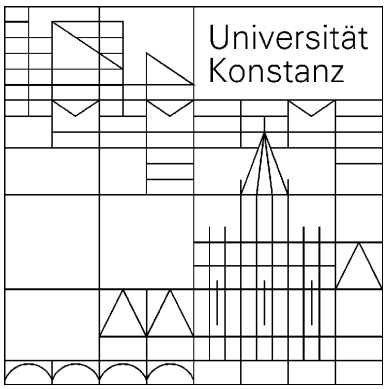


Fachschaft Physik

GnuPlot

November 2008

Dieses Skript dient als Handout zum GnuPlot-Kurs der Fachschaft Physik an der Universität Konstanz. Der Kurs ist eine Ergänzung des langjähringen LaTeX-Kurses der Fachschaft Physik und behandelt die Auswertung und Darstellung von Messdaten mit dem Open-Source-Programm GnuPlot mit Beispielen aus dem Praktikumsbetrieb. Bisher haben folgende Personen zu diesem Werk beigetragen: OLIVER GRÄSER, CLAUDIUS RIEK UND FRANZISKA MAIER. Die Autoren erheben keinen Anspruch auf Vollständigkeit und Fehlerfreiheit. Lob, Kritik und Anregungen bitte per Mail an:Fachschaft.Physik@uni-konstanz.de.

 Viel Spaß mit GnuPlot!

11.3

Inhaltsverzeichnis

1 Einführung

GnuPlot ist ein Programm zur Auswertung und Visualisierung von Messdaten. Im Gegensatz zu Tabellenkalkulationsprogrammen wie OpenOffice Calc oder Microsoft Excel bietet es ein Vielfaches sowohl an Darstellungsoptionen und vor allem Auswertungsfähigkeiten.
Gegenüber Oo Calc und MS Excel ist das Programm jedoch im Nachteil, vor allem in Punkto Verständlichkeit: So ist die offizielle Dokumentation noch unvollständig, und da GnuPlot *rein* kommandozeilenbasiert ist, ist das gerade für den geübten Windows-User ungewohnt. Allerdings lassen sich schon mit recht wenigen Grundkenntnissen hochwertige Graphen erstellen und so den Daten eine Menge Information entnehmen.
GnuPlot liegt momentan in Version 4 vor (http://www.gnuplot.info) und hat mittlerweile eine recht brauchbare Benutzerdokumentation, auch wenn sich diese nicht unbedingt für Einsteiger geignet. GnuPlot ist vollständig kommandozeilen-basiert, man muss also jeweils Befehle eintippen, um eine Reaktion zu bekommen – Kontextmenüs oder Assistenten wie bei Windows gibt es hier leider nicht. Allerdings sind die meisten Kommandos sehr simpel, und durch die Skriptfähigkeit lassen sich schon mit wenigen Zeilen große Datenmengen auswerten.

2 GnuPlot

2.1 Grundlagen

Gnuplot kann bei Linux nur von der Kommandozeile gestartet werden, d.h. man ruft den Befehl `gnuplot` entweder in der Konsole oder einem Terminal auf. Man arbeitet dann direkt in die Kommandozeile oder man fasst die Befehle in einer sog. Parameterdatei zusammen. Dies ist vor allem dann hilfreich, wenn man dieselbe Auswertung mehrere Male machen möchte.
Befehle müssen bei GnuPlot weder durch Sonderzeichen eingeleitet noch abgeschlossen werden, einfache Eingabe des Befehls, dessen Optionen und Parameter und anschließende Bestätigung durch Enter genügen. Das Programm springt dann automatisch in eine neue Zeile und wartet auf weitere Befehle.

```
gnuplot>  Befehl 1
gnuplot>  Befehl 2
```

2.2 Befehle

2.2.1 help - Hilfe

Mit der Eingabe `help` kann das Hilfe-Menü aufgerufen werden. Gibt man in die Kommandozeile nur `help` ein, so wird man unter Linux auf allgemeine Hinweise zu GnuPlot geführt, unter Windowslandet man auf einem Index, den man durchsuchen kann. Gibt man aber den Hilfe-Befehl in Kombination mit einem bestimmten Befehl ein

```
gnuplot>  help Befehl
```

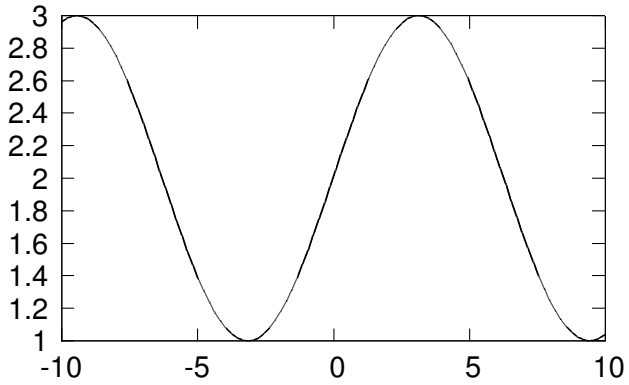
so wird man direkt auf die Seite des Hilfe-Menüs geführt, die diesen Befehl näher erklärt. Praktisch ist, dass das Menü mit zahlreichen Beispielen ausgestattet ist.

2.2.2 plot - Plotten einer Funktion

Der wohl zweitwichtigste Befehl bei GnuPlot nach `help` ist der Befehl `plot` . `plot` erzeugt einen neuen Frame (d.h. ein weißes Blatt mit Koordinatensystem), das auch mehrere Graphen enthalten kann. Um eine einfache Funktion zu Plotten, gibt man sie nach dem `plot` -Befehl in derselben Zeile in Abhängigkeit von *x* ein. Hier ein Beispiel zum Darstellen eines einfachen Sinus:

```
gnuplot>  f(x)=sin(a*x)+b
gnuplot>  a=0.5
gnuplot>  b=2
gnuplot>  plot f(x)
```

Was dann folgendermaßen aussieht:



In der ersten Zeile haben wir die Funktion *f(x)* in Abhängigkeit von *x* und den Parametern *a* und *b* definiert, sie bleibt über den gesamten Programmablauf so definiert, es sei denn wir geben *f(x)* oder *a* neu ein. In der nächsten Zeile wurde *a* ein fester Wert zugeordnet und die Funktion in der nächsten Zeile geplottet.
GnuPlot versteht sich auf alle mathematischen Ausdrücke, die der Gnu C Compiler verarbeitet. Dies sind:

die Grundrechenarten	`+,-,*,/`
die Potenz	`a**b` (nicht `a^b` wie bei LaTeX)
die 'einfachen' Funktionen	`exp, sin, cos, log ,ln`
die Quadratwurzel	`sqrt()`
Klammersetzung	`(`und`)`
Punkt vor Strich	·vor −
Konstanten	wie `pi`

Mit diesen Ausdrücken kann man Variablen auswerten, GnuPlot als Taschenrechner verwenden und vor allem Funktionen definieren und plotten.
Die unabhängige Variable ist immer *x*, alles andere sieht Gnuplot als Parameter an, der bei der Auswertung bekannt sein muss. So akzeptiert GnuPlot die Eingabe *f(x)=a*sin(x)+b*

ohne Murren, ein nun folgendes *plot f(x)* jedoch nicht – hier erhält man eine Warnung wegen eines undefinierten Parameters. GnuPlot bricht die Verarbeitung ab, sobald es *einen* undefinierten Parameter findet - auch wenn die Funktion f momentan zwei solche enthält. Es reicht also nicht aus, im folgenden *a=5* zu definieren, dann würde sich GnuPlot wegen *b* beschweren. Mit *a=5* und *b=2* ist der Ausdruck vollständig definiert und f kann geplottet werden.

2.2.3 set - Einrichten des Graphen

Lassen wir uns deneben geplotteten Sinus nochmals zeichnen, nur diesmal mit dem Parameterwert a=3, so stellen wir fest, dass er nicht sonderlich schön aussieht, sondern gezackt und unsymmetrisch ist. Das liegt daran, dass Graph standardgemäß in einen Wertebereich gezeichnet wird, dem GnuPlot für diese Funktion sinnvoll erscheint, der x-Bereich reicht normalerweise von -10 bis 10.
Wollen wir den Sinus so darstellen, wir wir es eigentlich erwartet haben, müssen wir erst den dargestellten Wertebereich unserer Koordinaten ändern. Dies geschieht über den Befehl `set`, mit dem sämtliche Umgebungsvariablen in GnuPlot gesetzt werden. In Kombination mit weiteren Optionen und Parametern kann man das Aussehen des kompletten Graphen ändern oder den Standardoutput ändern.

```
gnuplot>  set Variable [Variablenwert]
```

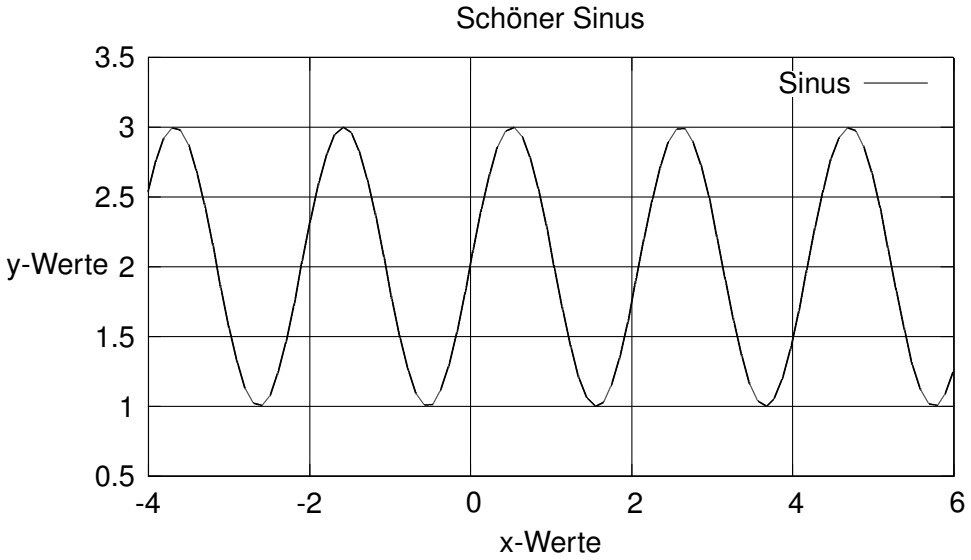
Die Verwendung ist sehr einfach daher listen wir hier nur die wichtigsten Variablen und ihre möglichen Werte:

`set xrange [a:b]`	Legt Wertebereich von x mit $a \leq x \leq b$ fest
`set yrange [a:b]`	Legt Wertebereich von y mit $a \leq y \leq b$ fest
`set xlabel 'xyz'`	Benennt x-Achse mit *xyz*
`set ylabel 'xyz'`	Benennt y-Achse mit *xyz*
`set title 'abc'`	Benennt gesamten Plot mit *abc*
`set log x`	Stellt die x-Achse logarithmisch dar
`set log y`	Stellt die y-Achse logarithmisch dar
`set grid`	Zeichnet ein Gitternetz in den Graphen

Die Tabelle zeigt allerdings nur einen kleinen Teil der Einstellungen, die man ändern kann, eine vollständige Liste kann man mit `help set` abrufen. Hat man einmal bestimmte Optionen mit dem `set`-Befehl gesetzt, so kann man mit `reset` wieder auf die Standardeinstellungen zurückgehen, der Befehl `show option` zeigt, welche Einstellung einer Option gerade aktuell ist, `show all` zeigt alle aktuellen Einstellungen an.
Hier ein Beispiel, wie die Befehlsfolge für einen fertig eingerichteter Graph aussehen kann:

```
gnuplot> set xrange [-4:6]
gnuplot> set yrange [0.5:3.5]
gnuplot> set xlabel 'x-Werte'
gnuplot> set ylabel 'y-Werte'
gnuplot> set title 'Schöner Sinus'
gnuplot> set grid
gnuplot> plot sin(3*x)+2 title 'Sinus'
```

Der dann so aussieht:

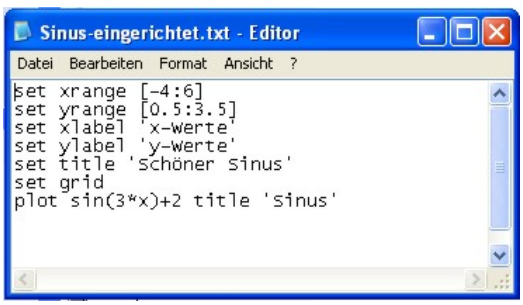


2.2.4 load - die Parameterdatei

Man sieht, dass für einen gutaussehenden Graphen meist mehr als nur eine einzige Befehlszeile notwendig ist. Man kann aber eine einmal direkt in GnuPlot eingegebene Befehlsfolge nicht abspeichern. Damit man aber, falls man bei einem Graphen nur einen Parameter ändern will oder ähnliches, nicht nochmals alles neu eintippen zu müssen, gibt man den kompletten Befehlsablauf in eine Text-Datei ein. Hier entspricht jedem Befehl eine einzelne Zeile (wie im Programm selbst) und die jeweiligen Änderungen werden dann an dieser Datei (`'Plot1.txt'`) vorgenommen. Diese kann dann über

```
gnuplot>  load 'Plot1.txt'
```

eingelesen werden und wird dann verarbeitet. Vorteil ist, dass man Parameter und Einstellungen sehr schnell und einfach verändern kann, ohne dass nochmals der gesamte Befehls-Ablauf getippt werden muss.
Als kleines Beispiel ist hier eine der Paramterdateien gezeigt, die zum Erstellen eines der vorigen Plots diente:



2.2.5 using - Einlesen von externen Daten

Mit GnuPlot kann man aber noch mehr als nur hübsche Graphen zeichnen. Führt man ein Experiment durch, so erhält man üblicherweise nur einzelne Datenpunkte, die zwar auch meist auf einer Kurve liegen (sollten), dies aber nicht immer tun.
Um diese Werte mit Gnuplot verarbeiten zu können, müssen sie tabellarisch in Spalten angeordnet in einer .dat oder .txt-Datei vorliegen. GnuPlot ist relativ gut im erkennen der einzelnen Spalten, damit jedoch keine Fehler passieren sollte sie aber durch einen einfachen Tab voneinander abgetrennt sein. Dies kann dann wie folgt aussehen:

Will man nun die einzelnen Messpunkte in einen Graphen eintragen lassen so lautet der Befehl

```
gnuplot>  plot 'Daten.txt'
```

wobei der komplette Pfad zu der Datei `Daten.txt` angegeben werden muss, der immer in Anführungszeichen (' ' oder „“) angegeben wird. Es wird dann automatisch die zweite über der ersten Spalte aufgetragen.
Hat man mehrere Spalten mit Daten zur Verfügung und will nur einige davon nutzen, so kann man diese mit dem Befehl `using` heraussuchen. Dieser Befehl beschreibt die Herkunft der Daten, indem er angibt, welche Spalte über welcher Aufgetragen werden soll.
So zeichnet zum Beispiel

```
gnuplot>  plot 'Daten.txt' using 3:2
```

die Werte der zweiten Spalte über die der dritten auf.
Will man die Werte noch verändern, z.B. die y-Werte mit -1 multiplizieren, so kann man folgendermaßen in die Auswertung eingreifen:

```
gnuplot>  plot 'Daten.txt' using 1:($3*(-1))
```

Hier gibt man mit dem $-Zeichen an, dass die Werte der Einträge in Spalte 3 genommen und jeweils mit -1 multipliziert werden sollen.
Man kann auch mehrere Datensätze in einen Graphen einzeichnen lassen, indem man die einzelnen Datenblöcke, die gezeichnet werden sollen, nach dem `plot` -Befehl durch Kommata abtrennt und aneinanderreiht. Die kann dann so aussehen:

```
gnuplot>  plot 'Daten.txt' using 1:($3*(-1)),
          'Daten.txt' using 1:2,
          'Daten2.txt' using 2:4
```

Nicht alle Arten von Graphen verlangen nur einem Datenpaar. Für Fehlerbalken in y-Richtung benötigt GnuPlot drei Zahlen: X, Y und ΔY. Ein solches Datentupel übergibt man z.B. als `using 1:2:5`, wenn die Daten für den Fehlerbalken in Spalte 5 gespeichert sind. Die Daten für den Fehler müssen dazu natürlich in der gleichen Datei stehen.
Den Namen einer Reihe von Messwerten oder einer Funktion (nicht des gesamten Plots!) kann man mit `title` festlegen.

```
gnuplot>  plot 'Daten.txt' u 1:2 title 'q=0.8'
```

erzeugt z. B. die Legende 'q=0.8' für die Messwerte aus 'Daten.txt' die auch in den Plot eingefügt wird. Ansonsten verwendet GnuPlot standardmäßig die Datenherkunft (im vorigen Beispiel also `'Daten.txt' u 1:2` als Legende. Möchte man ganz auf die Legende verzichten, kann man dies mit doppelten Apostrophen (`title ''`) erreichen.

Als besonderes Schmankerl versteht `using` auch logische Ausdrücke. Will man z.B. nur die Datenpunkte plotten, die nicht größer als eine bestimmte obere Schranke sein sollen, so kann man dies mit

```
gnuplot>  plot 'Daten.txt' using 1:($2<1?$2:1/0)
```

erreichen. Zunächst wird mit dem Fragezeichen die If-Frage $\$2 < 1$ gestellt, also ob der Wert der Zahl in Spalte 2 kleiner als 1 ist. Ist dies der Fall, plottet GnuPlot den Wert der 2.Spalte ($2). Andernfalls verwendet GnuPlot den Wert $1/0$, den es als Nicht-Zahl (nan, not-a-number) erkennt und auslässt. Dies ist gerade auch bei Fits sehr hilfreich.
Wer sich damit genauer beschäftigen will, sei hier auch wieder auf das Hilfe-Menü verwiesen.

2.2.6 fit - Fitten

Hat man nun die einzelnen Datenpunkte eingelesen, so will man meist einen mathematischen Zusammenhang zwischen den einzelnen Messwerte herstellen. Dabei muss man allerdings von vornherein wissen, welcher Art von Gleichung der Zusammenhang entspricht (z.B. bei atomarem Zerfall die Exponentialfunktion) oder entsprechen sollte.
Beim genauerem Bestimmen der Parameter der Funktion, die unsere Datenreihe bestmöglich beschreibt, kommt ein weiterer Vorteil von GnuPlot ins Spiel: GnuPlot kann *fitten*. Das heißt, dass GnuPlot in der Lage ist, eine Kurve durch die eingelesenene Messpunkte zu legen, die nach dem *Least Squares Fit*-Verfahren bestimmt wird.
Zuerst muss allerdings die äußere Gestalt der Funktion, für die das Verfahren durchgeführt werden soll, allgemein mit Parametern festgelegt werden. GnuPlot sucht dann diejenigen Parameter der Funktion, für die die quadratische Abweichung der Funktion von allen wirklichen Messwerten am kleinsten ist (was eben *Least Squares Fit* ist)
Dieses Verfahren ist vor allem nötig, wenn man *nichtlineare* Kurven an Meßdaten anpassen möchte. Die Grenzen von Tabellenkalkulationen sind hierbei schon schnell erreicht. Auch die klassische Methode – Lineal auf die x-Achse und messen – scheidet hier schnell aus.
Der entsprechende Befehl lautet `fit`, allerdings erfordert er eben die Vorgabe einer

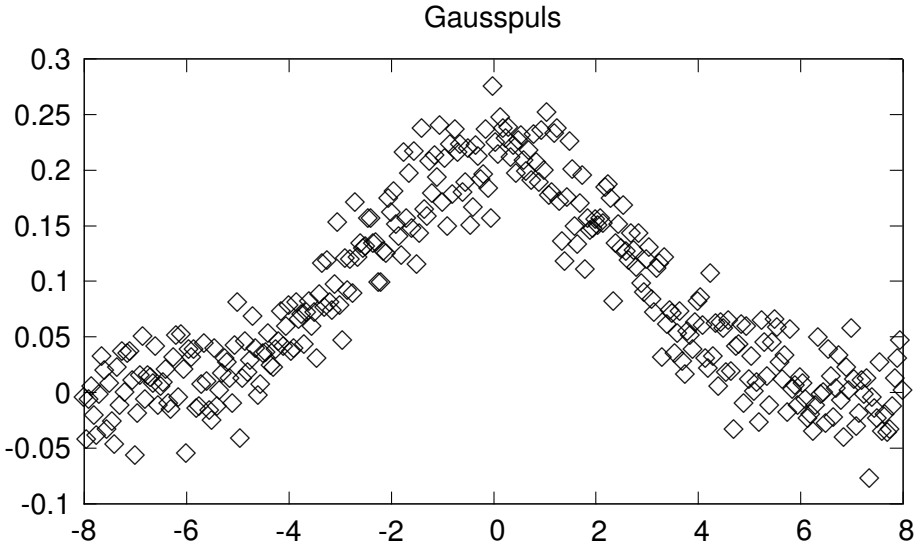
Funktion und der Messwerte, an die diese gefittet werden soll. Optional kann auch noch ein Wertebereich angegeben werden, in dem das Verfahren durchgeführt werden soll.
Hier ein Beispiel für einn Fit-Verfahren mit GnuPlot

```
gnuplot>  f(x)=a*x**3+b*x**2+c*x+d
gnuplot>  fit f(x) 'Daten.txt' u 1:2 via a,b,c,d
```

`via` gibt hierbei an, welche Parameter das Fit-Verfahren durchlaufen sollen, denn es kann ja sein, dass einige der Variablen schon mit festen Werten belegt sind, die sich auch während des fittens nicht ändern sollen.
Nachdem GnuPlot die Parameter berechnet hat, werden sie zusammen mit ihren Fehlern ausgegeben. Sie sind dann für *f(x)* festgelegt und man kann nun die Fit-Funktion in den Graphen mit den Messwerten einzeichnen lassen.
Zudem werden Korrelationen zwischen den Fitparametern angezeigt. Diese sind auch ein Maß dafür, wie verlässlich die Fitwerte sind. Je stärker die Werte korreliert sind, desto stärker kann eine leichte Veränderung in den Quelldaten auf die Fitparameter durchschlagen.

2.2.7 Numerisches Fitten

GnuPlot verwendet für Fits die Gnu Scientific Library GSL. Damit erbt GnuPlot auch alle Stärken und Schwächen, sprich, was der Algorithmus nicht hinbekommt, schafft auch Gnuplot nicht. Deshalb hier zusätzlich eine kleine Einführung ins numerische Fitten:
Generell ist der beste Fit an eine Funktion derjenige, für den die Summe der quadratischen Abweichungen minimal wird (Methode der *Least Mean Squares*, häufig als LMS abgekürzt).
D.h., das Programm bildet für folgendes Beispiel von Messpunkten



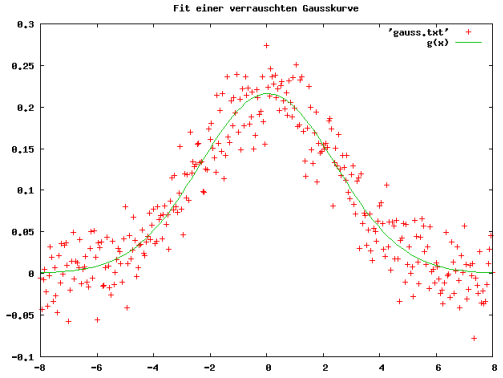
an die nun eben folgendermaßen die Gauss-Funktion $f(x) = \frac{A}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$ gefittet werden soll

```
gnuplot>  f(x)=A/sqrt(2*pi*sigma**2)
          *exp(-.5*((x-mu)/sigma)**2)
gnuplot>  fit f(x) 'noisygauss.dat'
          u 1:2 via A, sigma, mu
```

die Funktion

$$\Delta(A,\sigma,\mu) = \sum_{i=1}^{N} \left(y_i - \frac{A}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{1}{2} \frac{(x_i-\mu)^2}{\sigma^2} \right) \right)^2$$

und versucht diese zu minimieren. Hierzu geht man immer dem Gradienten nach, bis dieser in jeder Richtung nach oben zeigt – und dies ist dann das Minimum.
Der fertige Fit sieht dann folgendermaßen aus: Dabei treten verschiedene Probleme



auf. Zum einen ist das so gefundene Minimum nicht notwendig das einzige Minimum. Dies ist speziell dann der Fall, wenn die Fitparameter korreliert sind. Nehmen wir hierzu als Beispiel eine Potenzreihe mit Potenzänderungen <1, also z.B. $f(x) = ax^{0.6} + bx^{1.2}$. Diese Entwicklung will man über ein bestimmtes Intervall an eine Messreihe fitten. Hier ist die Korrelation offensichtlich. Wenn man b vergrößert, kann man das in einem gewissen Maß ausgleichen, indem man a verringert. Dies wird dazu führen, daß die Minima der Funktion Δ deutlich flacher ausfallen als bisher. Hat man dazu noch stark nichtlineare/nichtmonotone Funktionen, so kann es durchaus vorkommen, daß die Funktion mehrere Minima hat. Welches der Fit dann zurückliefert, hängt von den Startwerten ab – dies kann mitunter etwas völlig anderes sein als das gesuchte Minimum, der Plot des Fits hat manchmal nichts mehr mit den gefitteten Daten gemeinsam.
Ein anderes Problem besteht darin, daß der Gradient zunächst überhaupt am Startwert existieren muß, damit der Algorithmus funktioniert. Ist dies nicht der Fall, bricht der Fit-Algorithmus ab.
Im Allgemeinen kann man mit GnuPlot Funktionen mit zwei Parametern meist problemlos fitten, drei Parameter erfordern gelegentlich Hilfe bei den Startwerten. Mit fünf oder mehr Parametern hat man eher wenig Chance, speziell Potenzreihen bis zur Ordnung fünf sind eher aussichtslos.

2.2.8 with - Styles

Eine weitere Standardeinstellung bei GnuPlot ist, dass der erste Graph rot gezeichnet wird. Will man die Farbe eines Graphen oder die Form der Punkte, die gezeichnet werden, ändern, so muss man den *Style* des Graphen ändern.
Die Art des Graphen wird mit `with` bestimmt. So gibt es zum Beispiel `lines` (Datenpunkte sind mit gerade Strichen verbunden), `linepoints` (wie Lines, nur mit Punkten auf den Linien), `errorbars` (Fehlerbalken in y±-Richtung, erfordert ein 3er-Datentupel) und einige mehr (Eine vollständigere Liste findet man wieder unter `help`).
Als kleines Beispiel

```
gnuplot> plot 'Gauss.txt' with lines
```

Hier kommt dann auch die größte Schwäche von GnuPlot ins Spiel: Es ist nicht so einfach möglich, den Stil `linepoints` beispielsweise mit einer roten Linie und hohlen Kreisen als Punkten zu realisieren. GnuPlot kennt für jede Art von Graph verschiedene Stile, die sich meist nur in der Art des Punktes und der Farbe unterscheiden. Diese verschiedenen Stile sind durchnummeriert - d.h., selbst wenn der gewünschte Stil existiert, muss man auch noch seine Nummer kennen, um ihn verwenden zu können. Verschlimmert wird dies noch dadurch, daß sich die Stile je nach Ausgabegerät unterscheiden. Wechselt man vom Terminal X11 (also der graphischen Ausgabe auf dem Bildschirm unter Linux) zum Terminal PostScript (um die Ausgabe drucken zu können), so hat man mitunter statt eines satten Braun ein quasi unsichtbares Gelb auf dem Papier. Dies bedeutet häufig einen Abstrich an Qualität, weswegen man GunPlot auch eher nicht für für Publikationen verwendet.
Die standardmäßige Einstellung von `with` ist `points`, wobei der Punktstil von Graph zu Graph hochgezählt wird.
(Duch googlen und Arbeiten mit der `help`-Funktion kann man sich diesbezüglich noch weiter informieren, dies ist aber nicht Aufgabe dieses Skripts)

2.2.9 splot - 3D-Plots

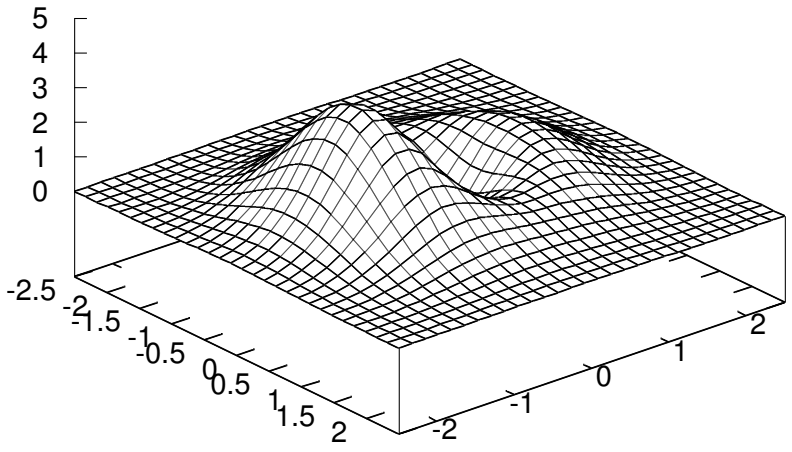
Mit GnuPlot ist nicht nur das Plotten von 2-dimensionalen Kurven, sondern auch von „3-dimensionalen“-Flächen möglich (also Projektionen auf 2D-Flächen).
Der Schritt in die dritte Dimension ist nicht schwer: Zusätzlich muss ein Wertebereich der z-Achse festgelegt werden (`set zrange`), sowie ein Blickwinkel und -höhe (Azimuth und Elevation mit `set view`). Statt `plot` heißt der Zeichenbefehl nun `splot`. Die Datenpunkte werden einer Datei entnommen bei der es sich nun um eine Textdatei mit drei Spalten für x- y- und z-Werte handelt.
Als kleine Besonderheit muss man den Blickwinkel mit dem auf die Projektion geschaut werden soll, definierten. Das geschieht mit `set view`, dem die Winkel α_x (Rotation um die x-Achse eines fiktiven Koordinatensystems, die horizontal im Bildschirm liegt), β_z (Rotation um die z-Achse, die senkrecht auf dem Bildschirm steht), und die Skalierung des gesamten Plots (S_s) und der z-Achse (S_{sz}) zugeordnet sind.

```
gnuplot>  set xrange [a:b]
gnuplot>  set yrange [c:d]
gnuplot>  set zrange [e:f]
gnuplot>  set view α_x, β_z, S_s, S_sz
gnuplot>  splot 'Daten3D.txt'
```

Anstatt von Messpunkten können auch hier Funktionen, die von einem oder zwei Parametern abhängen, geplottet werden. Sie werden dann als Gitternetz-Flächen im dreidimensionalen Koordinatensystem dargestellt. Nützliche Befehle sind hier noch `set isosamples 40`, der festlegt, dass hier auf den x- und y-Wertebereich jeweils 40 Gitternetzlinien verteilt sein müssen (je mehr Gitternetzlinien berechnet werden müsse, desto länger dauert die Darstellung als Plot), `set hidden3d` der die aus dem jeweiligen Blickwinkel nicht „sichtbaren“ Gitternetzlinien ausblendet und `set xtics` der die Schrittweite der Achsenmarkierungen bestimmt.
Hier eine Befehlsfolge als Beispiel:

```
gnuplot>  set hidden3d
gnuplot>  set isosamples 30
gnuplot>  set xrange [-2.5:2.5]
gnuplot>  set yrange [-2.5:2.5]
gnuplot>  set zrange [0.0:5.0]
gnuplot>  set xtics 0.5
gnuplot>  set ytics 1.0
gnuplot>  set ztics 1.0
gnuplot>  set view 40,50,1.0,1.5
gnuplot>  f(x,y)=(x**2+2.5*y**2-y)*exp(1-(x**2+y**2))
gnuplot>  splot f(x,y)
```

Und so sieht dann der Plot aus:



2.2.10 set output - Ausgabe in Dateien

Wenn man nun Messwerte eingelesen, einen passenden Graphen gefittet, das ganze auch noch geplottet hat, so stellt man fest, dass man diesen nicht ohne weiteres drucken oder abspeichern kann. Auch dies muss als Befehl in GnuPlot eingegeben werden. Dazu muss man zuerst das Ausgabeformat, indem man den Plot haben möchte, festlegen. Dies geschieht mit `set terminal`, in dessen Anschluss das gewünschte Format (hier PNG) angegeben wird. Eine komplette Liste der verfügbaren Dateiformate kann wie immer über `help terminal` abgerufen werden.
Der Name der Datei, in der der Graph abgespeichert werden soll, wird mit `set output` festgelegt, anschließend muss der Graph nochmals geplottet werde, dies kann aber über `replot` abgekürzt werden.

```
gnuplot>  set terminal png
gnuplot>  set output 'Dateiname.png'
gnuplot>  replot
```

Als besonderes Extra für die LaTeX-User gibt es die Möglichkeit, als Ausgabeterminal `latex` zu wählen, d.h. der Graph wird dann in eine .tex-Datei geschrieben, die sehr einfach über den LaTeX-Befehl

```
\input{Plot.tex}
```

in ein bestehendes TeX-File eingebunden werden kann. In den Beschriftungen von Titel, X-Achse etc. kann man dann auch TeX-Befehle verwenden, um z.B. griechische Buchstaben schreiben zu können.
Als Beispiel:

```
gnuplot> set terminal latex
gnuplot> set output 'sinus-eingerichtet.txt'
gnuplot> set xrange [-4:6]
gnuplot> set yrange [0.5:3.5]
gnuplot> set xlabel 'x-Werte'
gnuplot> set ylabel 'y-Werte'
gnuplot> set title 'Schöner Sinus'
gnuplot> set grid
gnuplot> plot sin(3*x)+2 title 'Sinus'
gnuplot> set size 1.0,0.7
```

Wichtig ist hierbei aber, dass man die relative Größe des Plots schon vorher über `set size` festlegen muss, da man die Grafik im äußeren Tex-File nicht mehr skalieren kann.