Gnuplot

Merkmale

Gnuplot (ein Wortspiel auf "Newplot") ist ein frei erhältliches, sehr flexibles kommandozeilenorientiertes Graphikpaket zur Darstellung zwei- und dreidimensionaler Daten und Funktionen. Darüber hinaus können mit Gnuplot auch beliebige benützerdefinierte Funktionen an Daten angepaßt ("gefittet") werden.

Obwohl es auch graphische Interfaces (z.B. xgfe) zu Gnuplot gibt, ermöglicht die Steuerung des Programms über eine Kommandozeile ein sehr rasches und effizientes Arbeiten. Kommandos können außerdem in derselben Weise, wie man sie per Hand eingibt, in Script-Files zusammengefaßt werden. Dies stellt eine sehr kompakte Möglichkeit dar, aufwendig gestaltete Graphiken zu speichern und jederzeit, durch Ausführung des Scripts in Gnuplot, zu reproduzieren. Ein weiterer Vorteil der Kommandozeilenorientierung besteht darin, daß man Gnuplot von anderen Programmen aus ansteuern kann und sich damit u.U. die Entwicklung einer eigenen Graphiksoftware erspart.

Gnuplot läuft auf fast allen Architekturen (Unix, VMS, Windows, DOS, ...) und unterstützt die Ausgabe in einer großen Anzahl von Formaten. Die wichtigsten Formate ("Terminals") sind X11, Postscript und verschiedene Bitmap-Formate. Je nach den Darstellungsmöglichkeiten der einzelnen Terminals sehen die Graphiken allerdings verschieden aus.

Prinzipiell kann man in Gnuplot zwei Arten von Kommandos unterscheiden, nämlich die verschiedenen `set`-Kommandos zur individuellen Gestaltung von Graphiken und die eigentlichen Plot- bzw. Fit-Befehle `plot`, `splot` und `fit`. Dazu kommen noch Definitionen, z.B. im Fall von benützerdefinierten Funktionen oder Konstanten. Gibt man keine speziellen Anweisungen hinsichtlich des Aussehens einer Graphik, so trifft Gnuplot geeignete Standardannahmen, sodaß man sich meist mit einem oder zwei Kommandos schon einen schnellen ersten Überblick über Daten verschaffen kann.

Je nach zu erstellender Graphik kommen bei Gnuplot drei verschiedene Arten von Files zur Verwendung: ein oder mehrere Datenfiles (sofern nicht nur analytisch gegebene Funktionen gezeichnet werden), ein Script-File (falls verwendet) und ein oder mehrere Ausgabefiles (wenn als Terminal etwas anderes als Ausgabe am Monitor gewählt wird).

Datenfiles

Die Datenfiles sind in der Regel ASCII-Files und können mit einem Editor erstellt oder der Ouput eines anderen Programms sein. Die Daten müssen in einer oder mehreren Spalten angeordnet sein, die durch Leerzeichen, Komma oder `<tab>` getrennt sind. Zeilen, die mit dem `"#"`-Zeichen beginnen, gelten als Kommentar und werden ignoriert. Die numerischen Werte können ganzzahlig, Kommazahlen oder im Exponentialformat sein.

2D Plots

Zweidimensionale Graphiken werden auch als XY-Plots oder 2D Plots bezeichnet. Datenfiles, die als Grundlage für 2D Plots dienen, bestehen i.a. aus zwei oder (wenn auch Fehlerbalken

gezeichnet werden sollen) mehr Spalten

$$\begin{array}{lll} x_1 & y_1 & \dots \\ x_2 & y_2 & \dots \\ x_3 & y_3 & \dots \\ \vdots & \vdots & \dots \\ x_N & y_N & \dots \end{array}$$

Wenn nicht durch die `using`-Option des `plot`-Befehls (s. unten) anders angegeben, werden, wie angedeutet, die ersten beiden Spalten als x- und y-Werte interpretiert.

3D Plots

Bei dreidimensionalen Graphiken kann es sich um "Punktwolken" (Scatter Plots) oder die Darstellung zweidimensionaler Flächen im Raum handeln. Im ersten Fall wird das Datenfile in der Regel mindestens drei Spalten enthalten, die als x-, y- und z-Werte der Datenpunkte aufgefaßt werden. Sollen die Daten jedoch als Drahtgitter- (Wireframe) oder schattierte Fläche (`pm3d`-Stil) dargestellt werden, so müssen die x- und y-Werte ein Gitter von $M \times N$ Punkten bilden und die Daten im File in durch je eine Leerzeile voneinander getrennten Blöcken angeordnet sein, wobei innerhalb jedes Blocks z.B. nur die x-Werte variieren, y aber fest ist

$$\begin{array}{llll} x_1 & y_1 & z_{11} & \dots \\ x_2 & y_1 & z_{21} & \dots \\ \vdots & \vdots & \vdots & \dots \\ x_M & y_1 & z_{M1} & \dots \\ \\ x_1 & y_2 & z_{12} & \dots \\ x_2 & y_2 & z_{22} & \dots \\ \vdots & \vdots & \vdots & \dots \\ x_M & y_2 & z_{M2} & \dots \\ \\ \vdots & \vdots & \vdots & \dots \\ \\ x_1 & y_N & z_{1N} & \dots \\ x_2 & y_N & z_{2N} & \dots \\ \vdots & \vdots & \vdots & \dots \\ x_M & y_N & z_{MN} & \dots \end{array}$$

Natürlich können die Rollen von x und y auch vertauscht werden.

Erste Kommandos

Gnuplot wird von der Unix-Kommandozeile mit Hilfe des Befehls

```
$ gnuplot
```

aufgerufen ("$ steht für das Shell-Prompt) und meldet sich mit

```
gnuplot>
```

Beschreibungen zu den einzelnen Gnuplot-Themen erhält man mit

`gnuplot> help` *topic*

oder nur

```
gnuplot> help
```

Gnuplot wird verlassen und zur Shell zurückgekehrt mit

```
gnuplot> quit
```

Statt `"quit"` genügt auch `"q"`, da alle Befehle abgekürzt werden können. Mit der Eingabe von `"q"` auf der Tastatur kann man auch ein aktives Graphikfenster schließen (ohne Gnuplot zu verlassen).

Der Befehl

`gnuplot> plot '`*filename*`'`

zeichnet die ersten beiden Spalten (interpretiert als x- und y-Werte) des Files *filename*, wobei zur Darstellung Symbole verwendet und Skalierung sowie Beschriftung an die Daten angepaßt werden. Analog liefert

`gnuplot> splot '`*filename*`'`

eine dreidimensionale Punktwolke der ersten drei Spalten von *filename*, die als x-, y- und z-Koordinaten der Datenpunkte aufgefaßt werden. Die Darstellung von analytisch gegebenen Funktionen $f(x)$ bzw. Flächen $f(x, y)$ erfolgt ebenfalls mit diesen Befehlen, wobei statt des Filenamens die gewünschte Funktion angegeben wird, also

`gnuplot> plot` *f(x)*

oder

`gnuplot> splot` *f(x,y)*

Als Standardintervall/gebiet wird in diesem Fall $[-10, 10]$ bzw. $[-10, 10] \times [-10, 10]$ angenommen.

Die Plot-Befehle

Der `plot`-Befehl

Der `plot`-Befehl kann durch verschiedene Optionen modifiziert werden. Einige (aber bei weitem nicht alle) Möglichkeiten sind

`plot '`*filename*`' using` *m*`:`*n* `title '`*text*`' with` *style*

bzw.

`plot` *function* `title '`*text*`' with` *style*

`"using"`, `"text"` und `"with"` können durch `"u"`, `"t"` und `"w"` abgekürzt werden.

Die `using`-Option dient dazu, andere als die ersten beiden Spalten des Datenfiles als x- und y-Werte auszuwählen. Im einfachsten Fall sind dies Spalte m als x- und Spalte n als y-Koordinate. Darüber hinaus gibt es aber auch noch die Möglichkeit, beliebige Kombinationen und Transformationen der Werte einer Zeile im Datenfile zu bilden. In diesem Fall spricht man den Wert in Spalte m mit $\$m$ an und setzt die als x- bzw. y-Werte zu verwendenden Ausdrücke in Klammern, z.B.

```
using (sqrt($1**2+$2**2)):(abs($3))
```

Mit Hilfe von `title` kann ein Text für die Legende spezifiziert werden. (Sofern mit {`un`}`set key` nicht anders angegeben, erscheint in der oberen rechten Ecke zu jedem gezeichneten Datensatz bzw. zu jeder Funktion der entsprechende Text mit dem zugeordneten Symbol- oder Linientyp.)

Die Option `with` gibt an, wie die Daten oder die Funktion darzustellen sind. Hier gibt es eine ganze Reihe von Möglichkeiten (s. `help plot with`), die ihrerseits wieder modifiziert werden können. Am wichtigsten sind

`with lines` {`linetype` *lt*} {`linewidth` *lw*}
`with points` {`linetype` *lt*} {`pointtype` *pt*} {`pointsize` *ps*}
`with dots` {`linetype` *lt*}
`with errorbars` {`linetype` *lt*} {`linewidth` *lw*} {`pointtype` *pt*} {`pointsize` *ps*}

Daneben gibt es auch noch `linespoints`, `impulses`, `steps` usw. Bei `errorbars` muß das Datenfile drei oder mehr Spalten enthalten, bzw. müssen drei mit `using` ausgewählt werden. Es können wieder Abkürzungen verwendet werden, z.B. genügt statt der ersten Zeile

`w l lt` *lt* `lw` *lw*

lt, *lw* usw. sind, je nach Bedeutung, ganzzahlige Codes oder reelle Parameter. Diese Codes haben allerdings für verschiedene Ausgabemedien (Terminals) verschiedene Bedeutung!

Soll eine Graphik mehrere, übereinander zu zeichnende Datensätze und/oder Funktionen enthalten, so kann man dem `plot`-Befehl auch eine durch Beistriche getrennte List von Filenamen/Funktionen + Optionen geben

```
plot 'file1' using...,\
     'file2' using...,\
     'file3' using...
```

Überlange Zeilen können mit "\" abgeteilt werden.

Der `splot`-Befehl

Der `splot`-Befehl ist dem `plot`-Kommando ähnlich, kennt aber weniger bzw. andere Stiloptionen (z.B. keine `errorbars`). Er dient außer zum Zeichnen von dreidimensionalen Datensätzen der Darstellung von Funktionen von zwei Variablen, $z = f(x, y)$ und wird weiter unten näher beschrieben.

Der `replot`-Befehl

Der Befehl

```
replot
```

führt das letzte `plot`- oder `splot`-Kommando nochmals aus.

Gestaltung einer Graphik

Der `set`-Befehl

Der `set`-Befehl (oder eigentlich *die* `set`-Befehle, denn es gibt weit über 100!) dient zur individuellen Gestaltung von Graphiken bzw. zum Setzen interner Parameter von Gnuplot. Zu jedem `set` gibt es i.a. auch ein `show`, das den momentanen Wert des Parameters angibt, sowie ein `unset`, das eine getroffene Definition widerruft bzw. auf einen Standardwert zurücksetzt. So legt z.B.

`set size` *xsize*`,`*ysize*

Skalierungsfaktoren für die Graphik fest, die, wie man mit

```
show size
```

feststellt, standardmäßig `1,1` sind.

Die Wirkung der `set`-Befehle wird i.a. erst nach dem nächsten `plot`, `splot` oder `replot` sichtbar.

Achsen und Achsenbeschriftung

Standardmäßig skaliert und beschriftet Gnuplot die Achsen abhängig vom Bereich der darzustellenden Daten. Will man das Aussehen und die Beschriftung der Achsen selbst definieren, so gibt man zunächst das Intervall, das auf die Achse abgebildet werden soll, mit

`set xrange [`*xmin*`:`*xmax*`]`

an. Hier, wie auch in den folgenden Kommandos, wurde nur der Befehl für die x-Achse angeführt; die Befehle für die anderen Achsen erhält man, indem man einfach `"x"` durch `"y"` oder `"z"` ersetzt, also z.B. `set yrange` oder `set zrange`.

Wo auf den Achsen größere Unterteilungsstriche (Tick Marks) und eine Beschriftung angebracht werden sollen, legt man mit

`set xtics` *start*`,`*step*`,`*end*

fest. Die Anzahl der dazwischenliegenden kleineren Unterteilungsstriche (Minor Tick Marks) definiert man mit

`set mxtics` n

Das Intervall zwischen zwei größeren Strichen wird dabei in n Teile geteilt. Ist `mxtics` nicht gesetzt, so ist das gleichbedeutend mit $n = 1$.

Das numerische Format für die bei den größeren Teilstrichen anzubringenden Zahlen kann man mit

`set format x '`*format*`'`

festlegen. Dabei ist *format* eine Formatbeschreibung in C-Syntax, also z.B. `'%.0f'` für Zahlen ohne Kommapunkt oder `'%.2f'` für Zahlen mit zwei Nachkommastellen.

Tick Marks und Beschriftung an beliebigen Stellen, x_1, x_2, ..., der Achsen kann man mit dem Kommando

`set xtics ("`*text1*`"` x_1 `, "`*text2*`"` x_2`, ...)`

anbringen.

Die Legende für eine Achse gibt man mit

`set xlabel '`*text*`'`

an.

Schließlich kann man noch mit

{`un`}`set logscale x`

zwischen linearen und logarithmischen Achsen wählen.

Beschriftung

Neben der Achsenbeschriftung besteht noch die Möglichkeit, die Graphik mit einer Überschrift zu versehen, an verschiedenen Stellen Texte oder Hinweispfeile anzubringen und die Legende ein- und auszuschalten bzw. an verschiedenen Positionen anzuordnen.

Der Befehl

`set title '`*text*`'`

versieht die Graphik mit der Überschrift *text*.

Eine Beschriftung an einer beliebigen Stelle in der Graphik bringt man mit

`set label` *n* *'text'* `at` *x*,*y*{,*z*} {*justification*} {`rotate`}

an. Dabei ist $n = 1, 2, 3, \ldots$ eine Marke (Tag) zur Identifikation des Labels, x, y (und z) seine Position (normalerweise im Koordinatensystem der Achsen), und *justification* (`left`, `right` oder `center`) gibt an, ob der Text dort linksbündig, rechtsbündig oder zentriert ausgerichtet werden soll. Mit `rotate` kann bei manchen Terminals der Text um 90 Grad gedreht werden.

Die Legende (Zuordnung der Punktsymbole oder Linientypen zum Datensatz oder zur Funktion, wie sie entweder automatisch oder mit Hilfe der `title`-Option des `(s)plot`-Kommandos angegeben wird) erscheint normalerweise in der rechten oberen Ecke der Graphik. Man kann die automatisch erzeugte Legende mit

`unset key`

unterdrücken bzw., falls in der rechten oberen Ecke kein Platz ist, die Legende mit

`set key` {*position*} {*justification*}

an einen anderen Ort setzen. Dabei kann *position* eine Kombination von `left`, `right`, `top` und `bottom` oder eine Koordinatenangabe *x*,*y*(,*z*) sein und *justification* `left` oder `right`. Daneben gibt es noch eine Reihe anderer Optionen (s. `help set key`).

Script-Files

Alle `set`-, `plot`-Befehle und sonstigen Kommandozeileneingaben in Gnuplot können auch von einem Script-File ausgeführt werden. Das Script-File ist ein gewöhnliches ASCII-File, das mit einem beliebigen Editor erstellt werden kann und in das man die Befehle in derselben Art, wie man sie auch interaktiv absetzen würde, der Reihe nach einträgt. Zeilen, die mit dem `"#"`-Zeichen beginnen, sind Kommentarzeilen und werden ignoriert; überlange Zeilen können mit `"\"` am Zeilenende abgeteilt werden.

Um ein Script in Gnuplot auszuführen, gibt man

`load` *'scriptfile'*

ein. Umgekehrt kann man mit

`save` *'scriptfile'*

alle zum momentanen Zeitpunkt einer interaktiven Gnuplot-Sitzung gültigen Definitionen und Optionen sowie das letzte `(s)plot`-Kommando in einem Script-File speichern.

Ein Script-File kann auch direkt von der Kommandozeile der Shell (als eine der Kommandozeilenoptionen an Gnuplot) ausgeführt werden

```
$ gnuplot {other_options} scriptfile
$
```

Die Kontrolle wird nach Beendigung des Scripts an die Shell zurückgegeben.

Terminals

Der etwas irreführende Begriff "Terminal" gibt an, auf welchem Ausgabemedium bzw. in welchem Ausgabeformat man die Graphik erstellen möchte. Gnuplot unterstützt eine Vielfalt von Ausgabemedien und -formaten; welche in einer bestimmten Installation verfügbar sind, kann man mit dem Befehl

```
set terminal
```

herausfinden.

Beim Start von Gnuplot unter Unix wird normalerweise als Standardannahme das `x11`-Terminal (also Ausgabe auf dem Monitor unter X-Window) voreingestellt. Will man auf ein anderes Ausgabemedium wechseln, so erfolgt das mit dem Befehl

`set terminal` *terminal* {*options*}

Ab diesem Zeitpunkt wird die Ausgabe aller `(s)plot`- und `replot`-Kommandos auf *terminal* geleitet. Entspricht *terminal* allerdings einem File, so muß man vorher zusätzlich noch mit

`set output` '*filename*'

den Namen des zu erzeugenden Ausgabefiles angeben. Dieses File wird i.a. aber erst ordnungsgemäß geschlossen, wenn man Gnuplot verläßt oder mit `set output` ein neues Ausgabefile wählt.

Ausgabe in ein Bit/Pixelmap-File, z.B. im PNG-Format, erzeugt man mittels

`set terminal png` {*size*} {*color*}

wobei für *size* `small`, `medium` und `large` in Frage kommen und für *color* `monochrome`, `gray` und `color`.

Bei weitem das wichtigste Format, insbesondere zur Herstellung von Graphiken für Publikationen, ist jedoch Postscript. Hier hat man außerdem die Möglichkeit einer ansprechenden Beschriftung einschließlich griechischer Buchstaben und gewisser mathematischer Symbole. Der Befehl kennt eine größere Anzahl von Optionen

```
set terminal postscript {landscape | portrait | eps} \
                        {enhanced } \
                        {monochrome | color} \
                        {solid | dashed} \
                        {'fontname'} {fontsize}
```

Hier bedeutet `eps`, daß ein Encapsulated Postscript (EPS)-File erzeugt werden soll; `solid` und `dashed` beziehen sich auf die Auswahlt der Linientypen; *fontname* ist der gewünschte Druckerzeichensatz (z.B. `Times-Roman`) für die Beschriftung und *fontsize* die Zeichengröße (z.B. `16`) in Points.

Mit `enhanced` kommt man in einen erweiterten Postscript-Modus, in dem man in TeX-artiger Notation Zeichen hoch- (Exponenten) und tiefstellen (Indizes) kann. So ergibt z.B.

`x^y`	x^y
`x^{yz}`	x^{yz}
`x_i`	x_i
`x_{ij}`	x_{ij}

Griechische Buchstaben erhält man, indem man (für ein oder mehrere Zeichen) auf den "Symbol"-Zeichensatz wechselt, z.B.

```
{/Symbol=16 r}
```

Hier wurde auf den Symbol-Zeichensatz in der Größe 16 Points geschaltet; der Buchstabe "r" entspricht dort dem griechischen "ρ". Da dieser Zeichensatz auch mathematische Symbole enthält, kann man damit auch Wurzelzeichen u.ä. schreiben. Zeichen eines beliebigen Zeichensatzes kann man auch mit Hilfe des Oktalcodes angeben. So liefert etwa

```
{/Times-Roman \305}
```

das Å-Symbol. (Dazu muß man allerdings vorher mit

```
set encoding iso_8859_1
```

den erweiterten ISO-8859 Zeichensatz verlangen.)

Benützerdefinierte Funktionen

Numerische Ausdrücke und Funktionen

Da es in Gnuplot einen `print`-Befehl und die Möglichkeit gibt, Funktionen und Konstanten zu definieren und aus diesen numerische Ausdrücke zu bilden, kann man das Programm auch als Taschenrechner mißbrauchen. Normalerweise legt man damit aber zu zeichnende oder zu fittende Funktionen fest.

Die Werte von Konstanten (manche, z.B. `pi`, sind vordefiniert) setzt man nach dem Muster

```
a=1.0
```

wobei die rechte Seite natürlich auch ein numerischer Ausdruck gemäß der in den meisten Programmiersprachen üblichen Syntax sein kann. [Als Potenzierungsoperator wird `"**"` (Fortran!) verwendet, d.h. es wird x^y als `x**y` geschrieben.] Außerdem steht für numerische Ausdrücke ein recht umfangreicher Satz von "eingebauten" Funktionen zur Verfügung (s. `help expressions functions`).

Eine Funktion definiert man z.B. mittels

```
f(x)=a*exp(-0.5*((x-x0)/b)**2)+c
```

oder auch als

```
f(x,a,b,c,x0)=a*exp(-0.5*((x-x0)/b)**2)+c
```

wobei der Name der Funktion nicht `"f"` sein muß, sondern frei gewählt werden kann. (Man kann außerdem beliebig viele Funktionen definieren.) Der Name der unabhängigen Variablen *muß* allerdings `"x"` (im zweidimensionalen Fall `"x"` und `"y"`) sein. Auch benützerdefinierte Funktionen dürfen in numerischen Ausdrücken verwendet werden.

Parameterdarstellung

Neben der üblichen Form expliziter Funktionen vom Typ $f(x)$ bzw. $f(x,y)$ gibt es in Gnuplot auch noch die Möglichkeit, Funktionen in Parameterdarstellung zu definieren, also mit Hilfe zweier Funktionen $x(t)$ und $y(t)$ bzw. dreier Funktionen $x(u,v)$, $y(u,v)$ und $z(u,v)$. (Die Funktionen müssen aber nicht "x","y" und "z" heißen.) Dies erweitert die Menge der definierbaren Funktionen beträchtlich.

Der Übergang in den Parameterdarstellungsmodus bzw. die Rückkehr in den expliziten Modus erfolgt mit dem Befehl

{un}`set parametric`

Die Namen der unabhängigen Variablen (Parameter) sind dann im Fall eindimensionaler Kurven `t` und im Fall zweidimensionaler Flächen `u,v`. Die Intervalle, in denen die Parameter variieren sollen, müssen mit

`set trange [`*tmin*`:`*tmax*`]`

usw. festgelegt werden. Der `plot`-Befehl verlangt im Parametermodus dann statt *einer* expliziten Funktion $f(x)$ *zwei* Funktionen $x(t)$ und $y(t)$ (durch Beistrich getrennt). Analog treten bei `splot` an Stelle von $f(x,y)$ *drei* Funktionen $x(u,v)$, $y(u,v)$ und $z(u,v)$.

Das folgende Beispiel erzeugt eine Ellipse mit Halbachsen a und b:

```
set parametric
set a=...
set b=...
set trange[0:1]
x(t)=a*cos(2*pi*t)
y(t)=b*sin(2*pi*t)
plot x(t),y(t) ...
```

Dreidimensionale Graphiken

Flächen und Punktwolken

Der Befehl `splot` ("surface plot") zeichnet analytisch gegebene Funktionen standardmäßig als Drahtgittermodell. Die Auflösung dieses Gitters (Anzahl der Linien m bzw. n pro Koordinatenachse) kann mit

`set isosamples` m`,`n

gesteuert werden. Je nach vorheriger Angabe von

{un}`set hidden3d`

wird das Drahtgitter durchsichtig oder undurchsichtig gezeichnet.

Eine wesentlich ansprechendere Darstellung als schattierte Fläche erreicht man mit Hilfe des ab der Version 4.0 von Gnuplot für die wichtigsten Terminals (z.B. `x11`, `postscript`, `png`) verfügbaren `pm3d`-Stils

```
set pm3d
splot f(x,y) with pm3d
```

Hierbei wird die z-Koordinate der Fläche $z = f(x, y)$ in einen Index in eine Farbtabelle (Palette) transformiert, gemäß der die Fläche schattiert wird. Im einfachsten Fall wählt man mit

```
set palette gray | color
```

zwischen einer voreingestellten Grauton- oder echten Farbdarstellung. Mit Hilfe weiterer Optionen des Befehls `set palette` ist es jedoch möglich, sehr allgemeine Zuordnungen der z-Werte zu Farbtönen zu definieren. Diese Zuordnung wird in einer "Color Box" sichtbar gemacht, deren Aussehen und Positionierung mit dem Befehl `set colorbox` gesteuert werden kann (s. `help` zu den entsprechenden Befehlen).

Im Fall von Datenfiles zeichnet `splot` standardmäßig eine Punktwolke. Als Symbole stehen für die `with`-Option nur `lines`, `linespoints`, `dots` und `impulses` zur Verfügung. Um eine Drahtgitter- oder schattierte Fläche durch die Punkte zu legen, genügen allerdings nicht die Optionen `with lines(points)` oder `with pm3d` allein, sondern es müssen, wie im Abschnitt über die Struktur von 3D Datenfiles ausgeführt, die Daten in gleich großen, durch Leerzeilen getrennten Blöcken angeordnet sein. D.h. im Gegensatz zur Darstellung von Punktwolken, wo für die Anordung der x- und y-Werte keine Einschränkung besteht, müssen für die Darstellbarkeit als Fläche diese Werte ein geordnetes Gitter bilden.

Gibt man bei `splot` im `pm3d`-Modus bei der `using`-Option *vier* statt der üblichen drei Spalten an, so werden die Werte der dritten Spalte nur für die räumliche Darstellung der Fläche verwendet, die Farbtöne jedoch aus den Werten der vierten Spalte berechnet. Damit hat man eine Möglichkeit zur Visualisierung "vierdimensionaler" Daten.

Die Ansicht einer 3D Graphik kann mit

```
set view xrot{,zrot}
```

variiert werden. Dabei ist *xrot* (in Grad) eine Drehung um die in der Zeichenebene liegende, horizontale x-Achse und *zrot* eine (danach ausgeführte) Drehung um die senkrecht zur xy-Ebene liegende z-Achse. Für den Spezialfall , daß man senkrecht auf die xy-Ebene blicken will, kann man statt `set view 0,0` auch

```
set view map
```

angeben.

Standardmäßig beginnen Tick Marks und Achsenbeschriftung der z-Achse relativ hoch über der xy-Ebene. Dies kann mit

```
set ticslevel level
```

korrigiert werden, indem man *ticslevel* auf einen kleinen Wert setzt (*ticslevel* wird in Einheiten des mit `zrange` definierten Intervalls gemessen).

Contour Plots

Bei der Darstellung einer Fläche mit `splot` kann mit

```
set contour {base | surface | both}
```

erreicht werden, daß zusätzlich Höhenschichtlinien eingezeichnet werden, und zwar mit `base` (Standardannahme) projiziert auf die xy-Ebene, mit `surface` auf der darzustellenden Fläche und mit `both` auf beiden. (Mit `unset contour` schaltet man Höhenschichtlinien wieder aus.) Wo Höhenschichtlinien gezeichnet werden, steuert man mit

```
set cntrparam {levels {auto {n}} \
                      | {discrete z1{, z2,...}} \
                      | {incremental start,step{,end}}}
```

(wobei es noch einige andere Optionen gibt). Mit `auto` (Standardannahme) werden die Höhenschichtlinien bei n äquidistanten z-Werten zwischen $z_{\min}$ und $z_{\max}$ gezeichnet; mit `discrete` bei vorgegebenen Werten z_1, z_2, ...; und mit `incremental` in Schritten von *step* zwischen *start* und *end*. Mit

```
set clabel {'format'}
```

kann man das Format der Beschriftung bestimmen (oder mit `unset clabel` die Beschriftung überhaupt unterdrücken).

Häufig will man, wie bei einer Landkarte, nur die Höhenschichtlinien sehen, aber nicht eine Ansicht die Fläche selbst. Das erreicht man, indem man die Darstellung der Fläche ausschaltet und die Graphik so dreht, daß man senkrecht auf die xy-Ebene schaut

```
unset surface
set view map
splot ...
```

Fits

Zur Anpassung der Parameter einer benützerdefinierten Funktion an Daten bietet Gnuplot eine recht bequeme Möglichkeit, diese Parameter durch Minimierung einer gewichteten Summe von Abweichungsquadraten zu bestimmen (Methode der kleinsten Quadrate). Es wird das iterative Verfahren von Marquardt–Levenberg verwendet.

Im Fall einer Funktion einer Variablen x

$$y = f(x, \alpha_1, \alpha_2, \ldots)$$

die von einer Reihe von Parametern, α_1, α_2, ..., (i.a. in nichtlinearer Weise) abhängt, ist die minimierte Größe

$$\chi^2 = \sum_{i=1}^{N} \left[y_i - f(x_i, \alpha_1, \alpha_2, \ldots) \right]^2 / \sigma_i^2$$

Dabei sind $x_1, x_2, \ldots, x_N$ die "Meßpunkte" mit zugeordneten "Fehlern" (Standardabweichung) σ_i. Im Fall einer Funktion von zwei Variablen x und y gilt eine analoge Formel.

Um einen Fit durchzuführen, muß man zunächst die anzupassende Funktion definieren und Startwerte für die darin vorkommenden Parameter spezifizieren. Außerdem wird man i.a. einen kleineren als den Standardwert

```
FIT_LIMIT=1.0e-05
```

für das Abbruchkriterium (Differenz der Abweichungsquadratsummen zwischen aufeinanderfolgenden Iterationen) setzen. Der eigentliche Fit erfolgt für eine Funktion einer Variablen üblicherweise mit einem Befehl der Gestalt

`fit` { `[x=`*xmin*`:`*xmax*`]` } *function* `'`*filename*`'` `using` *expr1*`:`*expr2*`:`*expr3* `via` *par1*`,`*par2*`,...`

(analog für eine Funktion von zwei unabhängigen Variablen; wegen weiterer Optionen im `fit`-Befehl s. `help fit`). Hier kann man mit `[x=`*xmin*`:`*xmax*`]` noch die im Fit betrachteten x-Werte auf einen kleineren Bereich als die ganze x-Achse einschränken; *expr1*, *expr2* und *expr3* geben—ähnlich wie beim `plot`-Befehl—an, welche Spalten oder numerischen Ausdrücke als x-, y- und σ-Werte verwendet werden sollen; und *par1*, *par2*, ... sind die anzupassenden Parameter. Man beachte, daß zwischen Fitfunktion und Datenfile kein Beistrich gesetzt wird.

Der Verlauf des Fits, einschließlich der Endwerte der Fitparameter, wird laufend am Schirm ausgegeben. Eine "Mitschrift" aller Fitprotokolle wird in das File `"fit.log"` im momentanen Verzeichnis kopiert.