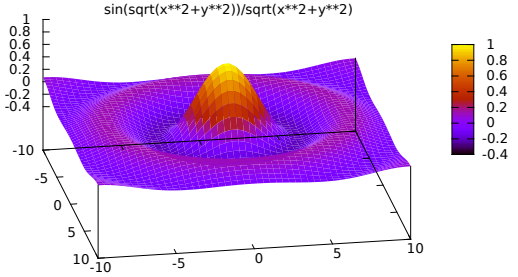


Graphische Analyse mit gnuplot[1]

[1] Tutorial von Thorsten Tepper García

Literatur zu gnuplot

- `http://gnuplot.info`
- "Gnuplot in action"; Philipp K. Janert

Analyse von Daten und Funktionen

mit Hilfe ihrer graphischen Darstellung

Was ist gnuplot?

`gnuplot`[2] ist ein **kommandozeilen-orientiertes**, *interaktives* wissenschaftliches Plotprogramm zur Darstellung von Funktionen und Daten:

- kostenlos
- ständig gewartet (aktuell: Version 6.0 vom November 2023, im Pool: 5.4.3 vom Januar 2022)
- verfügbar für Unix/Linux, MacOS und sogar Windows (wgnuplot)
- Code (C-basiert) frei verfügbar; kann für jede Rechnerarchitektur kompiliert werden
- verbraucht wenig Resourcen (d.h. Speicher)
- SEHR einfach zu bedienen

[2] Aussprache wie *newplot*

Was kann gnuplot? I

`gnuplot` kann folgendes graphisch darstellen (engl. *plotten*):

- (x, y)-Datenpaare
  z.B. Messdaten; erfolgt durch Einlesen aus Datei (i.d.R. ASCII)
- Funktionen der Form $f(x)$
  eingebaute, z.B. trigonometrische; benutzerdefinierte, z.B. Polynome
- 3D-Objekte
  (x, y, z)-Werte; Hyperflächen im 3D-Raum, also Funktionen $f(x, y)$
- 4D-Darstellung durch Farbkodierung (Farbpaletten)
- geometrische Objekte (Linien, Polygone)

Was kann gnuplot? II

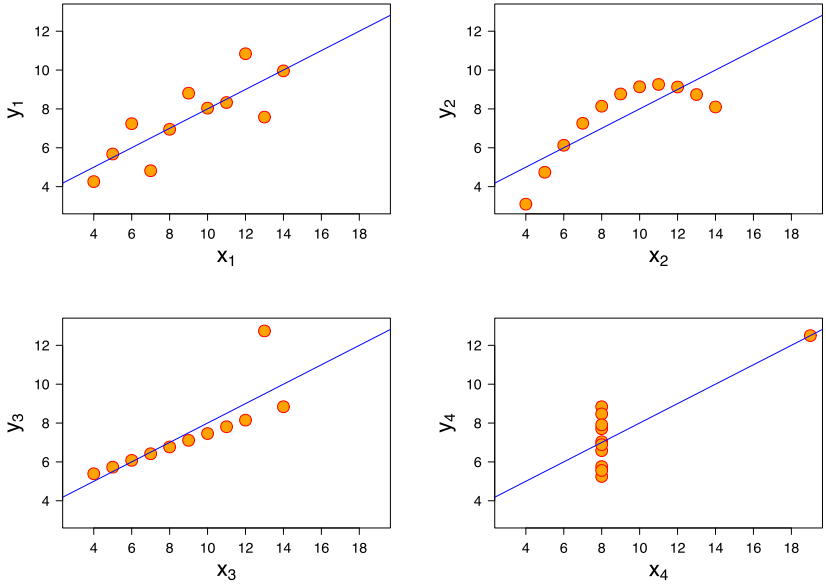
- Interpolation bzw. Anpassen (sog. "fitten") beliebiger Funktionen an Wertepaare
  3D-Interpolation (sog. "gridding") ebenfalls möglich
- Darstellung in kartesischen (x, y) oder Polar-Koordinaten (r, t)
- Parametrische Plots, d.h. Kurven $[x(t), y(t)]$ in der xy-Ebene,
  beliebige Kurven $[x(t), y(t), z(t)]$ oder
  Flächen $[x(u, v), y(u, v), z(u, v)]$ im 3D-Raum

Was kann gnuplot nicht?

Nicht unterstützt werden u.a.:

- Tortendiagramme (“pie charts”)
- freies Zeichnen (“free-hand” drawing) $\rightarrow$ z.B. mit `xfig`
- Volume rendering, Ray-tracing

Warum graphische Darstellung wichtig ist: Anscombes Quartett I



Warum graphische Darstellung wichtig ist: Anscombes Quartett II

Für alle gezeigten Datensätze sind die folgenden Werte gleich:

Statistische Größe	Wert
Anzahl der Punkte	11
Mittelwert $\langle x \rangle$	9 (exakt!)
Varianz $\sigma^2(x)$	11 (exakt!)
Mittelwert $\langle y \rangle$	7.50
Varianz $\sigma^2(y)$	4.122
Korrelation $\langle xy \rangle$	0.816
Linearer Fit	$y = 3.00 + 0.500x$

Beispiel-Session: Die Sinus-Funktion I

`gnuplot` im Terminal (also in der Shell) aufrufen:

```
gnuplot
```

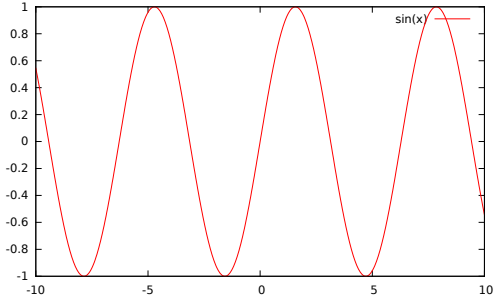
Interaktiver Modus startet, der Prompt zeigt an:

```
gnuplot>
```

und erwartet eine Eingabe, gefolgt von ENTER, z.B.

```
gnuplot> plot sin(x)
```

Ausgabe:

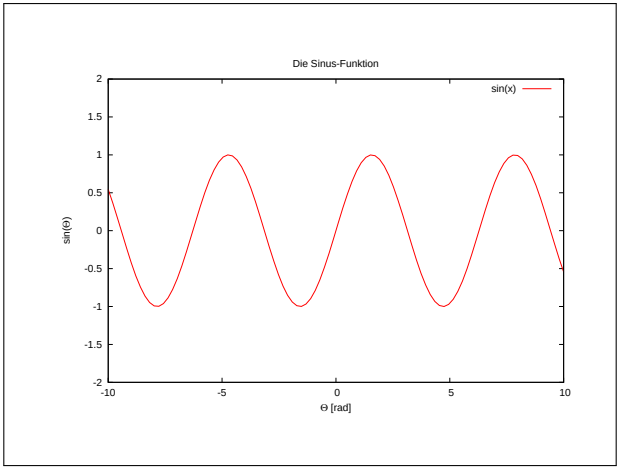


`gnuplot` verlasen mittels

```
quit
```

Aufgabe 2.1 Plotten von Funktionen

Starten Sie `gnuplot` und plotten sie die Sinus-Funktion.
Welche weiteren trigonometrischen Funktionen können Sie mittels `gnuplot` darstellen?



Plot-Elemente

- Terminal: graphische Ausgabe
  entweder Fenster, in dem der Plot erscheint (z.B. wxt, x11); oder Dateiformat, in dem der Plot gespeichert wird (z.B. `.ps`, `.pdf`, `.png`)
  Tipp:

```
gnuplot> help term
```

  oder auch

```
gnuplot> set term
```

- Canvas: Plotfläche ("Leinwand") innerhalb des Terminals
- Graph: bildliche Darstellung der Funktion
- Achsen: x-,y- (ggf. z-) Achse
- Labels: Beschriftung des Plots, der Achsen, der Daten
- Titel: Überschrift für einen Canvas

Voreinstellungen

- Es gelten bereits einige vordefinierte, oft sinnvolle Standard-Einstellungen, z.B.: Terminaltyp; *xy*-Achsenbereiche; Art und Größe der Schrift; Einteilung der Achsen-Ticks; Achsen-Verhältnis (“aspect ratio”)
- Stil/Farbe der Datensätze abhängig von deren Reihenfolge
- Voreinstellungen anzeigen durch

```
gnuplot> show all
```

- Voreinstellungen nicht immer ideal
- Jede Einstellung kann durch den Befehl `set` jederzeit überschrieben werden (s. u.).
- Die Standard-Einstellungen können jederzeit durch den Befehl `reset` wieder hergestellt werden.

Der Plot-Befehl I

`plot` *<Ranges> Was <Wie> <Beschriftung>*

- Ranges (optional): dargestellter *xy*-Bereich, z.B.

```
gnuplot> plot [-20:20][-1.2:1.2] sin(x)
```

- Was: Funktion (eingebaut, selbstdefiniert) oder Daten (aus Datei)
  → Optionen beim Einlesen aus Datei: `index`, `using`, `every` ...
- Wie (optional): Stil (Linien, Punkte, Balken, usw.), Farbe
  → Direktive `with` mit Optionen `style`, `width`, `size`, `color`, ...
- Beschriftung (optional): Beschreibung des dargestellten Datensatzes
  → Direktive `title`
- weitere, durch Komma getrennte Funktionen/Datensätze können mit demselben `plot` Befehl geplottet werden z.B.

```
gnuplot> plot 'beispiel.dat', sin(x), cos(x), f(x)
```

plot-Beispiele

```
gnuplot> plot cos(x) with points pointsize 0.1 linecolor rgb "pink"
gnuplot> plot [-1:*][] "beispiel.dat", sin(3*x)+2.3*x
```

Bemerkungen:

- Ranges: Angaben mit [min:max], erstes Paar für x-Range, 2. für y-Range; leer oder mit $*$ für Autoskalierung
- Styles: z.B. `lines` (Standard bei Funktionen), `points` (Standard bei Daten), `boxes` (ausgefüllbar), etc. → `help with`
- Dateinamen sind Textstrings und müssen – wie alle Textstrings in `gnuplot` – von einfachen oder doppelten Anführungsstrichen eingeschlossen werden: `'beispiel.dat'` oder `"beispiel.dat"`
  Die Datei wird mit einem Unix-Pfad angegeben (hier: aktuelles Verzeichnis).

Aufgabe 2.2 Daten plotten

Plotten Sie analog zum o.g. Beispiel die Daten aus `beispiel.dat` mit Punkten der Größe 2 in blau. Plotten Sie die Daten zusammen mit der Funktion `2*x`. Dabei soll der dargestellte y-Bereich von 0 bis 110 laufen.

Hinweis: Die benötigte Datei `beispiel.dat` befindet sich im Homeverzeichnis von `htodt`. Kopieren Sie sie also zunächst mittels `cp` in Ihr aktuelles Verzeichnis.

Weitere wichtige Befehle

- `set`: Zuweisung Einstellung $\rightarrow$ Wert

```
set xlabel "Temperatur [K]"
```

- `unset`: löscht den einer Einstellung zugewiesenen Wert

```
unset xlabel
```

- `help`: bietet *online* (im Unix-Sinne) Hilfe zu einem Befehl

```
help xlabel
```

- `show`: zeigt den zugewiesenen Wert einer Einstellung

```
show xlabel
```

- `test`: stellt alle verfügbaren Styles, Liniendicken, Textmodi, usw. im definierten Terminal dar

Beispiel-Session: Die Sinus-Funktion

Einstellungen:

```
gnuplot> set title "Die Sinus-Funktion"
gnuplot> set xlabel "Winkel [rad]"
gnuplot> set ylabel "Amplitude [cm]"
```

Nochmals plotten:

```
gnuplot> plot sin(x)
```

Hinweis:

Mehrere Befehle können auch in einer Zeile, durch ein Semikolon getrennt, eingegeben werden:

```
gnuplot> set xrange [-20:20] ; set yrange [-1.2:1.2]
```

Desweiteren kann eine Befehl in der nächsten Zeile mittels Backslash \ fortgesetzt werden:

```
gnuplot> plot sin(x) with lines linewidth 3  \
> linecolor rgb "yellow"
```

Den Zustand der Session abspeichern

Die im interaktiven gnuplot-Modus getätigten Eingaben und Ergebnisse kann man in einer Datei abspeichern:

```
gnuplot> save "meinPlot.gplt"
```

Und später, z.B. nach dem Start einer neuen gnuplot-Sitzung wieder laden:

```
gnuplot> load "meinPlot.gplt"
```

Aufgabe 2.3 Beschriftungen und Speichern

Plotten Sie die Sinusfunktion und beschriften sie die Achsen. Speichern Sie die Session, beenden sie gnuplot und starten Sie gnuplot erneut. Laden Sie die zuvor gespeicherte Session.

Funktionen I

Es können (beliebige, aber wohldefinierte) Funktionen definiert und dargestellt werden, die:

- von einer oder mehreren Variablen
- von einer beliebigen Anzahl von Parametern

abhängen. Zum Beispiel $f(x) = ax^2 + b$ (man beachte die Syntax!):

```
gnuplot> f(x) = a*(x**2) + b
gnuplot> a=2 ; b=2
gnuplot> plot f(x) with lines
```

Die Direktive `with` ermöglicht die Wahl verschiedener Plot-Stile (Linien, Punkte, Balken, ...), sowie die Einstellung der Linienstärke, Linienfarbe, usw. (Tipp: `help with`; `test`)

Beispiel

```
gnuplot> plot f(x) with lines linetype 1 linewidth 2 linecolor rgb "blue"
```

Wichtig:

- Der Name einer Funktion ist beliebig (und kann beliebig lang sein); sollte aber sinnvoll gewählt werden, z.B. `druck(x)`
- Die Reihenfolge, in welcher Funktionen und Parameter definiert werden, spielt keine Rolle; allerdings müssen vor dem Plotten ALLE für `gnuplot` unbekannten Variablen definiert werden, entweder als Funktion oder als Parameter.
- Der Wert einer definierten Variablen (z.B. "a") kann ausgegeben werden durch

  ```
  gnuplot> print a
  ```

- Von `gnuplot` genutzte unabhängige Variablen sind üblicherweise x, y, z bzw. t, u, v für parametrische Plots (Änderung dieser Variablennamen mittels z.B. `set dummy p`).

- Darstellung in Polar-Koordinaten (r, t)

```
gnuplot> set polar
gnuplot> set xrange [-2:2] ; set yrange [-2:2]
gnuplot> plot [0:12*pi] 0.04*t, cos(2*t)
gnuplot> unset polar
```

- Parametrische Plots $[x(t), y(t)]$, z.B. Kreise in 2D $[\sin(t), \cos(t)]$

```
gnuplot> set parametric
gnuplot> plot sin(t), cos(t)
gnuplot> unset parametric
```

Aufgabe 2.4 Polarkoordinaten und Kurven

Plotten Sie die o.g. Beispiele. Wie sehen die Graphen aus?

Eingabehilfen in gnuplot

`gnuplot` erleichtert die Eingabe von Einstellungen durch folgende Features

- `history`: wie in der Shell, d.h. mittels ↑ kann die letzte Zeile(!) wiederholt werden.
  Desweiteren zeigt der Befehl `history` die letzten Befehle an;
  `history !plot` wiederholt letzten Befehl, der mit `plot` beginnt
- `replot`: der letze Befehl, der mit `plot` beginnt, wird wiederholt
- Abkürzungen: bestimmte Einstellungen können abgekürzt werden, solange die Abkürzung eindeutig ist

Beispiel

```
... with lines linecolor rgb "blue"
```

kann abgekürzt werden zu

```
... w l lc rgb "blue"
```

Daten einlesen und plotten I

`gnuplot` kann jede (fast beliebige) Menge an Daten aus einer Datei einlesen und diese graphisch darstellen; i.d.R. wird dabei vorausgesetzt, dass

- die Datei im ASCII-Format vorliegt (obwohl auch binäre Dateien, z.B. `jpeg` eingelesen werden können; Tipp: `help binary`)
- die Daten in Spalten angeordnet sind, die durch ein oder mehrere Leerzeichen getrennt sind
- jede Zeile einem Datenpaar (x, y) entspricht

Beispiel

```
gnuplot> plot "beispiel.dat" with lines \
         linetype 1 linewidth 2 linecolor rgb "blue"
```

Daten einlesen und plotten II

Besonderheiten:

- Zeilen die mit # anfangen (also Kommentare) sowie ungültige Werte (NANs) werden stillschweigend ignoriert
- Durch einfache Leerzeilen getrennte Datensätze: *Unstetigkeiten*; betroffene Punkte werden nicht verbunden (`lines`)
- doppelte Leerzeilen: Datensatztrenner (s.u.)
- Falls die Datei:
  - eine einzige Spalte enthält, wird diese gegen die Zeilennummer geplottet
  - mehr als zwei Spalten enthält,
    oder aber die Reihenfolge der Daten in der Zeile nicht der (x, y)- Anordung entspricht,
    so kann eine beliebige Spalte gegen jede andere mittels der Direktive `using` geplottet werden

Daten einlesen und plotten III

Beispiel

Die Datei "beispiel.dat" enthält vier Spalten;
Um die 1. gegen die 4. Spalte zu plotten:

```
gnuplot> plot "beispiel.dat" using ($4):($1)
```

Desweiteren können (beliebige) Transformationen auf die Daten angewandt werden:

```
gnuplot> plot "beispiel.dat" using (3.1*($4)):(log10($2))
```

Aufgabe 2.5 Daten einlesen und bearbeiten

Plotten Sie o.g. Beispiele.

Daten einlesen und plotten IV

Wenn eine Datei mehrere Datensätze, getrennt durch *zwei Leerzeilen* (sog. "Blocks") enthält, kann ein bestimmter Block mittels der Direktive `index` eingelesen werden, wobei der erste Datensatz den Index "0" erhält, z.B.

```
gnuplot> plot "beispiel.dat" index 0 using ($4):($2)
```

Um die 2. gegen die 1. Spalte im 3. Block zu plotten:

```
gnuplot> plot "beispiel.dat" index 2 using ($1):($2)
```

Beschriftung der Daten / Legende

Im Plot kann für jeden dargestellten Datensatz eine Beschriftung in der Legende (engl. *key*) gesetzt werden, z.B.:

```
gnuplot> plot "beispiel.dat" index 0 using ($4):($2) title "Datensatz 1"
```

Die Einstellungen (Positionierung, Schrift, usw.) der Legende erfolgt mittels

```
gnuplot> set key <Optionen>
```

Beispiel: Optionen für Legende

```
gnuplot> set key top right font "Helvetica, 18" spacing 4
```

Bespiel: Ein vollständiger Plot (interaktiver Modus)

Die folgende Anweisungen erzeugen einen “vollständigen” Plot im interaktiven Modus:

```
gnuplot> set title "Winkelfunktion"
gnuplot> set key bottom left
gnuplot> set xlabel "Winkel [rad]"
gnuplot> set ylabel "Amplitude [cm]"
gnuplot> set xrange [-4:4]
gnuplot> set yrange [-3:3]
gnuplot> g(x) = 2*sin(3*x**2)
gnuplot> plot g(x) with lines linetype 2 linewidth 3 \
         linecolor rgb "green" title "g(x) = 2*sin(3*x^2)"
```

Histogramm

ist die **graphische** Darstellung der Häufigkeitsverteilung einer Größe x

- erfordert die Einteilung der Daten (Messgröße x) in Klassen, sog. **Bins**, deren **Breite** (Δx) kann konstant oder variabel sein
- dabei werden über den Bins direkt aneinander angrenzende Rechtecke errichtet, deren Höhe die Häufigkeit des entsprechenden Wertes darstellt

Histogramme können auch als Abschätzung der
Wahrscheinlichkeitsdichte $p(x)$
einer kontinuierlichen Zufallsvariablen x aufgefasst werden.

Histogramme II

`gnuplot` ermöglicht die Berechnung und geeignete Darstellung eines Histogramms aus einer gegebenen Datenmenge.

Beispiel

Gegeben sei eine Datei `"gauss.dat"` mit 10^3 Zufallszahlen, gezogen aus einer Gaußverteilung

$$N(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

mit Mittelwert $\mu = 0$ und Streuung $\sigma = 1$, also der Verteilung $N(x,0,1) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2} \cdot x^2)$

10-DM-Schein mit Gaußverteilung

Histogramme III

Man möchte die Häufigkeitsverteilung dieser Werte graphisch darstellen

1. Definiere die Breite der Bins ("bin width", Δx):

```
gnuplot> bw=0.2
gnuplot> set boxwidth bw
```

   ... und eine sog. "binning"-Funktion:

```
gnuplot> bin(x,s)=s*ceil(x/s)
```

   Die Funktion ceil(x) rundet den Wert von x auf

2. Die Anzahl der Datenpunkte (für die Normierung)

```
gnuplot> N=1000
```

   erhählt man ab gnuplot-Version 4.6 auch mittels

```
gnuplot> stats "gauss.dat" ; N = STATS_records
```

3. Das Histogramm wird erzeugt und dargestellt durch:

```
gnuplot> plot "gauss.dat" u (bin($1,bw)-0.5*bw):(1./(N*bw)) \
         smooth frequency with boxes lc rgb "blue"
```

Dabei werden die Daten in Bins aufgeteilt ("gebinnt") und durch die Direktive smooth frequency die Treffer eines jeden Bins aufsummiert; das Ergebnis wird als Funktion des Bins aufgetragen, und zwar mit Balken ("boxes").

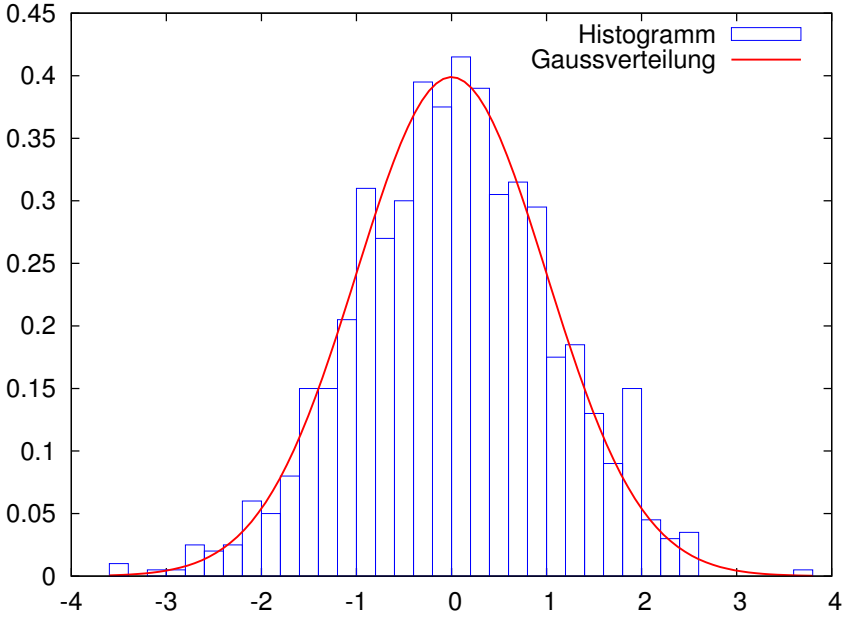
4. Man kann dazu noch die tatsächliche Verteilung explizit angeben

```
gnuplot> gauss(x) = 1./sqrt(2*pi) * exp(-0.5*x**2)
```

und sie mit dem Histogramm zusammen plotten:

```
gnuplot> replot gauss(x) with lines linewidth 3 linecolor rgb "red"
```

Histogramme V



Aufgabe 2.6 Histogramm

Plotten Sie die Verteilung der Daten `gauss.dat` (aus dem Homeverzeichnis von `htodt`) gemäß den o.g. Schritten. Plotten Sie auch die passende Gaußverteilung als Funktion dazu.

Hinweis:

In `gnuplot 4.4` kann man auch mit folgendem Trick die Anzahl `N` der Datenpunkte für die Normalisierung erhalten:

```
gnuplot> N  = 0
gnuplot> s(x) = ((N=N+1), 0)
gnuplot> plot "gauss.dat" u ($1):(s($1))
gnuplot> plot "gauss.dat" u (bin($1,bw)-0.5*bw):(1./(N*bw)) \
         smooth frequency with boxes lc rgb "blue"
```

Fitten I

gnuplot stellt den Befehl fit zu Verfügung, mit dem sich eine beliebige, von einem oder mehreren Parametern abhängige Funktion an eine Datenmenge anpassen ("fitten") lässt (Tipp: help fit).

Beispiel: linearer Fit

Man möchte eine *lineare* Funktion (zwei *freie* Parameter) an eine gegebene Datenmenge (in der Datei "beispiel.dat") fitten:

Definiere die zu fittende Funktion:

```
gnuplot> h(x) = b + a*x
```

Fit:

```
gnuplot> fit h(x) "beispiel.dat" via a,b
```

Im Grunde also:

`fit` <*Bereich*> *Funktion Daten* `via` *Parameter*

Man kann Daten und Fit zum Vergleich übereinander plotten:

```
gnuplot> plot "beispiel.dat", h(x)
```

Fitten mittels *Levenberg-Marquardt*-Algorithmus zur Minimierung von χ^2:

$$\chi^2 \equiv \sum_{i=1}^{N} \left(\frac{y(x_i, a) - y_i}{\sigma_i} \right)^2$$

mit N Messwerten (x_i, y_i), jeweils mit Messfehler σ_i, und Modell $y(x, a)$ mit einer unabhängigen Variablen x und einem freien Parameter a. Anpassung des Parameters a so, dass χ^2 ein Minimum erreicht.

Fitten III

`gnuplot` gibt während des Fit-Prozesses eine Fülle relevanter Information am Bildschirm aus, z.B. Anzahl der Iterationen, die aktuellen Werte der Parameter, die endgültigen Werte der Parameter und deren Unsicherheiten (1σ), sowie die sog. Korrelationsmatrix.

Die Ausgabe der Fit-Ergebnisse erfolgt auch in die Datei `fit.log`

Durch die Anweisung:

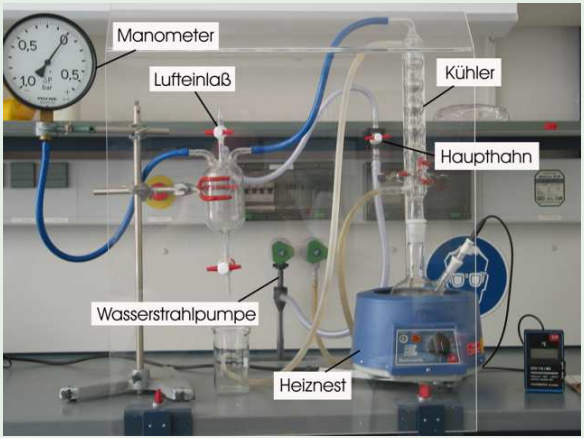
```
gnuplot> set fit errorvariables
```

werden die Unsicherheiten in den Parametern in der Variablen `a_err`, usw. gespeichert → Ausgabe mittels `show var`

Zu beachten: Anzahl der *freien* Parameter darf nicht die Anzahl von Datenpunkten überschreiten (→ Entartung, nicht wohldefiniert, sinnlos)

Tricks: Durch Wahl einer logarithmischen Skala können komplizierte Fits im linearen Raum durchgeführt werden (s. folgendes Beispiel)

Bestimmung der Phasenumwandlungsenthalpie Δh_v von Wasser in Dampf mittels einer Dampfdruckkurve:



Quelle: `http://www.uni-potsdam.de/u/phys_gprakt/img/t7.jpg`

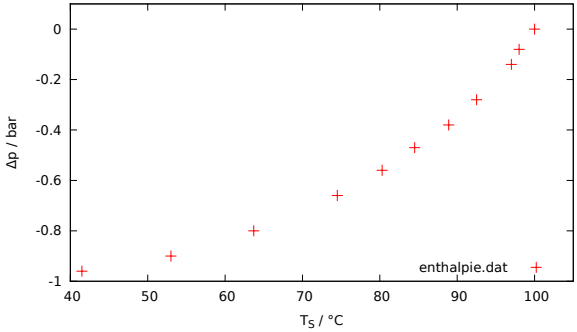
Fitten: Beispiel Dampfdruckkurve II

Gemessen wird der Druck $p(T)$ (bzgl. eines Referenzwertes p_0) als Funktion der Temperatur, und dann $\ln(p/p_0)$ über $1/T$ auftragen. Clausius-Clapeyronsche Gleichung:

$$\frac{dp}{dT} = p\frac{\Delta h_v}{RT^2}$$

Falls Δh_v nicht von T abhängt, kann man die Gleichung integrieren:

$$\ln\left(\frac{p}{p_0}\right) = -\frac{\Delta h_v}{R} \cdot \frac{1}{T} + \text{const.}$$




```
gnuplot> plot [40:105][-1:0.1] "enthalpie.dat" u ($2):($1) t "enthalpie.dat"
```

Fitten: Beispiel Dampfdruckkurve III

Idee: Transformiere Daten um lineare Gleichung zu fitten, d.h.

$$\begin{array}{rcccccc} \ln\left(\frac{p}{p_0}\right) & = & -\frac{\Delta h_v}{R} & \cdot & \frac{1}{T} & + & \text{const.} \\ f(x) & = & a & \cdot & x & + & b \end{array}$$

Ergibt also eine lineare Beziehung zwischen $\ln(p/p_0)$ und $1/T$, wobei der Anstieg die gesuchte Größe geteilt durch R ist.
Man möchte eine *lineare* Funktion an diese Messwerte fitten, die sich in der Datei `"enthalpie.dat"` befinden:

1. Definiere die Funktion, mit $x = 1/T$:

```
gnuplot> ln_p(x) = a * x + b
```

2. Die Unsicherheiten in den Fit-Parametern sollen gespeichert werden:

```
gnuplot> set fit errorvariables
```

Fitten: Beispiel Dampfdruckkurve IV

3. Variablen: R in kJ/mol, Referenzdruck p_0 in bar

```
gnuplot> R = 8.314e-3 ; p_0 = 1.019
```

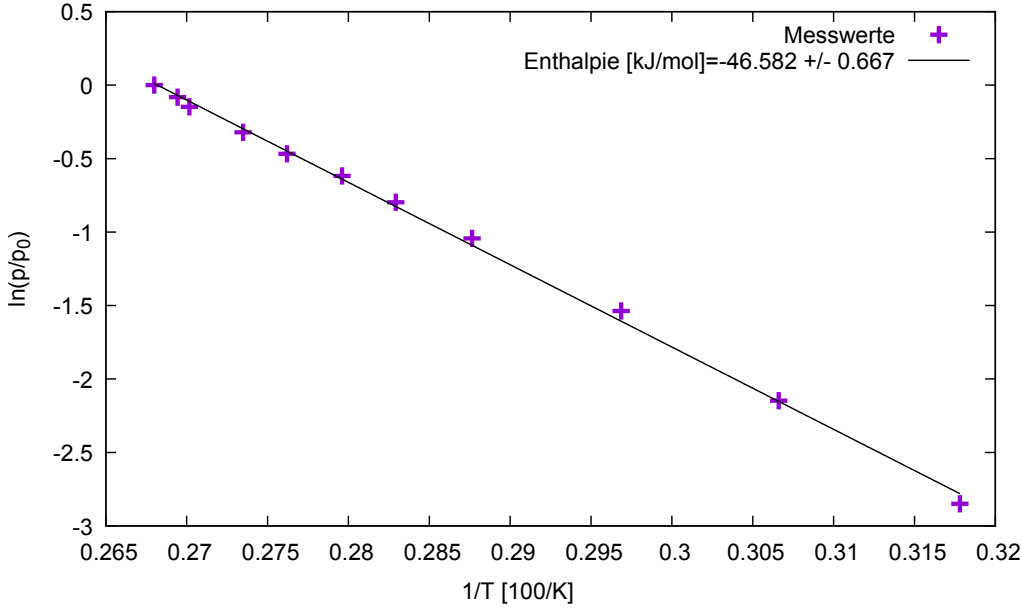
4. Fit:

```
gnuplot> fit [*:*] ln_p(x) "enthalpie.dat" \
 using (1./(($2)+273.15)):(log((p_0+($1))/p_0)) via a,b
```

5. Man kann Daten und Fit vergleichen, und dazu den Wert von a und dessen Unsicherheit a_err als Beschriftung verwenden:

```
gnuplot> plot "enthalpie.dat" using \
 (1e2/(($2)+273.15)):(log((1.019+($1))/1.019)) \
 with points ps 2 linewidth 2 title "Messwerte", \
 ln_p(1e-2*x) with lines linecolor "black" \
 t sprintf("Enthalpie [kJ/mol]=%5.3f +/- %5.3f",a*R,R*a_err)
```

Fitten: Beispiel Dampfdruckkurve V



Fitten: Beispiel Dampfdruckkurve VI

Bemerkungen:

- formaler Fehler `a_err` $\ll$ Messunsicherheiten (Unterschätzung!)
  $\rightarrow$ Literatur: 40.657 kJ/mol bei 100 °C
- x-Achse skaliert mit Faktor 100 für bessere Lesbarkeit
- Achsenbeschriftung:

```
gnuplot> set xlabel "1/T [100/K]"
gnuplot> set ylabel "ln(p/p_0)"
```

- formatierte Ausgabe der Beschriftung (3 Nachkommastellen) mittels C-artigem Befehl `sprintf`:

```
gnuplot> ... title sprintf ("pi mit 2 Nachkommastellen: %3.2f", pi)
```

Aufgabe 2.7 Eine Funktion an Daten fitten

Erstellen Sie den oben gezeigten Plot zur Verdampfungsenthalpie, indem sie das Beispiel mit der Datei `enthalpie.dat` (aus dem Homeverzeichnis von `htodt`) nachvollziehen.

Plot als Datei speichern I

Plot in einer Datei speichern, um diesen in ein Dokument, z.B. ein Versuchsprotokoll, Bachelor-Arbeit usw. einzubinden.

Plot in einer `.pdf`-Datei speichern durch folgende Anweisungen:

```
gnuplot> set terminal pdf enhanced color
```

→ Ausgabeformattyp PDF setzen. Optionen `enhanced` und `color` erlauben Nutzung besonderer Schriftarten (z.B. griechische Buchstaben) oder auch Indizes, Exponenten usw., bzw. Farb-Ausgabe.

```
gnuplot> set output "meinplot.pdf"
```

→ Ausgabe in Datei `"meinplot.pdf"` umlenken (sonst: Eingabeterminal)

Erneutes Plotten mit neuem Ausgabeformat und Ausgabekanal:

```
gnuplot> replot
```

Plot als Datei speichern II

Achtung!

I.d.R. ist die so erzeugte Datei erst verwendbar, nachdem `gnuplot` beendet wurde, *oder* das Terminal auf einen anderen Typ gesetzt wurde. (Datei wird sonst offen gehalten.)

Weitere Hinweise:

- Das Terminal bleibt solange bei dem eingestellten Typ (hier: PDF), bis es mittels `set term` wieder auf einen anderen Typ gesetzt wird, z.B. `set term wxt`
- Sonderzeichen für Terminals `pdf enhanced` und `postscript enhanced`:

Eingabe	pdf/ps-Ausgabe
`T_0`	T_0 (Subscript)
`e^{-x}`	e^{-x} (Superscript)
`{/Symbol Qp}`	$\Theta\pi$

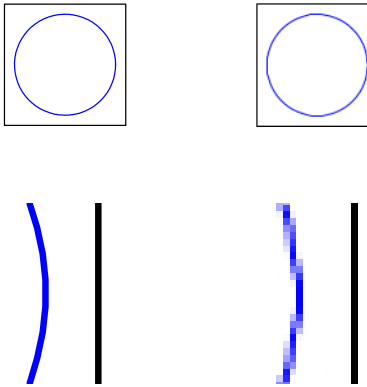
- Weitere unterstütze Formate sind:
  `ps`, `png`, `jpeg`, u.v.m (Tipp: `help terminal`)

Plot als Datei speichern III

Aufgabe 2.8 Ausgabe als PDF

Lassen Sie sich den Graphen zur Verdampfungsenthalpie als PDF-Datei ausgeben. Beachten Sie die Subskripte der Achsenbeschriftung.

Exkurs: Vektorgrafik und Pixelgrafik



Vektorgrafik (PS, links) und Pixelgrafik (jpg, rechts) und vergrößertes Detail (darunter)

- Vektorgrafik: Datei enthält Anweisung zum Zeichnen eines Kreises (`x y r 0 360 arc stroke`), bei der Darstellung auf einem pixelbasierten Gerät (Monitor, Drucker) wird die Grafik mit der verfügbaren Auflösung dargestellt. Vorteil: beliebig vergrößerbar. Geeignet für Diagramme
- Pixelgrafik: Datei enthält komprimierte RGB-Matrix mit vorgegebener Auflösung (z.B. 360x360 Pixel). Nachteil: Beim Vergrößern zu einer höheren Auflösung werden die einzelnen Pixel vergrößert, die Grafik sieht "pixelig" aus. Geeignet für Fotos

Man kann im interaktiven Modus solange die Einstellungen bearbeiten, bis der Plot so aussieht, wie man möchte, um ihn dann als Datei zu speichern. Dabei sollte man aber beachten, dass nicht alle Terminals (also z.B. x11 gegenüber PostScript) alle Schriften, Styles, usw. unterstützen, sodass die Darstellung zwischen dem interaktiven Modus und der gespeicherten Datei i.d.R. voneinander abweichen werden.

Einen Ausweg aus diesem Dilemma bietet `gnuplot` mit der Möglichkeit Skripte zu schreiben. Ähnlich einem Shell-Skript ist ein `gnuplot`-Skript nichts anderes als eine (ASCII-) Datei, die eine Reihe von `gnuplot`-Anweisungen enthält, die von `gnuplot` interpretiert und ausgeführt werden.

Eine solche Datei kann dann sooft aufgerufen und die Einstellungen in dieser Datei modifiziert werden, bis man das gewünschte Ergebnis erreicht hat. Dabei wird z.B. der Plot in eine PostScript-Datei geschrieben, die man sich wiederum nach jedem Aufruf des `gnuplot`-Skripts angucken kann.

Ein weiterer Vorteil eines solchen Skripts ist, dass im Falle einer Veränderung der eingelesenen Daten, der Plot nicht komplett neu erstellt werden muss (wie es der Fall bei Xmgrace ist), sondern lediglich die Datei mit den neuen Daten im Skript eingelesen werden muss, was natürlich die Arbeit sehr erleichtert und effizienter macht.

Ein solches Skript könnte folgendermaßen aussehen (beachte, Zeilen mit # werden als Kommentare aufgefasst und daher von `gnuplot` ignoriert, dienen aber der Information und Lesbarkeit eines Skripts!):

Beispiel: Skript-Datei 'beispiel.gplt' I

```
# Terminal-Einstellungen: PostScript, bunt, Sonderzeichen
set terminal postscript enhanced color eps

# Datei-Name
set output "beispiel.eps"

# Daten Beschriftung (unten, links)
set key bottom left

# Achsen Beschriftung
set xlabel "Winkel [rad]"
set ylabel "Amplitude [cm]"

# Wertebereiche
set xrange [-4:4]
set yrange [-3:3]

# Eine Funktion
g(x) = 2*sin(3*x**2)

# Plot
plot g(x) with lines linetype 2 linewidth 3 linecolor rgb "red"  title "g(x) = 2sin(3x^2)"
```

Beispiel: Skript-Datei 'beispiel.gplt' II

Die Skript-Datei kann im interaktiven Modus mit dem Befehl `load` ausgeführt werden, also:

```
gnuplot> load "beispiel.gplt"
```

Oder in der Linux-Shell:

```
$> gnuplot beispiel.gplt
```

In beiden Fällen wird eine Datei "beispiel.eps" erzeugt, die man sich z.B. mit gv angucken kann, also in der Shell:

```
$> gv beispiel.eps &
```

3D-Plots I

Das Plotten in drei Dimensionen bedeutet die Darstellung von Punkten der Form (x, y, z). Diese können *diskrete* Punkte sein, oder aber zu einer *kontinuierlichen* Menge der Form $z = f(x, y)$, also einer 2D-Fläche, eingebettet im 3D-Raum ($\rightarrow$ Hyperfläche), gehören.

Im ersten Fall erwartet `gnuplot` eine datei (z.B. "`beispiel.dat`") mit drei Spalten und mindestens einem Datensatz (x, y, z); die Darstellung erfolgt einfach durch

```
gnuplot> splot "beispiel.dat"
```

Analog zu den x-, und y-Koordinaten kann man die Achsenbeschriftung, den Wertebereich, usw. der z-Koordinate einstellen durch

```
gnuplot> set zlabel "z-Koordinate"
gnuplot> set zrange [zmin:zmax]
```

mit geeigneten Werten für `zmin`, `zmax`

```
gnuplot> set xyplane 0
```

setzt den Offset der xy-Ebene auf 0 (gemeinsamer Ursprung der Achsen).

Die Darstellung von zwei-dimensionalen Flächen verlangt lediglich die Definition einer entsprechenden Funktion.

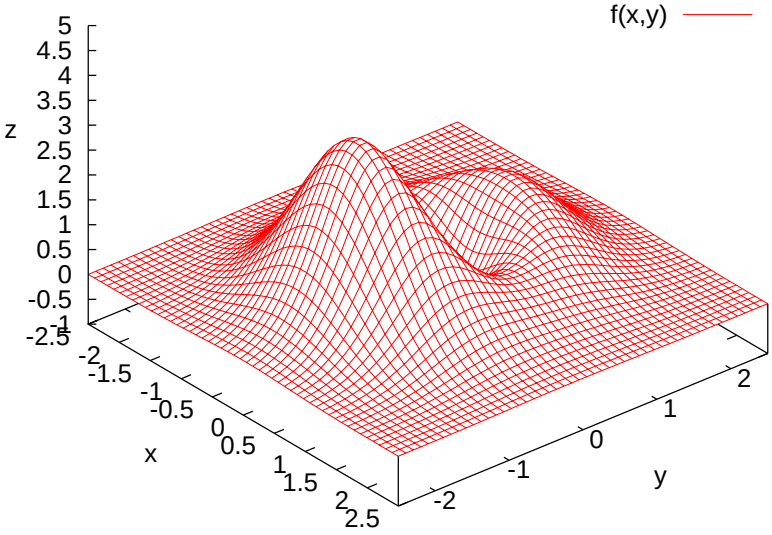
Es gibt drei mögliche Darstellungsarten

- Gitter (`isosamples`)
- Höhenlinien (`contours`)
- Farbpalette (`pm3d`)

Wir werden hier die Möglichkeiten die `gnuplot` bietet anhand des folgenden Beispiels erläutern:

$$f(x,y) = \left(x^2 + 2.5y^2 - y\right) \cdot \exp\left(1 - \left(x^2 + y^2\right)\right)$$

3D-Plots (Gitter) I



3D-Plots (Gitter) II

Der vorige Plot wurde erzeugt durch (im interaktiven Modus):

1. Wertebereiche:

```
gnuplot> set xrange [-2.5:2.5]
gnuplot> set yrange [-2.5:2.5]
gnuplot> set zrange [-1:5]
```

2. Blickwinkel (rot$_x$ {, rot$_z$, scale, scale$_z$}) Im interaktiven Modus bietet gnuplot die Möglichkeit, den Blickwinkel mit der Computer-Maus einzustellen!:

```
gnuplot> set view 40, 50, 1.0, 1.5
```

3. Auflösung des Gitters (Isolinien pro Achse; iso$_u$ {, iso$_v$}):

```
gnuplot> set isosamples 50
```

4. "Undurchsichtige" Fläche (vgl. `gnuplot> unset hidden3d`):

```
gnuplot> set hidden3d
```

5. Die Funktion

```
gnuplot> f(x,y)=(x**2+2.5*y**2-y)*exp(1-(x**2+y**2))
```

6. Und der Plot wird erzeugt durch:

```
gnuplot> splot f(x,y) with lines
```

Aufgabe 2.9 Eine Funktion $z(x, y)$ plotten

Plotten Sie dem Beispiel entsprechend die Hyperfläche
$f(x, y) = \left(x^2 + 2.5y^2 - y\right) \cdot \exp\left(1 - \left(x^2 + y^2\right)\right)$.

Kurven im 3D-Raum $[x(u), y(u), z(u)]$ bzw.
Hyperflächen $[x(u, v), y(u, v), z(u, v)]$:

```
gnuplot> set parametric
gnuplot> splot [0:20] cos(u), sin(u), u
gnuplot> splot cos(u)*cos(v),cos(u)*sin(v),sin(u)
gnuplot> unset parametric
```

Aufgabe 2.10 Kurven und Hyperflächen in 3D plotten

Plotten Sie die die o.g. Kurve und die Hyperfläche mittels der `parametric`-Plot-Option.

3D-Plots (Konturen) I

gnuplot bietet auch die Möglichkeit, eine Fläche durch Höhenlinien (engl. *contours*) darzustellen.

Wenn die Fläche durch $z = f(x, y)$ definiert ist, dann sind Höhenlinien gegeben durch $f(x, y) \equiv c_i$, wobei c_i konstante Werte sind.

Die entsprechende Anweisung in gnuplot lautet:

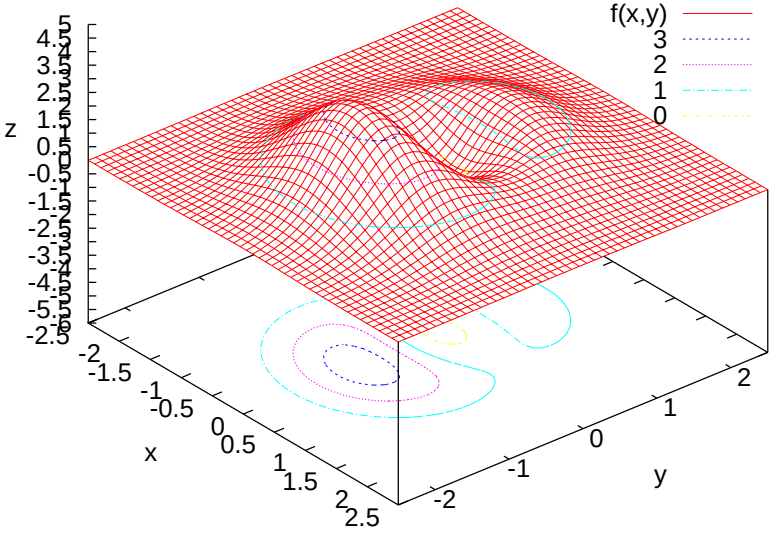
```
gnuplot> set contour <Option>

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

wobei *<Option>* durch `base`, `surface`, oder `both` zu ersetzen ist.
Es gibt eine Reihe von Optionen, mit denen man die Darstellung von Höhenlinien verarbeiten kann, dazu einfach

```
gnuplot> help contour
```

eingeben. Falls man `set contour both` angibt, sieht es so aus:

3D-Plots (Konturen) II



3D-Plots (Farbpaletten) I

Schließlich kann man 2D-Flächen mittels Farbkodierung (sog. Palette) darstellen.

Dazu muss man lediglich den entsprechenden Plot-Stil nehmen, nämlich `pm3d`, also "palette map in 3D".

Die enstsprechende Anweisung in `gnuplot` lautet also:

```
gnuplot> splot f(x,y) with pm3d
```

3D-Plots (Farbpaletten) II