LaTeX and the GNUPLOT Plotting Program[1]

David Kotz
David.Kotz@Dartmouth.edu

July 3, 1991

[1] This document describes GNUPLOT version 3.0. All plots in this document were made with that version of GNUPLOT.

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. Former GnuTeX users are referred to Section 4 for information about adapting to GNUPLOT. 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 a new 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.

There is a small package of auxiliary files (makefiles and scripts) that I find useful for making LaTeX plots with GNUPLOT. This is available for `ftp` as `pub/gnuplot-latex.shar` from `cs.duke.edu`. I can mail copies (see the end of this paper for information).

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. For example, output from the terminal type `unixplot` may be sent to a program interpreting the Unix plotting standard. The terminal types `imagen` and `postscript` may be used for output to printers understanding those languages. (A shell script (`lasergnu`) is supplied with the distribution that will accept a GNUPLOT command or input file and send the output to an Imagen or Postscript laser printer. This script may have been adapted to your site.) The terminal type `fig` outputs FIG code that can be read by the Fig graphics program and translated into forms usable in both TeX and LaTeX documents.

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

```
gnuplot gnu.input
```

In this example, all of the commands to GNUPLOT are contained in the file `gnu.input`. 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` command). GNUPLOT continues to print error messages to the terminal (stderr).

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)
```

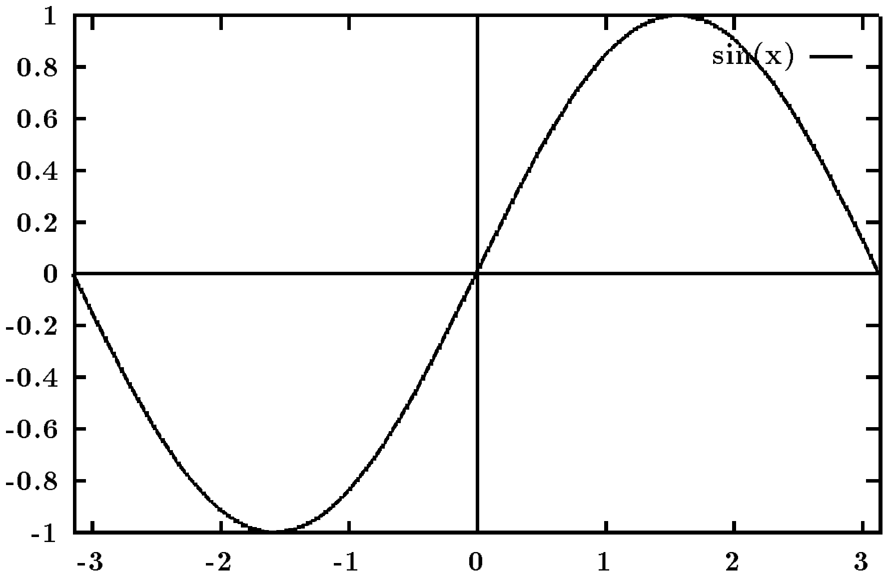

Figure 1: A first example: $y = sin(x)$

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 `set nokey`).

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.

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

```
set terminal latex
set output "eg2.tex"
set size 5/5., 4/3.
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)
```

We have specified the plot to be 5 inches wide and 4 inches tall with the `set 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. By specifying the scaling factors of 1 (or 5/5) and 1.3333 (or 4/3), we obtain the desired plot size.

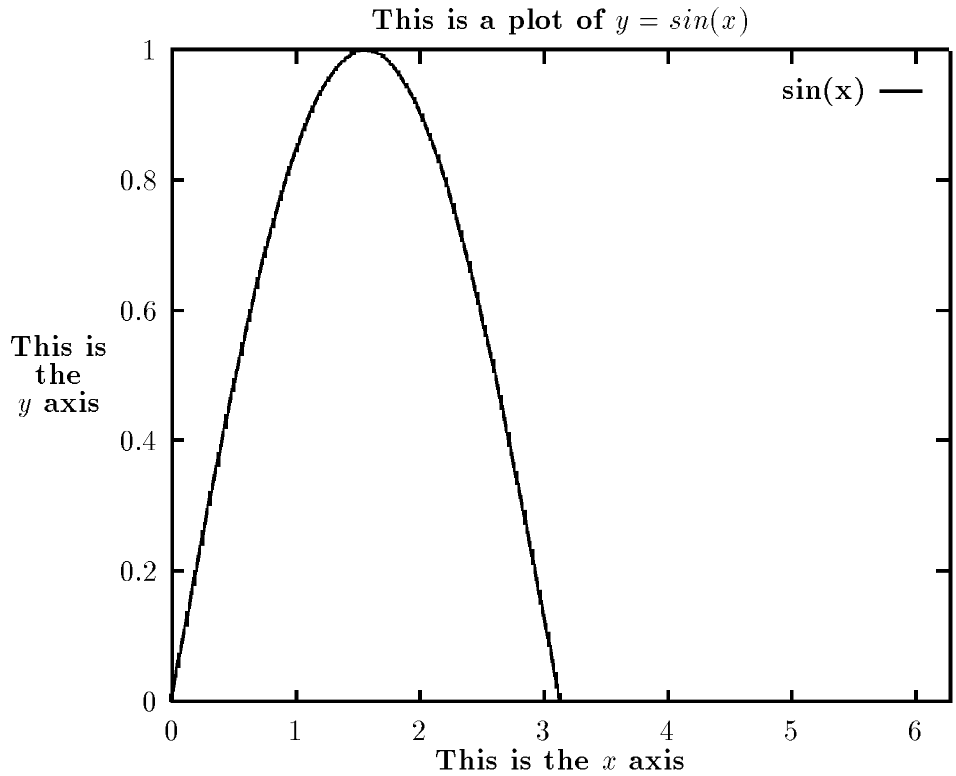

Figure 2: A more fancy example.

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 15,-10
plot x with lines, "eg3.dat" with linespoints
```

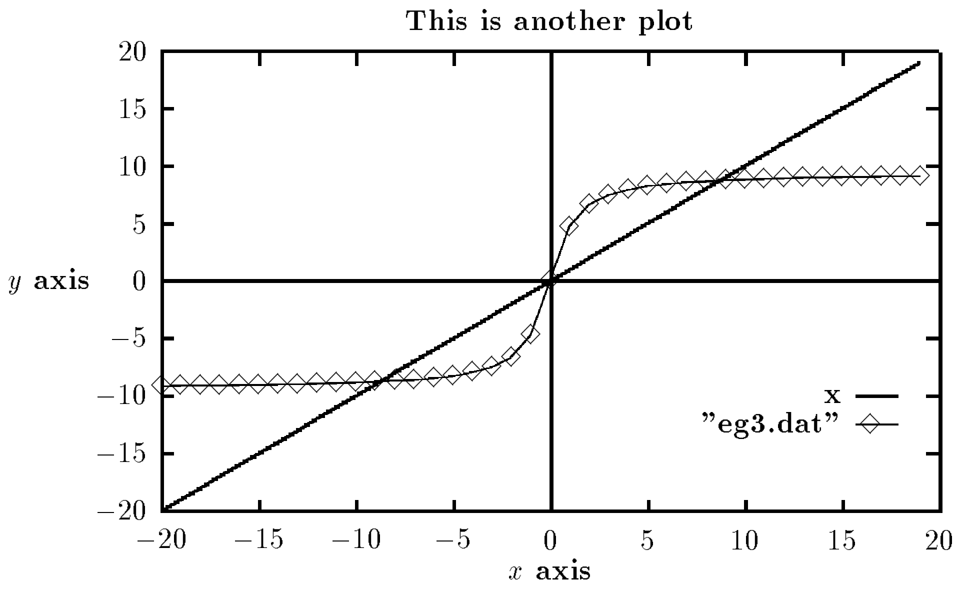

Figure 3: An example with two curves on the same plot.

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 7 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 "$\sin(x)$"
set nokey
set xtics -pi, pi/4
plot [-pi:pi] [-1:1] sin(x)
```

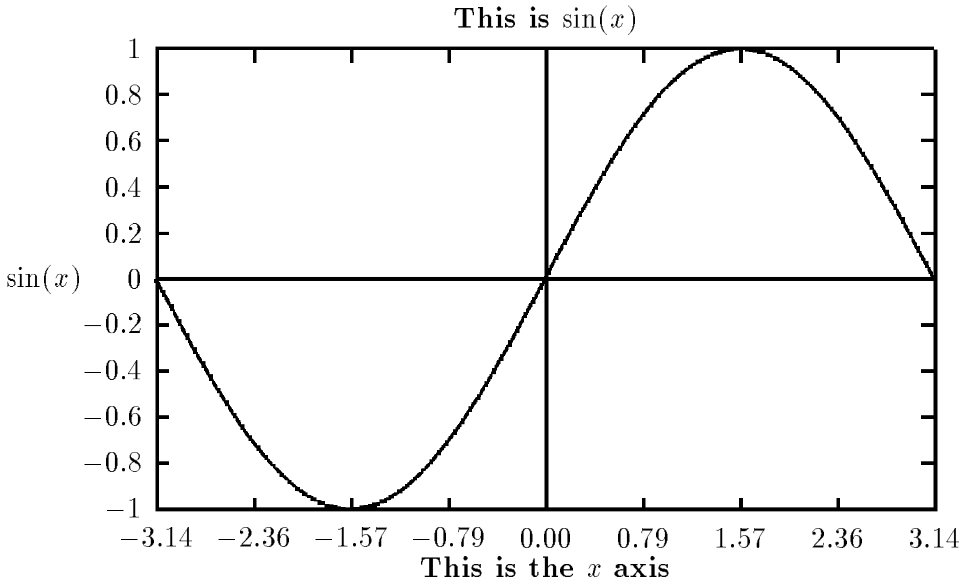

Figure 4: An example of the `set xtics` command.

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.

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)
```

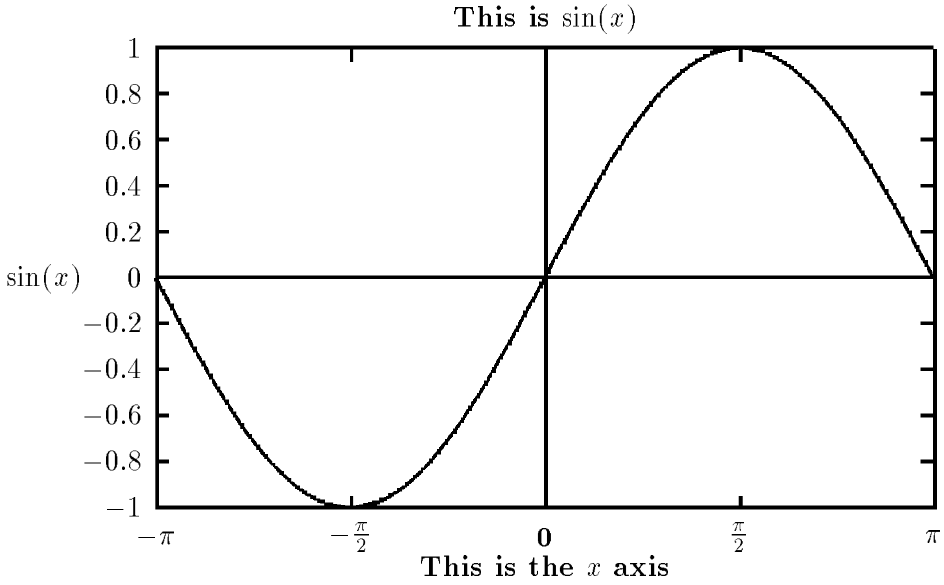

Figure 5: A fancy example of the `set xtics` command.

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.

```
set terminal latex
set output "eg6.tex"
set size 3.5/5, 3/3.
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"
set label "Data" at -5,-5 right
set arrow from -5,-5 to -3.3,-6.7
set key -4,8
set xtic -10,5,10
plot [-10:10] [-10:10] "eg3.dat" title "Data File"  with linespoints 1 7,\
   3*exp(-x*x)+1  title "$3e^{-x^{2}}+1$" with lines 4
```

Line and point types: For reference, we show all of the line and point types available in Figure 7.

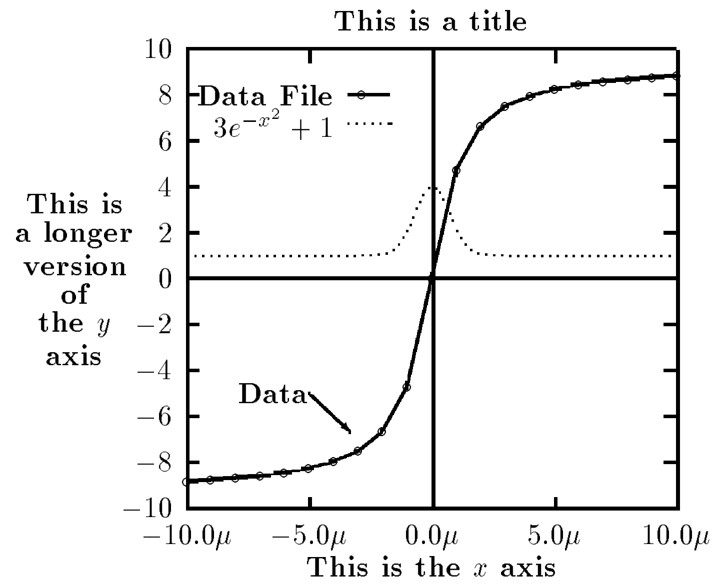

Figure 6: An example of many features.

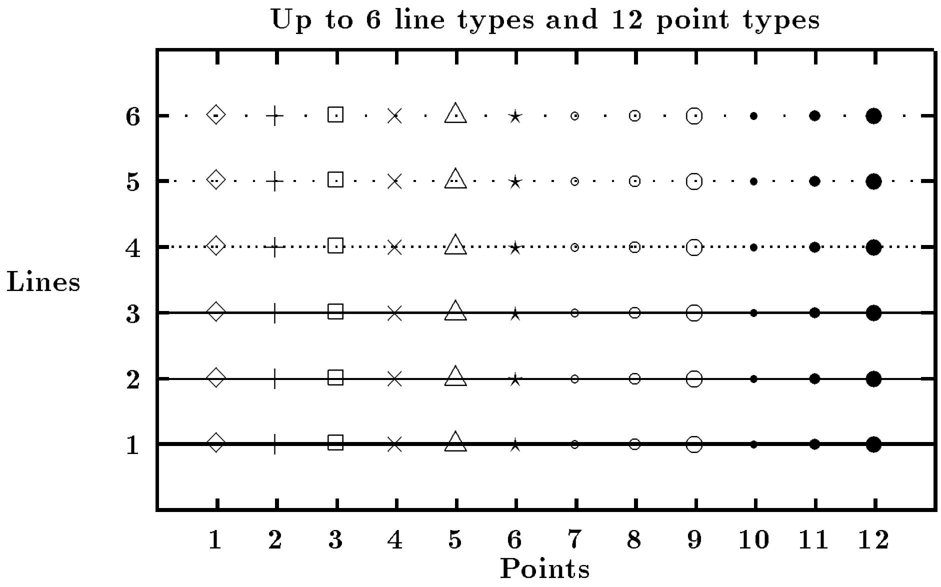

Figure 7: All of the line and point types in the LaTeX driver.

2.1 Summary — Use with LaTeX

In summary, to use the LaTeX facilities 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. Then the size of the plot should be given. For example, the command

```
set size 1,2
```

tells GNUPLOT to use a 5 inch wide by 6 inch high box for the plot. The numbers given are *scale factors*, not the actual size. The default is 5 inches by 3 inches. This is the size of the complete plot, including all labels.

When finished, 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.*,

```
\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* `\special`s, 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 For Former GnuTeX Users

Former GnuTeX users may be pleased with many of the new features (many inspired by your suggestions!), but will also find many changes. GNUPLOT will *not* run all GnuTeX input files unchanged. Several GnuTeX features were not included in GNUPLOT because they were specific to the LaTeX driver. I encourage you to use the newer GNUPLOT. A translator is available that attempts to translate your old GnuTeX 1.6 input files into GNUPLOT 3.0 files. You can ftp it from `cs.duke.edu` as `dist/sources/gnuplot/gnut2p.tar.Z`. This file also contains directions and a list of changes from GnuTeX to GNUPLOT.

5 Contact

Please contact me at `David.Kotz@Dartmouth.edu` with any comments you may have on GNUPLOT's LaTeX driver. For general GNUPLOT questions, send mail to the GNUPLOT mailing list (`pixar!info-gnuplot@ucbvax.berkeley.edu`).