Graphiken für LaTeX mit xfig und gnuplot erstellen

Rainer Ruprechtsberger (0056402)

Seminararbeit im Rahmen der Lehrveranstaltung KV Praktisches Arbeiten mit LaTeX
WS 04/05

***Abstract*— LaTeX BenutzerInnen stehen mehrere Packages zur Verfügung mit denen eine Menge verschiedener Graphikformate in LaTeX eingebunden werden können. Als grösseres Problem stellen sich das Setzen der Beschriftung und mathematischer Formeln in den Graphiken dar. Das Vektorgraphikprogramm xfig und das Plot Programm gnuplot unterstützen LaTeXund ermöglichen so eine ebenso elegante wie effiziente Lösung dieses Problems.**

Hier wird neben kurzen Benützungshinweisen zu den Programmen vor allem auf die Integration in LaTeX Projekte eingegangen.

I. Kurze Einführung in xfig

FIG ist ein Text basierendes Vektor Format und wurde ca. 1985 an der Universität von Texas von Supoj Sutanthavibul entwickelt. 1992 wurde das fig Programm von SunView auf X portiert. Mittlerweile gibt es eine ganze Menge Editoren und Konvertierungssoftware für das fig Format (vergleiche [1]). Im folgenden bezieht sich dieser Artikel auf die Weiterentwicklung des ursprünglichen xfig Ports [2] und die damit verbundene Konvertierungssoftware transfig[1].

Wie in der Abbildung 1 zu sehen ist bietet das Interface auf den ersten Blick Funktionen die von einem Vektorgraphik Programm erwartet werden. Dennoch gibt es einige Besonderheiten auf die kurz eingegangen werden sollte.

Das Interface von xfig ist von Grund auf für die Benutzung mit 3-Tasten Mäusen ausgelegt. Z.B. die Zeichen Funktion Polyline setzt mit der linken Maustaste einen ersten Punkt, mit der mittleren wird in den Freihand Modus gewechselt. Anschliessend kann mit der mittleren Maustaste der abschliessende Punkt gesetzt werden oder mit der rechten Maustaste gecanceled, während die linke Taste einen neuen Punkt festsetzt. Die Belegung der Tasten ist vom Kontext abhängig und wird im Interface links oben immer angezeigt (in der Abbildung 1 mit *Mouse Button Funktion* bezeichnet).

Die Optionen der einzelnen Funktionen sind über den ebenfalls Kontext sensiblen Bereich *Optionen der aktuellen Operation* zugänglich. Z.B Strichdicke und Farbe. Wenn diese Optionen einmal gesetzt wurden wird dies als Voreinstellung für die nächste Operation, aber auch als Voreinstellung für verwandte Operatioen benützt. So wird die Stricheinstellung auch für Kreise etc. übernommen.

xfig akzeptiert Tastatureingaben nur für Dialogfelder (wie z.B das Comments Feld in Abbildung 2) die sich direkt unter dem Mauszeiger befinden (Sloppy Focus). Dieses Verhalten wird von xfig direkt implementiert und lässt sich nicht durch die Windowmanager Einstellungen verändern. Irritierend ist dass dies auch für Tastaturkürzel gilt die nur akzeptiert werden wenn sich der Mauszeiger über der Arbeitsfläche befindet.

[1]Der Vollständigkeit wegen sei erwähnt dass es einen Port für M$ Systeme [3] gibt.

II. Integration von xfig und LaTeX

A. Notwendige Anpassungen in xfig

Die Standardeinstellungen von xfig unterstützen das Setzen von LaTeX Text in keiner besonderen Weise. Das kann mit folgenden Optionen geändert werden:

latexfonts — Damit werden nicht mehr PostScript, sondern LaTeXfonts verwendet. Die Fontspezifikationen wie Fettdruck können im Fontselection Dialog (Abbildung 3) ausgewählt werden. Escapesequenzen für z.B. Backslashes werden von xfig wo nötig automatisch gesetzt.

spezialtext — Diese Option schaltet das Maskieren von speziellen Zeichen im Text ab. LaTeX-Befehle in Textobjekten werden damit nach dem Einbinden in ein Dokument interpretiert. xfig selbst bietet kein Interface zum Rendern dieser und verliert damit zum Teil seine Eigenschaft als WYSIWYG Editor (vgl. Abschnitt III).

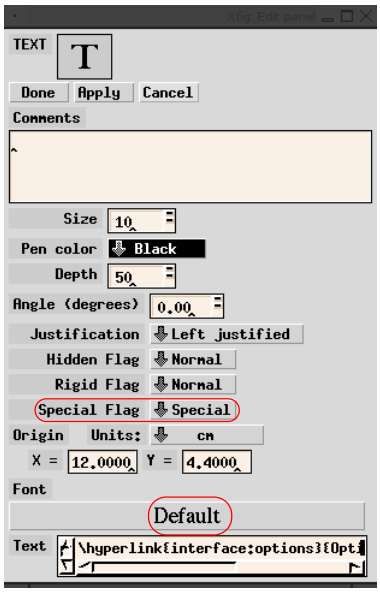

Fig. 2. Fontsettings in xfig

Diese Optionen sind über verschiendene Wege zugänglich

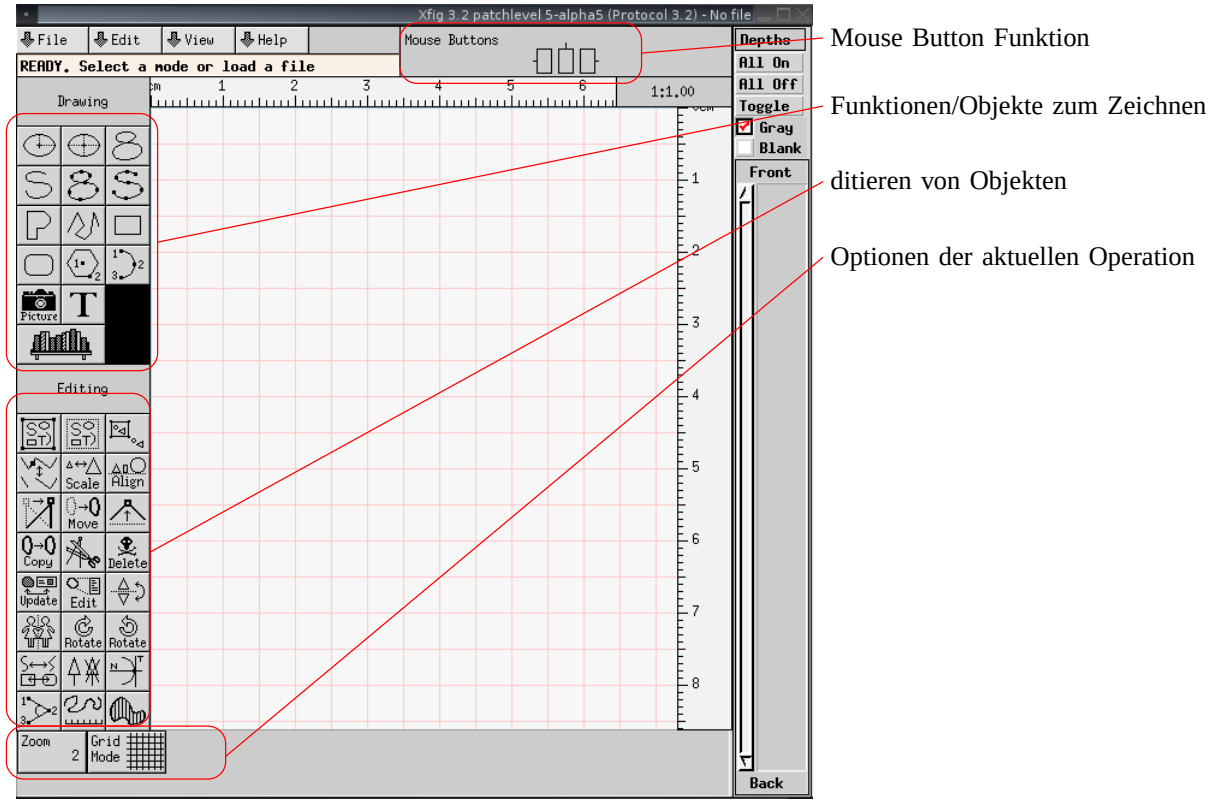

Fig. 1. Userinterface

- Die Optionen (siehe Abbildung 1) von Text Objekten bieten einen Button zu den Textfonts. Dort kann auf *use LaTeX Fonts* umgestellt werden (in Abbildung 3 hervorgehoben). Unter dem Namen *Text Flags* ist auch die specialtext Option zugänglich.
- Der Einstellungen Dialog für Textobjekte (Abbildung 2) bietet die Möglichkeit die entsprechenden Flags gegebenenfalls auch im nachhinein zu setzen.
- `xfig` kann mit den Optionen als Voreinstellung gestartet werden: `xfig -latexfonts -spezialtext`
- Über Xrecources können diese Einstellungen als Default für `xfig` gesetzt werden[2].

Auch wenn specialtext verwendet wird setzt `xfig` die Parameter für die verwendeten Text Teile explizit. Fontfamily, -size und -shape sind davon nur betroffen wenn sie in `xfig` besonders definiert werden, jedoch setzt `xfig`die Grösse der Schrift in Points und nicht mit relativen Grössen. Das sollte entweder beim Konvertieren der Graphik oder direkt in `xfig` an die im jeweiligen LaTeX-Dokument verwendete Basisgrösse angepasst werden.

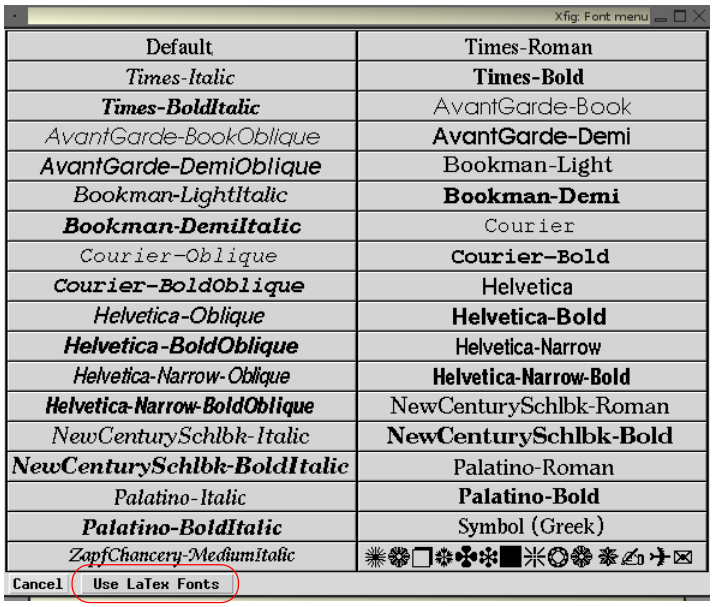

Fig. 3. Fontselection in `xfig`

B. *Export und Konvertierung von figs*

Um fig Graphiken in LaTeXeinbinden zu können werden sie in 2 Teile geteilt: einen LaTeXund entweder einen PostScript oder PDF Teil. Dies wird jedoch auch wenn der Export über das Interface von `xfig` angewählt wird von der `transfig` Software erledigt[3].

Um die 2 Teile aus einem fig File zu generieren sollte zu erst der PostScript Part mit

```
$ fig2dev –L pstex myfigure.fig \
```

[2]Nachdem Name und Pfade der Konfigurationsdatein hier je nach verwendetem System stark variieren wird hier nicht näher darauf eingeganen. Es sei jedoch darauf hingewiesen dass dies der effizienteste Weg ist `xfig` zu konfigurien.

[3]`transfig` wird gemeinsam mit `xfig` entwickelt und ist auch über [2] zu beziehen

```
myfigure.pstex
```

generiert werden. Beim anschliessenden Erstellen des LaTeXTeils muss der PostScript Part angegeben werden[4].

```
$ fig2dev -L pstex_t -p myfigure.pstex \
myfigure.fig myfigure.pstex_t
```

Alternativ dazu kann auch `transfig` verwendet werden um ein Makefile zu erzeugen.

Die Erstellung von PDFs funktioniert analog zu PostScript, der PDF Part muss hier jedoch auf .pdf enden.

Wesentlich für die Entscheidung welche der beiden Varianten gewählt werden soll ist die Art wie das LaTeX-Dokument weiter bearbeitet wird: wenn mit PDFLaTeX ein PDF Dokument erzeugt wird muss auch der Graphik teil in PDF exportiert werden. Hier ist besonders zu beachten, dass auch in `xfig` selbst nur Bildformate importiert werden die in PDF Dokumente eingebunden werden können (z.B. PNG oder JPEG).

C. Einbindung in LaTeX

Um die so erstellten Graphiken auch in LaTeXeinbinden zu können sind 2 Packages notwendig:

- graphicx — denn `xfig` bindet aus dem TeXTeil den anderen mittels `\includegraphics` ein.
- color — wird immer benötigt, auch wenn keine besonderen Farben verwendet werden, da `xfig` alles in rgb Values codiert.

Das Einbinden der Graphik selbst erfolgt mit einem `\input` Statement. Wie das folgende Beispiel zeigt kann dies auch mit if-Statements (siehe auch Anhang I) geschachtelt werden um ein einfaches Wechseln zwischen latex und pdflatex zu ermöglichen ohne den Code umschreiben zu müssen:

```
\begin{figure}[h!]
    \centering
    \ifpdf
        \input{sampleimport.pdftex_t}
    \else
        \input{sampleimport.pstex_t}
    \fi
    \caption{Beispiel: Importierte Grahpik}
    \label{fig:bspfig2}
\end{figure}
```

III. `xfig` Beispiel

Zur Illustration der Verwendung von Formeln in `xfig` ist hier ein kleines Beispiel angegeben. Abbildung 4 zeigt einen Screenshot von `xfig`, Abbildung 5 die importierte Graphik.

IV. `gnuplot` Einführung

Wie der Name impliziert ist `gnuplot` ein Plotting Programm. Funktionen oder Datenreihen können als Graphiken ausgegeben werden. Sowohl 2D als auch 3D Plots können generiert werden. Die Software wurde ursprünglich 1986 von Thomas Williams und Colin Kelley entwickelt und kam nach längerer Pause letzten April mit einer neuen Version 4.0.

[4]Die in Abschnitt II-A erwähnte Einstellung der Fontgrössen kann hier mit einer zusätzlichen Commandlineoption angegeben werden, z.B. `-s 10`

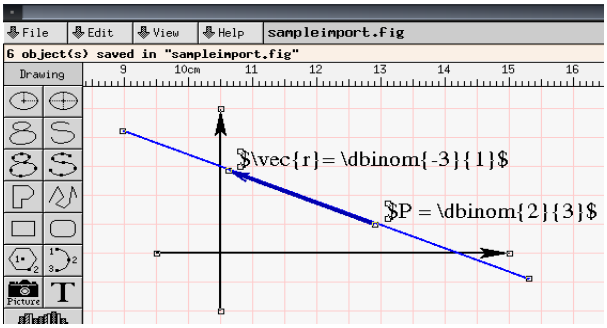

Fig. 4. `xfig` Beispiel: Screenshot

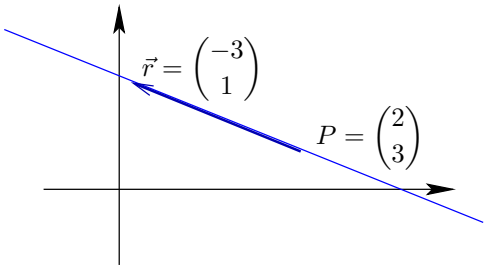

Fig. 5. `xfig` Beispiel: Importierte Grahpik

Anders als der Name impliziert steht dieses Programm nicht unter der GPL, es darf aber ohne Einschränkung verwendet werden. Näheres dazu findet sich auf der Hompage von `gnuplot`[4]. Das Programm ist für eine Reihe von Betriebssystemen verfügbar und kann über SourceForge[5] bezogen werden.

`gnuplot` bietet sowohl einen interaktiven Modus wie die Möglichkeit Batchdatein zu verarbeiten. Da der Syntax der gleiche ist eignet sich der interaktive Modus gut zum Testen der gewählten Einstellungen. Ausserdem bietet er mittels `help` schnellen Zugang zur Hilfe und Befehlsrefenz. Da die einzelnen Parameter eine Fülle von Optionen besitzen die in keinem Tutorial gedeckt werden kann wird hier die ausführliche und gut organisierte Hilfe von `gnuplot` mit Nachdruck empfohlen.

Die grundsätzliche Arbeitsweise mit `gnuplot` ist einfach: ähnlich einer C-shell Syntax werden mit `set` die gewünschten Parameter gesetzt um anschliessend mit `plot` für 2D oder `splot` für 3D Graphen die Ausgabe zu erzeugen. Die wichtigsten Parameter sind dabei

- terminal — bezeichnet hier die Ausgabe und kann beispielsweise X11 für die Bildschirmausgabe unter X oder latex für die Ausgabe als LaTeX-Datei[5].
- output — definiert das File in dem die Ausgabe gespeichert werden soll. Hier ist auf die Angabe von Anführungszeichen zu achten.
- format — bezeichnet die Beschriftung der Achsen.
- *Achse*label — ist eine (optionale) Bezeichnung der Achsen. *Achse* sollte durch die gewünschte Achse ersetzt werden, z.B. *x*

[5]`gnuplot` kann auch fig Files generieren, dies wird hier aber nicht weiter behandelt.

size ist für die Grösse der Ausgabe zuständig.

Mit dem `plot` Befehl kann der Inhalt eines Datenfiles ausgegeben werden. Die Anführungszeichen sind hier wieder unbedingt notwendig.

```
plot range "datafile" optionen
```

Oder wenn eine Funktion geplottet werden soll:

```
plot range function optionen
```

`gnuplot` bietet eine ganze Reihe von Funktionen an, im wesentlichen kann aber auch jeder mathematische Ausdruck wie er in C programmiert werde würde geplottet werden.

Kommentierte Beispiele für die verschiedenen Varianten finden sich im nächsten Abschnitt V.

a) Weitere Tutorials und Einführungen: Das Gnuplot 4.0 Tutorial[6] bietet einen sehr kurz gehaltenen Einstieg, es eignet sich jedoch um einen ersten Überblick zu gewinnen.

Die offizielle Dokumentation zu `gnuplot` deckt sich weitgehend mit der eingebauten Hilfe, es findet sich jedoch eine recht hilfreiche Referencecard auf der Dokumentationsseite[7].

Eine ausgezeichnete Einführung in `gnuplot` wurde in den IBM DeveloperWorks[8] veröffentlicht.

Die University of Northern Iowa hat eine Einführung in `gnuplot`[9] veröffentlicht die sich zwar noch auf die Version 3.7 von `gnuplot` bezieht aber eine kleine Sammlung von wertvollen Tipps enthält.

V. LaTeX Graphiken mit `gnuplot` erzeugen

Das `gnuplot`-LaTeX-Tutorial von David Kotz [10] ist schon etwas älter, bietet aber dennoch eine gute Einführung in die Verwendung von `gnuplot` mit LaTeX. Zwar bezieht es sich auf die obsolete Version 3 von `gnuplot`, sehr grosse Änderungen gab es bezüglich LaTeX, abgesehen vom Umgang mit Backslashes jedoch nicht.

Grundsätzlich ist LaTeXfür `gnuplot` ein Ausgabeformat wie alle anderen auch, hier wird aber eine vollständig in LaTeX-Code formatierte Ausgabe erzeugt.

Die Arbeit mit `gnuplot` wird am besten mit Beispielen erläutert.

A. Beschriftungen in `gnuplot`

```
# gnuplot soll LaTeX-Code ausgeben
set terminal latex
# und in dem File sample1.plot_t speichern
set output "sample1.plot_t"

# die Ausgabe soll verkleinert werden
set size 0.7,1

# Beschriftung der Achsen:
# die $-Zeichen schalten auf den
# displaymath mode um - damit wird die
# Beschriftung der Achsen ''mathematisch''
# gesetzt.
# Die Formatierung der Zahlen mit %3.1f
# und %g wird von gnuplot wie aus der
# C-Funktion printf gewohnt vorgenommen.
set format x "$%g$"
set format y "$%3.1f$"

# zusaetzliche Beschriftung der x-Achse
set xlabel "Zeit (min)"

# und fuer die y-Achse:
# das Drehen der Beschriftung erfodert das
# rotation Package
# backslashes muessen escaped werden
# der einfache \ maskiert den Zeilenumbruch
set ylabel "\\begin{sideways}\
        Temperatur ($\\tccentigrade$)\
    \\end{sideways}"

# das Datenfile sample1.dat soll
# geplottet werden, die Beschriftung
# der Kurve im Index der Graphik wird
# üunterdrckt und die Kurve geglaettet
plot "sample1.dat" notitle smooth unique
```

Listing 1. Beispiel Datenreihe: plot File

Wie im Listing zu sehen ist kann in den Beschriftungen LaTeX-Code wie gewohnt verwendet werden. Nur Backslashes müssen doppelt ausgeführt werden, da sie von `gnuplot` sonst eliminiert werden. Abbildung 6 zeigt das Resultat dieses Batchfiles[6].

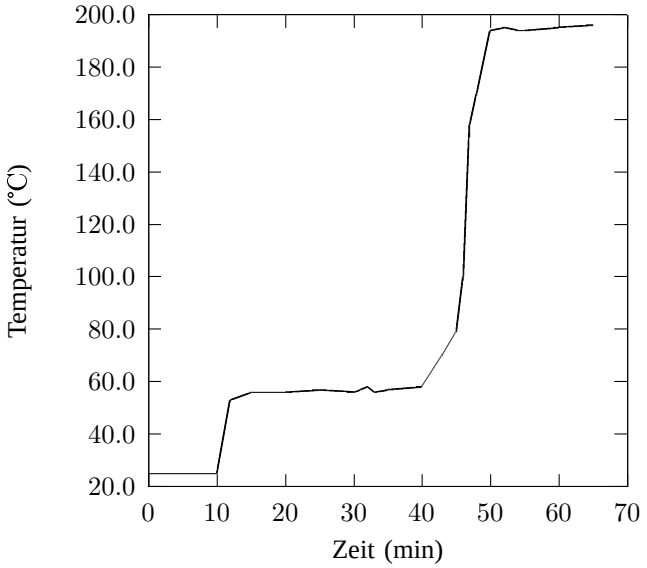

Fig. 6. Beispiel Datenreihe: Ausgabe

Das so generierte LaTeX-File kann mit einem `\input` Statement in das Projekt integriert werden:

```
\begin{figure}[h]
    \centering
    \input{sample1.plot_t}
    \caption{Beispiel Datenreihe: Ausgabe}
    \label{fig:bspplot1}
\end{figure}
```

B. Achsen und Linentypen

Die Art wie `gnuplot`die Achsen zeichnet ist nicht immer die gewünschte, um die x und y-Achse an einen anderen Platz

[6]Der Inhalt des Datenfiles kann Annhang II entnommen werden

zu bringen ist etwas Handarbeit notwendig. Folgendes Listing zeigt dies:

```
# Ausgabe als LaTeX in ein File
set terminal latex
set output "sample2.plot_t"

# skalieren der Graphik
set size 0.7,1

# der Kasten rund um die Graphik wird
# ausgeschalten
set border 0

# Die neuen Achsen werden als Pfeile
# definiert

# zuerst der Stil der Pfeile:
#  ein Stil 1 wird mit Strichtyp 1
#  und schwarz gefuellten Spitzen
#  definiert
set style arrow 1 \
    linetype 1 head back filled

# die neuen Achsen werden unter
# Verwendung dieses Stils gezeichnet
set arrow from 0,0 to 2*pi,0 as 1
set arrow from 0, -1.1 to 0,1.1 as 1

# die Beschriftung der Achsen

# x-Achse: Beschriftung an den
# Achsen (nicht dem Rand)
# wir definieren die einzelnen
# 'tics'
set xtics axis \
    ( "0" 0,\
    "$\\tfrac{\\pi}{2}$" pi/2,\
    "$\\pi$" pi,\
    "$\\tfrac{3\\pi}{2}$" 3*pi/2,\
    "$2\\pi$" 2*pi )

# dazwischen soll jeweils ein
# Strich ausgegeben werden
# (in 2 Teile geteilt)
set mxtics 2
# die Beschriftung
set format x "$%.2f$"

# y-Achse: vom Border auf die
# Achse verschoben und von
# -1 startend im
# Abstand von jeweils 0.2
set ytics axis -1, 0.2
# und beschriftet
set format y "$%g$"

# range von 0-2 pi auf der x-Achse
# und -1.1 bis +1.1 auf der y-Achse
# fuer die Funktionen wird ein
#  Titel angegeben der im Index
#  der Graphik angegeben wird.
# gnuplot gibt mehrere Funktionen
#  automatisch unterschiedliche
#  Strichtypen.
#  tan wird mit einem explizit
#  angegebenen Linienstyle
#  ausgegeben
plot [0:(2*pi)] [-1.1:1.1] \
    sin(x) title "$\\sin(x)$",\
    cos(x) t "$\\cos(x)$" , \
    tan(x) t "$\\tan(x)$" with lines 4
```

Listing 2. Beispiel Funktion: plot File

Abbildung 7 zeigt die Ausgabe dieses Batchfiles.

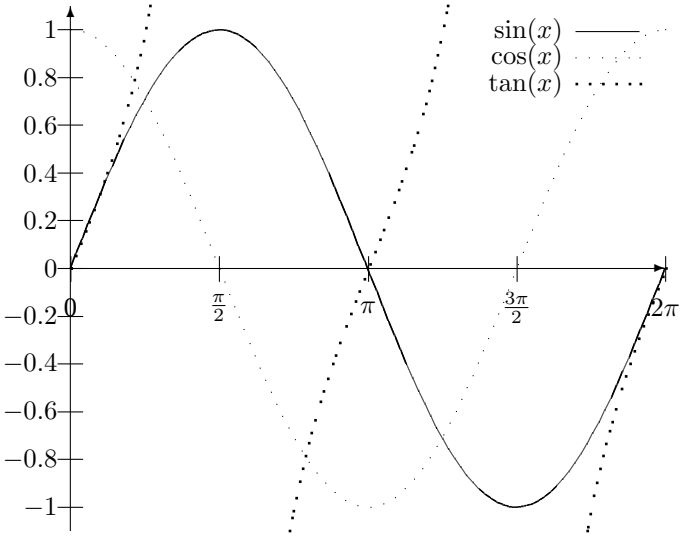

Fig. 7. Beispiel Funktion: Ausgabe

Ein Überblick über die einzelnen Linien und Punkt Varianten die für LaTeX zur Verfügung stehen findet sich im Tutorial[10].

C. Farben mit eps und LaTeX

Das in Version 4 neu hinzugekommene Terminal epslatex ermöglicht auch farbige Ausgaben für LaTeX.

```
# Definition von epslatex
# Die Voreinstellung liefert die
# selben Ergebnisse wie der latex
# Terminal: Farbe und durchgezogene
# Linien ümssen explizit angegeben
# werden. Die standard Groesse der
# Schrift wird ebenfalls spezifiziert
set terminal epslatex color solid 10

# Ausgabename - der LaTeX Teil wird
# automatisch als .tex gespeichert
set output "sample3.eps"

# skalieren
set size 0.7,1

# Beschriftung der Achsen
set format xy "$%g$"

# Definition eigener Funktionen
f(x) = x**2 + 2*x - 4
g(x) = x**3

plot g(x) title "$g(x) = x^3$", \
    f(x) with lines linetype 2 \
    title "$f(x) = x^2 + 2x - 4$"
```

Listing 3. Beispiel epslatex: plot File

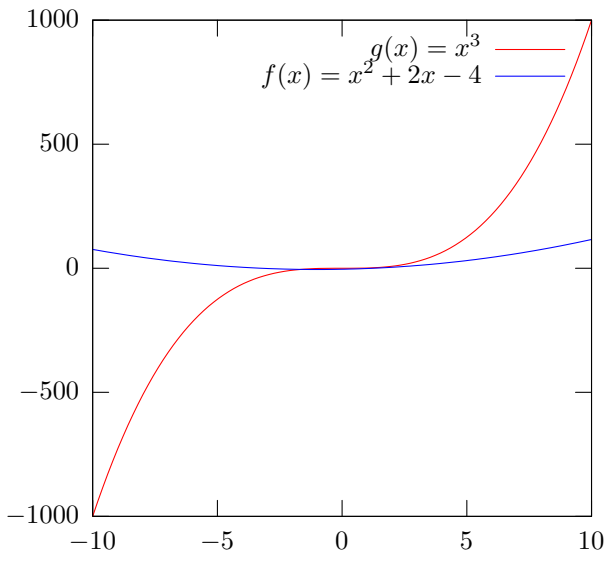

Fig. 8. Beispiel epslatex: Ausgabe

Bei der Verwendung von epslatex Terminals und pdflatex ist allerdings das resultierende eps File noch in ein pdf umzuwandeln, da pdflatex keine eps Graphiken direkt importieren kann. Da `gnuplot` ein `\includegraphics{basename}` Statement verwendet um den 2. Teil einzubinden ist es nicht notewendig den LaTeX-Part umzuschreiben.

VI. Automatisation mit `make`

Makefiles für das Übersetzen von LaTeX-Dokumenten zu schreiben ist im Allgemeinen keine triviale Aufgabe. Die Anzahl der notwendigen LaTeX-runs variiert je nach verwendeten Optionen. Zusatzprogramme wie `makeidx` und `bibtex` sind ebenfalls zu beachten.

Da `make` jedoch ausserordentlich praktisch ist finden sich eine Reihe von Versuchen dieses Problem allgemein zu lösen online. Exemplarisch dafür seien das Makefile von LaTeX Essentials von Lukas Ruf [11] und das „Ultimate Latex Makefile“ von Matti Airas [12] genannt.

Ein sehr interessanter Ansatz findet sich bei Lars Marowsky-Brée [13]. Lars hat ein perl Skript geschrieben dass LaTeX Dokumente auswertet und Abhängigkeiten generiert. Seine Version unterstützt jedoch nicht die Konvertierung von figs in einen PostScript und LaTeXteil, noch `bibtex` oder `gnuplot`. Eine modifizierte Version die zumindest ersteres kann findet sich unter [14], `gnuplot` und `bibtex` Unterstützung sind hier allerdings (noch) nicht integriert. Nach Meinung des Authors eignet sich dieses Makefile allerdings nur eingeschränkt für Projekte die im Source weitergegeben werden, da perl nicht uneingeschränkt vorausgesetzt werden kann.

Als Beispiel für ein komplettes Makefile ist das für die Übersetzung dieses Dokuments verwendete in Anhang III zu finden.

A. Generische Regeln zum Konvertieren von figs

Eine unterschiedliche Namensgebung für PDF und PostScript Varianten der exportierten LaTeX-Teile (und vor allem die Vermeidung der Endung .tex, die in generischen Makefiles Ärger verursachen würde) ermöglicht die Angabe von generischen Regeln zum Konvertieren:

```
%.pstex_t: %.fig
    fig2dev -L pstex $< $*.pstex
    fig2dev -L pstex_t -p $*.pstex $< $@

%.pdftex_t: %.fig
    fig2dev -L pdftex $< $*.pdf
    fig2dev -L pdftex_t -p $*.pdf $< $@
```

Beide Teile werden hier in einem Schritt generiert, Abhängigkeiten sind nach den jeweiligen LaTeX-Teilen zu definieren.

B. `gnuplot` *in Makefiles*

Die auch in [10] vorgeschlagene Benennung der Datein kann bei der Verwendung von make zu Problemen führen, der Autor schlägt daher die Benennung der LaTeX-Datein die von `gnuplot` generiert werden in .plot_t vor. So ist die Integration von `gnuplot` in Makefiles ist relativ einfach zu bewerkstelligen ohne dass `make` versucht andere LaTeX-Dokumente mit `gnuplot` zu erzeugen:

```
%.plot_t: %.plot
    gnuplot $<
```

Wenn das epslatex Terminal verwendet wird ist die Automatisation nicht so einfach zu bewerkstelligen, da sich `gnuplot` nicht davon abbringen lässt den LaTeX-Teil mit .tex zu benennen. Allgemein formulierte Regeln die gegebenenfalls auch die Konvertierung von eps in pdf berücksichtigen führen so leicht zu Schleifen.

Für einzelne Plots kann dies allerdings wie folgt angegeben werden:

```
sample3.tex: sample3.plot
    gnuplot $<
ifdef PDFTEX
    epstopdf $(<:%.plot=%.eps)
endif
```

Appendix I
ifpdf Makro

```
\newif\ifpdf
% we are not running pdflatex
\ifx\pdfoutput\undefined
    \pdffalse
\else % we are running pdflatex
    \pdfoutput=1
    \pdftrue
\fi
```

Listing 4. ifpdf Makro

Appendix II
Datenfile zum `gnuplot` Beispiel

```
0   25
5   25
10  25
12  53
15  56
20  56
```

```
25   57
30   56
31   57
32   58
33   56
35   57
40   58
43   70
45   79
46   100
47   158
48   170
50   194
52   195
54   194
55   194
60   195
65   196
```

Listing 5. sample1.dat

Appendix III
Vollständiges Makefile

Als Beispiel für ein vollständiges Makefile ist hier das für dieses Dokument verwendete inkludiert (Abbildung 9)

References

[1] I. MacPhedran. (2005, Feb.) Fig, xfig and assoziated software. [Online]. Available: http://duke.usask.ca/~macphed/soft/fig/

[2] xfig.org. (2005, Feb.) Xfig drawing program for the x window system. [Online]. Available: http://www.xfig.org/

[3] A. Schmidt. (2005, Feb.) Winfig homepage. [Online]. Available: http://user.cs.tu-berlin.de/~huluvu/WinFIG.htm

[4] G. Team. (2004, Apr.) gnuplot homepage. [Online]. Available: http://www.gnuplot.info/

[5] ——. (2004, Apr.) gnuplot files on sourceforge. [Online]. Available: http://sourceforge.net/project/showfiles.php?group_id=2055

[6] H. P. Gavin. (2004, Nov.) Gnuplot 4.0 - a brief manual and tutorial. [Online]. Available: http://www.duke.edu/~hpgavin/gnuplot.html

[7] G. Team. (2004, Apr.) Gnuplot documentation. [Online]. Available: http://www.gnuplot.info/documentation.html

[8] N. Sastry. (2004, July) Visualize your data with gnuplot. [Online]. Available: http://www-106.ibm.com/developerworks/library/l-gnuplot/

[9] C. of Natural Sciences Computing Laboratories. (1996, Nov.) Introduction to gnuplot. [Online]. Available: http://www.cs.uni.edu/Help/gnuplot/

[10] D. Kotz. (1991, July) Latex and the gnuplot plotting program. [Online]. Available: http://vieta.math.tu-cottbus.de/programme/gnuplot/latextut/latextut.html

[11] L. Ruf. (2004, Oct.) Latex essential - how to create your documentation by using latex. [Online]. Available: http://www.topsy.net/TeX/

[12] M. Airas. (2005, Jan.) The ultimate latex makefile. [Online]. Available: http://www.acoustics.hut.fi/u/mairas/UltimateLatexMakefile/

[13] L. Marowsky-Brée. (2003, Jan.) Latex-make. [Online]. Available: http://lars.marowsky-bree.de/tech/latex-make.html

[14] R. Ruprechtsberger. (2003, Nov.) Latex-make. [Online]. Available: http://rantanplan.org/~rupi/scriptwork/latex_make-0.01.tgz

```
all: xfig-gnuplot-latex.pdf

figs := userinterface.fig fontsettings.fig fontselection.fig sampleimport.fig
plot := sample1.plot_t sample2.plot_t

# comment out if you don't wanna use pdflatex
PDFTEX = 1

ifdef PDFTEX
    texfigs := $(figs:.fig=.pdftex_t)
else
    texfigs := $(figs:.fig=.pstex_t)
endif

# define dependencies here
xfig-gnuplot-latex.pdf: xfig-gnuplot-latex.tex samplescreen.png $(texfigs) \
    $(plot) sample3.tex Makefile

# dependecies of the datafile plot
sample1.plot_t: sample1.plot sample1.dat

# using pdflatex - or don't
ifdef PDFTEX
# this is sorta hack, especially the bibtex part
%.pdf: %.tex
    @ AUXFILE="$(basename $@).aux" ; \
    NEWMD5=`md5sum $$AUXFILE` ; OLDMD5="FOO"; \
    until [ "$$OLDMD5" == "$$NEWMD5" ] ; do \
        OLDMD5="$$NEWMD5" ; \
        pdflatex $< ; \
        bibtex $(<:%.tex=%) ; \
        NEWMD5=`md5sum $$AUXFILE`; \
    done ;
else
%.dvi: %.tex
    @ AUXFILE="$(basename $@).aux" ; \
    NEWMD5=`md5sum $$AUXFILE` ; OLDMD5="FOO"; \
    until [ "$$OLDMD5" == "$$NEWMD5" ] ; do \
        OLDMD5="$$NEWMD5" ; \
        latex $< ; \
        bibtex $(<:%.tex=%) ; \
        NEWMD5=`md5sum $$AUXFILE`; \
    done ; \

%.pdf: %.dvi
    dvips -Ppdf -Pcmz -Pamz $.*.dvi
    ps2pdf14 $*.ps $@
endif

# convert fig files into their ps an tex parts
%.pstex_t: %.fig
    fig2dev -L pstex $< $*.pstex
    fig2dev -L pstex_t -p $*.pstex $< $@
# same thing if we use pdflatex
%.pdftex_t: %.fig
    fig2dev -L pdftex $< $*.pdf
    fig2dev -L pdftex_t -p $*.pdf $< $@

# plotting in a latex ''terminal''
%.plot_t: %.plot
    gnuplot $<
# epslatex plot
sample3.tex: sample3.plot
    gnuplot $<
ifdef PDFTEX
    epstopdf $(<:%.plot=%.eps)
endif

# cleanup targets
.PHONY: clean distclean
clean:
    -rm -f *.bak *~ *.aux *.log *.dvi *.pstex* *.pdftex* *.ps *.out *.blg \
        *.bbl $(figs:.fig=.pdf) *.plot_t sample3.pdf sample3.eps sample3.tex
distclean: clean
    -rm -f xfig-gnuplot-latex.pdf
```

Listing 6. Makefile

Fig. 9. Vollständiges Makefile