Anleitung zum Fitten mit gnuplot[1]

Diese Anleitung soll als Einstieg ins Arbeiten mit gnuplot dienen. Hervorragende weiterführende Tips gibt es unter:

- http://t16web.lanl.gov/Kawano/gnuplot/intro/index-e.html

Generell ist es meist günstig alle Befehle in eine Textdatei zu schreiben und diese mit gnuplot zu laden.
`load "datei.txt"`
Zur Zeit nicht benötigte Befehle können mit einem `#` auskommentiert werden.

1 Darstellung von Daten und Funktionen – Plot

Mit gnuplot können sowohl Datenpunkte als auch Gleichungen – oder beides dargestellt werden.

1.1 Darstellung von Daten

- Die Daten müssen in einer Textdatei in durch `Tab` getrennten Spalten stehen.
- Einzelne Zeilen können durch `#` auskommentiert werden.
- `plot "daten.txt"` ergibt ein Diagramm bei dem die Daten der ersten Spalte auf der x-Achse, die der zweiten Spalte auf der y-Achse aufgetragen sind.
- Mit `plot "daten.txt" using 3:2` wird die zweite Spalte über der dritten aufgetragen.
- Vor dem Plotten kann auch mit den Spalten gerechnet werden:
  `plot "daten.txt" using (1000/$3):(log($2))`
- `plot` und `using` können durch `p` und `u` abgekürzt werden.
- Der Bereich der x-Werte der aufgetragen wird, kann beschränkt werden:
  `p [3:11] "daten.txt"`

[1] die Anleitung bezieht sich auf die Windows-Version von gnuplot – sollte aber auch für Linux-Systeme weitestgehend richtig sein

1.2 Darstellung von Funktionen

- Standardfunktionen wie `sin, cos, exp, log...` sind vordefiniert (im Zweifelsfall Hilfe bemühen)
- Es können eigene Funktionen definiert werden:
  `f(x)=a+H_a*exp(x**2/b**2)`[2]
  Dabei muss die unabhängige Variable immer `x` heißen.
- Parameter müssen vor dem Plotten gesetzt werden `a=1;H_a=2;b=3`
  Die Parameter und Funktionennamen sind case-sensitiv.
- `p "daten.txt" , f(x)` stellt Datenpunkte und Funktionsgraph in einem Diagramm dar.

2 Least-Squares-Fit

- Eine vorher definierte Funktion mit beliebig vielen Parametern kann durch einen Least-Squares-Fit an einen Datensatz angepasst werden:
  `fit f(x) "daten.txt" u 3:2 via a, H_a, b`
- Zuvor sollten sinnvolle Startwerte für die Parameter gesetzt werden (sonst ist der Startwert immer Eins):
  `a=1;H_a=2;b=3`
- Es ist möglich vor dem Fit mit den Spalten zu rechnen und den x-Wertebereich an den gefittet werden soll zu verändern.
  `fit [3:11] f(x) "daten.txt" u (1000/($3)):(log($2)) via a, H_a, b`
- `p [0:15] "daten.txt" u (1000/($3)):(log($2)), f(x)` stellt die Daten von $x = 0$ bis $x = 15$ zusammen mit der an die Werte zwischen $x = 0 \ldots 15$ gefitteten Funktion in einem Diagramm dar.

3 Verändern der Ausgabe

- Es können Titel für die einzelnen Graphen angegeben werden, die in der Legende auftauchen:
  `p "daten.txt" t "Titel der Funktion"`
- Die Achsen können beschriftet werden:
  `set xlabel "Reziproke Temperatur 1000/T (1/K)", set ylabel...`

[2] gebrochene Potenzen müssen als Dezimalzahlen angegeben werden `x**1.5` statt `x**(3/2)`

- Der dargestellte Wertebereich der Achsen kann geändert werden:
  `set yrange [1e13:1e17]`
- Die Achsen können logarithmisch aufgetragen werden:
  `set logscale <???>`
  `<???>` kann je nach Achse `x`, `y` oder `xy` sein.
- Alle, mit `set...` gemachten Angaben können mit `reset` zurückgesetzt werden.

Um die erzeugte Graphik als Bilddatei auszugeben muss in das entsprechende `terminal` umgeschaltet werden. Es existiert kein `terminal` zur Erzeugung von .jpg-Dateien. Es soll nun im Folgenden kurz dargestellt werden, wie .png- und .eps-Dateien erzeugt werden können.

`set term png`	Schaltet auf die Erzeugung von .png-Dateien um.
`set term postscript color enhanced font "Helvetica, 18"`	Schaltet auf die Erzeugung von farbigen postscript-Dateien um. Die Schrift wird auf Helvetica mit 18pt gesetzt
`set encoding iso_8859_1`	Nötig zur Darstellung von Umlauten etc.
`set output "neutral.eps"`	Name der Ausgabedatei
`replot`	Plottet die Ausgabe in die Datei
`pause 1`	Pause
`set term windows`	Setzt das `terminal` zurück auf Bildschirmanzeige
`replot`	Plottet die Graphen nochmal auf den Bildschirm