interpolation. Die X-Werte der Variablen müssen monoton steigend sein, andernfalls wird linear interpoliert.

2 : glatte (einmal stetig differenzierbare) Funktionsinterpolation. Die X-Werte der Variablen müssen monoton steigend sein, andernfalls wird linear interpoliert.

3 : Darstellung durch Treppenkurven. Die X- oder Y-Werte der Variablen müssen monoton sein, andernfalls wird linear interpoliert.

4 : Kurveninterpolation mit Generalized Drawing Primitive (GDP)† des ZEDAT/GraS-GKS.

= : keine Änderung

Default: 0

Mit dem Profile-Kommando können weitere Kurven-Voreinstellungen verändert werden.

```
&PROFILE -CURVE <name> <parameterliste>
```

†GKS-Terminus; es handelt sich um eine Kurveninterpolation, die in der im ZIB vorhandenen GKS-Implementation vorhanden ist.

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	0	|	1	Modus, ob der Name der Kurve in der Legende erscheinen soll.
P 2	:	>	0.0		Linewidth der Kurve mit* Markergrößenanpassung.
P 3	:	-1	...	n	Marker-Farbe*.
P 4	:	1	...	n	Marker-Textfont*.
P 5	:	0	...	2	Marker-Textprecision*.
P 6	:	1	...	5	Numerierung oberhalb(5) oder unterhalb(1) des Markers.†
P 7	:	1	...	4	Treppeninterpolationsmodus
P 8	:	>	0.0		Marker-Größenfaktor

7.10 DATA

Dieses Kommando leitet die Eingabe der Daten ein. Erläuterung der Syntax in Kap. 5.4, S.17.

7.11 DEF

Das Kommando def Legt die Struktur der Eingabedaten fest. Erläuterung der Syntax in Kap. 5.3, S.13.

7.12 END

Dieses Kommando beendet den Eingabestrom eines Kommandos.

Kommandosyntax

```
&END
```

7.13 GRID

Dieses Kommando schaltet die Unterlegung der Zeichenfläche mit Gitterlinien ein oder aus und setzt deren Attribute. Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

Beispiel

Ziel : Es sollen senkrechte Gitterlinien über der X-Achse gezeichnet werden.

Kommando : &grid +x Properties → Grid → Vertical→ on

*GKS-Terminus, s. Glossar, Anhang C

†Diese Einstellung wirkt nur, wenn zusätzlich auch ein *Marker* ausgegeben wird.

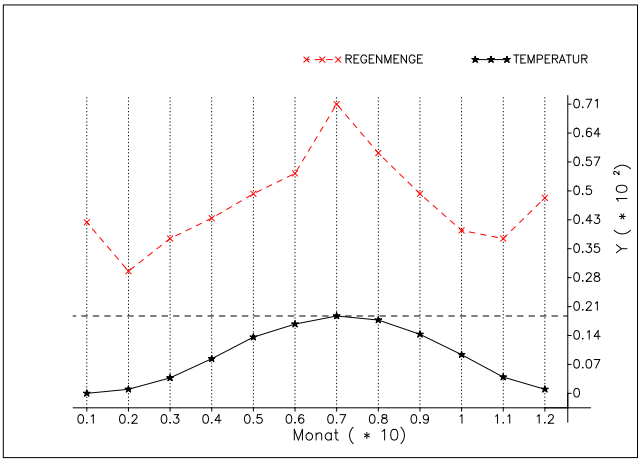

Fig. 13 Bild mit vertikalen Gitterlinien

Kommandosyntax (Form 1, zum Zeichnen eines kartesischen Gitters)

```
&GRid <on|off> [ <linetype> [ <colour> [ <linewidth> ] ] ]
```

on :: Die Zeichenfläche wird mit einem kartesischen Gitter unterlegt.

off :: Kein Gitter wird gezeichnet.

Default :: off

Kommandosyntax (Form 2, zum Zeichnen von Gitterlinien)

```
&GRID <[op]achsenbezeichner> [ <linetype> [ <colour>
      [ <linewidth> ] ] ]
```

op ::
- \+ : Auf der angegebenen Achse werden Gitterlinien gezeichnet.
- \- : Es werden kein Gitterlinien auf der angegebenen Achse gezeichnet.

achsenbez. ::
- X | VERTICAL : Es werden vertikale Gitterlinien über der X-Achse gezeichnet.
- Y | HORIZONTAL : Es werden horizontale Gitterlinien neben der Y-Achse gezeichnet.

linetype ::
- >0 : Die Gitterlinien werden mit dem angegebenen Linientyp* gezeichnet.
- $=$: keine Änderung
- Default : 3

colour ::
- ≥ 0 : Die Gitterlinien werden mit der angegebenen Linienfarbe* gezeichnet.

= : keine Änderung

Default : 1

linewidth :: >0.0 : Die Gitterlinien werden mit der angebenen Liniendicke* gezeichnet.

= : keine Änderung

Default : 1

7.14 HEADLINE

Das Kommando `HEADLINE` liest eine neue Überschrift ein.

Beispiel

Ziel : Die Zeichnung soll mit einer Überschrift versehen werden. Diese soll mit dem Textfont 12 gezeichnet und mit einem Rahmen versehen werden.

Kommando :
```
&headline 1
'Temperatur und Regenmenge'
&profile -headline
2 = = = = = = = = = 1
```

Properties → Headline
→ Headlinetext
→ Framecolour
→ Text Font

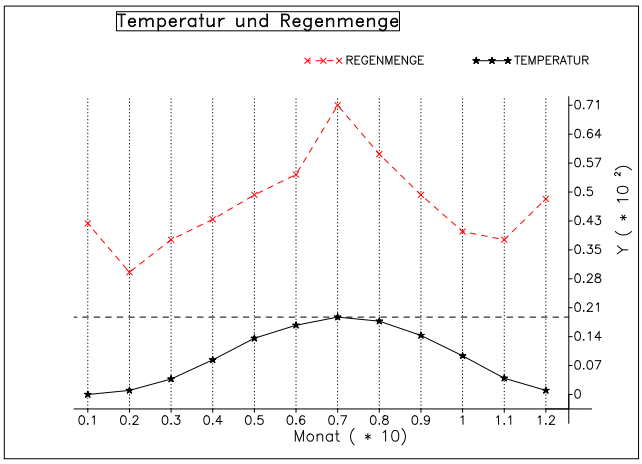

Fig. 14 Bild mit eingerahmter Überschrift

Kommandosyntax

```
&HEADLine <n> [ <'text'> ]
```

n :: 1 ... 10 : Die eingelesene Textzeile wird als n-te Überschriftszeile ausgegeben. Die Position der ersten Textzeile ist am oberen Rand der Zeichnung, mittig über den Kurven. Alle weiteren

*GKS-Terminus, s. Glossar, Anhang C

Zeilen werden jeweils mit einem bestimmten Abstand darunter ausgegeben. Zwischen dem oberen Rand und den Kurven ist standardmäßig Platz für zwei Zeilen.

text :: Text der Überschrift (max. 50 Zeichen, längere Texte werden abgeschnitten).

Mit dem Profile-Kommando können weitere Headline-Voreinstellungen verändert werden.

```
&PROFILE -HEADLINE <n> <parameterlist>
```

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	1	...	n	Textfont*			
P 2	:	0	...	2	Textprecision*			
P 3	:	0	...	n	Textfarbe			
P 4	:	>	0.0		Faktor der Character Height*			
P 5	:	>	0.0		Character Expansion Factor*			
P 6	:	>	0.0		Character Spacing*			
P 7	:				Textwinkel			
P 8	:	0	...	3	Horizontales Alignment*			
P 9	:	0	...	3	Textpath*			
P 10	:	0	...	1	X-Position	(0.0	=	linker Rand,
						1.0	=	rechter Rand)
P 11	:	0	...	1	Y-Position	(0.0	=	unterer Rand,
						1.0	=	oberer Rand)
P 12	:	-1	...	n	Textrahmenfarbe			
P 13	:	-1	...	n	Texthintergrundfarbe			

7.15 HELP

Ohne Angabe eines Parameters listet HELP die Kommandos auf; mit einem Kommandonamen als Parameter wird die Syntax des betreffenden Kommandos angezeigt.

Kommandosyntax

```
&Help [ <kommando> ]
```

kommando :: Ein gültiger Kommandoname.

*GKS-Terminus, s. Glossar, Anhang C

7.16 LAYER

Dieses Kommando wählt den aktuellen Layer aus. Ein Layer ist ein vollständiges Bild, welches nur angezeigt wird, wenn für den betreffenden Layer Variablen zur Darstellung ausgewählt wurden (siehe Kommando VARIABLES, S. 71). Die Bilder können sich dabei ganz oder teilweise überlappen.
Folgende Kommandos stellen Attribute ein, die nur für den aktuellen Layer gelten :

ATEXT, PROFILE-ATEXT
AXIS, PROFILE-AXIS, PROFILE-AXTICK
BOUNDS
GRID
MINRATIO
PROFILE-DRAWINGAREA
VARS

Beispiel

Ziel : Die beiden Kurven sollen auf zwei Layer verteilt werden, wobei die Y-Achse für die Temperatur am linken Rand der Zeichnung und die für die Regenmenge am rechten Rand gezeichnet werden sollen.

Die Hilfslinie soll nicht mehr gezeichnet werden.

Kommando :

```
&vars - regenmenge
&auxline 1 del
&axis y = min
&atext y 'Temperatur'
&layer 2
&axis y = max
&atext y 'Regenmenge'
&axis x del
&vars = regenmenge
```

Properties → Auxline → Direction

Properties → Axis → Layer2 → Y-axis → Intersection

Properties → Axis Text → Axis Text

Settings → Vars → Layer1 → temp → Layer2 → rain_m

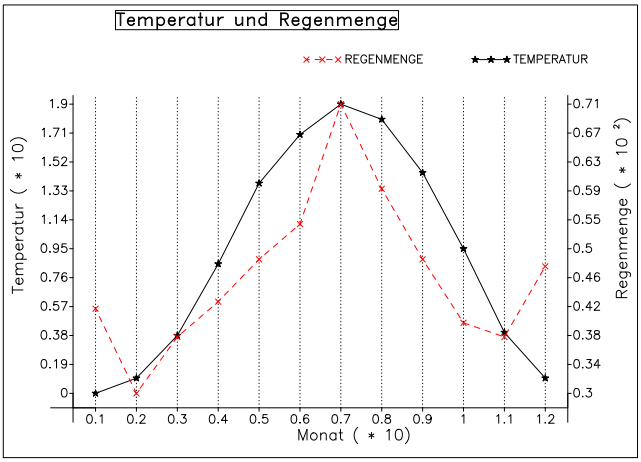

Fig. 15 Bild mit zwei übereinander liegenden Layern.

Kommandosyntax

`&LAYer <layerno>`

layerno	:: 1 ... 4 :	Nummer des, für die nachfolgen den Kommandos gültigen Layers.
	* :	Die nachfolgenden Einstellungen sind für alle Layer gültig.
		Default : 1

7.17 LEGENDTYPE

Dieses Kommando stellt die Art der Legende und deren Position ein. Wenn für eine Variable, deren Name in der Legende erscheinen soll, ein Alias-Name angegeben wurde, so wird dieser in der Legende gezeigt. Wurde für mehrere Variablennamen der gleiche Alias-Name definiert, so erscheint in der Legende der Alias-Name nur einmal und zwar für die erste Variable in der Auswahlreihenfolge (siehe Kommando VARIABLES, S. 71).
Bei Legendentyp 1 werden die Attribute der Kurve der ersten Variable verwendet. Bei Legendentyp 2 wird ausgehend von dem Alias-Namen der ersten Variablen auf alle weiteren Kurven mit einem Pfeil gezeigt, deren Variablennamen denselben Alias-Namen haben (siehe Kommando ALIAS, S. 27)

Beispiel

Ziel	:	Die Legende soll am linken, oberen Rand beginnen.
Kommando	:	&legend = left top Properties → Legend → Position

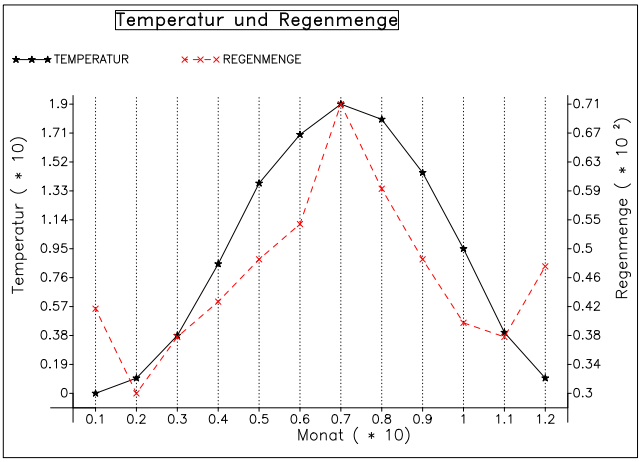

Fig. 16 Legende, in der linken oberen Ecke beginnend.

Kommandosyntax

```
&LEGtype <typ> [ <hor_pos> [ <vert_pos> [ <richtung>
               [ <spalten/reihen> ] ] ] ]
```

typ	::	0	:	Es wird keine Legende gezeichnet.
		1	:	Die Namen der gezeichneten Kurven werden tabellarisch mit einer kurzen Linie, wie sie in der Kurve verwendet wurde, aufgeführt.
		2	:	Die Kurven werden an einem Extremwert mit ihrem Namen versehen. Wenn die am Rand der Zeichnung stehenden Namen zu weit von der Kurve entfernt sind, wird ein Pfeil vom Namen zur Kurve gezeichnet (*Point to Curves*).
		=	:	keine Änderung
		Default	:	1
hor_pos	::	LEFT	:	Die Legende beginnt am linken Rand.
		RIGHT	:	Die Legende beginnt am rechten Rand.
		=	:	keine Änderung
		Default	:	RIGHT
vert_pos	::	TOP	:	Die Legende beginnt am oberen Rand.
		BOTTOM	:	Die Legende beginnt am unteren Rand.
		=	:	keine Änderung
		Default	:	TOP

richtung :: HORIZONTAL : Die Legende wird ausgehend von der angegebenen Position waagerecht weitergeführt.

VERTICAL : Die Legende wird ausgehend von der angegebenen Position senkrecht weitergeführt.

= : keine Änderung

Default : VERTICAL

spalt./reih. :: : Bei horizontaler Richtung Anzahl der Spalten, bzw. bei vertikaler Richtung Anzahl der Reihen mit Kurvennamen, nach deren Erreichen auf die nächste Spalte bzw. Reihe umgeschaltet wird.

-1 : Anzahl der Spalten bzw. Reihen ist gleich der Anzahl der Kurvennamen.

= : keine Änderung

Default : -1

Mit dem Profile-Kommando können weitere Legenden-Voreinstellungen verändert werden.

```
&PROFILE -LEGEND <parameterliste>
```

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	1	...	n	Textfont*
P 2	:	0	...	2	Textprecision*
P 3	:	0	...	n	Textfarbe
P 4	:	>	0.0		Faktor der Character Height*
P 5	:	>	0.0		Linewidth des Legendenrahmens
P 6	:	-1	...	n	Legendenrahmenfarbe
P 7	:	-1	...	n	Legendenhintergrundfarbe
P 8	:	0.	...	1.	Schwellwert für Pfeile bei Leg.-Typ 2; Bei 1.0 erscheinen keine, bei 0.0 erscheinen immer Pfeile.

*GKS-Terminus, s. Glossar, Anhang C

7.18 LIST

Dieses Kommando listet die aktuellen Einstellungen für den angegebenen Kommandonamen auf.

Kommandosyntax

```
&LISt <kommandoname>
```

kommandoname :: GRAZIL-Kommando, für das die aktuellen Einstellungen angezeigt werden sollen.

7.19 MINRATIO

Mit diesem Kommando kann verhindert werden, daß bei logarithmischer Y-Achse der wichtige Teil der Kurven sich im oberen Teil der Zeichnung„drängelt“. Dies geschieht durch Angabe eines Schwellwertes, den das Verhältnis von kleinsten zum zweitkleinsten Wert, bezogen auf das Verhältnis vom kleinsten zum größten Wert der darzustellenden Variablen unterschreiten muß. Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

Kommandosyntax

```
&MInratio <rwert>
```

rwert :: Realzahl im Intervall 0.0 ... 1.0 .
Default : 0.33

7.20 NAME

Mit diesem Kommando werden die Namen der darzustellenden Kurven (maximal 128) deklariert.

Kommandosyntax (Form1)

```
&name <name-1>, <name-2> ..., <name-n>
```

name-1, ... -n :: Namen der darzustellenden Kurven (alphanumerisch, jeweils max. 12 Zeichen).
n $\leq$ 128

Kommandosyntax (Form 2)

```
&name <n>
```

n :: Zahl der Kurven, die mit den Defaultnamen (V1 ... Vn) belegt werden (n $\leq$ 128).

7.21 PLOT

Dies ist ein Synonym für das Kommando &run (s. S. 67).

7.22 PROFILE

Mit diesem Kommando können eine Reihe von Voreinstellungen für verschiedene Zeichenobjekte, wie Achsen, Achsentexten, Kurven, usw. verändert werden.

Beispiel

Ziel	: Eine Texttabelle soll eingerichtet werden. Diese wird dann beim nächsten Beispiel verwendet.	
Kommando	: &profile -textab 1 'Jan' 'Feb' 'Mrz' 'Apr' 'Mai' 'Jun' 'Jul' 'Aug' 'Sep' 'Okt' 'Nov' 'Dez'	Settings → Texttable

Kommandosyntax

```
&PROfile -<objekt> [ <verteiler> ] <parameterliste>
```

objekt :: ATEXT : Voreinstellungen für den Achsentext. Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

- *verteiler* :: X | Y | *
- *Parameterliste* :: Siehe Kommando ATEXT, S. 30.

objekt :: AUXLINE : Voreinstellungen für die Hilfslinien.

- *verteiler* :: 1 ... 10 | *
- *Parameterliste* :: Siehe Kommando AUXLINE, S. 32.

objekt :: AXIS : Voreinstellungen für die Achsen. Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

- *verteiler* :: X | Y | *
- *Parameterliste* :: Siehe Kommando AXIS, S. 36.

objekt :: AXTICK : Voreinstellungen für die Beschriftung der Achsenticks. Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

- *verteiler* :: X | Y | *

Parameterliste :: Siehe Kommando AXIS, S. 36 .

objekt :: BARCHART : Voreinstellungen für die Barchartdarstellung.

verteiler :: Variablenname | *

Parameterliste :: Siehe Kommando RENDER, S. 58 .

objekt :: CURVE : Voreinstellungen für die Kurvendarstellung.

verteiler :: Variablenname | *

Parameterliste :: Siehe Kommando CURVE, S. 41 .

objekt :: DRAWINGAREA : Voreinstellungen für die Kurvenzeichenfläche. Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

Parameterliste ::

P 1	:	T	|	F	Flag, ob die Position der Kurvenzeichenfläche den nachfolgenden 4 Parametern entnommen werden soll
P 2	:	0	...	1	X-Position des linken unteren Punktes der Kurvenzeichenfläche. (0.0 = links, 1.0 = rechts)
P 3	:	0	...	1	Y-Position des linken unteren Punktes der Kurvenzeichenfläche. (0.0 = unten, 1.0 = oben)
P 4	:	0	...	1	X-Position des rechten oberen Punktes der Kurvenzeichenfläche. (0.0 = links, 1.0 = rechts)
P 5	:	0	...	1	Y-Position des rechten oberen Punktes der Kurvenzeichenfläche. (0.0 = unten, 1.0 = oben)

P 6	:	0	...	n	Farbe des Rahmens um die Zeichnung.
P 7	:	0	...	n	Dicke des Rahmens um die Zeichnung.
P 8	:	0	...	n	Hintergrundfarbe der Zeichnung.
P 9	:	-1	...	n	Farbe eines Rahmens um die Kurvenzeichenfläche.
P 10	:	-1	...	n	Hintergrundfarbe der Kurvenzeichenfläche.
P 11	:	T	|	F	Flag, ob eine Schneidekante in der gewünschten Zeichnungsgröße markiert werden soll.

objekt :: HEADLINE : Voreinstellungen für die Textzeilen.

verteiler :: 1 ... 10 | ∗

Parameterliste :: Siehe Kommando HEADLINE, S. 47 .

objekt :: LEGEND : Voreinstellungen für die Legendentexte

Parameterliste :: Siehe Kommando LEGENDTYPE, S. 50 .

objekt :: STAMP : Voreinstellungen für die Beschriftung des Datums- und Zeitstempels.

Parameterliste :: Siehe Kommando STAMP, S. 69 .

objekt :: TEXTTABLE : Erstellen einer Texttabelle.

verteiler :: 1 ... 10 | ∗ (Texttabellennummer)

Parameterliste ::

P 1	:	1.	Texteintrag (max. 20 Zeichen)
P 2	:	2.	Texteintrag
P 3	:	3.	Texteintrag
	.		
	.		
P n		n.	(max. 128) Texteintrag Die Anzahl der angegebenen Texte bestimmt die Länge einer Texttabelle. Es können maximal 10 Tabellen mit insgesamt 128 Texteinträgen erstellt werden.

Ein Gleichheitszeichen in der Parameterliste läßt den bestehenden Wert unverändert.

7.23 QUIT

Synonym für das Kommmando STOP (s. S. 71).

7.24 RECEIVE

Dieses Kommando dient zum Empfangen und Darstellen von Daten aus einem gleichzeitig laufenden Anwendungsprogramm. Voraussetzung ist, daß GRAZIL und das Anwendungsprogramm auf Rechnern laufen, zwischen denen eine Prozesskommunikation mittels UNIX-Sockets möglich ist. Bevor das Anwendungsprogramm die Kommunikation mit GRAZIL beginnt, muß das &receive-Kommando in GRAZIL aufgerufen werden. Siehe dazu auch Kap. 8.
Bei Verwendung des Kommandos &receive wird nur der aktuelle Layer angezeigt.

Kommandosyntax

```
&RECeive <operator> <var1> <var2> ... <varn>
```

operator :: = : Die empfangenen Daten für die angegebenen Variablen werden so gespeichert, daß evtl. vorhandene Daten überschrieben werden.

\+ : Die empfangenen Daten für die angegebenen Variablen werden zusätzlich zu evtl. bereits vorhandenen Daten gespeichert.

var-n :: Variablenname einer mit dem &name definierten Variablen (= Kurve) oder deren Alias-Name (siehe ALIAS, S. 27).
Wildcards sind hier nicht möglich! Die Stellung eines Namens in der Parameterliste korrespondiert mit der Variablennummer in der GRAZIL-Kommunikationsbibliothek (vgl. hierzu auch Kap. 8).

7.25 RENAME

Dieses Kommando verändert den Namen von Variablen oder Kommandos.

Kommandosyntax (Form 1)

```
&REName -c <alter Name> <neuer Name> [ <iwert> ]
```

alter Name :: Ein schon vorhandener Kommandoname.

neuer Name :: Neuer Kommandoname.

iwert :: Mindestanzahl der einzugebenden Buchstaben des neuen Namens.

Kommandosyntax (Form 2)

```
&REName -n <alter Name> <neuer Name>
```

alter Name :: Ein bereits deklarierter Variablenname.

neuer Name :: Neuer Variablenname.

7.26 RENDER

Mit diesem Kommando wählt man die Art der Darstellung aus und stellt die dazu gehörigen Darstellungsattribute ein. Die Darstellungsart bestimmt auch, in welcher Weise die Daten und ihre Struktur interpretiert werden. Folgende Darstellungsarten sind implementiert:

CURVE**:** Die Variablenwerte werden als Kurven ausgegeben (= Default). Die zum Zeichnen der Kurve einstellbaren Attribute sind die gleichen, wie im Kommando &curve. Mit folgendem Kommando kann die Kurvendarstellung noch beeinflußt werden:
&area = Fläche unter Kurven

VBARCHART**:** Die Variablenwerte werden als senkrechte Balken dargestellt. Dabei entspricht die Balkenhöhe den Y-Werten an den Datenpunkten*.

HBARCHART**:** Die Variablenwerte werden als waagerechte Balken dargestellt. Dabei entspricht die Balkenlänge den X-Werten an den Datenpunkten*.

SERRBAR**:** An die Stützpunkte der Variablen *name* werden symmetrische Fehlerbalken gezeichnet, deren Größe durch eine zweite Variable gegeben ist.

AERRBAR**:** An die Stützpunkte der Variablen *name* werden asymmetrische Fehlerbalken gezeichnet, deren Größe durch eine oder zwei weitere Variablen gegeben ist.

Beispiel

Ziel : Nur die Variable *temperatur* soll als Balkendiagramm ausgegeben, und die einzelnen Balken mit der bereits eingerichteten Tabelle beschriftet werden.

*Mit dem &bounds-Kommando erstellte Achsengrenzen haben keinen Einfluß, es werden stets alle Datenwerte berücksichtigt. Für beide Balkendarstellungen gilt: Die in Auswahlreihenfolge erste Variable eines Layers legt die Balkenstärke aller als Balkendiagramm auszugebenden Variablen fest; die Variable mit der größten Anzahl von Werten sollte daher als erste ausgewählt werden. Mit dem Kommando &area werden die Darstellungsattribute Farbe und Füllart der Balken eingestellt.

Kommando : &layer *
&vars =
&layer 1
&vars = temperatur
&axis x del
&legen 0
&grid off
&rende temp -vbarchart = = = 1
&prof -barchart temp = = = 1.4

Settings → Vars
→ Layer1 und 2
→ Clear Request-List
→ Layer1 → temp
Properties → Axis
→ X-Axis → Mode
Properties → Legend
→ Legendtype
Properties → Grid
→ Off
Properties → Render Attributes
→ temp
→ Render Bar Cart
→ Textmode
→ Text Height

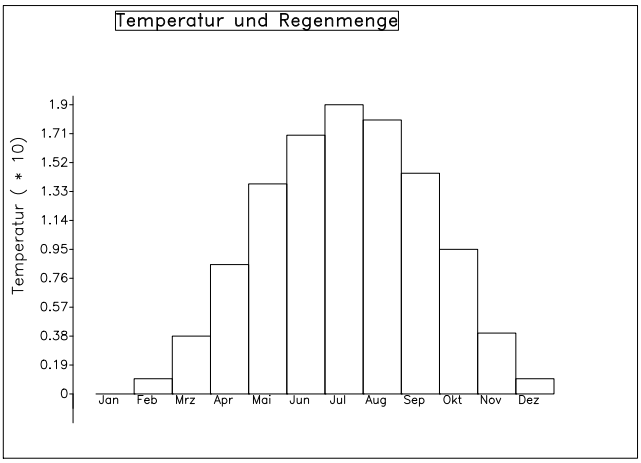

Fig. 17 Temperaturkurve als Balkendiagramm

Kommandosyntax

```
(1)  &RENDer <name> -CURVE [ <linecolour> [ <linetype>
             [ <markertype> [ <markerfrequenz>
             [ <interpolation> ] ] ] ] ]

(2)  &RENDer <name> -VBARCHART [ <balkenoffset> [ <varboffset>
             [ <startmodus> [ <beschriftungsmodus> ] ] ] ]

(3)  &RENDer <name> -HBARCHART [ <balkenoffset> [ <varboffset>
             [ <startmodus> [ <beschriftungsmodus> ] ] ] ]

(4)  &RENDer <name> -SERRBAR <errorvarname>[.X|.Y] [ <imodus>
             [ <ebcolour> [ <linewidth> [ <bmodus> ] ] ] ]

(5)  &RENDer <name> -AERRBAR <errorvarname1>[.X|.Y]
                             <errorvarname2>[.X|.Y]
```

[<ebcolour> [<linewidth> [<bmodus>]]]

name :: char-string Name einer mit dem Kommando &name definierten Variablen (= Kurve). Wildcards sind analog dem &vars-Kommando erlaubt.

linecolour ::

-1 : Die bei dem jeweiligen Ausgabegerät möglichen Farben werden zur Darstellung der Kurven genutzt, d.h. Kurven werden, soweit möglich, durch verschiedene Farben unterschieden.

$\geq$0 : Die Kurve wird in der angegebenen Farbe gezeichnet. Bei Ausführung des Kommandos &run wird überprüft, ob diese Farbe für das gewählte Ausgabegerät existiert. Wenn nicht, so wird mit Farbe 1 gezeichnet.

= : keine Änderung

Default: -1

linetype ::

-1 : Die im GKS genormten Linientypen werden zur Darstellung der Kurven genutzt, d.h. mehrere Kurven werden zur Unterscheidung mit verschiedenen Linienarten gezeichnet. Die punktierte Linie wird nicht verwendet, da sie für das Gitter reserviert ist.

0 : Es wird keine Linie gezeichnet.

$\geq$0 : Die Kurve wird mit dem angegebenen Linientyp† gezeichnet. Bei Ausführung des Kommandos &run wird überprüft, ob dieser Linientyp für das gewählte Ausgabegerät existiert. Wenn nicht, so wird mit Linientyp 1 gezeichnet.

= : keine Änderung

Default: -1

markertype ::

-3 : Die ersten 26 Kurvenstützpunkte werden mit 'A' bis 'Z' als Markierung durchnumeriert.

-3x : Die Stützpunkte werden mit dem Markertyp x markiert; die ersten 26 Stützpunkte werden zusätzlich mit 'A' bis 'Z' durchnumeriert (die Numerierung steht oberhalb oder unterhalb der Stützpunkte).

-2 : Die Stützpunkte werden zur Markierung von 1 bis n durchnumeriert.

-2x : Die Kurvenstützpunkte werden mit dem Markertyp x markiert und zusätzlich die Stützpunkte durchnumeriert (die Numerierung steht oberhalb oder unterhalb der Stützpunkte).

-1 : Die im GKS genormten Markertypen werden zur Markierung der Kurven genutzt, d.h. mehrere Kurven werden durch verschiedene Markertypen unterschieden. Der Markertyp 1 (Punkt) wird nicht verwendet, da er auf Linien nicht zu sehen ist.

0 : Es wird kein Marker gezeichnet.

>0 : Die Kurve wird mit dem angegebenen Markertyp markiert. Bei Ausführung des Kommandos &run wird überprüft, ob dieser Markertyp für das gewählte Ausgabegerät existiert. Wenn nicht, so wird mit Markertyp 2 (+) markiert.

zeichen : Die Kurve wird mit dem angegebenen nichtnumerischen 'Zeichen' markiert.

= : keine Änderung

Default: -1

markerfreq. :: 0 : Es werden keine Marker gezeichnet.

1 : Jeder Stützpunkt wird markiert.

n : Jeder n-te Stützpunkt wird markiert. Es wird mit dem ersten darstellbarem Stützpunkt begonnen und auch der letzte darstellbare Stützpunkt wird markiert.

= : keine Änderung

Default: 1

interpolation :: 0 : Lineare Interpolation

1 : Stückweise kubische Hermitesche Interpolation. Die X-Werte der Variablen müssen monoton steigend sein; andernfalls wird linear interpoliert.

2 : glatte (einmal stetig differenzierbare) Funktionsinterpolation. Die X-Werte der Variablen müssen monoton steigend sein, andernfalls wird linear interpoliert.

3 : Darstellung durch Treppenkurven. Die X- oder Y-Werte der Variablen müssen monoton sein; andernfalls wird linear interpoliert.

4 : Kurveninterpolation mit Generalized Drawing Primitive (GDP)† des ZEDAT/GraS-GKS

= : keine Änderung

Default: 0

balkenoffset :: >0.0 : Abstand zwischen zwei Balken einer Variablen in Einheiten der Balkenstärke. Die Balkenstärke wird in Abhängigkeit von der Anzahhl der Kurvenstützpunkte so eingerichtet, daß die Balken die zur Verfügung stehende Kurvenzeichenfläche möglichst ausnutzen.

= : keine Änderung

Default: 0.0

varboffset :: >0.0 : Abstand zwischen dem Anfang eines Balkens zu dem entsprechenden Balken der vorher auszugebenden Variablen. Es wird angegeben, wie groß der Abstand im Verhältnis zur Balkenstärke ist. „0." bedeutet, daß die Balken übereinander liegen. „1." bedeutet, daß die Balken nebeneinander liegen.

†GKS-Terminus; es handelt sich um eine Kurveninterpolation, die in der im ZIB vorhanden GKS-Implementation vorhanden ist.

= : keine Änderung

Default: 0.0

startmodus :: 0 : Die Grundlinie der Balken liegt am Variablenminimum.

1 : Die Grundlinie der Balken liegt am mit &Bounds eingestellten Minimum.

2 : Die Grundlinie der Balken liegt an der orthogonalen Achse.

3 : Die Grundlinie der Balken liegt auf dem entsprechenden X- bzw. Y-Wert der vorherigen Variablen. Die Grundlinie der Balken der ersten Variablen liegt bei deren Minimum.

4 : Die Grundlinie der Balken liegt auf dem entsprechenden X- bzw. Y-Wert der vorherigen Variablen. Die erste Variable fängt beim mit &bounds eingestellten Minimum an.

5 : Die Grundlinie der Balken liegt auf dem entsprechenden X- bzw. Y-Wert der vorherigen Variablen. Die Grundlinie der Balken der ersten Variablen liegt an der orthogonalen Achse.

= : keine Änderung

Default: 0

beschr.modus :: -2 : Die Balken werden mit den die Balkenlänge bestimmenden Variablenwerten beschriftet, d.h. vertikale Barcharts werden mit den Y-Werten, horizontale mit mit X-Werten beschriftet.

-1 : Die Balken werden mit den Werten der Balkenlaufrichtung beschriftet, d.h. vertikale Barcharts werden mit den X-Werten, horizontale mit den Y-Werten beschriftet.

0 : Keine Beschriftung der Balken.

1 ... 10 : Die Balken werden mit der Tabelle n beschriftet.

= : keine Änderung

Default: -1

errorvarname :: Variable, deren absolute Werte symmetrisch als Fehlerbalken an die entsprechenden Datenpunkte von *name* gezeichnet werden. Die Anzahl der Werte muß gleich der Anzahl der Werte von *name* sein. Ist die Variable zweidimensional und wurde keine Komponente (.X oder .Y) angegeben, so werden Fehlerkreuze gezeichnet. Wird eine Komponente angegeben, so wird ein Fehlerbalken der entsprechenden Richtung gezeichnet. Eine eindimensionale Variable wird als vertikaler Fehlerbalken interpretiert, wenn keine Komponente angegeben wurde.

errorvarname1 :: Variable, deren Werte als die positive Hälfte der Fehlerbalken an die Datenpunkte der entsprechenden Kurven gezeichnet werden. Ein '-' als Name unterdrückt das Zeichnen dieser Fehlerbalkenhälfte. Sonst wie *errorvarname*.

errorvarname2 :: Variable, deren Werte als die negative Hälfte der Fehlerbalken an die Datenpunkte der entsprechenden Kurven gezeichnet werden. Ein '-' als Name unterdrückt das Zeichnen dieser Fehlerbalkenhälfte. Sonst wie *errorvarname*.

imodus ::

1 : Der Absolutbetrag der Werte von *errorvarname* wird als Länge der Fehlerbalken interpretiert und beidseitig an die Datenpunkte der Kurve gezeichnet.

2 : Der Absolutbetrag der Werte von *errorvarname* wird als Länge der Fehlerbalken interpretiert und mittenzentriert an die Datenpunkte der Kurve gezeichnet.

= : keine Änderung

Default: 1

ebcolour ::

-1 : Die Farbe des Fehlerbalkens entspricht der Farbe* der Kurve von *name*.

≥0 : Die Fehlerbalken werden in der angegebenen Farbe gezeichnet.

= : keine Änderung

		Default:	-1
linewidth	::	>0. :	Liniendicke* der Fehlerbalken.
		= :	keine Änderung
		Default:	1.0
bmodus	::	0 :	Die Fehlerbalken werden als Linie gezeichnet.
		1 :	Die Fehlerbalken enden mit einem kleinen Querbalken.
		2 :	Die Fehlerbalken enden mit einem kleinen Pfeil.
		= :	keine Änderung
		Default:	0

Mit dem Profile-Kommando können weitere Kurven-Voreinstellungen verändert werden. Der erste Parameter kann bei allen Darstellungsarten verwendet werden, die weiteren Parameter haben nur eine Wirkung, wenn im &render-Kommando die Darstellungsart -CURVE gewählt wurde.

```
&PROFILE -CURVE <name> <parameterliste>
```

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	0	|	1	Modus, ob der Name der Kurve in der Legende erscheinen soll.
P 2	:	>	0.0		Linewidth* der Kurve mit Anpassung der Markergröße.
P 3	:	-1	...	n	Marker-Farbe.
P 4	:	1	...	n	Marker-Textfont*.
P 5	:	0	...	2	Marker-Textprecision*.
P 6	:	1	|	5	Numerierung oberhalb(5) oder unterhalb(1) des Markers.†
P 7	:	1	...	4	Treppeninterpolationsmodus
P 8	:	>	0.0		Marker-Größenfaktor

Mit dem Profile-Kommando können weitere Balken-Voreinstellungen verändert werden.

```
&PROFILE -BARCHART <name> <parameterliste
```

*GKS-Terminus, s. Glossar, Anhang C

*GKS-Terminus, s. Glossar, Anhang C

†Diese Einstellung wirkt nur, wenn zusätzlich auch ein *Marker* ausgegeben wird.

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	1	...	n	Textfont*
P 2	:	0	...	2	Textprecision*
P 3	:	0	...	n	Textfarbe
P 4	:	>	0.0		Faktor der Character Height*
P 5	:	>	0.0		Character Expansion Factor*
P 6	:	>	0.0		Character Spacing*
P 7	:				Textwinkel
P 8	:	1	...	5	Position des Textes in Balkenlängsrichtung: 1 : unterhalb/links des Balkens 2 : am unteren/linken Rand des Balkens 3 : in der Mitte des Balkens 4 : am oberen/rechten Rand des Balkens 5 : oberhalb/rechts des Balkens
P 9	:	1	...	3	Position des Textes quer zum Balken 1 : links/unterhalb des Balkens 2 : in der Mitte des Balkens 3 : rechts/oberhalb des Balkens
P 10	:	-1	...	n	Farbe* des Textrahmens
P 11	:	-1	...	n	Farbe* des Texthintergrundes

7.27 RESET

Dieses Kommando setzt Einstellungen auf einen vorherigen Zustand zurück.

Beispiel

Ziel : Die Layouteinstellungen sollen auf ihre Defaultwerte zurückgesetzt werden.

Kommando : &reset layout Reset → Layout to default

*GKS-Terminus, s. Glossar, Anhang C

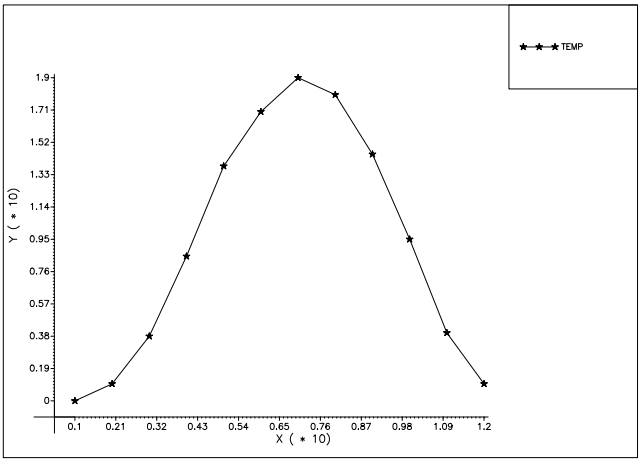

Fig. 18 Temperaturkurve nach dem Rücksetzen der Layouteinstellungen.

Kommandosyntax

`&RESet <state>`

state :: CURVE : Es werden alle Einstellungen, die die Darstellungen der Variablen bestimmen, auf ihren Defaultwert gesetzt (z.B. &curve-, &render-Einstellungen).

LAYOUT : Wie *curve*; zusätzlich werden alle weiteren, die Bilddarstellung bestimmenden Einstellungen auf ihren Defaultwert gesetzt (z.B. &legend-, &headline-, &axis-, usw. Einstellungen). Die Variablenauswahl bleibt erhalten.

ALL : Wie *layout*; zusätzlich werden das Ausgabefenster und alle Variablen gelöscht.

LASTRUN : Alle Einstellungen werden auf die Werte gesetzt, die sie vor dem letzten Aufruf von &run hatten. Es werden die Zustände vor den letzten zehn &run-Aufrufen gespeichert. Durch mehrmaliges Aufrufen von &reset lastrun (bis zu zehnmal) kann somit auf einen noch früheren Zustand zurückgesetzt werden.

7.28 RUN

Dieses Kommando stellt nach Überprüfung aller Einstellungen das Bild auf dem aktuellen Ausgabegerät dar. Es werden nur diejenigen Layer gezeigt, aus denen Variablen zur Darstellung gewählt wurden (siehe VARIABLES, S. 71). Bei Ausführung des Kommandos wird eine Datei angelegt, in die die aktuellen Einstellungen gesichert werden. Diese Datei wird vom Kommando &reset benutzt. Sie kann auch mit dem Kommando &comfile gezielt eingelesen werden

(siehe auch Kap. 4.3, Dateireferenzen von GRAZIL).

Kommandosyntax

`&Run`

7.29 SAVE

Sichert den aktuellen Stand der Einstellungen, nicht aber die Daten, in eine Datei. In UNIX-Systemen kann der Dateiname mit Pfad angegeben werden. Wenn die Datei nicht vorhanden ist, wird sie neu eingerichtet. Wird der Dateiname nicht in Apostrophe eingeschlossen, werden alle Kleinbuchstaben in Großbuchstaben übersetzt.

Beispiel

Ziel	: Alle Einstellungen sollen in die Datei *rain_temp.com* gesichert werden.	
Kommando	: &save * 'rain_temp.com' &save -prof 'rain_temp.com'	File → Save

Kommandosyntax

`&SAVe <kommando> <filename>`

kommando	:: Name :	Es werden die Einstellungen dieses Kommandos gesichert. Existiert die Datei *filename* schon, so wird an das Ende der Datei geschrieben.
	* :	Es werden die Einstellungen für alle Kommandos mit Ausnahme von &profile gesichert. Existiert die Datei *filename* schon, so wird die Datei überschrieben!
filename	::	Dateiname; bei Einschließung durch Apostrophe keine Übersetzung in Großbuchstaben!

7.30 SIZE

Dieses Kommando stellt die Größe der Zeichenfläche auf dem aktuellen Ausgabegerät ein. Die Angaben sind numerisch (in Metern) oder symbolisch (A-Format) zu machen.

Beispiel

Ziel	: Die Ausgabegöße soll auf das Format *A6* gesetzt werden.	
Kommando	: &size a6	Settings → Workstation → Format

Kommandosyntax

```
&SIze <xwert ywert> | <wert> | <Ax>
```

xwert ywert	::	Größe der Zeichenfläche in X- und Y-Richtung.
wert	::	Stellt eine quadratische Zeichenfläche mit der Seitenlänge <*wert*> ein.
Ax	::	A2 - A8: stellt das entsprechende DIN-A-Format als Zeichenfläche ein (quer). Default: A4 für das voreingestellte Ausgabegerät Metafile*.

7.31 STAMP

Mit diesem Kommando können ein Datums- und / oder Zeitstempel, oder zwei kurze Texte (jeweils max. 8 Zeichen lang) in eine Ecke der Zeichnung gesetzt werden.

Beispiel

Ziel : In der rechten unteren Ecke soll das aktuelle Datum, gegenüber der voreingestellten Größe um den Faktor 1.5 vergrößert, geschrieben werden.

Kommando :
```
&stamp = #d
&prof -stamp = = = 1.5
&comfile 'doku.layout'
```

Properties → Stamp
→ Text 2
→ Text Height

File → Load

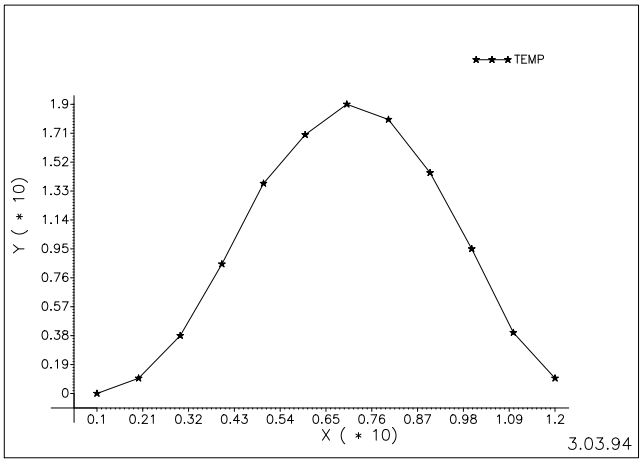

Fig. 19 Bild mit Datumstempel

Kommandosyntax

*GKS-Terminus, s. Glossar, Anhang C

`&STAMp [ <stamp_1> [ <stamp_2> [ <hor_pos> [ <vert_pos> ] ] ] ]`

stamp_1,stamp_2 :: #D : Es wird das aktuelle Datum eingetragen.

#T : Es wird die aktuelle Uhrzeit eingetragen.

'text' : Text (max. 8 Zeichen), der eingetragen wird.

' ' : Eintragung wird gelöscht.

= : keine Änderung

Default : ' ' , d.h. kein Datum- oder Zeitstempel wird ausgegeben.

hor_pos :: X-Koordinate für Anfangspunkt des Textes.

LEFT : der Stempel wird an den linken Rand gesetzt.

RIGHT : der Stempel wird an den rechten Rand gesetzt.

= : keine Änderung

Default : RIGHT

vert_pos :: Y-Koordinate für Textanfangspunkt.

BOTTOM : der Stempel wird an den unteren Rand gesetzt.

TOP : der Stempel wird an den oberen Rand gesetzt.

= : keine Änderung

Default : BOTTOM

Mit dem Profile-Kommando können weitere Stamp-Voreinstellungen verändert werden.

`&PROFILE -STAMP <parameterliste>`

Parameterliste :: Ein '=' (Platzhalter) läßt den Parameterwert unverändert.

P 1	:	1	...	n	Textfont*
P 2	:	0	...	2	Textprecision*
P 3	:	0	...	n	Textfarbe
P 4	:	>	0.0		Faktor der Character Height*
P 5	:	-1	...	n	Textrahmenfarbe
P 6	:	-1	...	n	Texthintergrundfarbe

*GKS-Terminus, s. Glossar, Anhang C

7.32 STOP | QUIT

Dieses Kommando beendet das Programm.

Kommandosyntax

`&STOp`

7.33 VARIABLES

Dieses Kommando wählt die darzustellenden Kurven aus. Es sind Wildcards erlaubt. Die Eingabe eines `"?"` ersetzt genau einen Buchstaben, die eines `"*"` ersetzt alle ab dem *. Durch Eingabe von `"**"` wird nach `"*"` gesucht (keine Wildcard).
Die hiermit eingestellten Attribute gelten nur für den aktuellen Layer (siehe Kommando LAYER, S. 49).

Beispiel

Ziel : Es sollen alle Kurven dargestellt werden.

Kommando : &vars = * Settings → Vars → Select all

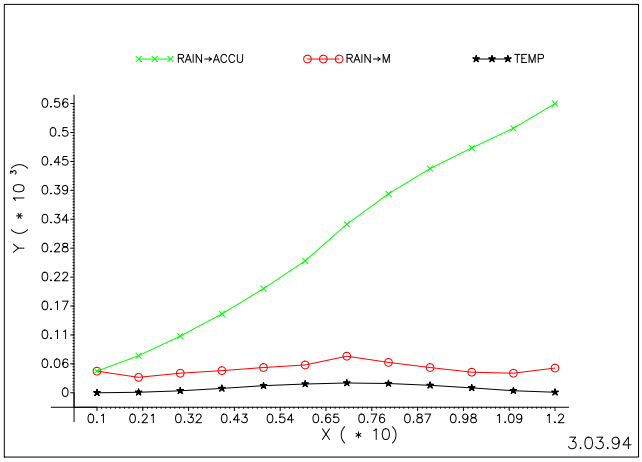

Fig. 20 Alle Kurven in einem Layer.

Kommandosyntax

`&Vars <operator> <var1> <var2> ... <varn>`

operator :: = : Trägt die angegebenen Variablen in die Liste der auszugebenen Kurven ein. Eine vorhandene Liste wird überschrieben. Wird nur der Operator `"="` gegeben, wird eine vorhandene Liste gelöscht.

\+ : Trägt die angegebenen Variablen zusätzlich in die Liste ein.

- : Entfernt die angegebenen Variablen aus der Liste.

var-x :: Kurven/Variablenname, der mit &name vereinbart wurde.

Default : keine

7.34 WORKSTATION

Dieses Kommando stellt das aktuelle Ausgabegerät ein. Ausgabegeräte können wirkliche Geräte sein, oder Files in denen das Bild in einem gewissen Format (z. B. CGM, PostScript, oder einem der verschiedenen GKS-Metafile-Formate) abgelegt wird. Wenn es sich bei dem Ausgabegerät um ein Sichtgerät handelt, wird dessen maximale Bildschirmgröße als Zeichenfläche eingestellt. Welche Ausgabegeräte angegeben werden können, ist von der zugrundeliegenden GKS-Implementation abhängig. Mit dem Kommando &list wktype können die aktuell vorhandenen Ausgabegeräte abgefragt werden, wobei in der Liste das eingestellte Ausgabegerät mit `"->"` markiert wird.

Kommandosyntax

```
&Wktype <iwert> [ <filename> | <rechnername> ]
```

iwert :: Workstation Typ,

1	:	NEWGKS coded Metafile OUTPUT
4	:	NEWGKS clear Text for editing
5	:	DEC Metafile
6	:	SEPP Metafile
7	:	GTSGRAL Metafile
8	:	Old Style FU-ZEDAT Metafile
10	:	CHARACTER CODED CGM
101	:	PostScript-Ausgabe DIN A4 hoch
102	:	PostScript-Ausgabe DIN A4 quer
103	:	Encapsulated PostScript-Ausgabe DIN A4 hoch
104	:	Encapsulated PostScript-Ausgabe DIN A4 quer
3100	:	SUN-Workstation 15“ Farbe
3110	:	SUN-Workstation 19“ Farbe
3111	:	SUN-Workstation 19“ S/W
3211	:	SGI-Workstation 19“ Farbe
74753	:	HP-Plotter 7475 DIN A3
74754	:	HP-Plotter 7475 DIN A4

filename :: Name einer Datei, in die der Output gelenkt wird. Wird *filename* nicht angegeben, so wird eine Defaultdatei angelegt (siehe Kap. 4, Dateireferenzen von GRAZIL).

Ist das ausgewählte Ausgabegeät ein Bildschirm und wird zusätzlich *filename* angegeben, so wird dieser Name als Rechnername interpretiert und die Ausgabe auf diesen Host gelenkt.

Namen zur Vermeidung der Übersetzung in Großbuchstaben mit Apostrophe einschließen.

Default : 1

8 Die GRAZIL-Kommunikationsbibliothek

Die GRAZIL-Kommunikationsbibliothek enthält Funktionen zur direkten Übergabe von Daten und Kommandos aus einem Anwendungsprogramm an GRAZIL. Hierbei ist es unerheblich, ob das Anwendungsprogramm und GRAZIL auf den gleichen Rechnern laufen oder auf verschiedenen.
Um diese Kommunikation zu ermöglichen, müssen die Funktionen mit dem Anwendungsprogramm gebunden und von diesem aufgerufen werden.
Die Kommunikation wird über UNIX-Sockets bewerkstelligt. Voraussetzung für die Kommunikation ist, daß der Rechner, auf dem das Anwendungsprogramm laufen soll, eine Netzverbindung zu dem Rechner hat, auf dem GRAZIL gestartet wurde, und diesen Rechner kennt. Die Kommunikation Anwendungsprogramm/GRAZIL erfolgt nach dem Client-Server-Modell. GRAZIL ist der Server. Dort muß zuerst das &receive-Kommando aufgerufen werden, bevor das Anwendungsprogramm seinerseits mit dem Aufruf von *gc_open* die Verbindung zu eröffnen versucht. Falls die Verbindung vor dem regulären Verbindungsabbau (Aufruf von *gc_close*) abbricht, geht GRAZIL nach einer gewissen Zeit wieder in den Kommandoeingabemodus über (Timeout).

8.1 Kommunikationsfunktionen

Im ZIB stehen die Funktionen bzw. Unterprogramme der GRAZIL-Kommunikationsbibliothek auf den SUN- und den CRAY-Rechnern sowohl für FORTRAN- wie auch für C-Programme zur Verfügung.
Die Funktion *gc_open* muß im Anwendungsprogramm vor allen anderen *gc_...*-Funktionen aufgerufen werden. Vor dem Aufruf von *gc_open* muß GRAZIL mit dem Kommando &receive in "Hörbereitschaft"versetzt werden.
Nachfolgend werden die einzelnen Funktionen in alphabetischer Reihenfolge beschrieben.

8.1.1 gc_bounds

Diese Funktion stellt den Wertebereich der Achsen in GRAZIL ein.

Aufruf

```
gc_bounds  (&error, axis, lowerbound, upperbound)
                                          ( C )
call gc_bou (error, axis, lowerbound, upperbound)
```

```
                                                ( SUN-FORTRAN )
call GCLBOU (error, axis, lowerbound, upperbound)
                                                ( CRAY-FORTRAN )
```

error :: wird auf -1 gesetzt, wenn der Aufruf nicht erfolgreich war; sonst wird 0 zurückgegeben.

axis :: Achsenbezeichner

0 : die Grenzwerte gelten für alle Achsen.

1 : die Grenzwerte gelten für die x-Achse.

2 : die Grenzwerte gelten für die y-Achse.

lowerbound :: unterer Grenzwert für den Darstellungsbereich der jeweils angegebenen Achse.

upperbound :: oberer Grenzwert für den Darstellungsbereich der jeweils angegebenen Achse.

8.1.2 gc_close

Die Funktion beendet die Verbindung zu GRAZIL. Hat GRAZIL bereits empfangene Daten noch nicht dargestellt, so werden diese bei Aufruf von *gc_close* dargestellt.

Aufruf

```
gc_close (&error)                ( C )
call gc_clo (error)              ( SUN FORTRAN )
call GCLCLO (error)              ( CRAY FORTRAN )
```

error :: wird auf -1 gesetzt, wenn der Aufruf nicht erfolgreich war; sonst wird 0 zurückgegeben.

8.1.3 gc_open

Die Funktion eröffnet eine Verbindung über UNIX-Sockets zu GRAZIL.

Aufruf

```
gc_open (&error, hostname)         ( C )
call gc_ope (error, hostname)      ( SUN FORTRAN )
call GCLOPE (error, hostname)      ( CRAY FORTRAN )
```

error :: wird auf -1 gesetzt, wenn der Aufruf nicht erfolgreich war; sonst wird 0 zurückgegeben.

hostname :: Name des Rechners, auf dem GRAZIL läuft und auf Eingabe über die Socketverbindung wartet.

Siehe Kommando &receive auf Seite 57.

8.1.4 gc_show

Die Funktion veranlaßt GRAZIL, die bereits empfangenen Daten darzustellen.

Aufruf

```
gc_show (&error)                    ( C )
call gc_sho (error)                 ( SUN FORTRAN )
call GCLSHO (error)                 ( CRAY FORTRAN )
```

error :: wird auf -1 gesetzt, wenn der Aufruf nicht erfolgreich war; sonst wird 0 zurückgegeben.

8.1.5 gc_svar

Die Funktion ordnet die zu sendenden Daten einer Variablen (Kurvenname) in GRAZIL zu.

Aufruf

```
gc_svar (&error, curvenumber)       ( C )
call gc_sva (error, curvenumber)    ( SUN FORTRAN )
call GCLSVA (error, curvenumber)    ( CRAY FORTRAN )
```

error :: wird auf -1 gesetzt, wenn der Aufruf nicht erfolgreich war; sonst wird 0 zurückgegeben.

curvenumber :: Position der Variablen in der Parameterliste von &receive siehe Seite 57 .

8.1.6 gc_2Ddata

Die Funktion sendet 2D-Daten an GRAZIL.

Aufruf

```
gc_2Ddat (&error, ndat, xval, yval)          ( C )
call gc_2Dd (error, ndat, xval, yval)        ( SUN FORTRAN )
call GCL2DD (error, ndat, xval, yval)        ( CRAY FORTRAN )
```

error :: wird auf -1 gesetzt, wenn der Aufruf nicht erfolgreich war; sonst wird 0 zurückgegeben.

ndat :: Anzahl der zu sendenden (x, y)-Werte (Datenpunkte).

xval :: Feld der Länge *ndat*, das zu übermittelnde X-Werte enthält.

yval :: Feld der Länge *ndat*, das zu übermittelnde Y-Werte enthält.

8.2 Beispiel

Nachfolgend ein Beispiel eines einfachen C-Programms, das Daten erzeugt und diese mit Hilfe der Kommunikationsroutinen an GRAZIL sendet, um sie grafisch darzustellen.

```
#include <stdio.h>
#include <math.h>

  main ()
    {
      int   num, ncurves, err, i, n;
      float a, b, x[3], y[3], delphi, alpha, xcen, ycen;

         xcen = 6.0;
         ycen = 6.0;
         a = 3.0;
         b = 3.0;
         ncurves = 3;
         num = 20;

         gc_open(&err, "host01");
```

/* Es wird eine Verbindung zu GRAZIL aufgebaut. GRAZIL ist vorher auf dem Rechner *host01* gestartet worden und bereit, mit einem Programm zu kommunizieren. */

```
         if (err >= 0 )
           {
             gc_bounds(&err, 0,  -1., 12.);
```

/* Der Darstellungsbereich im GRAZIL wird für beide Achsen auf ein Intervall von -1.0 bis 12.0 eingestellt. */

```
             for ( n=1; n<= ncurves; n++)
               {
                 gc_svar(&err, n);
```

/* Die Daten werden der n'ten im &receive-Kommando aufgeführten Variablen zugeordnet. */

```
                 a = a + 1.;
                 b = b + 1.;
                 delphi = 6.2831853 / num;
                 alpha = 0;
                 for ( i=1; i<= num +1; i++)
                   {
                     x[0] = xcen + a * cos(alpha);
```

```
                    y[0] = ycen + b * sin(alpha);
                    alpha = alpha + delphi;
                    gc_2Ddat( &err, 1, x, y);
```

/* Die jeweils errzeugten Datenpunkte werden an GRAZIL gesendet. */

```
                  }
                gc_show(&err);
```

/* GRAZIL zeigt die empfangenen Daten auf dem eingestellten Ausgabegerät an. */

```
              }
            gc_close(&err);
```

/* Die Verbindung zu GRAZIL wird beendet. */

```
          }
        else
          {
            exit(1);
          }
      }
```

8.3 Übersetzen und Binden

Unter der Voraussetzung, daß die GRAZIL-Kommunikationsbibliothek in dem Verzeichnis $GC_LIB liegt, wird ein Anwendungsprogramm *appl.c* bzw. *appl.f* wie folgt übersetzt und gebunden:

SUN C-Programm

%cc appl.c $GC_LIB/gc_clsunc.o -o main

SUN Fortran77-Programm

%f77 appl.f $GC_LIB/gc_clsunf.o -o main

CRAY C-Programm

%cc appl.c $GC_LIB/gc_clcrayc.o -o main -lm -lnet

CRAY Fortran77-Programm

1. Übersetzen mit CFT77

%cft77 appl.f

2. Binden

%segldr -lm -lnet -o appl appl.o $GC_LIB/gc_clcrayf.o

Bei der Installation am ZIB ist auf SUN-Rechnern

%setenv GC_LIB /usr/local/zib_lib

zu setzen. Auf den CRAY-Rechnern kann dies unterbleiben, da dort die Bibliothek im Standardpfad des Link-Editors liegt.

9 Tips und Tricks

Umlaute : Umlaute in Texten können mit den Textfonts

Roman Complex (= Font 3)

Gothic German (= Font 15)

erzeugt werden. In diesen Textfonts erzeugt der Backslash hochgesetzte Pünktchen ohne Buchstabenvorschub (z.B. \a = ä). Das Zeichen ß wird in diesem Textfont durch Eingabe des senkrechten Strichs (|) ausgegeben.

LaTeX-Texte : Mit Hilfe des TeX-Makros PSFRAG können Texte in Postscript-Bildern nachträglich verändert werden. Soll dies bei GRAZIL-Bildern benutzt werden, müssen die zu verändernden Texte kurz sein und mit der Textprecision *String* ausgegeben werden. Bei automatisch positionierten Texten kann das nachträgliche Verändern von Texten zu Komplikationen führen (im fertigen Bild können sich Texte überlappen).

Fonts : Für Plots guter Qualität sollte man die Adobe-Textfonts benutzen. Diese sind mit Textfontnummern von 22 bis 53 anzusprechen. Sie stehen nur für die Ausgabe nach Postscript zur Verfügung.

Fileausgabe : Bei Ausgabe auf Files (z.B. Ausgabegerät *Metafile*) sollte die Environment-Variable GRAZIL_RUN_ON_APPLY auf 0 gesetzt sein, um unabsichtliches Überschreiben zu vermeiden.

A Limitierungen

Maximale Anzahl der (X,Y)-Punkte pro Variable	:	4096	
Maximale Anzahl von Variablen	:	128	
Maximale Anzahl von darzustellenden Variablen	:	128	
Maximale Anzahl von Alias-Namen	:	128	
Maximale Anzahl von Hilfslinien	:	10	
Maximale Anzahl von Texttabellen	:	10	
Maximale Anzahl von Texten in allen Texttab.	:	128	
Maximale Anzahl von Layern	:	4	
Maximallänge der Variablennamen	:	12	Buchstaben
Maximallänge des Headline-Textes	:	50	Buchstaben
Maximallänge des Achsentextes	:	128	Buchstaben
Maximallänge des Hilfslinientextes	:	20	Buchstaben
Maximallänge des Stamp-Textes	:	8	Buchstaben
Maximallänge eines Alias-Namens für Variablen	:	20	Buchstaben
Maximallänge eines Dateinamens in Kommandos	:	256	Buchstaben
Maximallänge einer Eingabezeile	:	256	Buchstaben

B Layout-Automatismen

Um mit einem Minimum von Einstellungen stets ein sinnvolles Layout zu erhalten, sind in GRAZIL folgende Automatismen zur Steuerung des Layouts implementiert.

- Die Beschriftung der Achsenticks wird normiert, d.h. sie werden mit Zahlen $\in$ [-1.0, 1.0], beschriftet. Ist zur Normierung eine Zehnerpotenz erforderlich, wird diese an den Achsentext angehängt.

- Die X-Position des Y-Achsentextes ist abhängig von der Achsentickbeschriftung der Y-Achse.

- Die Y-Position des X-Achsentextes ist abhängig von der Achsentickbeschriftung der X-Achse.

- Die Position des Y-Achsentextes verschiebt sich an das obere Ende der Y-Achse, wenn eine horizontale Hilfslinie mit Beschriftung existiert.

- Die Position des X-Achsentextes verschiebt sich an das rechte Ende der X-Achse, wenn eine vertikale Hilfslinie mit Beschriftung existiert.

- Vertikale und horizontale Hilfslinien werden mit dem ihre Position definierenden X- bzw. Y-Wert beschriftet, falls die hierzu orthogonale Achse keine Achsenticks enthält und kein Hilfslinientext explizit angegeben wurde.

- Wenn als Schnittpunkt einer Achse nicht *min* eingestellt ist, wird der jeweilige Achsentext am Ende der Achse positioniert.

- Wenn keine Achsenticks verlangt werden, wird auch der Exponent nicht an den Achsentext angehängt.

- Bei dem Legendentyp 2 (*Point to Curves*) werden die Namen der Kurven stets so positioniert, daß sie sich nicht überschneiden. Überschreitet der Abstand vom Kurvennamen in der Legende zum nächsten Stützpunkt der Kurve einen Schwellwert, so wird zusätzlich ein Pfeil vom Kurvennamen zum Stützpunkt der Kurve gezeichnet.

- Balkendiagramme werden auf der Achse, auf der sie stehen, gleichmäßig verteilt. Mit &bounds eingestellte Grenzen wirken nur auf die Beschriftung der Achsen.

C Glossar von GKS-Begriffen

Die Erklärungen wurden unter Nutzung des Standardwerks zu GKS [BEC] erstellt; in diesem Werk finden sich zum Teil deutlich detailiertere Beschreibungen der Begriffe.

Alignment
: Das ist die Angabe, welcher Punkt am Text mit dem Startpunkt übereinstimmt. Das horizontale Alignment kann die Werte 1 (links), 2 (Mitte) und 3 (rechts) annehmen. Das vertikale Alignment kann die Werte 1 (oberer Rand der Textbox, 2 (obere Begrenzungslinie der Großbuchstaben), 3 (halbe Höhe der Textbox), 4 (untere Begrenzungslinie) der Großbuchstaben und 5 (unterer Rand) der Textbox annehmen. Die Begriffe “horizontal“ und “vertikal“ sind immer relativ zur Textlaufrichtung zu sehen.

Character Expansion Factor
: Der Character Expansion Factor steuert das Seitenverhältnis der Character Box, d.h. das einzelne Zeichen umrahmende Rechteck. Ist er gleich 1.0, dann ist die Box quadratisch, ist der Character Expansion Factor kleiner als 1.0, so ist die Box höher als breit. Ist er größer als 1.0, so ist die Box breiter als hoch.

Character Height
: Das ist die Höhe der Zeichen. In den GRAZIL-Kommandos kann immer nur ein Faktor eingegeben werden, der mit dem voreingestellten Wert für die Character Height multipliziert wird.

Character Spacing
: Das Character Spacing stellt den Abstand der Character Boxes ein. Ist das Spacing gleich 0.0, so werden die Boxes lückenlos aneinander gesetzt. Ist das Spacing größer als 0.0, so wird der Text gesperrt; ist es kleiner, überlappen sich die Character Boxes.

Colour
: Hiermit ist der “Polyline Colour Index“ bzw. “Polymarker Colour Index“

des GKS gemeint. Der Colour Index 0 repräsentiert immer die Hintergrundfarbe, alle anderen Werte eine auf dem aktuellen Ausgabegeät vorhandene Farbe. Der Colour Index 1 ist dabei meist die Standardfarbe d.h. Schwarz auf Druckern und Weiß oder Grün auf Monitoren.

In GRAZIL ist, bei Verwendung der ZEDAT/GraS-GKS-Implementation, auf Monitoren die Zeichenfläche weiß und die Farbe 1 schwarz.

Farbe

→ Colour

Line Type

Die Linientypen 1-4 sind im GKS genormt und demnach immer vorhanden:

1	=	durchgezogene Linie
2	=	gestrichelte Linie
3	=	punktierte Linie
4	=	strichpunktierte Linie

Weitere Linientypen können für bestimmte Ausgabegeäte vorhanden sein, wobei über ihr konkretes Aussehen keine Aussage getroffen werden kann.

Line Width

Das ist der "Linewidth Scale Factor" des GKS zur Angabe der Dicke einer Linie: es wird keine absolute Dicke angegeben, sondern ein Faktor zur "normalen" Liniendicke, welche wiederum abhängig ist von der Wahl des Ausgabegerätes.

Marker Type

Die Markertypen 1-5 sind im GKS genormt und müssen auf jeder GKS-Workstation vorhanden sein.

1	=	kleinstmöglicher Punkt	.
2	=	Kreuz	+
3	=	Sternchen	*
4	=	kleiner Kreis	o
5	=	Kreuz	x

Weitere Markertypen sind implementierungsabhängig.

Metafile

Das ist eine Datei, in der der graphische Output in wiederverwendbarer Form gespeichert werden kann. Das Format, in dem die einzelnen Informationen gespeichert werden, ist im GKS nicht genormt. Dies hat zur Folge, daß die Metafile-Formate verschiedener GKS-Implementierungen i.A. verschieden und nicht austauschbar sind.

normalisierte Koordinaten

Dimensionslose Koordinaten im Bereich [0., 1.]. →Weltkoordinaten werden im GKS in normalisierte Koordinaten transformiert.

Style
Das ist der "Interior Style" eines Füllgebietes im GKS. Er kann die Werte

0	=	Hollow	=	Umrandung einer Fläche,
1	=	Solid	=	Fläche in einheitlicher Farbe gefüllt,
2	=	Pattern	=	Fläche wird mit einem Muster gefüllt und
3	=	Hatch	=	Fläche wird schraffiert

annehmen.

Style Index
Der "Style Index" eines Füllgebietes gibt für die Interior Styles Pattern und Hatch eine genauere Spezifikation des Musters oder der Schraffur an. Muster und Schraffuren sind geräteabhängig.

Text Alignment
→ Alignment

Text Font
Der "Textfont" ist eine Nummer zum Auswählen verschiedener Schriftarten. Ihre Anzahl ist abhängig von der GKS-Implementation und vom Ausgabegerät. Im ZEDAT/GraS-GKS sind an allen Ausgabegeräten 21 Textfonts verfügbar. Zusätzlich stehen bei PostScript-Ausgabe 35 Adobe-Textfonts zur Verfügung.

Text Path
Das ist die Laufrichtung eines Textes. Er kann die Werte 0 (rechts), 1 (links), 2 (nach oben) und 3 (nach unten) annehmen.

Text Precision
Die "Textprecision" gibt die Qualität eines Textes an. Sie kann folgende Werte annehmen:

0	=	String	=	geringste Qualität,
1	=	Char	=	mittlere Qualität,
2	=	Stroke	=	höchste Qualität, der Text wird hierbei in Vektoren aufgelöst und alle Textattribute werden dargestellt.

Weltkoordinaten
Die Größenordnungen der Koordinaten und deren Dimension sind den darzustellenden Problemen angepaßt.

D Änderungen

GRAZIL Version 6.1 vom 31.12.93
Funktionserweiterungen gegenüber der vorherigen Version 6.0:

GUI: Diese GRAZIL-Version ist über ein grafisches User Interface zu bedienen.

Neuer Prompt: In der Kommandoschnittstelle meldet sich GRAZIL jetzt mit dem Prompt *grazil>*.

Default-Dateinamen: Bei der Ausgabe in Files wird als Default der Name der ersten beim Aufruf von GRAZIL angegebenen Eingabedatei oder der letzten mit &comfile eingelesenen Datei verwendet.

GRAZIL Version 6.0 vom 1.08.91

Funktionserweiterungen gegenüber der vorherigen Version 5.0:

Neue GKS-Version: Diese GRAZIL-Version wurde mit der neuen Implementierung des ZEDAT/GraS-GKS erstellt. Es müssen daher andere Environment-Variablen gesetzt werden! (siehe Kap. 4.3).

&layer**-Kommando:** neu, zum Auswählen und Setzen von bis zu 4 gleichzeitig darstellbaren (nebeneinander liegenden oder überlappenden) Bildern.

&render**-Kommando:** neu, zur Auswahl von Darstellungsarten (Balkendiagramme, Fehlerbalken)

&reset**-Kommando:** neu, zum Zurücksetzen von GRAZIL in einen früheren Zustand.

&auxline**-Kommando:** erweiterte Positionierung an Kurvenextrema möglich.

&def**-Kommando:** Erweiterungen in der Strukturbeschreibung.

&curve**-Kommando:** zusätzlich kann eine Kurve mit Treppeninterpolation dargestellt werden.

&axis**-Kommando:** zusätzliche Parameter zum Auswählen von Texttabellen zur Tickbeschriftung und zum Setzen der Tickpositionen.

Legende 2: Pfeile, die bei Legenden vom Typ 2 erscheinen können, können ausgeblendet, bzw. erst bei größeren Abständen zwischen Kurve und Kurvenbezeichnung gezeichnet werden.

Texttabellen: Es können mit dem &PROFILE-Kommando Texttabellen erzeugt werden, um Balkendiagramme und Achsenticks zu beschriften.

Defaultdatei: Bei Aufruf von GRAZIL wird eine Startup-Datei .grazil.zugr mit Voreinstellungen eingelesen.

E X-Resourcefile

Die nachfolgend genannten Einstellungen der X-Resourcen für das GUI bewirken daß

- in allen Fenstern die Aktionsknöpfe hellblau hinterlegt sind
- Der Quit-Knopf und die Cancel-Knöpfe zusätzlich einen roten Rahmen haben.

X-Resources für das GUI

ggui*cw1*background: LightSkyBlue
ggui*alias_pr_cw1*background: LightSkyBlue
ggui*area_pr_cw1*background: LightSkyBlue
ggui*atext_pr_cw1*background: LightSkyBlue
ggui*atick_pr_cw1*background: LightSkyBlue
ggui*auxl_pr_cw1*background: LightSkyBlue
ggui*auxltxa_pr_cw1*background: LightSkyBlue
ggui*axis_pr_cw1*background: LightSkyBlue
ggui*barch_pr_cw1*background: LightSkyBlue
ggui*bound_pr_cw1*background: LightSkyBlue
ggui*curve_pr_cw1*background: LightSkyBlue
ggui*errbar_pr_cw1*background: LightSkyBlue
ggui*file_pr_cw1*background: LightSkyBlue
ggui*render_pr_cw1*background: LightSkyBlue
ggui*dr_pr_cw1*background: LightSkyBlue
ggui*grid_pr_cw1*background: LightSkyBlue
ggui*headl_pr_cw1*background: LightSkyBlue
ggui*legnd_pr_cw1*background: LightSkyBlue
ggui*stmp_pr_cw1*background: LightSkyBlue
ggui*txtab_pr_cw1*background: LightSkyBlue
ggui*vars_pr_cw1*background: LightSkyBlue
ggui*wk_pr_cw1*background: LightSkyBlue
ggui*quit_but*borderColor: red
ggui*alias_cancel_but*borderColor: red
ggui*area_cancel_but*borderColor: red
ggui*atext_cancel_but*borderColor: red
ggui*atick_cancel_but*borderColor: red
ggui*auxl_cancel_but*borderColor: red
ggui*auxltxa_cancel_but*borderColor: red
ggui*axis_cancel_but*borderColor: red
ggui*barch_cancel_but*borderColor: red
ggui*bound_cancel_but*borderColor: red
ggui*curve_cancel_but*borderColor: red
ggui*errbar_cancel_but*borderColor: red
ggui*file_cancel_but*borderColor: red
ggui*render_cancel_but*borderColor: red
ggui*dr_cancel_but*borderColor: red
ggui*grid_cancel_but*borderColor: red
ggui*headl_cancel_but*borderColor: red
ggui*legnd_cancel_but*borderColor: red
ggui*stmp_cancel_but*borderColor: red
ggui*txtab_cancel_but*borderColor: red
ggui*vars_cancel_but*borderColor: red
ggui*wk_cancel_but*borderColor: red

```
ggui*squitbutton*background: LightSkyBlue
ggui*sapplbutton*background: LightSkyBlue
ggui*sinfotoggle*background: LightSkyBlue
ggui*squitbutton*borderColor: red
```

F Kommandofile

Die nachfolgend aufgeführten GRAZIL-Kommandos wurden benutzt, um die Voreinstellungen von GRAZIL so zu verändern, daß alle Texte vergrößert gezeichet werden, und die Größe der Zeichenfläche den Erfordernissen dieser Dokumentation angepasst ist.

GRAZIL-Kommandos aus *doku.layout*

```
&legtype 1 right top horizontal -1
&layer *
&profile -drawingarea t 0.11 0.11 0.89 0.80
&prof -atext * = = = 1.4
&prof -axtick * = = = 1.4
&layer 1
&prof -auxline * = = = 1.4
&prof -headline * = = = 1.4
&prof -legend = = = 1.4 = -1 -1
&prof -stamp = = = 1.4
```

G Danksagung

An dieser Stelle wollen wir Herrn U. Pöhle danken, der stets mit Tips und konstruktiver Kritik die Entwicklung dieses Plotprogramms begleitet hat.

Literatur

[BEC] J. Bechlars, R. Buhtz: *GKS in der Praxis*, Springer-Verlag, 1986.

Eine Neuauflage wird voraussichtlich im November 1994 erscheinen.

[GNU] T. Williams, C. Kelly: *GNUPLOT 3.2 Manual, An Interactive Plotting Programm*, 1993, Mail to info-gnuplot@ames.arc.nasa.gov.

[LAN] J. Langendorf, O. Paetsch: *GRAZIL - Ein graphisches Anwendungsprogramm zur Darstellung von Kurven und Funktionsverläufen im Koordinatensystem*, Konrad Zuse-Zentrum für Informationstechnik Berlin, Technical Report 87-7 1987

Index