1 Kurzanleitung zum Einstieg in Gnuplot

1.1 Was ist Gnuplot?

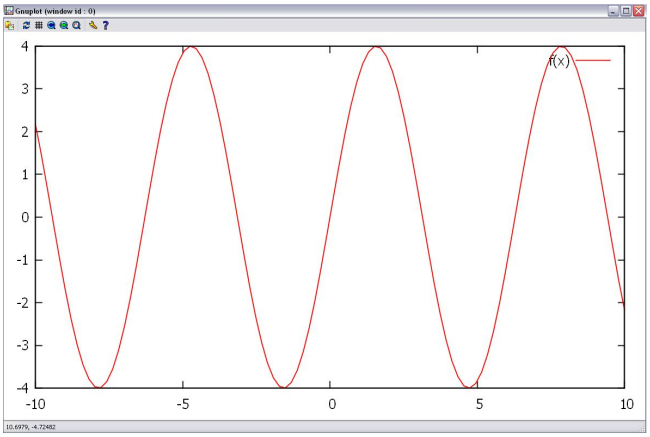
Gnuplot ist ein kommandozeilengesteuertes Computerprogramm zur graphischen Darstellung von Messdaten und mathematischen Funktionen. Mit Gnuplot können sowohl zwei- als auch dreidimensionale Grafiken erstellt und bearbeitet werden. Außerdem kann Gnuplot funktionale Zusammenhänge an Sätze von Messwerten anpassen (*Fitten*). Gnuplot stammt aus der Unix-Welt, läuft aber auf allen gängigen Betriebssystemen. Von http://www.gnuplot.info/ kann es kostenlos heruntergeladen werden. Für Windows kann eine Installationsdatei (aktuelle Version `gp460-win32-setup.exe`) heruntergeladen werden. Bei anderen Betriebssystemen muss das Programmpaket heruntergeladen und bei der Installation selbst kompiliert werden. Auf der oben genannten Seite findet sich eine Online Hilfe. Im folgenden soll hier lediglich eine kurze Einführung gegeben werden, die den Einstieg in die Benutzung erleichtern und die Erstellung einfacher Diagramme ermöglichen soll. Eine umfassende Anleitung wollen wir hier nicht geben.

1.2 Wie arbeitet man mit Gnuplot? Kommandozeile, Skripte

Sie starten Gnuplot unter Windows, indem Sie im Unterverzeichnis `\gnuplot\bin` die Datei `wgnuplot.exe` doppelklicken. Mit der Datei kann ebenfalls eine Verknüpfung auf dem Desktop erstellt werden. Sie erhalten ein Kommandozeilen-Fenster mit der Eingabeaufforderung

```
gnuplot> _
```

Ist eine Eingabe erfolgt und mit ENTER bestätigt, wird diese ausgeführt. Mit einem `plot`-Befehl erscheint ein weiterer Fenster, in dem die Graphik dargestellt wird. Andere




```
gnuplot> a=4
gnuplot> f(x)=a*sin(x)
gnuplot> plot f(x)
```

Abbildung 1: Graphische Darstellung einer Funktion in `Gnuplot`.

Befehle, beispielsweise Definitionen einer Funktion oder Angaben zur Formatierung einer Graphik, wirken sich erst später beim ausführen des `plot`-Befehl aus.
Mit den Eingaben

```
gnuplot> a=4
gnuplot> f(x)=a*sin(x)
gnuplot> plot f(x)
```

wird folgende Graphik erzeugt (s. Abb. 1).

Damit die Befehlsfolgen nicht immer wieder über die Kommando-Zeile eingegeben werden müssen, können diese in einer Script-Datei mit `save Datei.plt` gespeichert und jederzeit mit dem Befehl `load Datei.plt` aufgerufen werden. Es handelt sich um eine einfache Textdatei, die ebenfalls in einem Text-Editor-Programm erstellt und bearbeitet werden kann. So kann die Skript-Datei an die Darstellung unterschiedlicher Datensätze angepasst und wiederverwendet werden.

Hinweis:

- Kommentare in einer Skript-Datei beginnen mit einem Hash-Zeichen (`#`)
- Kommandos können sich in einer `plt`-Datei über mehrere Zeilen erstrecken, das letzte Zeichen muss dann ein \(Backslash) sein

1.3 Graphische Darstellung von Funktionen

Einige wichtige Gnuplot-Befehle zur graphischen Darstellung von Funktionen sollen hier anhand von einem Beispiel (s. Abb. 2) gezeigt und erläutert werden.

- Eine Diagrammbeschriftung wird mit `set title` erzeugt

  ```
  gnuplot>  set title 'Sinus- und Cosinusfunktion'
  ```

- Eine Achsenbeschriftung wird mit `set xlabel` bzw. `set ylabel` erzeugt

  ```
  gnuplot> set xlabel 'x[rad]'
  gnuplot> set ylabel 'sin(x)'
  ```

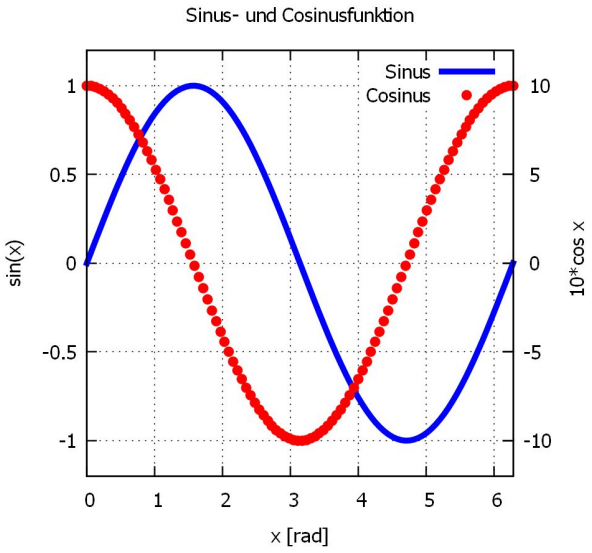
- Werden mehrere Funktionen in einem Diagramm dargestellt, ist es möglich, zwei Achsen unabhängig voneinander zu beschriften

  ```
  gnuplot> set ylabel 'sin(x)'
  gnuplot> set y2label '10*cos(x)'
  ```

- Hilfsstriche werden mit `'set y2tics'` gesetzt: `gnuplot> set y2tics 5`

- Gitternetzlinien werden mit dem Befehl `'set grid'` erzeugt

- Mit den Angaben `[n:m]` nach dem `plot`-Befehl werden die Grenzen festgelegt, innerhalb dessen eine Funktion dargestellt werden soll. Denselben Effekt erreicht man mit den Befehlen `set xrange [m:n]`, `set yrange [m:n]` bzw. `set y2range [m:n]`




```
set title 'Sinus- und Cosinusfunktion'
set xlabel 'x [rad]'
set ylabel 'sin(x)'
set y2label '10*cos x'
set xrange [0:2*pi]
set yrange [-1.2:1.2]
set y2range [-12:12]
set y2tics 5
set grid
set size ratio 1.0
plot sin(x) title 'Sinus' lw 4 lc 3, \
cos(x) title 'Cosinus' with points lt 46
```

Abbildung 2: Graphische Darstellung von Funktionen

```
gnuplot>  plot [0:2*pi] [-1.2:1.2] sin(x)
gnuplot>  set xrange [0:2*pi]
gnuplot>  set yrange [-1.2:1.2]
gnuplot>  set y2range [-12:12]
```

- Mehrere Kurven im selben Diagramm lassen sich darstellen, indem man die Funktionen durch ein Komma getrennt nach dem `plot` Befehl angibt

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

- Will man einer Funktion einen eigenen Namen geben, so benutzt man den Befehl `title` nach dem `plot`-Befehl

  ```
  gnuplot> plot sin(x) title 'Sinus'
  ```

- Kurven werden per Default als Linien dargestellt. Die Art der Darstellung kann jedoch angepasst werden - Linienbreite kann variiert werden, Kurven können mit Symbolen (`'points'`) oder mit `'linespoints'` - Symbolen, die mit Linien verbunden sind, gezeichnet werden.

  ```
  gnuplot> plot cos(x) with points
  ```

  Hinweis: Gibt man in der Kommandozeile den Befehl `'test'` ein, so wird ein Fenster angezeigt, in dem man alle verfügbaren Linientypen, Größen und Farben ablesen kann. Setzt man hinter einen `plot`-Befehl eine Angabe für die Linienbreite `'lw'` und eine Zahl, so bekommt man die entsprechende Linienstärke. Mit `'lt'` können der Linientyp und mit `'lc'` die Farbe gewählt werden.

- Das Größenverhältnis der Achsen kann mit `set size ratio` angepasst werden. Mit der Angabe `size ratio 1.0` sind die Achsen gleich groß. Denselben Effekt erreicht man mit dem Befehl `set size square`

```
gnuplot> set size ratio 1.0
```

Logarithmische Auftragung

Standardmäßig werden die Daten in linearen Koordinaten dargestellt. Für viele Zwecke ist die Darstellung in einem anderen als dem linearen System von Vorteil. Exponentielle Abhängigkeiten werden üblicherweise in einfach logarithmischen Koordinaten dargestellt.
In Abb. 3 ist die Exponentialfunktion $f(x) = e^x$ im Diagramm links in linearen Koordinaten dargestellt. Im Diagramm rechts wurde die Skalierung der y-Achse mit `set logscale y` auf eine logarithmische umgestellt.

```
gnuplot> set xlabel 'x'
gnuplot> set ylabel 'log y'
gnuplot> set grid
gnuplot> set logscale y
gnuplot> set size ratio 1.0
gnuplot> plot [-5:5]  exp(x)
```

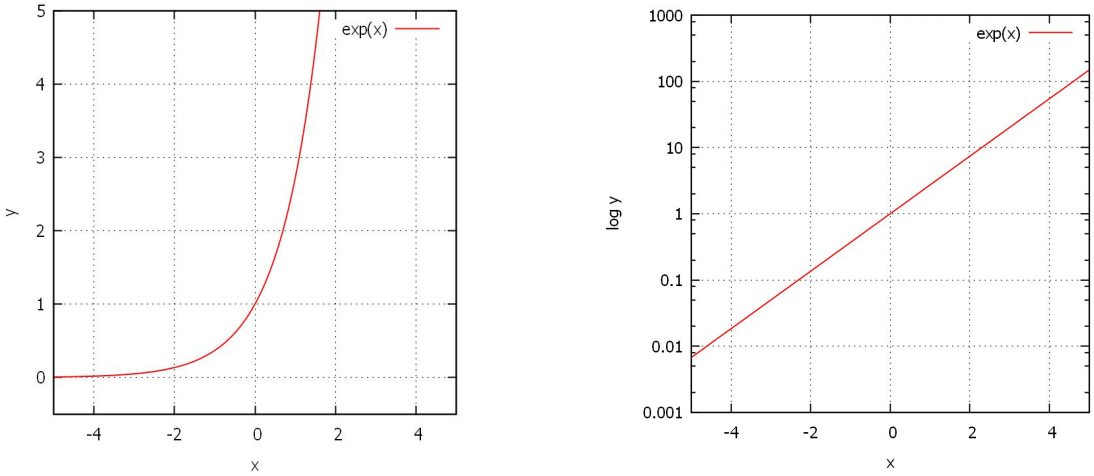


Abbildung 3: Lineare und logarithmische Auftragung einer Exponentialfunktion

Doppelt-logarithmische Auftragung

Auch doppelt-logarithmische Darstellungen, welche sich besonders für Abhängigkeiten nach Potenz-Gesetzen eignen, sind möglich. Die Funktion $f(x) = 2 \cdot x^3$ in doppelt-logarithmischer Darstellung (s. Abb. 4) erhält man durch die folgenden Befehle [1]:

```
gnuplot> set logscale x
gnuplot> set logscale y
```

[1] Man beachte auch die Darstellung von x^3 durch `x**3`.

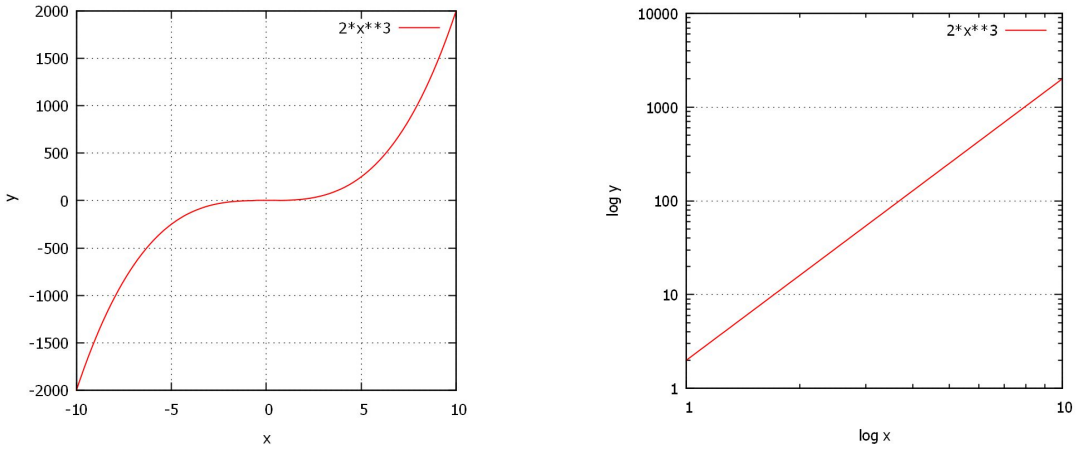


Abbildung 4: Lineare und doppellogarithmische Auftragung eines Potenz-Gesetzes

```
gnuplot> set xtics 0.1
gnuplot> set size ratio 1.0
gnuplot> plot [1:10] 2*x**3
```

Polarkoordinaten

In einem Polarkoordinatensystem ist jeder Punkt der Ebene `(t,r)` durch eine Winkel- `t` und eine Radialkoordinate `r` beschrieben. Polarkoordinaten sind oft nützlich, wenn eine Messgröße in Abhängigkeit vom Winkel aufgetragen werden soll.

In Abb. 5 ist ein Beispiel eines mit Hilfe von `gnuplot` erstellten Polarkoordinatendiagramms gezeigt.

- Zuerst wird ein Polarkoordinatendiagramm erzeugt

  ```
  gnuplot> set polar
  ```

- Es werden Gitternetzliniern gezeichnet

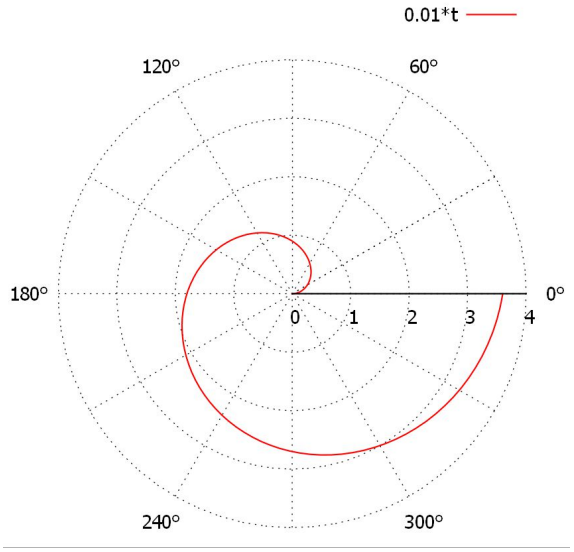
  ```
  gnuplot> set grid polar
  ```

- In `gnuplot` ist die Winkelkoordinate per Default in Radian im Intervall `[0:2*pi]` definiert. Mit folgendem Befehl kann der Winkel auch in Grad im Intervall [0:360] angegeben werden[2]

  ```
  gnuplot> set angles degrees
  ```

- Mit folgenden Befehlen werden die Achsen bzw. die Hilfsstriche aus dem Diagramm entfernt

[2] Mit `set trange [m:n]` kann der Winkelbereich `t` beliebig gewählt werden




```
set polar
set grid polar
set angles degrees
unset border
unset xtics
unset ytics
set size square
set rrange [0:4]
set label '0°' at ... center
set label '60°' at ... center
...
set key tmargin
plot 0.01*t
```

Abbildung 5: Graphische Darstellung einer Funktionen in Polarkoordinaten

```
gnuplot> unset border
gnuplot> unset xtics
gnuplot> unset ytics
```

- Folgender Befehl ist äquivalent zu `size ratio 1.0` - es wird ein Diagramm mit dem Achsenverhältnis 1:1 erzeugt

  ```
  gnuplot> set size square
  ```

- Es wird das darzustellende Intervall der Radialkoordinate gesetzt

  ```
  gnuplot> set rrange [0:4]
  ```

- Beschriftung der Winkel erfolgt mit dem Befehl `set label`. Mit diesem Befehl kann an einer beliebigen angegebenen Stelle im Diagramm eine Beschriftung hinzugefügt werden.

  ```
  gnuplot> set label '0°' at (4.5*cos(0)), (4.5*sin(0)) center
  ```

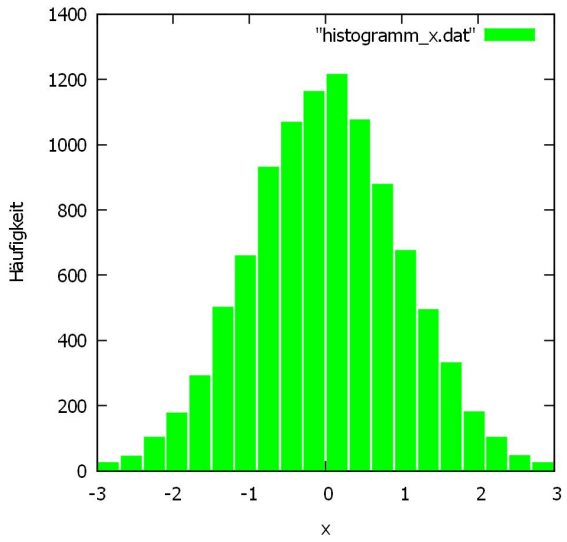
  Obige Zeile fügt den Text '0°' an der Stelle mit den Koordinaten ($x = r * \cos t$), ($y = r * \sin t$) und zentriert diesen. Entsprechend werden auch weitere Beschriftungen bei 60°, 120° usw. gesetzt.

- Der Befehl `set key` positioniert die Legende. Mit der Angabe `tmargin` wird die Legende am oberen Rand des Diagramms platziert.

  ```
  gnuplot> set key tmargin
  ```

Säulendiagramme

Ein Säulen- bzw. Balkendiagramm ist eine Art der graphischen Darstellung, die oft benutzt wird, wenn man beispielsweise eine statistische Verteilung in Form eines Histogramm abbilden will. In `gnuplot` wird ein Säulendiagramm durch das Hinzufügen des Befehls `'with boxes'` nach dem `plot`-Befehl erzeugt (s. Abb. 6). Mit `'set boxwidth'` kann die relative Breite der Balken eingestellt werden. Außerdem können die Farbe der Randlinie `'lc rgb 'green''`, die Füllfarbe `'set style fill solid 1.0'` bzw. das Füllmuster mit `'set style fill pattern'` gewählt werden.




```
set xlabel 'x'
set ylabel 'Häufigkeit'
set xrange [-3:3]
set size square
set boxwidth 0.85 relative
set style fill solid 1.0
plot 'histogramm_x.dat'  w boxes, \
     lc rgb 'green'
```

Abbildung 6: Graphische Darstellung einer Funktionen als Säulendiagramm.

Hinweis

Um beim nächsten `plot`-Befehl zu Voreinstellungen zurückzukehren benutzt man den `reset`-Befehl. Um einzelne Befehle zurückzusetzen wird der `unset`-Befehl benutzt:

```
gnuplot> reset
gnuplot> unset grid    #Das Netzgitter wird aus dem Diagramm entfernt
gnuplot> unset polar   #Man kehrt zu kartesischen Koordinaten zurück
```

1.4 Graphische Darstellung von Messwerten

Wichtig im Zusammenhang mit dem experimentellen Praktikum ist natürlich nicht nur die grafische Darstellung von Funktionen, sondern vor allem von Messdaten. Gemessene Daten können hierbei aus einer Datei bezogen werden. Die Daten sollten im Normalfall im so genannten `csv`-Format (character seperated values) vorliegen, das heißt die Werte stehen in einer Textdatei, wobei die Spalten durch Leerzeichen oder Tabulator getrennt werden. Eine solche Datei kann von einem Messprogramm (Measure Dynamics, IPS, usw.) ausgegeben werden. Handelt es sich um selbstständig (ohne PC) aufgezeichnete Daten, können diese mit Hilfe eines Tabellenkalkulationsprogramms in eine Tabelle eingetragen und als eine Textdatei exportiert werden. Die Messdatei kann auch mehr als zwei Spalten enthalten, wenn eine Messung beispielsweise bei verschiedenen Parametern durchgeführt wurde.

In unserem Beispiel wurde die Ausgangsspannung eines Verstärkers abhängig von der Eingangsspannung für zwei verschiedene Verstärkungsfaktoren gemessen. Die Eingangsspannung steht in Spalte eins, die Messergebnisse für die erste Messung wurden in Spalte zwei, und die Ergebnisse der zweiten Messung in die dritte Spalte eintragen. Ein Ausschnitt aus einer solchen Messdatei mit dem Namen `'Messung1.dat'` bzw. `'Messung1.txt'` sieht wie folgt aus:

```
#U_e    U_a_1   U_a_2
 0.0     0.0     0.0
 0.5     1.8     3.7
 1.0     3.5     7.3
 1.5     5.2     10.9
 2.0     7.1     14.5
 2.5     8.9     17.9
```

Möchte man die Daten in einem Diagramm darstellen, so muss nach dem `plot`-Befehl der Dateiname eingetragen werden. Befindet sich die Datei in einem anderen Verzeichnis, so muss der Pfad entsprechend ergänzt werden. [3]

```
gnuplot> plot 'C:\Messdaten\...\Messung1.txt'
```

Standardmäßig werden die Daten der 2. Spalte (y-Spalte) gegen die der 1. Spalte (x-Spalte) aufgetragen. Enthält die Datei mehr als zwei Spalten und möchte man andere als das erste Spaltenpaar für die Auftragung auswählen, so können über die Option `'using m:n'` die gewünschten Spalten angegeben werden:

```
gnuplot> plot "Messung1.txt"
gnuplot> plot "Messung1.txt"  using 1:2       #alternativ

gnuplot> plot "Messung1.txt"  using 1:3
```

[3] Der Pfad kann auch über die graphische Oberfläche von gnuplot mit `Plot-->Data filemname...` ausgesucht werden.

In unserem Beispiel wird das Diagramm folgenden Befehlen erzeugt:

```
set grid
set xlabel 'Eingangsspannung U_e [V]'
set ylabel 'Ausgangsspannung U_a [V]'
plot "Messung1.txt"  using 1:2, \
     "Messung1.txt"  using 1:3
```

Das Ergebnis ist in Abbildung 7 zu sehen.

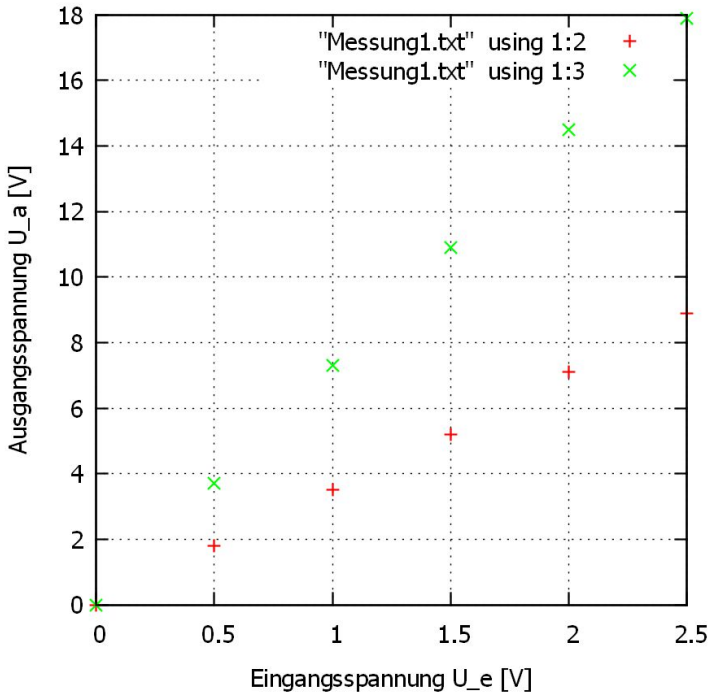


Abbildung 7: Graphische Darstellung der Daten aus einem Datensatz.

1.5 Berechnungen durchführen

Oft müssen die Messdaten noch verarbeitet werden, bevor das Ergebnis graphisch dargestellt werden kann. Möchte man mit dem Datensatz Berechnungen durchführen, so kann man es mit dem `using`-Befehl tun:

```
gnuplot> plot "Messung.dat" using ($1*2*pi/360):(sin($2))
```

Mit `$1` bzw. `$2` wird die Spalte gewählt, die als Variable verwendet werden soll. Die Befehle müssen in Klammern stehen.

Die Funktionen können auch schon im Vorfeld definiert werden:

```
gnuplot> k=2*pi/360
gnuplot> f(x)=sin(x)
gnuplot> plot "Messung.dat" using ($1*k):(f($2))
```

Bei der Definition von Funktionen in `gnuplot` ist folgendes zu beachten:

- Operatoren
  - Standard-Operatoren: `+ - * /`
  - Potenzen: `2**3`= 2^3
- einige vordefinierte Funktionen

`gnuplot`-Befehl	Formel	Beschreibung
`exp(x)`	e^x	Exponentialfunktion
`log(x)`	$\ln(x)$	natürlicher Logarithmus
`log10(x)`	$\log_{10}(x)$	10er Logarithmus
`sqrt(x)`	$\sqrt{x}$	Wurzelfunktion
`sin(x),cos(x),...`	$\sin(x)$, ...	trigonometrische Funktionen
`abs(z)`	$\|z\|$	Betrag
`arg(z)`	$\phi = \arg(z)$	Argument/Phase
...	...	...

  ... und viele mehr (s. `help-->functions`)

- Konstanten

  Konstanten können mit dem Zuweisungsoperator `=` definiert werden. Die Konstante π ist als `pi` bereits definiert. Die Eulersche Zahl e erhält man über `e=exp(1)`.

Die Ergebnisse der Berechnung können neben der graphischen Darstellung mit `'set table'` in einer (*.dat) Datei abgespeichert und für weitere Berechnungen verwendet werden.

```
gnuplot> set table "Ergebnisse1.dat"
gnuplot> plot "Messung.dat" using ($1*2*pi/360):(sin($2))
gnuplot> unset table
```

1.6 Messfehler - Fehlerbalken

Jede Messung ist üblicherweise mit Messfehlern behaftet. Bei der graphischen Darstellung müssen diese unbedingt berücksichtigt werden. In unserem Beispiel wird eine Messung gezeigt, bei der die Leistung P einer Batterie in Abhängigkeit vom Widerstand R gemessen wurde (s. Abb. 8, oben links). Aufgrund der Messungenauigkeiten weichen die gemessenen Werte von ihrem 'wahren' unbekannten Wert etwas ab. Werden nun einzelne Messpunkte einfach mit einer Zickzacklinie verbunden, so zeigt die Funktion keinen stetigen Verlauf, der an sich zu erwarten wäre. Eine solche Art der Auftragung bei der Darstellung von stetigen Zusammenhängen ist generell ungeeignet.

Messfehler müssen bei jeder Messung abgeschätzt und für jeden Messpunkt ermittelt werden. [4] In diesem Beispiel sind die Messdaten zusammen mit den ermittelten Fehlern für Δx bzw. Δy in einer Datei mit vier Spalten im Format `x, y, xerror, yerror` abgespeichert. Mit dem `using`-Befehl , z. B. `using 1:2:3`, wird die Spalte im Datensatz

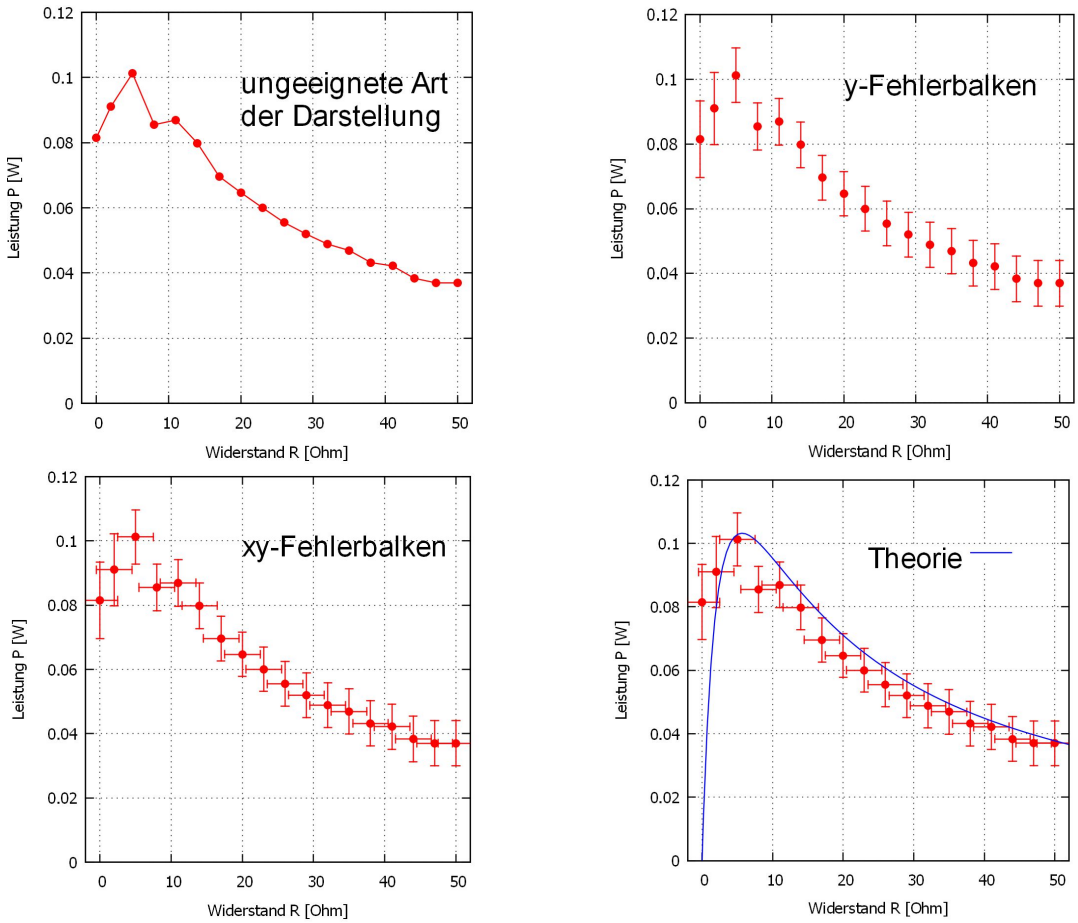


Abbildung 8: Graphische Darstellung einer mit Fehlern behafteten Messreihe: Verwendung der Fehlerbalken und anschließender Vergleich mit Theorie.

[4] s. Anleitung zur Fehlerrechnung: Standardabweichung des Mittelwerts, Fehlerfortpflanzung

ausgewählt, welche die Fehlerwerte enthält. Mit den Befehlen `xerrorbars`, `yerrorbars` bzw. `xyerrorbars` wird die Variable gekennzeichnet, der die Fehlerbalken zugeordnet werden sollen.

Beispiele für die Darstellung der Messdaten mit x-, y, oder xy-Fehlerbalken sind in Abb. 8 oben rechts bzw. unten links zu sehen. Die entsprechenden Befehle sehen wie folgt aus:

```
gnuplot> plot 'Messung1.dat' using 1:2:3 w  xerrorbars      #x-Fehlerbalken
gnuplot> plot 'Messung1.dat' using 1:2:4 w  yerrorbars      #y-Fehlerbalken
gnuplot> plot 'Messung1.dat' using 1:2:3:4 w  xyerrorbars   #xy-Fehlerbalken
```

Ist ein theoretischer Zusammenhang bekannt, der den Kurvenverlauf beschreiben soll, empfiehlt es sich diesen im gleichen Diagramm darzustellen. Dabei kann der Kurvenverlauf durch das Ausführen des `fit`-Befehls ermittelt werden (s. Abschnitt 1.7). In diesem Beispiel wird die Funktion $P(R)$ mit den bereits bekannten Parametern zusammen mit den Messdaten in Abb. 8 unten rechts dargestellt. Die entsprechenden Befehle lauten:

```
gnuplot> P(x) = 1.53**2 * x/(5.67+x)**2
gnuplot> plot 'Messung1.dat' using 1:2:3:4  w  xyerrorbars notitle, \
                             P(x) title 'Theorie' w lines ls 3
```

1.7 Anpassen einer Funktion an experimentelle Messwerte

Beim Durchführen einer Messung ist es oft von Interesse einen funktionellen Zusammenhang, der die Daten theoretisch beschreibt, zu ermitteln bzw. zu überprüfen. Mit Hilfe des `fit`-Befehls kann an eine Messreihe ein physikalisches Modell in Form einer Funktion angepasst werden. Das Ziel der Ausgleichsrechnung liegt darin, die Parameter des Modells zu bestimmen und deren Fehler anzugeben.

Wichtig: Beim Ausführen der `fit`-Funktion ist es entscheidend, zur Beschreibung der Messdaten ein Modell zu wählen, das physikalisch sinnvoll ist.

Es ist falsch, wenn man versucht, an die Messdaten eine beliebige Funktion `f(x)` anzupassen, die durch die Messpunkte gehen könnte (s. Abb. 9).

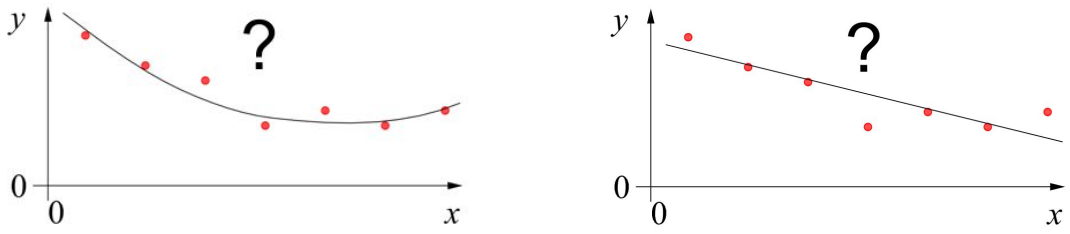


Abbildung 9: Dieselbe Messreihe mit zwei unterschiedlichen Fitlinien.
Ist f(x) = ax^2 + bx + c oder f(x) = ax + b das geeignete Modell?

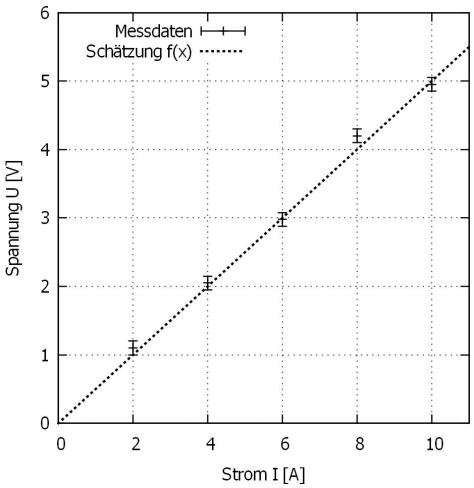
Das Vorgehen beim Anpassen einer Funktion an experimentelle Messwerte soll an einem Beispiel gezeigt werden. Es wurde der Spannungsabfall an einem ohmschen Widerstand als Funktion des Stroms, der durch den Widerstand fließt, gemessen. Die Messungenauigkeit ist für alle Messpunkte gleich und beträgt $\Delta U = 0.1\,\mathrm{V}$.

➔ Machen Sie sich zuerst den physikalischen Zusammenhang klar, der hinter der Messung liegt!

Es ist das Ohmsche Gesetzt $U = I \cdot R$, daher hängt die gemessene Spannung U linear vom Strom I ab. Um dies zu überprüfen, wird beim Fitvorgang eine lineare Funktion der Form `f(x)=a*x+b` verwendet.

➔ Anfangswerte für Fitparameter setzen

Eine erste Abschätzung der Fitparameter erfolgt anhand der aufgetragenen Messdaten. In diesem Fall sind die gesuchten Parameter Steigung der Geraden `a` bzw. der y-Achsenabschnitt `b`. Anhand der Abb. 10 wählt man für die Anfangswerte der Fitparameter `a=0.5` und `b=0`. Diese Werte werden während des Fitvorgangs noch optimiert.




```
f(x)=a*x+b
a=0.5
b=0
plot 'ohm.dat' using 1:2:3 w yerrorbars, \
      f(x)
```

Abbildung 10: Wahl der Anfangswerte für die Fitparameter

Werden in `gnuplot` vor dem Ausführen des `fit`-Befehls keine Anfangswerte für die Fitparameter gesetzt, so wird jedem Parameter automatisch ein Anfangswert von 1.0 zugewiesen. Meistens führt dies ebenfalls zu dem gewünschten Ergebnis. Ein Fit konvergiert jedoch schneller und liefert zuverlässigere Ergebnisse, wenn passende Anfangswerte gesetzt wurden. Anfangswerte müssen in diesem Fall vor dem Ausführen des `fit`-Befehls deklariert werden, wobei pro Zeile jeweils ein Parameter notiert wird.

Mit dem Keyword `#FIXED` hinter einem Parameter können Parameter gekennzeichnet werden, die beim Fitvorgang nicht variiert werden sollen. Beispielsweise:

```
gnuplot> b=0    #FIXED
```

➔ Ausführen des `fit`-Befehls

Mit dem `fit`-Befehl werden die Parameter nach der Methode der kleinsten Quadrate optimiert. Dabei gibt man den Namen der Messdatei sowie den funktionellen Zusammenhang f(x) an, der an die Messdaten angepasst werden soll. Fitparameter `a` und `b` werden mit dem Befehl `via` gekennzeichnet.

```
gnuplot> f(x)=a*x+b
gnuplot> fit f(x) 'ohm.dat' using 1:2:3 via a,b
```

Mit der Angabe `using 1:2:3` wird beim Ausführen des `fit`-Befehls die Messgenauigkeit der y-Messwerte von $\Delta U = 0.1\,\mathrm{V}$ aus der 3.Spalte des Datensatzes berücksichtigt.

Wichtig: `fit`-Befehl funktioniert auch ohne Angaben für die Messungenauigkeit der y-Messwerte (`fit using 1:2`). Jedem Messpunkt wird dann automatisch ein Messfehler von $\Delta y = 1$ zugeordnet. Alle Datenpunkte werden folglich gleich mit dem Wert 1 gewichtet. In diesem Fall wird jedoch die Fehlerangabe bei der Ermittlung der Fitparameter bedeutungslos.

Konvergiert die Anpassung, erhält man in etwa folgende Ausgabe:

```
    After 4 iterations the fit converged.
final sum of squares of residuals : 3.747
rel. change during last iteration : -3.81629e-006

degrees of freedom    (FIT_NDF)                        : 3
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 1.11759
variance of residuals (reduced chisquare) = WSSR/ndf   : 1.249

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a               = 0.4925           +/- 0.01767      (3.588%)
b               = 0.101            +/- 0.1172       (116.1%)


correlation matrix of the fit parameters:

               a      b
a               1.000
b              -0.905  1.000
```

Die nun angepassten Fitparameter der Funktion `f(x)` betragen `a=0.4925` und `b=0.101`. Außerdem wird für beide Parameter ein Fehler von jeweils '`+/- 0.01767`' bzw. '`+/- 0.1172`' angegeben. Diese Werte können verwendet werden, um Fehlergeraden im gleichen Diagramm darzustellen.

Ein anderer wichtiger Parameter, der die Qualität des Fits beschreibt, ist das so genannte 'reduzierte Chi-Quadrat' (`reduced chisquare=WSSR/ndf`). Ein Fit ist ok, wenn dieser Parameter ungefähr den Wert 1 beträgt [5].

Mit dem `plot`-Befehl werden die Messdaten zusammen mit der gefitteten Funktion im gleichen Diagramm dargestellt (schwarze Gerade in Abb. 11).

[5] s. Methoden der statistischen Ausgleichsrechnung

Darstellung der Fehlergeraden

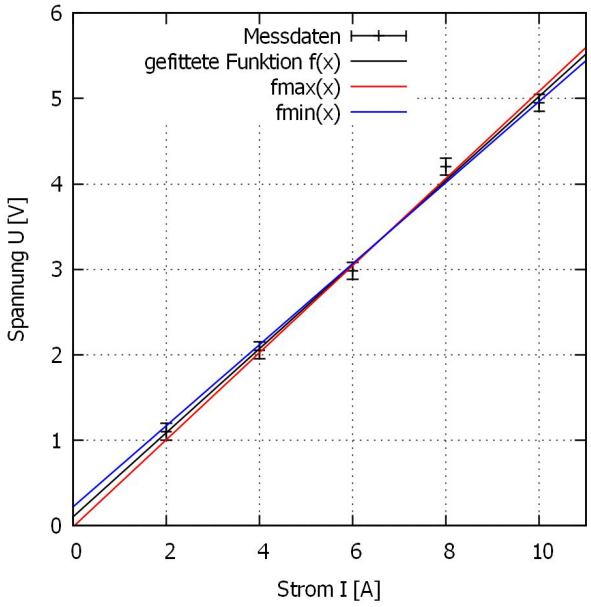
Fehlerwerte, die für die Fitparameter beim Ausführen des `fit`-Befehls ermittelt wurden, können verwendet werden, um Fehlergeraden im gleichen Diagramm einzuzeichnen. Fehlergeraden sind Geraden mit maximaler bzw. minimaler Steigung, die mit den Messwerten noch zu vereinbaren sind. Um diese Fehlerwerte abfragen zu können, muss vor dem `fit`-Befehl das Kommando

```
gnuplot>  set fit errorvariables
```

gesetzt werden. Die entsprechenden Fehlerwerte werden von `gnuplot` mit dem Suffix `_err` versehen. Diese können in Ausdrücke für `fmax(x)` bzw. `fmin(x)`, die jeweils Geraden mit maximaler bzw. minimaler Steigung beschreiben, eingesetzt werden

```
fmax(x)  =  (a+a_err)*x + b-b_err
fmin(x)  =  (a-a_err)*x + b+b_err
```

Mit dem `plot`-Befehl werden nun die Messdaten zusammen mit der linearen Fitfunktion `f(x)` und den ermittelten Fehlergeraden `fmax(x)` bzw. `fmin(x)` in einem Diagramm dargestellt (s. Abb. 11).




```
set fit errorvariables

f(x)=a*x+b
fit f(x) 'Messung.dat' using 1:2 via a,b

fmax(x)=(a+a_err)*x + b-b_err
fmin(x)=(a-a_err)*x + b+b_err

plot 'ohm.dat' using 1:2:3 w yerrorbars,\
f(x), fmax(x), fmin(x)
```

Abbildung 11: Darstellung der Messwerte, der Fitfunktion und der Fehlergeraden mit `gnuplot`.

Anhand der Abb. 11 kann die Qualität des Fits, ausgedrückt durch χ^2_{red}, auch graphisch veranschaulicht werden. 'Reduced chisquare' wurde zuvor zu $\chi^2_{\text{red}} = 1.249 \approx 1$ ermittelt. Wie es für einen gelungenen Fit zu erwarten wäre, liegen sowohl Fit- als auch Fehlergeraden in Grenzen, die mit der Messungenauigkeit der Messwerte zu vereinbaren sind.

Gewichteter Fit

Im vorigen Beispiel ist man von einem Messfehler ausgegangen, der für alle Messwerte konstant ist. Unterschiedliche Messpunkte einer Messreihe werden jedoch oft mit ungleicher Messgenauigkeit bestimmt. Beispielsweise:

- ➔ beim radioaktiven Zerfall (Poisson-Verteilung): $\Delta n = \frac{1}{\sqrt{n}}$
- ➔ Fehler werden mittels Fehlerfortpflanzung ermittelt und es gilt $\Delta y = \Delta y(x)$
- ➔ Messgenauigkeit für individuelle Messungen variiert experimentell bedingt

Die Fehlerangabe Δy für einzelne Messpunkte wird dabei als Standardabweichung s interpretiert und beim Ausführen des Fits dazu benutzt, die Gewichtung w nach $w = 1/s^2$ für jeden Datenpunkt auszurechnen. Dazu muss die entsprechende Fehlerangabe in einer der Spalten des Datensatzes stehen und beim Ausführen des `fit`-Befehls mit `using x:y:s` unbedingt angegeben werden.

Wichtig: Messpunkte mit einer kleineren Messunsicherheit sollen beim Ausführen des `fit`-Befehls stärker gewichtet werden, als solche mit einer großen Unsicherheit.

Angenommen, der zweite Messpunkt aus dem vorigen Beispiel wurde mit einer größeren Abweichung vom 'wahren' Wert und einem Fehler von $\Delta y_2 = 0.7$ gemessen, alle anderen Messpunkte wurden mit einem Fehler von $\Delta y = 0.1$ bestimmt. Wird dies beim Ausführen des `fit`-Befehls nicht berücksichtigt und ein konstanter Fehler für alle Messwerte angenommen, so verändert sich das `fit`-Ergebnis deutlich (s. Abb 12a, rot). Wird dagegen die größere Messunsicherheit von $\Delta y = 0.7$ für den zweiten Messpunkt durch die Angabe

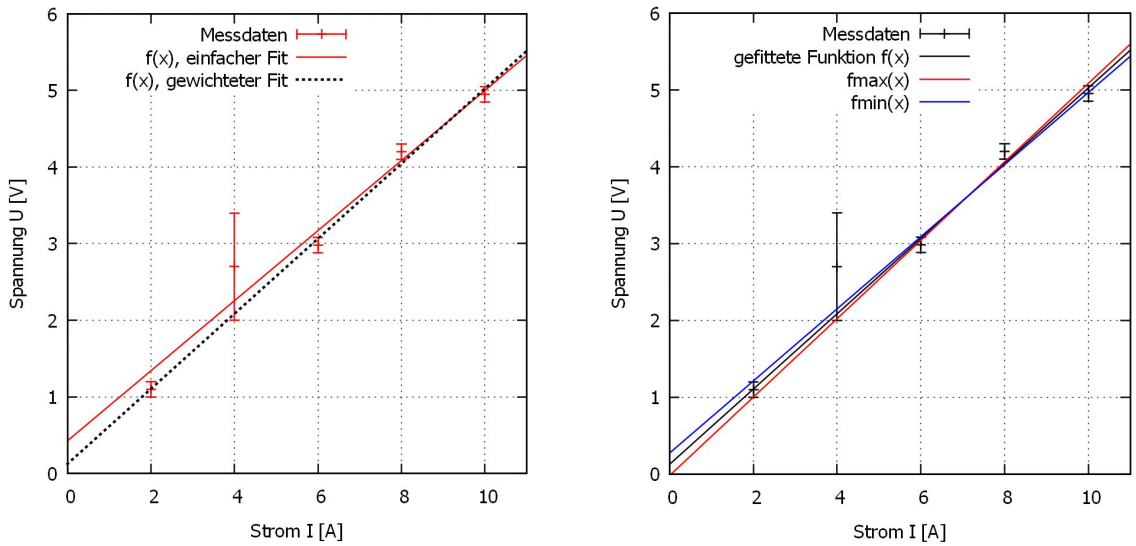


Abbildung 12: (a) Vergleich einfacher (rot) vs gewichteter Fit (gestrichelt) (b) gewichteter Fit mit Fehlergeraden

`using x:y:s` explizit angegeben, so wird dieser Punkt beim Ausführen des `fit`-Befehls weniger stark gewichtet. Das Ergebnis ist in Abb. 12a mit gestrichelter Linie bzw. in Abb. 12b zusammen mit den Fehlergeraden dargestellt.

Hinweis: Ist die Fehlerangabe für alle Messpunkte gleich ($\Delta y_i = \texttt{konst}$), gibt der `fit`-Befehl immer dieselben Abweichungen für Fitparameter `a,b` aus, unabhängig vom Wert, der für Δy angenommen wurde.

1.8 Speichern der erzeugten Diagramme

Für die Verwendung der Graphen in anderen Programmen können diese in die Zwischenablage kopiert und anschließend in einem Graphikprogramm, z. B. Corel Draw, bearbeitet werden. Es besteht auch die Möglichkeit, die Diagramme in einem der gängigen Grafik-Formate - `eps, jpeg, png`, usw. direkt zu exportieren und abzuspeichern. Mit

```
gnuplot> set terminal
```

bekommt man eine Liste der unterstützten Formate und der möglichen Optionen.

Grafiken für wissenschaftliche Publikationen werden üblicherweise, besonders wenn es um Kurvendiagramme handelt, im `postscript` Format abgespeichert (s. Abb. 13a). Solche Vektor-Graphiken können problemlos skaliert und in andere Formate umgewandelt werden. Außerdem erlaubt Gnuplot in diesem Format erweiterte Optionen bei der Beschriftung der Diagramme. Die `enhanced`-Option erlaubt es, griechische Buchstaben oder Formeln darzustellen. Es empfiehlt sich wiederum Raster-Formate wie `png` oder `jpeg` zu verwenden, wenn komplizierte Funktionen, z. B. eine zweidimensionale Intensitäts- oder Temperaturverteilung, dargestellt werden sollen (s. Abb. 13b).

Um ein Diagramm als Grafik zu speichern, muss man zunächst das Ausgabegerät von Gnuplot auf einen Dateityp (`<terminal-type>`) umstellen. Danach wird mit dem Befehl `set output` der Dateiname der mit dem anschließenden `plot`-Befehl erzeugten Datei spezifiziert. Die Ausgabe erfolgt nun nicht mehr auf dem Bildschirm sondern in der Datei:

```
gnuplot> set terminal <terminal-type>
gnuplot> set output 'datei.type'
gnuplot> plot f(x)
```

Wie man eine `postscript`-Datei erzeugt, soll an Hand eines Beispiels gezeigt werden:

```
gnuplot> set terminal postscript eps 25 color solid linewidth 3 enhanced
gnuplot> set output  'graph.eps'
```

Hier bedeutet `eps`, dass ein Encapsulated Postscript `eps`-File erzeugt werden soll. ´25´ gibt die die Zeichengröße für die Beschriftung an. `color solid` implementiert, dass die Kurven mit farbigen durchgezogenen Linien dargestellt werden sollen. `linewidth` steuert

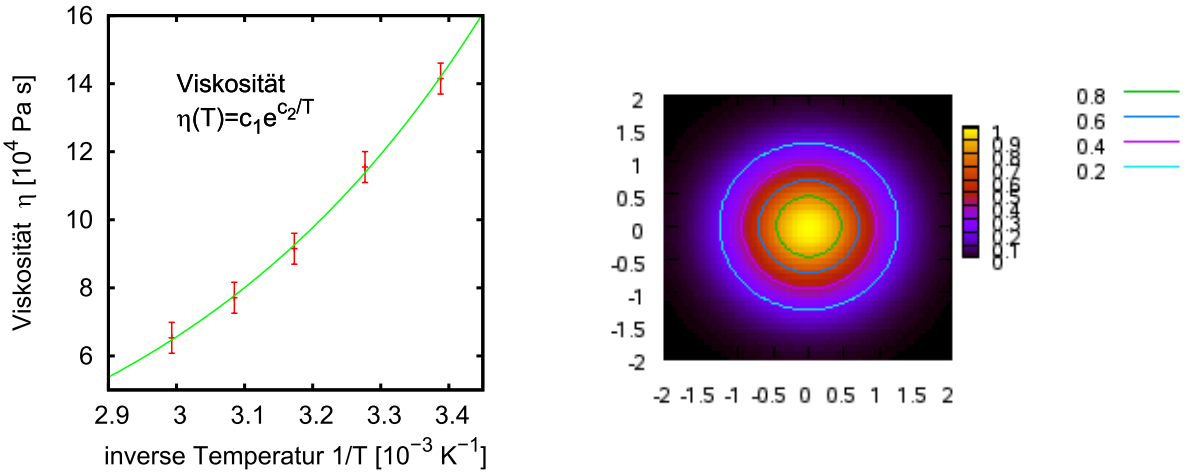


Abbildung 13: (a) `postscript`-Output (`eps`-Datei) (b) `png`-Datei

die Linienbreite der Kurven und Achsen. Mit `enhanced` kommt man in einen erweiterten `postscript`-Modus, in dem man in TeX-artiger Notation griechische Buchstaben und Formeln darstellen kann (für Syntax s. Tabelle):

Operator	Syntax	Ergebnis
griechische Buchstaben	`´{/Symbol h}´`	η
Hochstellen	`´10^{-3}´`	10^{-3}
Tiefstellen	`´F_{Stokes}´`	F_{Stokes}

Hinweise:

- Bei der Verwendung von griechischen Buchstaben erfolgt die Ausgabe auf dem Bildschirm immer unformatiert, d. h. `set xlabel ´{/Symbol a}´` zeigt zunächst auch `/Symbol a` als Achsenbeschriftung an. Erst beim Exportieren des Graphs als `eps`-Datei werden die Formatierungen wirksam und es wird `'α'` als Achsenbeschriftung angezeigt.
- Um Umlaute im `postscript`-Modus darstellen zu können, muss die Kodierung mit `set encoding iso_8859_1` angegeben werden.

Eine `png` bzw. `jpeg` Datei wird mit entsprechenden Befehlen erstellt. Dabei kann die Auflösung der Datei durch Anhängen von `size x,y` eingestellt werden.

```
gnuplot> set term png size 800,600
gnuplot> set output 'graph.png'
gnuplot> plot f(x)
```

1.9 Ausgabe auf Bildschirmdarstellung zurücksetzen

Um die Ausgabe wieder normal auf dem Bildschirm zu erhalten, genügt unter Windows der Befehl

```
gnuplot> set terminal windows
```

Alle Ausgaben erscheinen danach wieder auf dem Bildschirm. In die zuletzt geöffnete Datei wird nicht mehr geschrieben.

1.10 Hilfe

Gnuplot verfügt über ein Hilfesystem, das einfach über die Kommandozeile im Gnuplot Fenster aufgerufen werden kann.

```
gnuplot> help
```

öffnet ein Fenster mit dem Index der Hilfe, sowie der Möglichkeit zur Suche in der Hilfe-Datenbank. Möchte man die Hilfe-Seite eines bestimmten Befehls sehen, so kann man diese auch direkt durch die Eingabe

```
gnuplot> help set
```

oder noch spezifischer etwa durch

```
gnuplot> help set logscale
```

aufrufen.