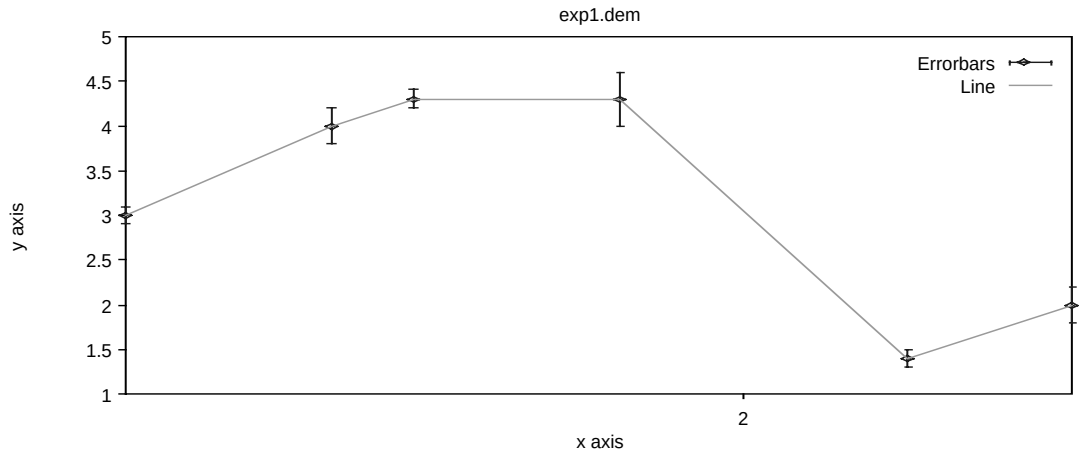
UNIVERSITÄT OSNABRÜCK

Rechenzentrum

Einführung in gnuplot

-

ein interaktives Programm zur grafischen Darstellung von Datenpunkten und math. Kurven und Flächen



Einführung in gnuplot

Universität Osnabrück
- Rechenzentrum -
Frank Elsner (F.Elsner@rz.uni-osnabrueck.de)
Albrechtstraße 28 AVZ
D-49076 Osnabrück

Stand: 11/03/00
Version: 1.3

Inhaltsverzeichnis

1. Einleitung

In diesem Skript wird das *Public Domain* Programm **`gnuplot`** einführend und mit Beispielen behandelt.

`gnuplot` läßt sich folgendermaßen charakterisieren:

> **`gnuplot`** ist ein kommando-orientiertes, interaktives Programm zum Zeichnen von Funktionen und Datenpunkten. Es kann zum Zeichnen von 2- und 3-dimensionalen Daten in verschiedenen Formaten verwendet werden. Es entspricht damit dem Bedürfnis von Wissenschaftlern, Daten visuell ansprechend darzustellen. gnuplot ist Copyrighted, jedoch frei distributierbar, d.h. Sie zahlen keine Lizenzgebühren.

Im einzelnen bietet **`gnuplot`** folgenden Funktionsumfang:

1. Zeichnen 2-dimensionaler Funktionen f(x) in unterschiedlichen Stilarten (Punkte, Linien, Fehlerbalken)
2. Zeichnen 3-dimensionaler Funktionen g(x,y) in unterschiedlichen Stilarten (Kontur, Oberfläche, Hinzufügen von Gittern)
3. Verwenden der mitgelieferten mathematischen Funktionen wie abs(x), sqrt(x) und von einfachen benutzer-definierten Funktionen
4. Verwenden von komplexen Daten {x, y} und Funktionen
5. Darstellen von 2D- und 3D-Daten
6. Einfaches Portieren von gnuplot Kommandodateien (Versionen für diverse Plattformen)
7. Exportieren von Abbildungen in diverse Grafik-Formate wie z.B. Postscript und Drucken auf diverse Nadel-, Tintenstrahl- und Laserdrucker
8. Bereitstellen von Online Hilfe Informationen
9. Hinzufügen von Gestaltungselementen (Titel, Achsenbeschriftungen, Achseneinteilung, Legende, Pfeile, ...)
10. Editieren der Kommandozeile und Zugreifen auf die Kommandohistorie

Die Abbildungen in diesem Skript sind über die Windows Zwischenablage importiert worden und deshalb nicht von optimaler Qualität. Sie können Abbildungen jedoch auch im `Postscript` bzw. `Encapsulated Postscript` Format in hoher Qualität abspeichern und in Textverarbeitungsprogramme wie z.B. **LaTeX** oder **Word für Windows** einbinden.

2. Ein einführendes Beispiel

Laden Sie das Programm über das Kommando **`gnuplot`** bzw. durch Anklicken des entsprechenden Symbols (*Icon*):

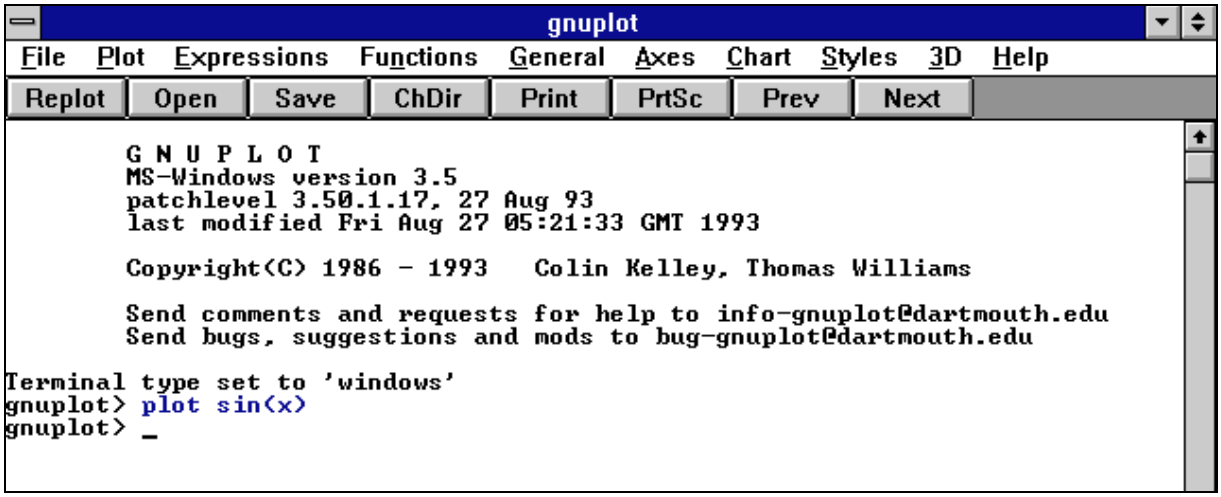
```
$ gnuplot                          # starts gnuplot.
```

`gnuplot` ist ein interaktives Programm. Sie geben Kommandos in der Kommandozeile ein, und **`gnuplot`** führt die gewünschten Berechnungen durch und zeigt ggf. Abbildungen (in der Regel in einem separaten Fenster) an.

Geben Sie in der **`gnuplot`** Kommandozeile folgendes **`plot`** Kommando ein:

```
gnuplot> plot sin(x)               # plots the function sin(x) in the
                                   default range.
```

Die folgende Abbildung zeigt die Benutzeroberfläche von **gnuplot unter Windows**, die eine Kombination von Tastatureingaben in der Kommandozeile und Eingaben über das Menüsystem ermöglicht:



Durch Anklicken eines Menüpunktes wird automatisch der zugehörige Kommandotext in die Kommandozeile geschrieben, so daß Sie die verfügbaren Kommandos nicht

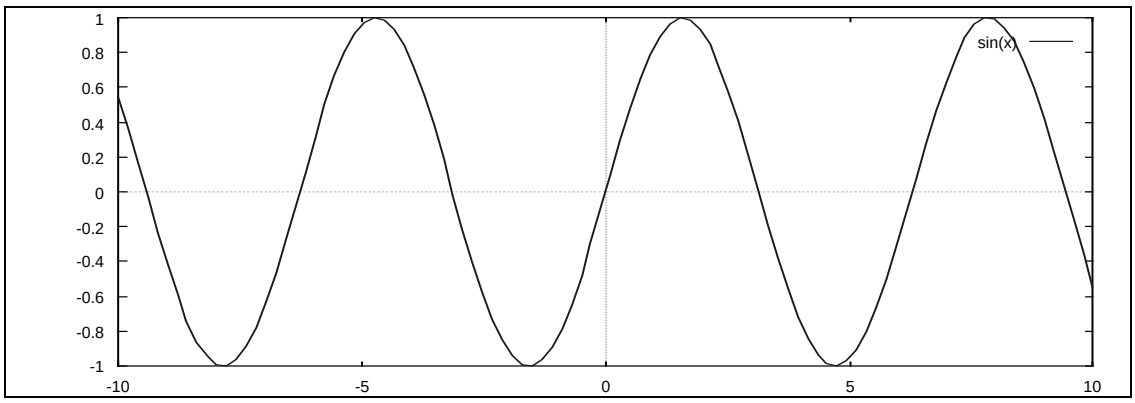
auswendig lernen müssen, sondern vielmehr die entsprechenden Menüpunkte auswählen können.

Das Ergebnis des Beispielkommandos `plot(sin(x))` ist eine grafische Darstellung (*function plot*) der Sinus-Funktion. **`gnuplot`** zeigt die Abbildung in einem separaten Grafik-Fenster an. Sie können die Abbildung im Grafik-Fenster von **gnuplot für Windows** über folgende Menüpunkte in die Zwischenablage kopieren:

Options -> Copy to Clipboard

Sie können nun aus einem anderen Windows Programm auf die Windows Zwischenablage zugreifen (*Copy from Clipboard*) und die Abbildung z.B. in einen Text einfügen.

Die folgende Abbildung der Sinus Funktion ist auf diese Art und Weise erzeugt, in die Zwischenablage kopiert und danach in dieses **Word für Windows** Dokument eingefügt und innerhalb von **Word für Windows** skaliert worden:



Beenden Sie Ihre **`gnuplot`** Sitzung durch Eingabe des **`quit`** Kommandos:

```
gnuplot> quit                          # terminates gnuplot session.
```

3. Darstellen von Punkten, Kurven und Flächen

Die folgende Tabelle enthält die grundlegenden Kommandos von **`gnuplot`** zum Darstellen von Datenpunkten (wie z.B. Meßdaten) aus externen Dateien und mathematischen Funktionen. Die Bezeichnung **`splot`** steht hierbei als Abkürzung für *surface plot*:

Kommando	**Funktion**	**Beispiel**
`plot [a:b] f(x)`	stellt die Funktion *f*(x) im Intervall [a,b] dar.	`plot [-5:5] x**2`
`splot [a:b] [c:d] \` `  h(x,y)`	stellt die Funktion *h*(x,y) im Rechteck [a,b]x[c,d] dar.	`plot [-1:1] [-1:1] x*y`
`plot "datafile"`	stellt x-y-Datenpunkte aus der Datei *datafile* dar.	`plot "exp1.dat"`
`splot "datafile"`	stellt x-y-z-Datenpunkte aus der Datei *datafile* dar.	`plot "exp2.dat"`

`plot "datafile1" \`
`  ...,`
`  "datafile2" \`
`  ..., \`
`  f1(x) \`
`  ...,`
`  f2(x) \`
`  ...`
	stellt in einer Abbildung mehrere Funktionen und/oder Datendateien dar, wobei jede Funktion/Datendatei und die zugehörigen Optionen durch ein Komma voneinander getrennt werden müssen.	`plot [0:5]`
		`  sin(x),`
		`  "yy",`
		`  x`

Diese Grundformen können auf vielfältige Art und Weise variiert werden.

4. Ändern von Voreinstellungen

Sie können die Voreinstellungen von **`gnuplot`** für die Darstellung von Abbildungen über Optionen verändern, die Sie entweder *lokal* für jedes **`plot`** Kommando oder *global* für alle folgenden Kommandos festlegen können. Globale Einstellungen nehmen Sie über das **`set`** Kommando vor. Im folgenden werden Voreinstellungen für die Wertebereiche, den Titeltext und den Stil jeweils nur für ein Kommando geändert:

Option	**Funktion**	**Beispiel**

`plot \`
`  [x1:x2] [y1:y2] \`
`  f(x)`
	legt die Wertebereiche für die x- und y-Achse fest, die berücksichtigt werden sollen.	`plot \`
		`  [0:10] [0:0.5] \`
		`  cos(x)`

`[s]plot ... \` `  title "text"`	legt den Titel fest.	`plot sin(x) \` `  title "Abb. 1"`
`[s]plot ...\` `  with style`	legt den Stil fest, möglich sind u.a. Linie, einzelne Punkte und Impulse.	`plot sin(x) \` `  with impulses`

5. Speichern und Zurückladen von Voreinstellungen

Speichern Sie zunächst folgendermaßen die Voreinstellungen des **`set`** Kommandos für eine spätere Verwendung, verwenden Sie danach eigene Einstellungen und laden Sie die ursprünglichen Einstellungen zurück:

```
save set "defaults.ini"              # saves default settings to file.
set ...                              # uses your settings ...
load "defaults.ini"                  # changes to default settings.
...                                  # uses the default settings ...
```

6. Darstellen von Datenpunkten aus einer oder mehrerer Datendateien

Sie können Daten in kartesischen Koordinaten (siehe aber auch: **`set mapping`**) aus einer externen Datei einlesen und grafisch darstellen. Jede Zeile der externen Datei enthält dabei eine Beobachtung in der Reihenfolge `<Wert für x> <Wert für y>` und ggf. `<Wert für z>`.

Kommando	**Funktion**	**Beispiel**

`plot "datafile1" \`
`  with style1 ,\`
`  "datafile2" \`
`  with style2`
`  ...`
	liest x-y-Daten aus 2 Datendateien ein und stellt sie unter Kontrolle der Stilarten *style1* und *style2* dar.	`plot "xy1" with lines,`
		`  "xy2" with dots`

`splot "datafile" \`
`  using c1:c2:c3 \`
`  with style \`
`  ... \`
	liest x-y-z Daten ein.	`splot "xyz"`

Laden Sie als Beispiel die Kommandodatei `exp1.dem`:

```
load "exp1.dem"                      # loads and executes the commands
                                     from the command file exp1.dem.
```

`# exp1.dem:`	`# name of command file`

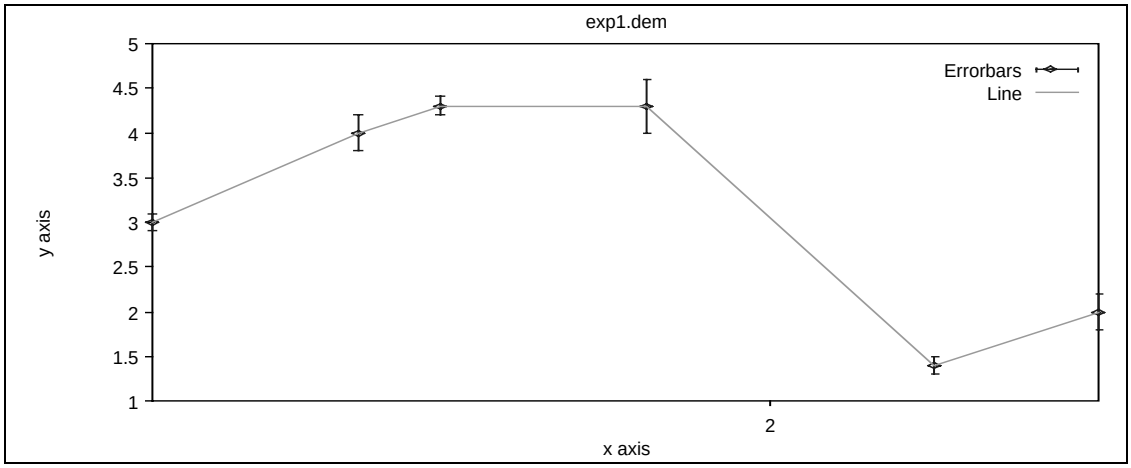
```
set title "exp1.dem"                    # sets some options.
set xlabel "x axis"
set ylabel "y axis"
plot "exp1.dat" \                       # uses data points from datafiles
     title "Errorbars" \
     with errorbars,\
     "exp1.dat" \
     title "Line" with lines

pause -1 "Hit any key or press \        # waits for user input
 OK to continue ..."
```

Die Datei `exp1.dat` enthalte folgende Beobachtungen:

```
# exp1.dat:                             # comment: name of datafile
# sample data                           # comment: short description:
# x, y, delta_y
0.5 3.0 0.1                             # 1. data record
1.0 4.0 0.2                             # 2. data record
1.2 4.3 0.1                             # ...
1.7 4.3 0.3
2.4 1.4 0.1
2.8 2.0 0.2
```

`gnuplot` produziert folgende Abbildung, in der der 3. eingelesene Wert als Fehlertoleranz *delty_y* interpretiert und als Fehlerbalken dargestellt wird:

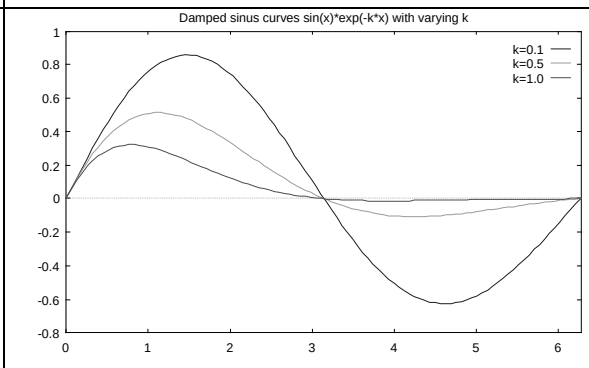


7. Darstellen von einer oder mehreren Funktionen

Laden Sie die Kommandodatei `exp2.dem`:

```
load "exp2.dem"                         # loads and executes the commands
                                        from the command file exp2.dem.
```

# exp2.dem	

# gnuplot command file 	
# damped sinus curves 	
 	
set title "Damped sinus curves \
 sin(x)*exp(-k*x) with varying k"
 	
plot [0:2*pi] sin(x)*exp(-0.1*x)\
  title "k=0.1",\
 sin(x)*exp(-0.5*x) \
  title "k=0.5",\
 sin(x)*exp(-1.0*x) \
  title "k=1.0"
	

8. Definieren von Funktionen

Neben den eingebauten mathematischen Funktionen und Operatoren (*C-like*) können Sie selbstdefinierte Funktionen verwenden. Zur Definition einer Funktion stehen Ihnen die eingebauten mathematischen Standardfunktionen (u.a. ***abs, sin, cos, exp, floor, log, rand, sin, sqrt, tan***) und die aus C bekannten Operatoren (u.a. `*, /, -, +, &&, ||, ?:`) zur Verfügung. **`gnuplot`** unterstützt allerdings mit Ausnahme von `?:` keine Kontrollstrukturen wie z.B. **`if`**, **`while`** o.ä. und keine externen Funktionen aus Funktionsbibliotheken.

Komplexe Zahlen können in geschweiften Klammern in der Form {*real_part*, *imag_part*} wie z.B. `{1,1}` für `1+1*I` eingegeben werden.

Definieren und verwenden Sie z.B. folgendermaßen eine Abstandsfunktion in der Ebene:

```
dist2D(x,y) = \                         # defines dist2D
         sqrt(x**2+y**2)
splot [0:1] [0:1] \                     # plots surface of function dist2D.
         dist2D(u,v)
```

9. Speichern von Abbildungen

Das Ausgabeformat wird über das **`set terminal`** Kommando festgelegt, z.B. sorgt die Einstellung **`set terminal postscript`** für eine Ausgabe im Format Postscript. In diesem Fall sollten Sie die Ausgabe in eine Datei lenken, die Sie auf einem Postscript Drucker ausgeben können.

Speichern Sie den Graphen der Sinus Funktion folgendermaßen in eine Postscript Datei:

```
plot [-2*pi:2*pi] sin(x)              # creates plot in graphics window.
set terminal postscript               # changes output characteristics:
set output "sin.ps"                   # directs output to external file
replot                                # re-plots the last plot to file.
quit
$ lpr -Pps-queue sin.ps               # prints PostScript file to printer
                                      queue.
```

10. Einbinden von Abbildungen in Texten

Die mitgelieferten **Frequently Asked Questions**[1] enthalten Fragen und kurze Antworten, zu diesem Thema z.B. folgende Antwort:

Basically, you save your plot to a file in a format your word processor XYZ can understand (using "set term" and "set output", see above), and then you read in the plot from your word processor.

Details depend on the kind of word processor you use; use "set term" to get a list of available file formats.

Many word processors can use Encapsulated PostScript for graphs. This can be generated by the 'set terminal postscript eps' command. Most MS-DOS word processors understand HPGL (terminal type hpgl).

With TeX, it depends on what you use to print your dvi files. If you use dvips or dvi2ps, you can use Encapsulated PostScript. For emTeX (popular for MS-DOS), you can use emTeX, otherwise use the LaTeX terminal type, which generates a picture environment.

If nothing else helps, try using the pgm or ppm format and converting it to a bitmap format your favourite word processor can understand. An invaluable tool for this is Jef Poskanzer's PBMPLUS package.

In **gnuplot für Windows** können Sie Abbildungen direkt über die Windows Zwischenablage in Texte einfügen (siehe Menüpunkt *Options->Copy to Clipboard* im **gnuplot** Grafikfenster), allerdings sind die Abbildungen nur von mittlerer Qualität.

11. Testen der mitgelieferten Demos

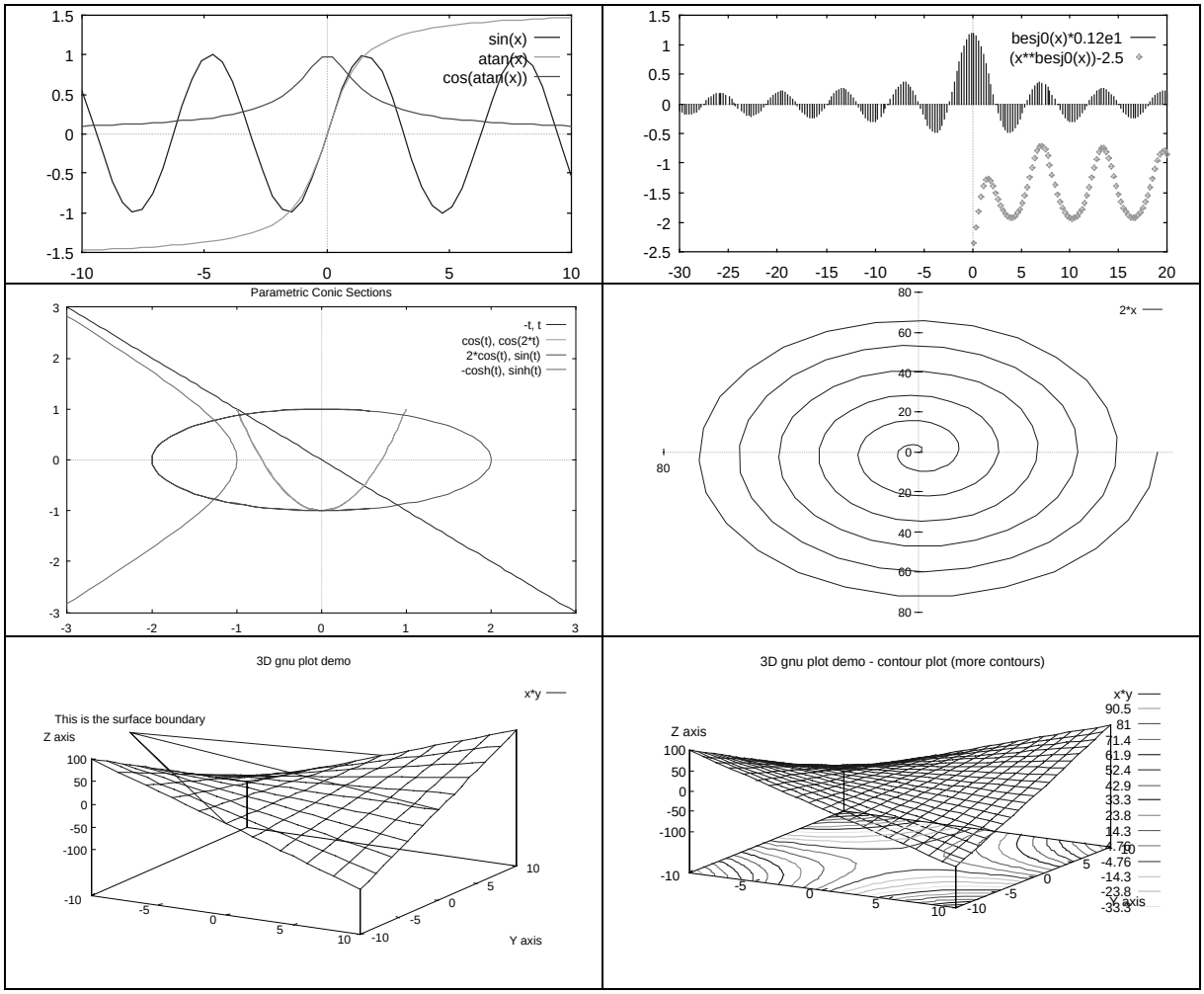
[1] Die FAQs werden wie die Dokumentation maschinenlesbar mit dem eigentlichen Programm gnuplot kostenlos verteilt.

Zum Lieferumfang von **gnuplot** gehört ein umfangreicher Satz an Beispielprogrammen (**gnuplot** Kommando-Dateien).

Laden Sie die mitgelieferten Demonstrationen über das **load** Kommando:

```
chdir 'c:\gp\demo'                    # changes directory.
load 'all.dem                         # loads demo batch file.
```

Die folgende Tabelle zeigt einige der während der Demo erzeugten Abbildungen:



Experimentieren Sie auch mit den mitgelieferten Beispielprogrammen.

12. Verwenden der Hilfefunktion

gnuplot für Windows enthält die gesamte Dokumentation als Text mit Verweisen (*Hypertext*). Wählen Sie den Menüpunkt *Help* in der Menüleiste aus, um in das Hilfesystem zu verzweigen.

In den **FAQs** findet sich folgender Hinweis:

Give the 'help' command at the initial prompt. After that, keep looking through the keywords. Good starting points are 'plot' and 'set'.

Read the manual, if you have it. Ask your colleagues, the system administrator or the person who set up gnuplot.

13. Darstellen von parametrisierten Kurven und Flächen

`gnuplot` ermöglicht neben der Darstellung von Funktionen zusätzlich die Darstellung von 2D-Kurven und 3D-Flächen.(Ein Kreis läßt sich beispielsweise nicht über eine Funktion darstellen.) In diesem Fall lautet die Syntax folgendermaßen:

Kommando	**Funktion**	**Beispiel**

`set parametric`
`plot [t=`*`a:b`*`] `*`x`*`(t),`*`y`*`(t)`
`set noparametric`
	stellt eine Kurve in der Ebene dar, die über t parametrisiert wird.	`set parametric`
		`plot [t=0:2*pi] \`
		`  cos(t), sin(t)`

`set parametric`
`splot [u=`*`a:b`*`] [v=`*`c:d`*`] \`
`  `*`x`*`(t), `*`y`*`(t), `*`z`*`(t)`
	stellt eine Fläche im Raum dar, die über u und v parametrisiert wird.	`set parametric`
		`splot [u=0:1] [v=0:1]\`
		`  u, v, u*v`

`set polar`	interpretiert Funktionen in der Form `r=r(phi)`.	`set polar`
		`r(phi) = exp(-phi)`
		`plot r(phi)`

Sie können im 3D-Fall nur oder zusätzlich Konturlinien erzeugen (siehe **`set contour base`**). Die Anzahl der Konturlinien und weiter Eigenschaften können Sie detailliert über Optionen festlegen.

14. Verwenden von kartesischen, Zylinder- und Kugelkoordinaten

Per Voreinstellung werden Daten in Datendateien als kartesische Koordinaten interpretiert. Sie können jedoch 3D-Daten als Polar- oder Zylinderkoordinaten interpretiert einlesen:

Kommando	**Funktion**	**Beispiel**

`set mapping \`
` {cartesian |`
`  spherical |`
`  cylindrical }`
	wählt kartesische, Kugel- oder Zylinderkoordinaten.	`set mapping \`
		`  cylindrical`

`splot "`*`datafile`*`"`	lädt und stellt Daten dar, die in der vorher festgelegten Darstellung (Koordinaten) vorliegen müssen.	`# x=r*sin(theta)`
		`# y=r*cos(theta)`
		`# z=z`

15. Darstellen von Funktionen mit unterschiedlichen Definitionsbereichen

Für das Zeichnen von 2 oder mehr Funktionen, deren Definitionsbereiche nicht übereinstimmen, können Sie folgende Methode verwenden:

How do I plot two functions in non-overlapping regions?

Use a parametric plot. An example:

```
a=1                                  # x1 ranges from a to b ,
b=3                                  # x2 from c to d
c=2
d=4
x1(t) = a+(b-a)*t                    # Linear transformations:
x2(t) = c+(d-c)*t
f1(x) = sin(x)
f2(x) = x**2/8
set parametric
plot [t=0:1] \                       # f1 : [a,b] -> R
 x1(t), f1(x1(t)) title "f1" \       # f2 : [c,d] -> R
 x2(t), f2(x2(t)) title "f2"
```

16. Literaturhinweise, Quellcode, ausführbare Programme

Die Dokumentation zu **`gnuplot`** ist wie das Programm kostenlos und bereits Bestandteil des Lieferumfangs (Dokumente im `Postscript` Format). Die Dokumentation ist ebenfalls *online* über das **`help`** Kommando verfügbar (speziell bei **gnuplot für Windows** im Windows Help Format).

Sie erhalten **`gnuplot`** u.a. für die Plattformen DOS, Windows und Unix im jeweiligen Rechenzentrum oder per anonymous FTP:

```
Host:              irisa.irisa.fr
Verzeichnis:       pub/gnuplot
```

In diesem Verzeichnis liegen u.a. die C-Quellen und ausführbare Programme für DOS und Windows.

17. Anhang

Voreinstellungen für set:

```
set terminal windows
set output
set noclip points
set clip one
set noclip two
set border
set boxwidth
set dummy x,y
set format x "%g"
set format y "%g"
set format z "%g"
set nogrid
set key
set nolabel
set noarrow
set nologscale
set offsets 0, 0, 0, 0
set nopolar
set angles radians
set noparametric
set view 60, 30, 1, 1
set samples 100, 100
set isosamples 10, 10
set surface
set nocontour
set clabel
set nohidden3d
set cntrparam order 4
set cntrparam linear
set cntrparam levels auto 5
set cntrparam points 5
set size 1,1
set data style points
set function style lines
set xzeroaxis
set yzeroaxis
set tics in
set ticslevel 0.5
set xtics
set ytics
set ztics
set title "" 0,0
set notime
set rrange [-0 : 10]
set trange [-5 : 5]
set urange [-5 : 5]
set vrange [-5 : 5]
set xlabel "" 0,0
set xrange [-10 : 10]
set ylabel "" 0,0
set yrange [-10 : 10]
set zlabel "" 0,0
set zrange [-10 : 10]
set autoscale r
set autoscale t
set autoscale xy
set autoscale z
```

```
set zero 1e-08
```

18. Übungen

1. Starten Sie **`gnuplot`** auf einer beliebigen Plattform.

2. Erzeugen Sie eine Abbildung der Sinus- und Cosinus-Funktion im Intervall von 0 bis 2 Pi.

3. Modifizieren Sie die Abbildung durch Hinzufügen von Titel, Text und Achsenbeschriftungen usw. (**`set title`**, **`set xlabel`**, **`set ylabel`**, **`set label`**, ...).

4. (*) Kopieren Sie die Abbildung in die Windows Zwischenablage oder speichern Sie sie im `latex`, `postscript eps` oder im `aifm` Format ab.

5. (*) Fügen Sie die Abbildung in ein **LaTex** oder **Word für Windows** Dokument ein und drucken Sie das Dokument aus.

6. Erzeugen Sie eine Kommando-Datei `sample1.plt`, die die Schritte 2-3 ausführt.

7. Erzeugen Sie eine Abbildung von Meßwerten mit Fehlerbalken (`... with errorbars`) und legen Sie die Gerade y=3x+4 durch die Datenpunkte.

Beispieldaten:

```
# x, y, d
# d: possible error, true value lies in [y-d,y+d]
0.0  2.7  0.5
1.0  7.0  0.7
2.0 11.4 0.8
3.0 15.1 0.3
4.0 19.6 0.9
5.0 22.4 0.8
```

8. Erzeugen Sie eine Abbildung mit der Betragsfunktion abs(x) und (*) der selbstdefinierten Heaviside-Funktion H(x). H(x) ist 0 für x kleiner oder gleich Null und 1 für x größer Null (Hinweis: Verwenden Sie den *expr1***?***expr2***:***expr3* Operator aus C).

9. Erzeugen Sie eine Abbildung mit sin(x), sin(2x), sin(3x), ..., sin(nx) im Intervall von -Pi bis Pi., n=6. (*) Wieviele unterschiedliche Linientypen/Farben stehen maximal zur Verfügung? Stellen Sie in einer weiteren Abbildung sin(3x) in allen möglichen Stilen dar.

10. Erzeugen Sie eine Kommando-Datei `sample2.plt`, die die Schritte 8-10 ausführt.

11. Erzeugen Sie eine Abbildung der Funktion h(x,y)=exp(-(x**2+y**2)) im Quadrat [0,1]x[0,1]. Fügen Sie 10 Konturlinien hinzu und blenden Sie versteckte Linien aus.

12. Erzeugen Sie einen Kreis unter Verwendung von Polarkoordinaten mit `r=r(phi) = 1` (siehe: **`set polar`**) und (*) eine Kugel unter Verwendung von Kugelkoordinaten mit `r=r(s,t)=1`.

13. Erzeugen Sie eine Kommando-Datei `sample3.plt`, die die Schritte 12-13 ausführt.

14. Überprüfen Sie nach einem Neustart die aktuellen Einstellungen von **`set`** mit **`show all`**. Speichern Sie die aktuellen Einstellungen in eine Datei und drucken Sie diese Datei als Referenz aus.

19. Musterlösungen

```
# Musterloesungen zu den Aufgaben im gnuplot Skript
# Univ. Osnabrueck, Rechenzentrum, F. Elsner, 11.10.94

# Defining constants and saving default settings:
save set "e:\doc\gnuplot\kurs\default.set"
PI = 3.1415

# Plotting with default options:
load "e:\doc\gnuplot\kurs\default.set"
set title ""
plot [x=0:2*PI] sin(x), cos(x)
pause -1

# Adding title, labels and positioning key (legend):
load "e:\doc\gnuplot\kurs\default.set"
set title "Graphen von sin(x) und cos(x)"
set xlabel "x axis"
set ylabel "y axis"
set key 4, 0.5
set label 1 " <- Schnittpunkt Pi/4"  at PI/4.0, sin(PI/4) left
plot [0:2*PI] sin(x), cos(x)
pause -1

# Saving to file, latex format:
set terminal latex
set output "e:\doc\gnuplot\kurs\abb1.tex"
replot

# Saving to file, encapsulated postscript format:
set terminal postscript eps
set output "e:\doc\gnuplot\kurs\abb1.eps"
replot

# Plotting data points x-y-d with errorbars:
load "e:\doc\gnuplot\kurs\default.set"
set title "x-y Messwerte mit Fehlerbalken"
plot "e:\doc\gnuplot\kurs\xyd" with errorbars
pause -1

# Plotting data points x-y-d with errorbars and line y(x):
load "e:\doc\gnuplot\kurs\default.set"
set title "x-y Messwerte mit Fehlerbalken und Gerade"
y(x) = 4.0*x + 3.0
plot "xyd" with errorbars, y(x)
pause -1

# Defining and plotting the Heavyside function H(x) and abs(x):
H(x) = (x > 0) ? 1 : 0
load "e:\doc\gnuplot\kurs\default.set"
```

```
set title "Graphen der Betrags- und der Heavyside-Funktion"
plot [-5:5] H(x), abs(x)
pause -1

# Plotting sin(nx) for n=1,...,12:
load "e:\doc\gnuplot\kurs\default.set"
set title "Graphen von sin(nx), n=1,...,12"
s1(x) = sin(x)
s2(x) = sin(2*x)
s3(x) = sin(3*x)
s4(x) = sin(4*x)
s5(x) = sin(5*x)
s6(x) = sin(6*x)

plot [-PI:PI] s1(x), s2(x), s3(x), s4(x), s5(x), s6(x)
pause -1

# Plotting sin(3x), all possible styles:
load "e:\doc\gnuplot\kurs\default.set"
set title "Graph von sin(3x), unterschiedliche Stile"
set samples 300        # using more nodes for sampling (x) values
plot [-PI:PI] sin(3*x) with lines
pause -1
plot [-PI:PI] sin(3*x) with points
pause -1
plot [-PI:PI] sin(3*x) with linespoints
pause -1
plot [-PI:PI] sin(3*x) with impulses
pause -1
plot [-PI:PI] sin(3*x) with dots
pause -1
plot [-PI:PI] sin(3*x) with steps
pause -1

# Plotting h(x,y):
load "e:\doc\gnuplot\kurs\default.set"
set title "Funktionsgebirge und Konturlinien von exp(-(x**2+y**2))
set hidden3d
set contour base                # possible: base, surface or both
set cntrparam bspline
set cntrparam levels 10
set xtics -1.0, 0.5, 1.0
set ytics -1.0, 0.5, 1.0
set ztics 0.0, 0.2
h(x,y) = exp(-(x**2+y**2))
splot [x=-1:1] [y=-1:1] h(x,y)
pause -1

# Plotting a circle different ways:
load "e:\doc\gnuplot\kurs\default.set"
set title "Darstellung eines Kreises mit Radius r=1, Polarkoordinaten"
set polar            # y=y(x) -> r=r(phi)
set angles radians   # using radians, not degrees
r(t)=1
plot [t=0:2*PI] r(t)
pause -1

load "e:\doc\gnuplot\kurs\default.set"
set title "Darstellung eines Kreises mit Radius r=1, Parametrisierung"
set parametric
x(t)=sin(t)
y(t)=cos(t)
plot [t=0:2*PI] x(t), y(t)
pause -1

# Plotting a sphere using spherical coordinates:
load "e:\doc\gnuplot\kurs\default.set"
set title "Darstellung einer Kugel mit Radius 1, Kugelkoordinaten"
set parametric
r(s,t) = 1                          # A sqhere has constant radius.
x(s,t) = r(s,t) * cos(s) * cos(t)
y(s,t) = r(s,t) * cos(s) * sin(t)
```

```
z(s,t) = r(s,t) * sin(s)
splot [s=-PI/2.0:PI/2.0] [t=0:2*PI] x(s,t), y(s,t), z(s,t)
pause -1
```