

Nice plots in PDF-based \TeX -engines such as \XeTeX and \pdfTeX using Octave, gnuplot and Xfig

Hans Kunkell

December 4, 2008

Contents		3 gnuplot examples	4
1 Simple Way for LyX Users	1	3.1 A first plot	4
2 More Advanced Plots	2	3.2 Terminal Test	5
2.1 Octave	2	3.3 Various plots	6
2.2 gnuplot	2	3.4 Histograms	12
2.3 Xfig	3	3.5 Multiplots	13
2.4 \XeTeX / \pdfTeX	3	A Script: <i>Figconverter</i>	15
2.5 Summarize	3	B Script: gnuplot \Rightarrow Xfig \Rightarrow PDF+\TeX	16

Introduction

We all come to situations when we want to insert graphs and plots into our documents. Of course it is possible to save the graph as an ordinary image and insert, but the result is for sure not worthy a real \LaTeX -user.

Until now I've been using \XeTeX and the reason for that is the ability to easily produce multilingual texts as well as the output to PDF. However, it took some time until I figured out how to best produce figures and graphs for \XeTeX . But now I know, and soon you will also know!

The aim of this document is to explain one way of including nice figures into \XeTeX -documents. Throughout this document I refer to \XeTeX but it will also work for other typesetting engines producing PDFs, such as \pdfTeX . First I will describe the way from Octave to typesetting and finally some examples are shown, which I use myself as quick reference.

This is not a beginners guide in Octave nor gnuplot. I expect you already know how to use Octave (or Matlab). Regarding gnuplot the first example is easy but the rest is not explained in *that* full detail. I recommend you having a look at the *Gnuplot \LaTeX Tutorial* if this is the first time and then the great *gnuplot manual*. They are both available at <http://www.gnuplot.info>. I have learnt and borrowed lots from those great sources!

As final words before we begin, I hope this guide will be useful along your way to beautiful typesetting!

1 Simple Way for LyX Users

For users of LyX (<http://www.lyx.org>) the steps are very easy, if you feel you don't need the advanced plotting functions of gnuplot. You can create a `fig`-file directly and use in LyX. To save current figure in Octave as `fig`, type in

```
print -dfig output.fig
```

to get the graph into the file `output.fig`. Add `-textspecial` before the filename to make text (labels, legends and titles etc) "special" so they will be typeset by \TeX . You may also specify other options such as `-mono` or `-color mono/color`, `-solid` or `-dashed` for line type, `-portrait` or `-landscape` for orientation.

After you have saved your `fig`-file, you just have to choose "External Material" in the menus of LyX (Insert > File > External Material) and select "Template: XFig" and your file. That's it!

2 More Advanced Plots

If you are not using L_yX or you have to create more advanced plots, this section describes the steps from Octave all the way down to typesetting the PDF.

2.1 Octave

First perform all numerical calculations in Octave (or Matlab). If you are making an ordinary 2D-plot we need one "y-value" for each "x-value". gnuplot wants the data in columns where each line corresponds to one point, like this:

```
100.000000    22.906142
125.000000    24.844343
160.000000    26.988542
200.000000    28.926742
250.000000    30.864943
315.000000    32.872353
```

Suppose we have two column vectors **X** and **Y** with our precious data. To save in a gnuplot-friendly format, type

```
A = [X Y];
save -ascii "xxx.dat" A
```

to save the matrix **A** in ascii format to the file `xxx.dat`. It is important to create the matrix **A** since Octave otherwise will save the two vectors in one column after each other – not good!

You can of course save more variables than two, and that will actually become necessary when making more advanced plots.

2.2 gnuplot

gnuplot is actually used by Octave but I find its use quite limited. Especially it has problems with typesetting equations. gnuplot can create L^AT_EX pictures directly or EPS pictures for insertion into documents, but to use together with pdf_TE_X or X_FT_EX the result is not that nice. But going via Xfig produces beautiful graphs, and text is typeset correctly with X_FT_EX.

To produce a `fig`-file we need to tell gnuplot. It can be done in interactive mode but if you want to reuse or you need to change, the best way is to write all commands in a `gnuplot`-file (hereafter named just `gp`-file). I begin the file with

```
set terminal fig monochrome textspecial
set output "xxx.fig"
```

which makes a monochrome output – good for printing – and the `textspecial` command so we can insert L^AT_EX commands like $\sin(x)$ in our figures (but actually we need two backslashes, like $\backslash\sin(x)$, since one backslash will be eaten by gnuplot, the other we save for L^AT_EX). Writing `help fig` in gnuplot's interactive mode gives the following information about the options:

```
set terminal fig {monochrome | color}
                 {landscape | portrait}
                 {small | big | size <xsize> <ysize>}
                 {metric | inches}
                 {pointsmax <max_points>}
                 {solid | dashed}
                 {fontsize <fsize>}
                 {textnormal | {textspecial texthidden textrigid}}
                 {{thickness|linewidth} <units>}
                 {depth <layer>}
                 {version <number>}
```

To get a custom size of the canvas use `size <x> <y>`. The default is inches so change to `metric` if you want to set size in centimeters. Please, refer to the help for more information.

Next line tells gnuplot where to save the file, in this case we choose `xxx.fig`.

This is followed by the commands of what to plot and how to plot it. It can be as easy as

```
plot "xxx.dat"
```

to plot the points in the file "xxx.dat". But usually we would like to add some title etc. See the examples of code and output which is found in section 3.

After writing your gp-file we process it with the command

```
gnuplot xxx.gp
```

It will then run the commands in the file and if the terminal is set to fig as written above, it will be saved as a fig-file.

2.3 Xfig

This step can luckily be done automatically. If we have created the file xxx.fig we just type, in a terminal,

```
fig2dev -L pdftex xxx.fig xxx.pdf
fig2dev -L pdftex_t -p xxx.pdf xxx.fig xxx.pdf_t
```

The command on the first line produces the PDF file without any \LaTeX commands, while the second line produces the text. The filename after `-p` should be the name of the file we want overlaid, including the relative path from your \TeX document to the pdf-file. The `xxx.pdf_t` file contains a picture environment inserting the pdf image file and your text strings that will be processed by $X_{\text{fig}}\TeX$. If you need any special character like `üö` you'll need to edit the file manually since `fig2dev` can only handle ascii. `fig2dev` also has some other options, like setting the font size with `-s 10` for 10pt fonts, but please have a look in the manual.

If any changes (of design, not results, I hope!) is necessary, editing the file in Xfig is also possible. Export as **Combined PDF/Latex (both parts)**. However, if you save your files to a different directory (usually a subdirectory) than your \TeX document, the image won't show up. The reason for that is that the relative path is wrong, and $X_{\text{fig}}\TeX$ can't find it unless you have informed $X_{\text{fig}}\TeX$ of where to find it. It can be done in two ways:

1. Using the command `\graphicspath{{path1/}{pathN/}}`. Then $X_{\text{fig}}\TeX$ will look for the graphics file, ie. PDF-file, in these directories.
2. Setting the path in the `pdf_t`-file. This requires less memory and is faster than the above method, but if you move the files they must be regenerated or edited. But since the `pdf_t`-file is a text file you can edit it in any text editor to make $X_{\text{fig}}\TeX$ find the PDF!

It is actually possible to use both methods, like in this document!

2.4 $X_{\text{fig}}\TeX$ /pdf \TeX

Importing into $X_{\text{fig}}\TeX$ /pdf \TeX is now easy. Just do as you would do with any image, but exchange `\includegraphics{file}` with `\input xxx.pdf_t`, like this:

```
\begin{figure}[h]
  \begin{center}
    \resizebox{14cm}{!}{\input xxx.pdf_t} % For resizing!
    \input xxx.pdf_t
    \caption{A nice figure.}
    \label{fig:xxx}
  \end{center}
\end{figure}
```

It depends on `graphic` (or `graphicx`) package, so include one of those packages in the preamble, as well as the `color` package if you're doing colored plots.

2.5 Summarize

There are quite a few steps to go through. Luckily, I have made a script that will make it easier to generate figures, once the gnuplot file is made. It is found in appendix B, and free to use.

You will soon realize that a lot of files will be created. For each figure at least four files is necessary – `data-file(s)`, `fig`, `pdf`, `pdf_t`. If it is just one figure in the document it won't be so messy, but with many plots it can be a good idea to keep them in separate directories. However, as mentioned above, the `pdf_t`-file must contain the path to the pdf-file or otherwise use of command `\graphicspath` is necessary. It is also covered a bit more in appendix B.

3 gnuplot examples

This section contains some sample plots, including multiplots (known as subplots in Octave) on page 13. Histograms are found on page 12.

3.1 A first plot

Even though the aim is not to be a gnuplot guide, let's start with a basic plot.

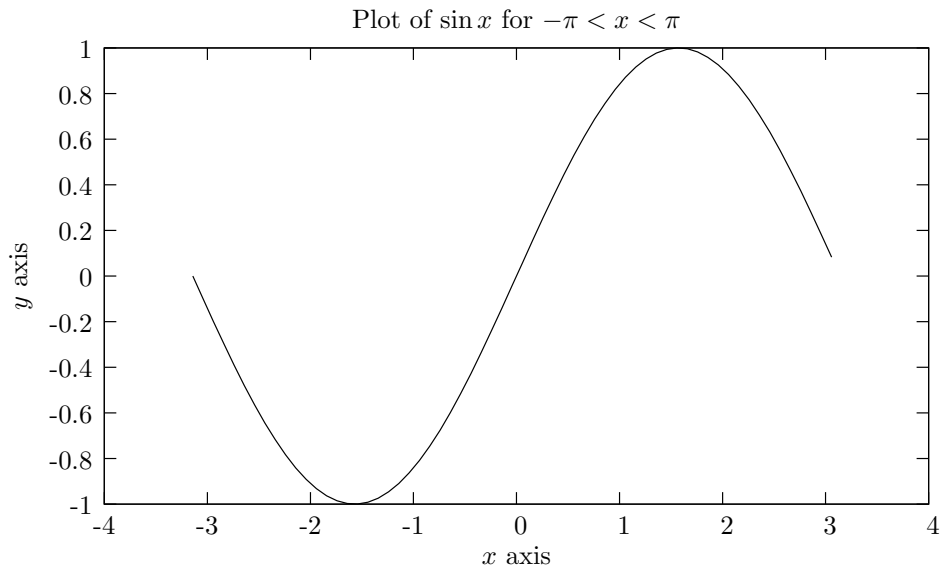


Figure 1: A sine.

The code for figure 1:

```
set terminal fig monochrome textspecial
set output "plot/simplesine.fig"

set xlabel "$x$ axis"
set ylabel "$y$ axis"
set title "Plot of $\sin x$ for  $-\pi < x < \pi$ "

plot "data/simplesine.dat" with lines notitle
```

It doesn't look fantastic but is a simple example for first time viewers. First and second line tells gnuplot to save the plot to the file `plot/simplesine.fig`. We set the labels on the x and y axis with command `set xlabel "text"`. Setting a title works similarly. Since we set `textspecial` on first line, all text is typeset with \LaTeX so anything between `$ $` will be typeset as formula, as expected. As mentioned before, commands for \LaTeX must be double-escaped, ie. use two backslashes like `\\`, since the backslash is an escape-character for gnuplot as well. Then we instruct to plot the data from the file `data/simplesine.dat` with lines. We don't need a key, or legend, so we write `notitle`. Easy peasy!

3.2 Terminal Test

The terminal test output, figure 2 and 3, was achieved by the `test` command. Please note the differences in line style whether or not the output is monochrome or color.

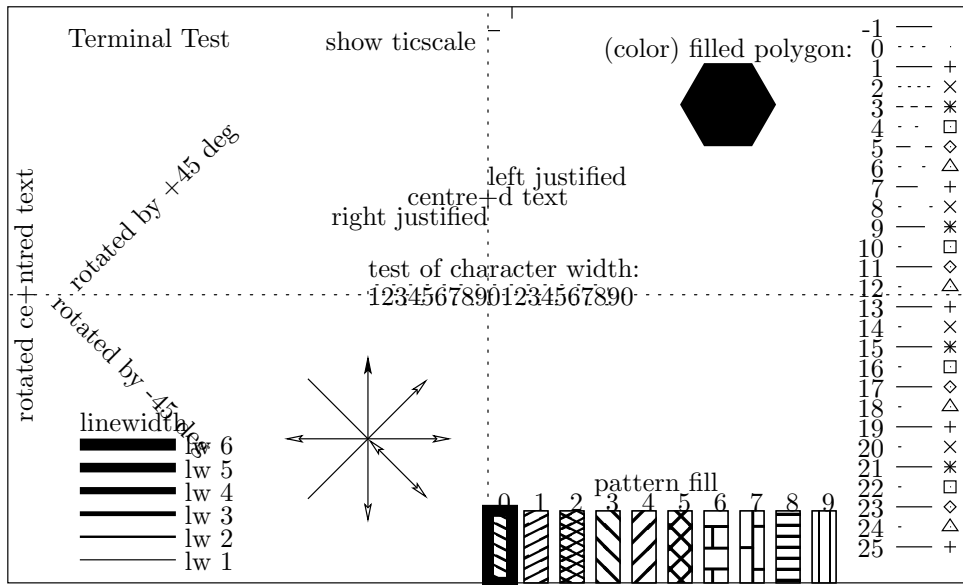


Figure 2: A terminal test showing linestyles and pointstyles, linewidth etc. in monochrome.

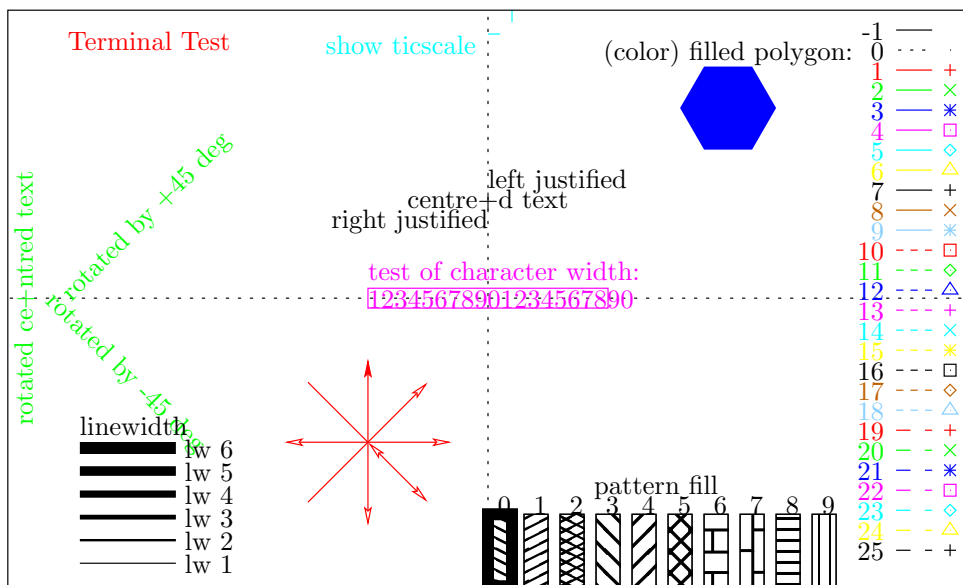


Figure 3: A terminal test showing linestyles and pointstyles, linewidth etc. in color.

3.3 Various plots

The following are some example figures and the gnuplot code required to produce.

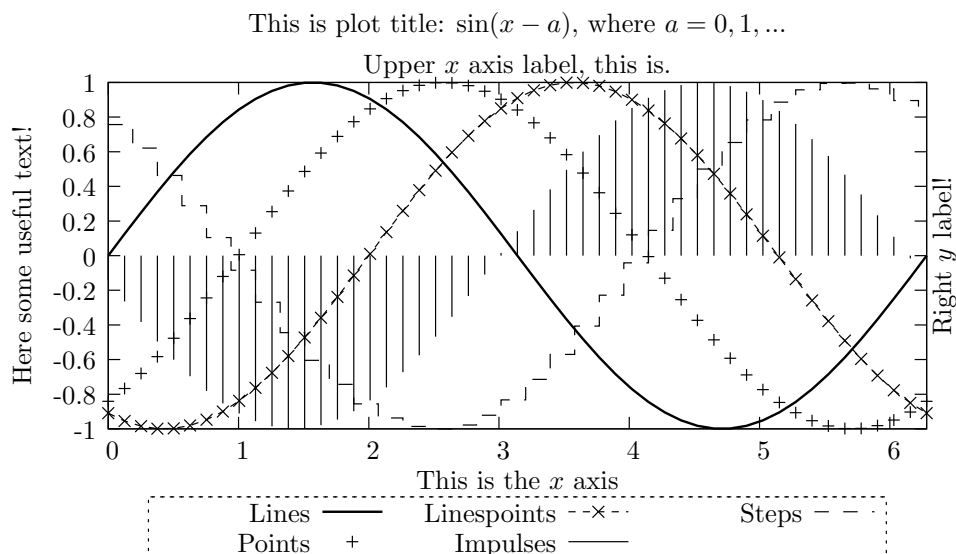


Figure 4: Sines with different styles.

Figure 4 mainly just shows different line types. Code used is:

```
set terminal fig monochrome textspecial
set output "plot/sin.fig"

set title "This is plot title: $ \sin(x-a)$, where $a=0,1,...$"
set xlabel "This is the $x$ axis"
set ylabel "Here some useful text!"
set x2label "Upper $x$ axis label, this is."
set y2label "Right $y$ label!"

# Place legend under graph, horizontal alignment, double spacing between items,
# surround by a box with linestyle 2, increase width of items by 2.5
set key bmargin center horizontal spacing 2 box ls 2 width 2.5

# Fix figure to the last point, ie do not continue to 7.
set autoscale xfixmax

# Plotting every 2nd point from datafile. Select columns with "using", for example
# "using 3:2" to plot column 2 ("y") against column 3 ("x").
plot "data/sin.dat" every 2 using 1:2 title "Lines" with lines linewidth 2, \
"data/sin.dat" every 2 using 1:3 title "Points" with points, \
"data/sin.dat" every 2 using 1:4 title "Linespoints" with linespoints, \
"data/sin.dat" every 2 using 1:5 title "Impulses" with impulses ls 1, \
"data/sin.dat" every 3 using 1:6 title "Steps" with steps
```

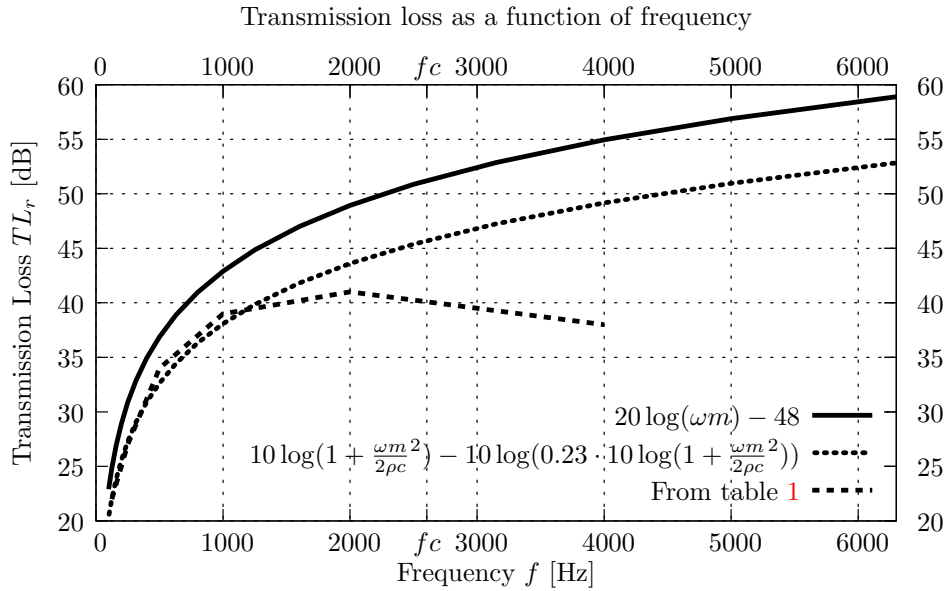


Figure 5: Transmission loss for random incidence, TL_r , for a 4.5 mm thick steel plate, numerically calculated from two different equations and compared to table data. Coincidence frequency, f_c , is marked in figure and is about 2.6 kHz.

Frequency f	[Hz]	125	250	500	1000	2000	4000
Transmission loss TL_r	[dB]	22	27	34	39	41	38

Table 1: Transmission loss TL_r for a 4.5 mm thick steel plate for different frequencies.

The code for figure 5:

```
set terminal fig monochrome textspecial
set output "plot/transloss.fig"

set title "Transmission loss as a function of frequency"
set xlabel "Frequency $f$ [Hz]"
set ylabel "Transmission Loss $TL_r$ [dB]"

# Stops plotting at end of data, not continuing to next xtic or x2tic (x=7000):
set autoscale xfixmax; set autoscale x2fixmax

# This loads the file containing the coincidence frequency, fc, which is around 2600 Hz.
# The file contains just "fc = 2603" or something like that.
load "data/transloss-fc.dat"

# Add a xtic and x2tic item for the coincidence frequency:
set xtics add ("fc" fc); set x2tics add ("fc" fc)
set y2tics # Set tics on right y-axis.

# Place legend at right bottom and increase spacing between items:
set key right bottom spacing 2.5

set grid # enable grid, default settings

# Plotting with lines and specified linewidth (lw) = 3:
plot "data/transloss.dat" using 1:2 title "$20\log(\omega m_i)-48$" with lines lw 3, \
    "data/transloss.dat" using 1:3 title "$10\log(1+\frac{\omega m}{2\rho c})^2)-10\log(0.23\cdot 10\log(1+\frac{\omega m}{2\rho c})^2))$" \
    with lines lw 3, \
    "data/transloss-tab.dat" title "From table \ref{tab:transloss}" with lines lw 3
```

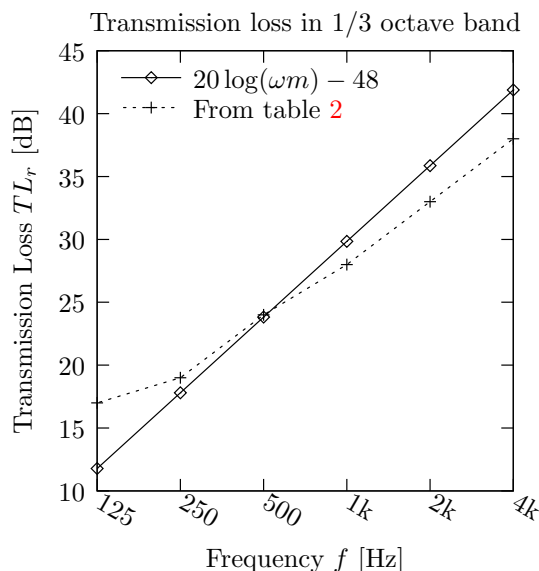


Figure 6: Transmission loss for random incidence, TL_r , for a 1 mm thick steel plate, numerically calculated from $TL_r = 20 \log(\omega m) - 48$ and compared to data in table 2.

Frequency f	[Hz]	125	250	500	1000	2000	4000
Transmission loss TL_r	[dB]	17	19	24	28	33	38

Table 2: Transmission loss TL_r for a 1 mm thick steel plate for different frequencies.

The code for figure 6:

```

set terminal fig monochrome textspecial
set output "plot/transloss2.fig"

# Title and x/y-labels:
set title "Transmission loss in 1/3 octave band"
set xlabel "Frequency  $f$  [Hz]" offset 0,character -1
set ylabel "Transmission Loss  $TL_r$  [dB]"

# Place legend at top left, change place of text and linetype with "reverse",
# align text left with "Left" and increase spacing between items:
set key top left reverse Left spacing 2

# Used to scale the figure:
set size square 1,1

# Logarithmic x-axis, base 10 (default, though)
set logscale x 10

# xtics sets slightly rotated. Define the tics; ("label" pos, "label" pos, ... )
set xtics rotate by -30 ("125" 125, "250" 250, "500" 500, "1k" 1000, "2k" 2000, "4k" 4000)

plot "data/transloss2.dat" using 1:2 title "$20\log(\omega m)-48$" with linespoints 1 5, \
"data/transloss2.dat" using 1:3 title "From table \ref{tab:transloss2}" with linespoints 2 1

```

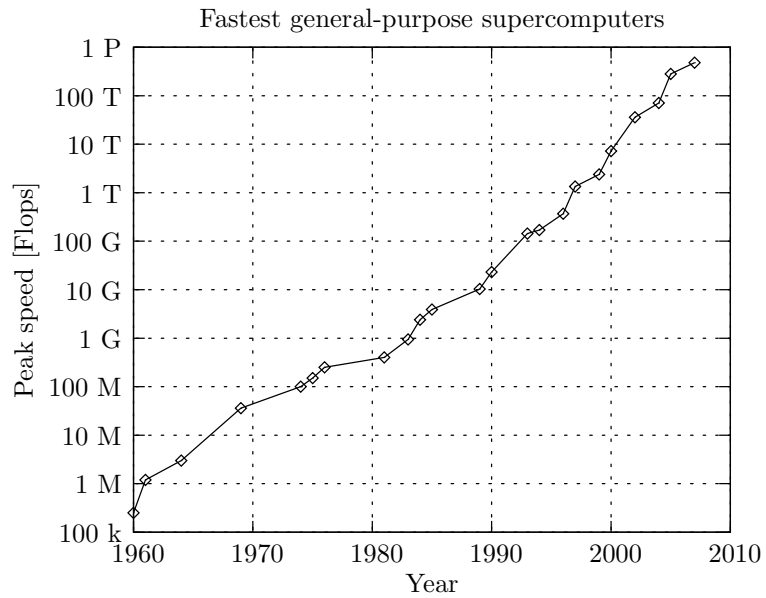


Figure 7: Peak speed for record-holders for fastest general-purpose supercomputers from 1960 til 2007.

The code for figure 7:

```

set terminal fig size 10 8 metric monochrome textspecial
set output "plot/superdator.fig"

set title "Fastest general-purpose supercomputers"
set xlabel "Year"
set ylabel "Peak speed [Flops]" offset character -1

set logscale y 10

set grid

# Remove legend:
unset key

set ytics ("100 k" 100e3, "1 M" 1e6, "10 M" 10e6, "100 M" 100e6, "1 G" 1e9, \
          "10 G" 10e9, "100 G" 100e9, "1 T" 1e12, "10 T" 10e12, "100 T" 100e12, "1 P" 1e15)

set xrange [1960:2010]
set yrange [100e3:1e15]

plot "data/superdator.dat" axes x1y1 with linespoints 1 5

```

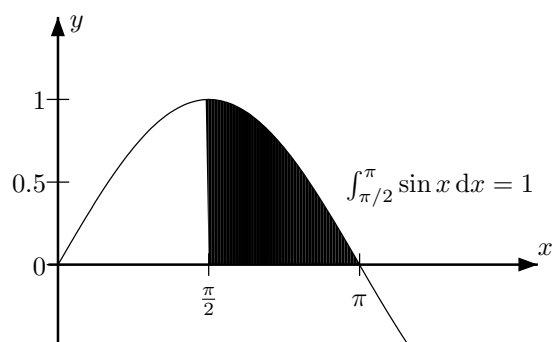


Figure 8: $y = \sin x$ plotted with boundaries $\pi/2$ and π for integration shown.

The code for figure 8:

```

set terminal fig monochrome size 8 5 metric textspecial
set output "plot/integral.fig"

x1 = -0.1
x2 = 5
y1 = -0.5
y2 = 1.5
# Set range to display:
set xrange [x1:x2]
set yrange [y1:y2]

# Set zero axis. Arrow head and label must be done manually I guess:
set zeroaxis lt 1 lw 2
# Arrows from origin to somewhere, with filled head and 1.5 char length, 20 deg
set arrow to x2,0 head size character 1.5,20 filled
set arrow to 0,y2 head size character 1.5,20 filled
# Labels at arrows head, offset 1 character length:
set label "$y$" at 0,y2 offset character 1
set label "$x$" at x2,0 offset 0, character 1

# Remove border
unset border

# Set tics at axis:
set xtics axis ("$\frac{\pi}{2}$" pi/2, "$\pi$" pi)
set ytics axis 0,0.5,1

unset key

set label "$\int_{\pi/2}^{\pi} \sin x \text{ d } x = 1$" at 3,0.5 left

# Must for some reason plot the line again when outputting to fig:
plot "data/integral.dat" using 1:2:3 with filledcurves, \
"data/integral.dat" using 1:2 with lines 1

```

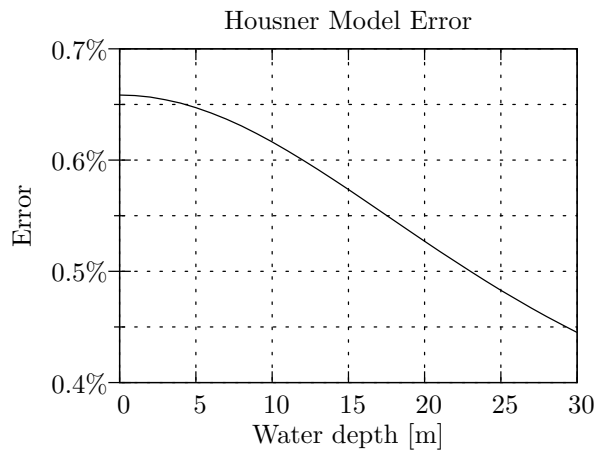


Figure 9: Difference between Housner Model and Fluid Dynamic Model as a function of water depth in tank.

Boring plot (I mean, compared to the other), but the %-sign was a bit tricky until solution was found. Anyway, here is the code for figure 9:

```

set terminal fig size 8 6 metric monochrome textspecial
set output "plot/housner.fig"

# Remove legend:
unset key

# We want a grid for both major and minor ytics and (major) xtics:
set grid ytics mytics xtics

# Set plot range for x and y:
set xrange [0:30]
set yrange [1.004:1.007]

# %-sign in tics must be doubled, like %% to prevent being treated
# by printf as a format changer. See more: gnuplot> help set format. Of course
# double backslashes are needed \\ so it will be transformed into \%
# in the TeX-file. Minor tics, here unlabeled, with an 1 after position.
set ytics axis ("0.7\\%" 1.007, "" 1.0065 1, "0.6\\%" 1.006, \
               "" 1.0055 1, "0.5\\%" 1.005, "" 1.0045 1, "0.4\\%" 1.004)

set title "Housner Model Error"
set xlabel "Water depth [m]"
set ylabel "Error"

plot "data/housner.dat" with lines

```

3.4 Histograms

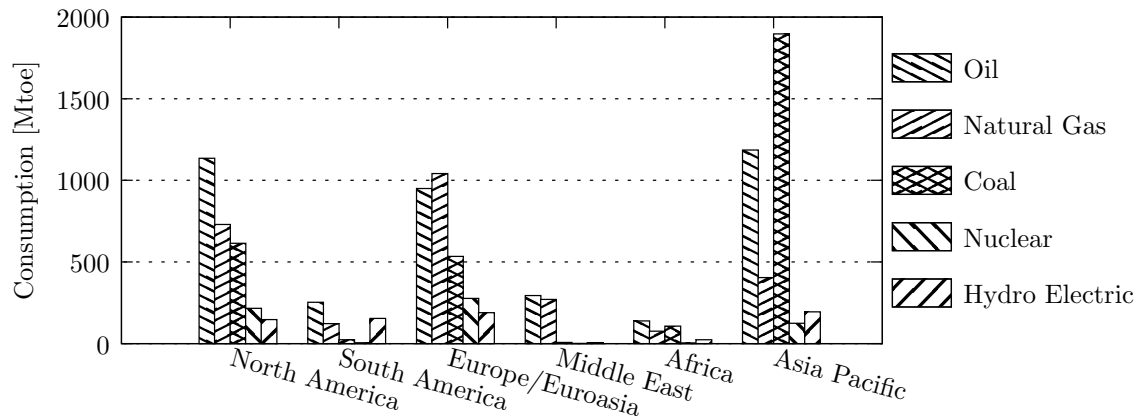


Figure 10: Consumption by fuel, in million tonnes oil equivalent (Mtoe), by region, during 2007. Data from *BP Statistical Review of World Energy June 2008*, <http://www.bp.com/statisticalreview>.

Figure 10 is a basic example of a histogram. Each column in the datafile corresponds to one energy resource and each row is a region. Like this:

	Oil	Gas	...
North America	1134	728	...
South America	251	121	...
⋮	⋮	⋮	⋮

Below is the gnuplot code to create above figure:

```
set terminal fig monochrome textspecial metric size 15 7
set output "plot/energislag07.fig"

# Y-label, x-label not necessary:
set ylabel "Consumption [Mtoe]"
# key in right margin, centered to graph, text Left-aligned,
# description boxes on reverse side, increased spacing
set key rmargin center Left reverse spacing 4

# We want histogram:
set style data histogram
# I think these are the defaults for histogram:
set style histogram clustered gap 2
# Cycle from pattern 0, use line as border:
set style fill pattern 0 border -1
# ytics in steps of 500
set ytics 500
# Set grid on y-axis, not on x:
set grid ytics noxtics nomxtics

# Label our data. We need rotate since text is too long. And then add a small
# offset so the first character doesn't "touch" the graph.
set xtics rotate by -15 ("North America" 0, "South America" 1, "Europe/Euroasia" 2, \
"Middle East" 3, "Africa" 4, "Asia Pacific" 5) offset 0, char -0.25

plot \
  "data/energislag07.dat" using 1 title "Oil", \
  "" using 2 title "Natural Gas", \
  "" using 3 title "Coal", \
  "" using 4 title "Nuclear", \
  "" using 5 title "Hydro Electric"
```

3.5 Multiplots

Multiplots, or subplots as known in Octave, is next!

Multiplotting!

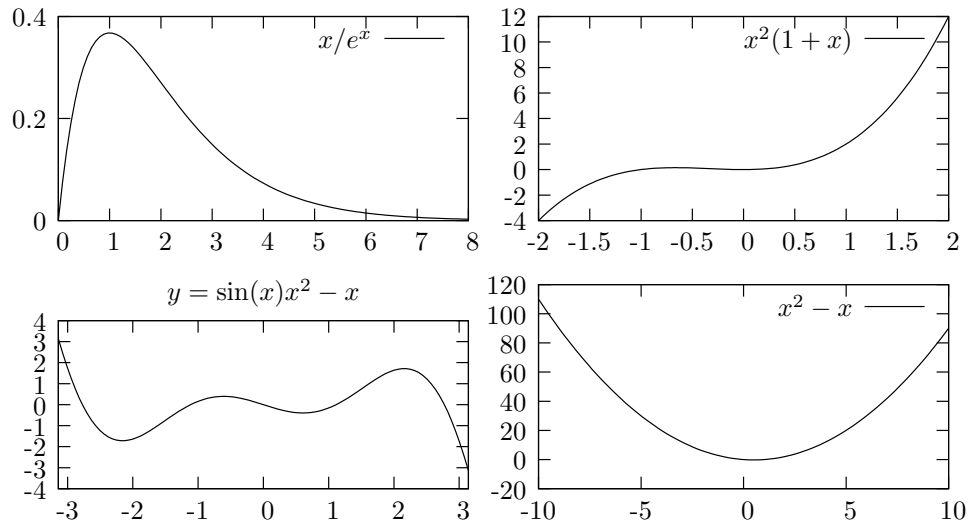


Figure 11: Example of multiplots.

The code for figure 11:

```
set terminal fig monochrome textspecial
set output "plot/multiplot.fig"

# Start multiplot mode with 2 rows, 2 columns.
# They are inserted rowsfirst downwards.
# You can also try columnsfirst and upwards!
set multiplot layout 2,2 rowsfirst downwards title "Multiplotting!"

# Set the left margin so the graphs are lined exact, otherwise they will be
# a little bit tilted due to different ytics:
set lmargin 3
#show margin

set ytics 0,0.2
plot [0:8] x/exp(x) title "$ x / e^x $"
set ytics autofreq # Let gnuplot decide ytics for following plots!

plot [-2:2] x**2*(1+x) title "$ x^2 (1+x) $"

set title "$y=\sin(x) x^2 - x$" # set title for next plot window
plot [-pi:pi] sin(x)*x**2 - x notitle
unset title # remove title, otherwise remain for following plots

plot x*x-x title "$x^2-x$"

unset multiplot

# Range can be given with [x1:x2], otherwise default range [-10:10].
```

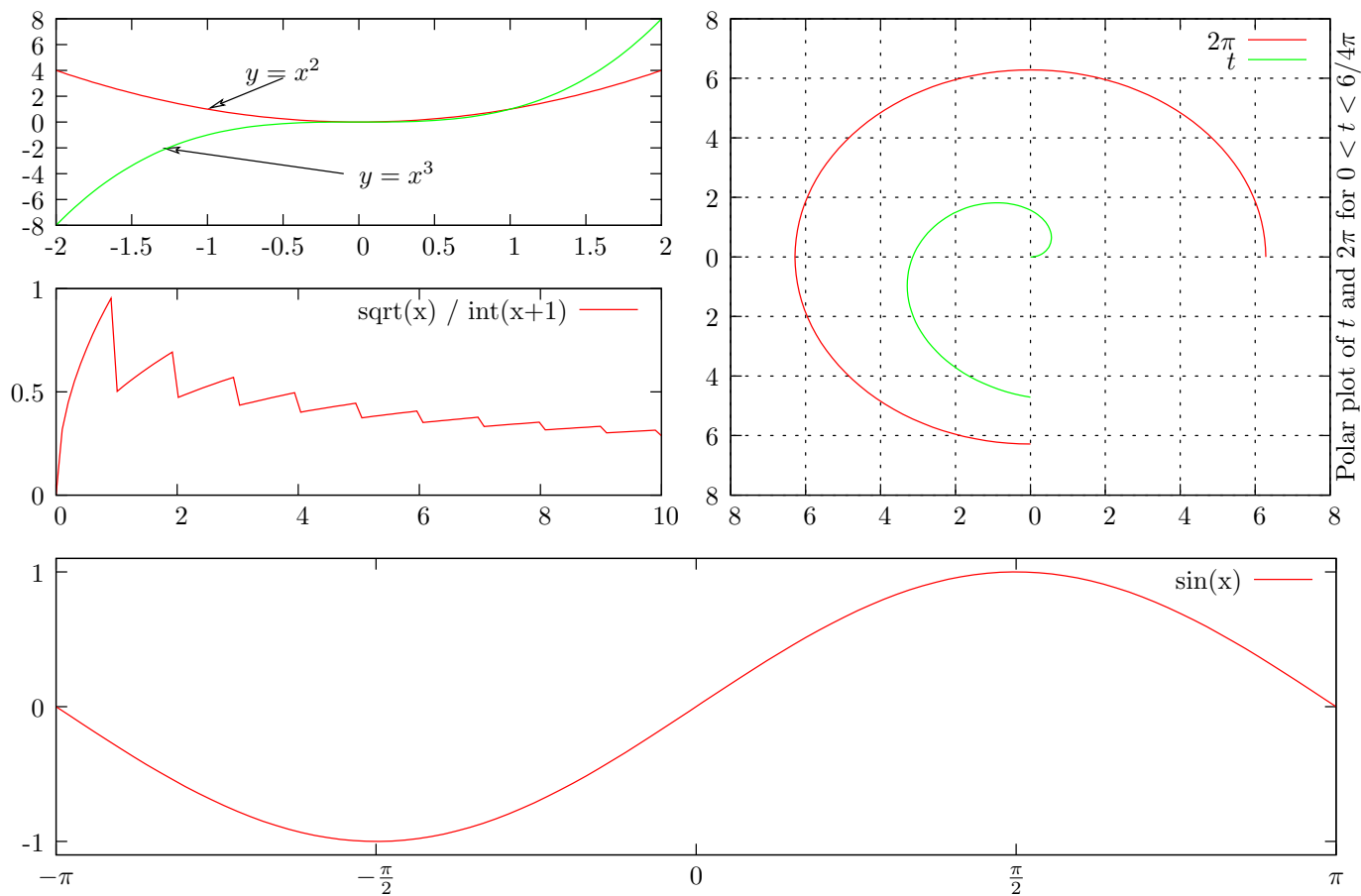


Figure 12: Example of a custom designed multiplot.

The code for figure 12:

```
# Set canvas size with: size <x> <y> in cm when "metric" is given.
# "inches" changes to inches!
set terminal fig color size 18 12 metric textspecial
set output "plot/multiplot2.fig"

set multiplot

# Set left margin to 3 character's width:
set lmargin 3

### Lower plot
# First, define size and where to plot the sine:
set size 1,0.4
set origin 0,0
# Set nice xtics:
set xtics ("-$\\pi$" -pi, "$-\\frac{\\pi}{2}$" -pi/2, "0" 0, \\
          "$\\frac{\\pi}{2}$" pi/2, "$\\pi$" pi)
set ytics -1, 1

plot [-pi:pi] [-1.1:1.1] sin(x)

# Set x- and ytics to default:
set xtics autofreq
set ytics autofreq

### Upper right plot
# Move to another location and set another size:
```

```

set size 0.5,0.6
set origin 0.5,0.4
set y2label "Polar plot of  $t$  and  $2\pi$  for  $0 < t < 6/4\pi$ "
set grid

set polar
plot [0:6*pi/4] 2*pi title " $2\pi$ ", t title " $t$ "

unset polar
unset title
unset grid
unset y2label

### Middle left plot
set size 0.5,0.3
set origin 0,0.4
set ytics 0, 0.5
plot [0:10] sqrt(x) / int(x+1)
set ytics autofreq

### Upper left plot
set size 0.5,0.3
set origin 0,0.7

# Make labels and arrows:
set label 1 " $y=x^2$ " at -0.5,4 center
set arrow 1 from -0.5,3.4 to -1,1
set label 2 " $y=x^3$ " at 0,-4 left
set arrow 2 from -0.1,-4 to -1.3,-2

plot [-2:2] x**2 notitle, x**3 notitle

# Unset labels, arrows to prevent display on (eventual) next plot:
unset label 1; unset label 2
unset arrow 1; unset arrow 2

unset multiplot

```

A Script: *Figconverter*

A simple, all-round script to convert files are given below. Put it in executable path or for example in home directory (it will then be run by, for example, `~/figconverter`). It will then create the output files in the current directory.

```

#!/bin/bash

if [[ $# -eq 0 ]]
then
    echo "Usage: $0 file1 [file2 ... fileN]"
    exit 1
fi

for filename; do
    file=${filename%.fig} # Removes file ending ".fig"
    file_fig="$filename"
    file_pdf="$file.pdf"
    file_pdf_t="$file.pdf_t"

    fig2dev -L pdftex $file_fig $file_pdf
    fig2dev -L pdftex_t -p $file_pdf $file_fig $file_pdf_t

```

done

If you want you may even insert the following line at the end of your `gnuplot` files and the `figs` will be created instantly.

```
system "~/figconverter *.fig"
```

This converts all files so please change if necessary!

B Script: `gnuplot` \Rightarrow `Xfig` \Rightarrow `PDF+TEX`

This is a more advanced Bash script that will save your plots and convert the `fig`-file to `pdf`- and `pdf_t`-files and put them in folder of your choice. You should place it in same directory as your `gp`-files and run it from there.

It has been used in the creation of this document and its settings suppose the `gp`-files are located in a directory called `gp`, directly under the `TEX`-document. It will look for the `fig`-files in the directory `gp/plot` and also save all plots to that directory. This results in typing `\input gp/plot/xxx.pdf_t` to insert a figure.

```
#!/bin/bash

#### Set your paths:
# Path to fig-file relative this script's location, ie where gnuplot saves the fig:
figpath="plot/"
# Path to pdf- and pdf_t-files relative this script's location, where to output pdf*:
pdfpath="plot/"
# Path to pdf- and pdf_t-files relative your TeX document:
texpath="gp/plot/"

#### Settings done!

if [[ $# -eq 0 ]]
then
    echo "Usage: $0 file1 [file2 ... fileN]"
    exit 1
fi

for filename; do
    file=${filename%.gp} # Removes file ending ".gp"
    file_gp="$filename"
    file_fig="$file.fig"
    file_pdf="$file.pdf"
    file_pdf_t="$file.pdf_t"

    gnuplot $file_gp
    fig2dev -L pdftex $figpath$file_fig $pdfpath$file_pdf
    fig2dev -L pdftex_t -p $texpath$file_pdf $figpath$file_fig $pdfpath$file_pdf_t
done
```