LaTeX and the Gnuplot Plotting Program

David Kotz

Principal author of this tutorial for gnuplot 3.0, July 3, 1991

current gnuplot team

Updates of this tutorial for gnuplot 4.0, March 2004
Update of this tutorial for gnuplot 4.2, August 2006
Update of this tutorial for gnuplot 4.4, September 2010
Update of this tutorial for gnuplot 4.6, February 2012

Contents

1 Introduction and History

Gnuplot was originally developed by Colin Kelley and Thomas Williams in 1986 to plot functions and data files on a variety of terminals. In 1988 and 1989 I created an alternate version, known as GnuTeX, that supported a new "terminal type" called `latex`, so gnuplot would output LaTeX code. The plot could then be included in a LaTeX document. I added a number of embellishments, supported only by the `latex` terminal, allowing the user to produce publication-quality plots.

In late 1989 and early 1990 GnuTeX and a number of other gnuplot variants were merged together into a new release of gnuplot, 2.0. This includes, among many other improvements, a LaTeX driver derived from the one in GnuTeX. Anyone interested in using gnuplot with LaTeX should read the next section, a tutorial, and the primary gnuplot manual.

The reader should note that the LaTeX picture environments output by gnuplot can be quite large and complicated, and can easily exceed the memory capacity of TeX. If an enlarged version of TeX is available, it is wise to use it. Otherwise, keep your plots simple and add `\clearpage` to your document where necessary.

There is also an EEPIC driver (`eepic`), intended for use with the EEPIC macro package for LaTeX. EEPIC allows for much more efficient line-drawing, runs through LaTeX faster, and uses less memory. See Section 3 for more information.

Other gnuplot terminal types have been added that take advantage of newer TeX and LaTeX packages and variants, including `emtex`, `metafont`, `epslatex`, `tikz`, and `context`.

2 Using gnuplot for LaTeX: a Tutorial

Gnuplot is by nature an interactive program. Users making plots for LaTeX will generally not use gnuplot interactively. Whenever hard copy is desired from gnuplot, the program need not be run on a graphics terminal. In this case the output is directed to a file or pipe, then sent to the appropriate output device.

We now ignore the interactive nature of gnuplot and provide the input to gnuplot from a file, *i.e.*,

```
gnuplot plotcommands.gp
```

In this example, all of the commands to gnuplot are contained in the file `plotcommands.gp`. Multiple filenames may be supplied to gnuplot this way, read in the order they are given. The output (one or more plots) may be piped to another program or redirected to a file. Usually, however, we direct the output explicitly with an instruction to gnuplot (the `set output "outfile.tex"` command). Gnuplot continues to print error messages to the terminal (stderr). After printing, the output file has to be closed by `set output`, *i.e.* without the file name specification.

Example 1: Here is a first example, producing a plot for this document. The gnuplot input file is given below, and the output appears as Figure 1. The input file defines the output to be in LaTeX, gives a file name for the output, and plots $y = \sin(x)$ for x on $[-\pi, \pi]$. To produce the figure, I simply `\input{eg1}` in a `center` environment in a `figure` environment. In following examples, I will enclose the figure in a box to make it look a little better.

```
set terminal latex
set output "eg1.tex"
plot [-3.14:3.14] sin(x)
```

Note that gnuplot has drawn in the axes, labeled the tic marks for us, scaled the y axis automatically, and added a key in the upper-right-hand corner (this may be moved with the `set key` command, and removed with `unset key`[1]).

This is the default line style for the LaTeX driver. Because of the limited picture capabilities of LaTeX, many dots are required to approximate drawing a solid line. This may overload the memory of many TeX implementations. There are other line types available that draw dotted lines and use much less memory. The EEPIC driver draws solid lines with much less memory usage.

[1] In gnuplot version 4.0, the syntax `set noXXX` changed to `unset XXX`.

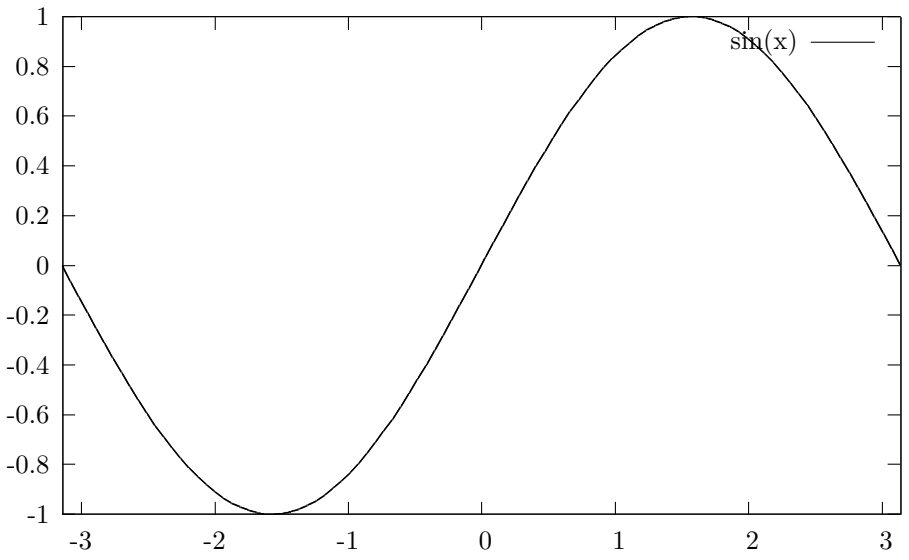

Figure 1: A first example: $y = \sin(x)$.

Example 2: Now we will embellish the plot a little with some labels. This input file produces Figure 2.

```
set terminal latex size 7cm, 5cm
set output "eg2.tex"
set format xy "$%g$"
set title 'This is a plot of $y=\sin(x)$'
set xlabel 'This is the $x$ axis'
set ylabel 'This is\\the\\$y$ axis'
plot [0:6.28] [0:1] sin(x)
```

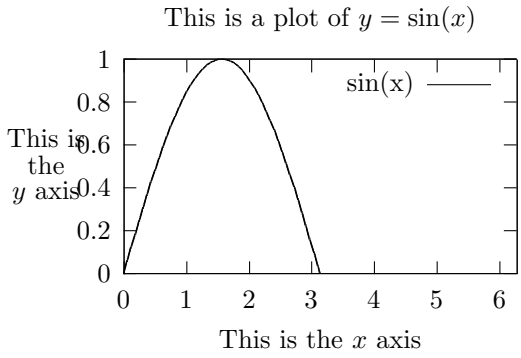

Figure 2: A fancier example.

We have specified the plot to be 7 cm wide and 5 cm tall with the `set term latex size ...` command. This is the size of the area used by the plot, *including* space for the labels. In the first example, this size was the default 5 inches by 3 inches.

We have requested that the format used by the x- and y-axis tic mark labels be in LaTeX math mode. This makes the labels look a little better. The default is `set format xy "%g"`. The `%g` represents the general-purpose floating point formatting specification for the `printf` function in C. Any valid floating-point formatting specification, or LaTeX command, is allowed in the format.

A title for the plot and labels for the axes were set up in the next three commands. Note that they are processed by LaTeX and so may have math mode and other symbols in them. The ylabel may have multiple

lines, delineated with \\. The ylabel can be moved around with optional offset parameters (see `set ylabel` in the gnuplot manual). Typically, the ylabel needs to be moved to the left to avoid interfering with the left-hand side of the plot. Once these labels are set up, they will be used for all subsequent plot commands until they are changed. These labels are also supported by the other terminal types, but (of course) any LaTeX code in the string will not be interpreted. We have also defined the range of both x (now $[0, 2\pi]$) and y (here $[0, 1]$).

So far we have plotted one curve, $y = \sin(x)$, on one plot. In gnuplot, each `plot` command generates a new plot. If the output is to a screen, the screen is cleared. If to a printer, a new page is produced. In the `latex` case, a new picture is started. It is not likely that LaTeX users will want this to happen, so generally each plot has its own input file and is kept in a separate output (`.tex`) file for inclusion at different places in the document.

Example 3: To place more than one curve on a plot, use one `plot` statement and separate the description of each curve by a comma. In our next example, we will plot both a function and a data file on the same plot. This plot is shown in Figure 3.

```
set terminal latex
set output "eg3.tex"
set format xy "$%g$"
set title "This is another plot"
set xlabel "$x$ axis"
set ylabel "$y$ axis"
set key at 15,-10
plot x with lines, "eg3.dat" with linespoints
```

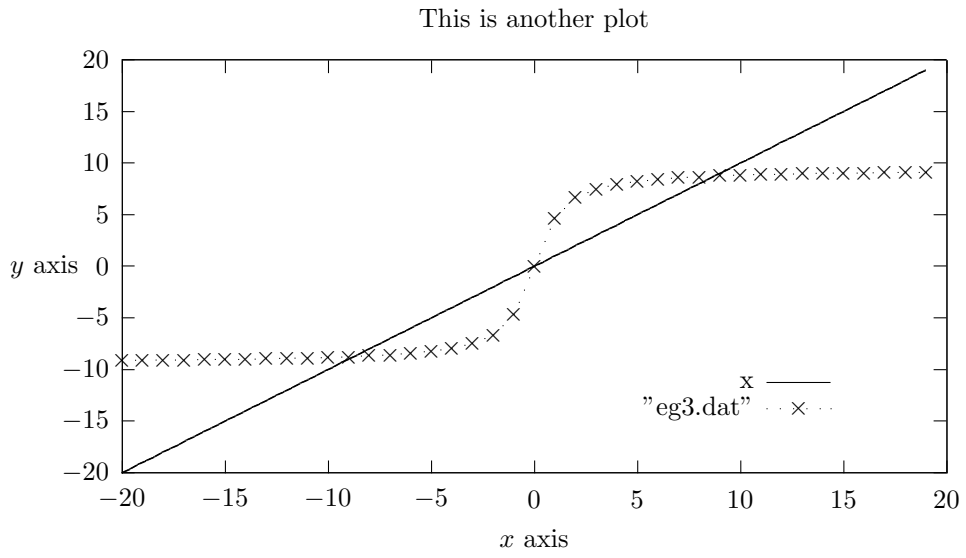

Figure 3: An example with two curves on the same plot.

Here you will see that the x range was not specified. The x range is determined automatically, unless specified by the user. In this case, it is defined by the range of the data file `"eg3.dat"`. The function is plotted over the same range. If no data files or x range are supplied, the default range of $[-10 : 10]$ is used. We have also moved the key to a different position. The function $y = x$ is plotted "with lines", which is the default plot style for functions, and is shown here to illustrate the plot style option. The data file `eg3.dat` is plotted with style `linespoints`, a style like `lines` that also plots a symbol at each data point.

There is a style called `points` that only plots the symbols at data points, and another called `dots` that plots a tiny dot for each data point. The `points` and `linespoints` styles produce a different point symbol for each curve on the plot (for up to twelve symbols, after which they are re-used; see Figure 8 for a complete list). The `lines` and `linespoints` styles use a different line style for each curve on the plot (in this example the dots have different spacing). The style `impulses` draws a perpendicular from each point to the x-axis. Finally, the `errorbars` style can draw error bars at each data point (see the gnuplot manual).

Example 4: In the above plots of $\sin(x)$, it would make more sense to label the axis in units of π. The position and labels of the tic labels may be specified by the user, with the `set xtics` and `set ytics` commands. This is demonstrated by the following example, shown in Figure 4.

```
set terminal latex
set output "eg4.tex"
set format y "$%g$"
set format x "$%.2f$"
set title 'This is $\sin(x)$'
set xlabel "This is the $x$ axis"
set ylabel "$\\sin(x)$"
unset key
set xtics -pi, pi/4
plot [-pi:pi] [-1:1] sin(x)
```

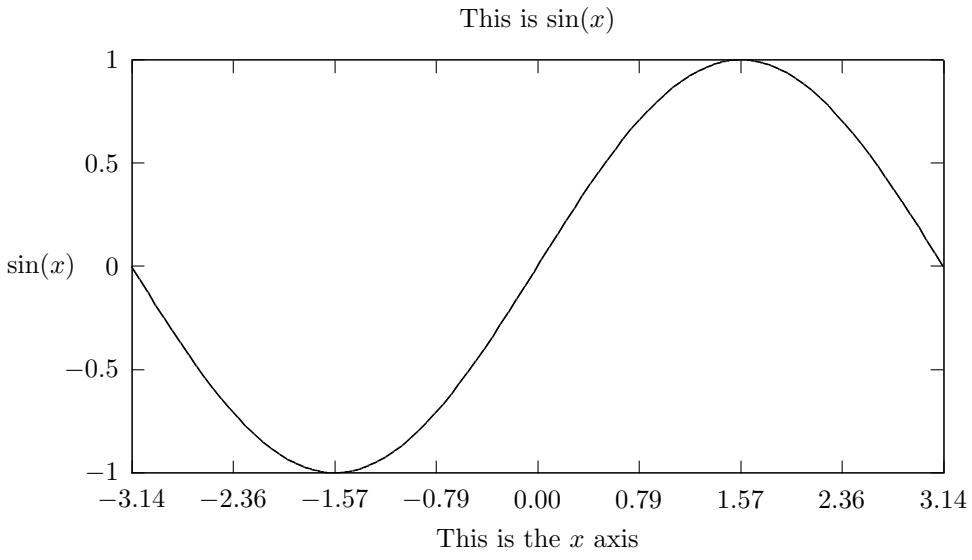

Figure 4: An example of the `set xtics` command.

Since `pi` is a predefined variable in gnuplot, we can use it anywhere we may use an expression. The `set xtics` command here specifies that the tics on the x axis start at $-\pi$ and increment by $\pi/4$. Since no end point is given, the tics continue to the right edge. We have also turned off the key, and changed the format to restrict the x-axis tic labels to 2 decimal places. Note that the y axis label was delimited by double quotes, so the backslash had to be escaped. Within single quotes, as in the title, gnuplot passes the backslashes through with no changes. (The exception: a backslash at the end of a line—even within single quotes—is used by gnuplot for line continuation.)

With a little more work, the plot can look even better. Another form of this command allows us to specify the label and position of each tic individually. Replacing the above `set xtics` command with the following gives us Figure 5. We also make use of the line continuation character, the backslash (\), to spread out this command for readability.

```
set xtics ('$-\pi$' -pi,\
 '$-\frac{\pi}{2}$' -pi/2,\
 "0" 0,\
 '$\frac{\pi}{2}$' pi/2,\
 '$\pi$' pi)
```

Going further: You should now be able to make a variety of plots for your LaTeX document. We will present a final example without explanation that showcases some of the capabilities of gnuplot. You may find documentation for the various commands in the gnuplot manual, though hopefully this example is somewhat self-explanatory. This is shown in Figure 6.

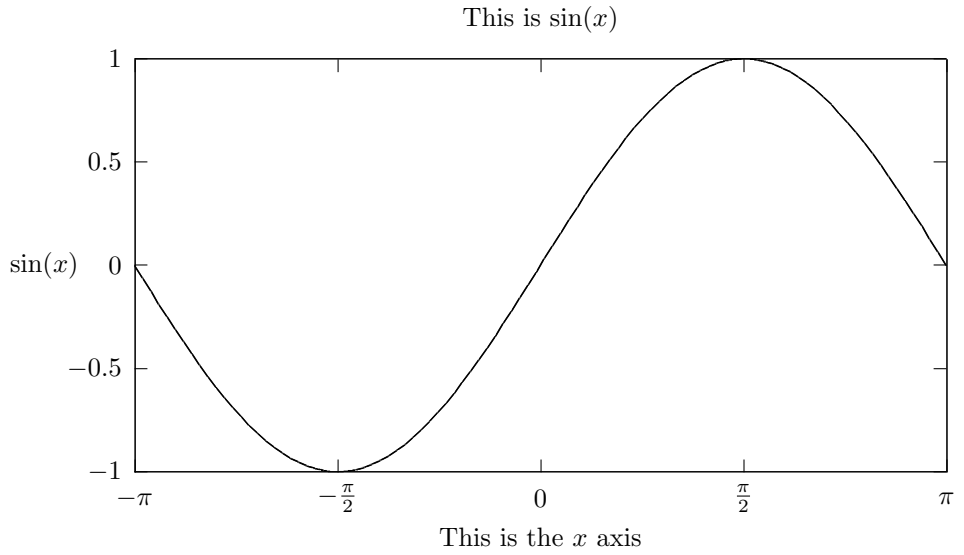

Figure 5: A fancy example of the `set xtics` command.

```
set terminal latex size 5.0, 3.0
set output "eg6.tex"
set format y "$%g$"
set format x '$%5.1f\mu$'
set title "This is a title"
set xlabel "This is the $x$ axis"
set ylabel 'This is\\a longer\\version\\ of\\the $y$\\ axis' offset -1
set label "Data" at -5,-5 right
set arrow from -5,-5 to -3.3,-6.7
set key top left
set xtic -10,5,10
plot [-10:10] [-10:10] "eg3.dat" title "Data File"  with linespoints lt 1 pt 7,\
   3*exp(-x*x)+1  title '$3e^{-x^{2}}+1$' with lines lt 4
```

Line and point types: For reference, we show all of the line and point types available in Figure 8.

2.1 Summary — Use with LaTeX

In summary, to use the most basic LaTeX mode of gnuplot, the first command to gnuplot should be

```
set terminal latex
```

and the output of gnuplot should be directed to a file, for example,

```
set output "plot.tex"
```

This may be anything you like but it should have a `.tex` extension, of course. The default plot size is 5 inches by 3 inches. You can change this be specifying a size in either cm or inches when you select the terminal

```
set terminal latex size 7cm, 5cm
```

Then you do the (s)plot, and finally issue commands to close the file and switch the terminal back to the default by

```
set output
set terminal pop
```

Finally, the file will contain all of the plots you have specified (you probably only want one plot per file). This file can then be used in a LaTeX document, *e.g.*,

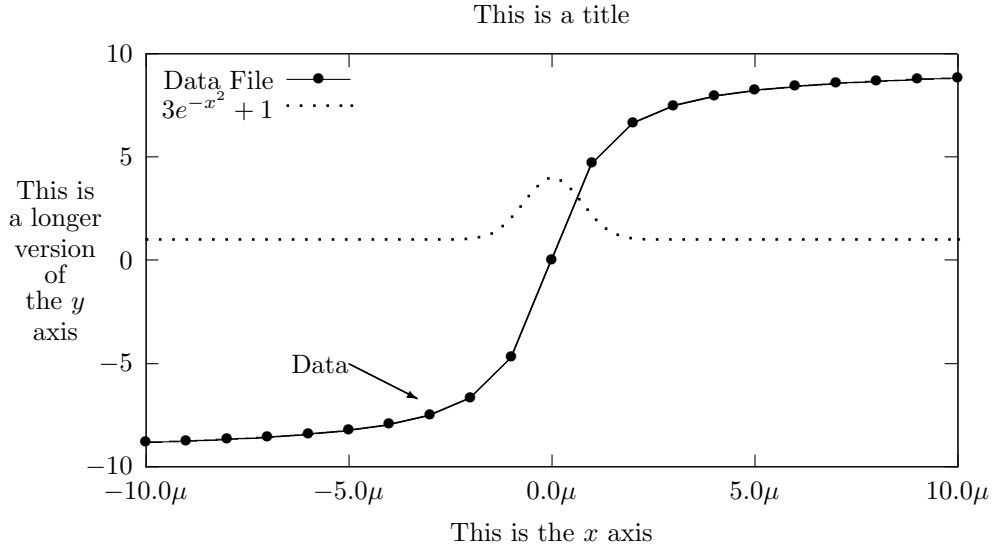

Figure 6: An example of many features.

```
\begin {figure}
  \begin{center}
    \input{plot}
  \end{center}
\end {figure}
```

This puts the plot into a figure.

You will also want to read about the following commands: `set title, set xlabel, set ylabel, set key, set label, set xtics, set ytics`, and `set clip`. These are all described in the regular gnuplot manual.

3 Use with EEPIC

EEPIC is a macro package extending the picture environment of LaTeX. If you have the EPIC or EEPIC macros, and your `dvi` translator supports the *tpic* `\specials`, then you can save LaTeX memory. With EEPIC pictures, the `plot.tex` file will be smaller, LaTeX will run much faster (and need much less memory), and the `dvi` file will be smaller. The quality of the output is about the same. If you change the source, you can generate some more interesting line styles.

To use EEPIC, set gnuplot's terminal type to `eepic` instead of `latex`, and use gnuplot as before. The line styles will change. Include the file `plot.tex` in your document as before, along with the document style options `[epic,eepic]`.

4 Other TeX-based terminal types in gnuplot

In addition to the `latex` terminal, the following LaTeX-friendly terminals are available. The gnuplot documentation contains more information about the options and usage for each of these.

- `emtex`: Like the `latex` terminal, but supports emtex specials: any line slopes contrary to a very limited set of LaTeX slopes.
- `epslatex`: Combined LaTeX and postscript parts for text and lines, respectively, with the postscript part included by `\includegraphics{...}` command.
- `pstex` and `pslatex`: Combined TeX / LaTeX and postscript parts for text and lines, respectively, included by `\special{psfile=...}` command.

- `mf` and `mp`: Produces metafont and metapost outputs.
- `pslatex and epslatex`: New versions of these terminals were introduced in gnuplot 4.4. See the example below.
- `tikz`: Much improved graphics compaired to earlier LaTeXterminals. Use of the terminal requires a lua interpreter that is called by gnuplot. The resulting .tex file can be processed using pdflatex.
- `context`: Produces output for the ConTeXt environment.

The `tikz` and `context` terminals support newer gnuplot features such as transparency and embedded images. The *standalone* terminal option allows you to create a *.tex document that can be processed by itself without being embedded in a wrapping document.

5 Example of using the epslatex terminal driver

The epslatex terminal driver allows you to mix the best features of TeX and PostScript. Text elements are typeset by TeX while the graphic elements are created and positioned in parallel by gnuplot's PostScript driver. The plot can use either color or grayscale. The driver produces two different files, one for the eps part of the figure and one for the LaTeX part. The name of the LaTeX file is taken from the 'set output' command. The name of the eps file is derived by replacing the file extension (normally ".tex") with ".eps" instead.

With the epslatex terminal you cannot use "\\" in a plain string to denote newline. You can either put each line in a separate label and specify vertical spacing manually, or put the entire label in a `\parbox`. For example, the labels in Figure 7 were generated with the following commands:

```
x=.10; y=.15; dy=.05
set label "left torus:"                              at screen x,y; y=y-dy
set label '$x=\cos u+\frac{1}{2}\cos u \cos v$'   at screen x,y; y=y-dy
set label '$y=\sin u+\frac{1}{2}\sin u \cos v$'   at screen x,y; y=y-dy
set label "$z=\\frac{1}{2}\\sin v$"                  at screen x,y
x=.65; y=.08
set label '\parbox{2.5in}{right torus:\\$x=1+\cos u+\fr\
ac{1}{2}\cos u \cos v$\\$y=\frac{1}{2}\sin v$\\\
$z=\sin u + \frac{1}{2}\sin u \cos v$}' at screen x,y left
```

6 Contact for help

For general gnuplot questions, you can post to the gnuplot newsgroup `comp.graphics.apps.gnuplot`. Additional sources of information are listed on the gnuplot homepage `http://www.gnuplot.info`.

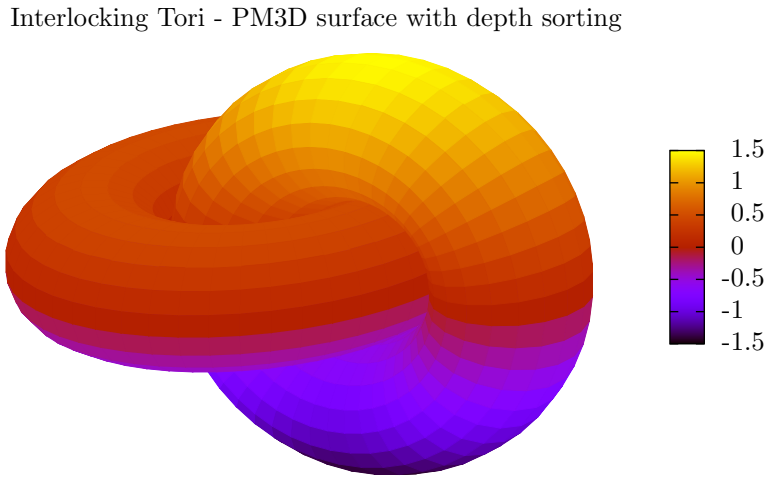

Figure 7: Interlocking tori demo, drawn using the epslatex driver.

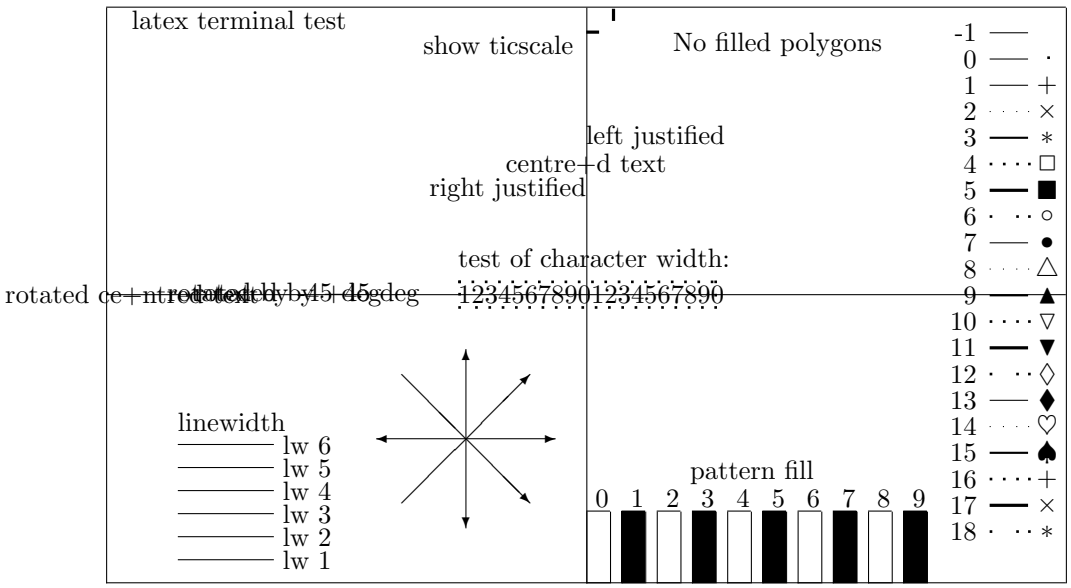

Figure 8: Line and point types for the basic latex terminal. Note lack of text rotation.