egplot:
Encapsulated **gnuplot** for LaTeX*

Axel Probst†

c/o
Federal Institute for Materials Research and Testing
Unter den Eichen 87
D-12205 Berlin
Germany

November 30, 2010

Abstract

The **egplot** package allows to encapsulate **gnuplot** commands in LaTeX sources. This is very useful for keeping illustrations in sync with the text. It also frees the user from inventing descriptive names for PostScript files. Additionally the package provides commands that enable the user to let **gnuplot** do calculations and insert the result values into the generated output.

Contents

---

*This is `egplot.sty`, version v1.02a, date 1998/07/08.

†e-mail: `Axel.Probst@bam.de`

Copying

1 Introduction

When adding illustrations to documents, one faces two bookkeeping problems:

1. How to encourage oneself to keep the illustrations in sync with the text, when the document is updated?
2. How to make sure that the illustrations appear on the right spot?

For both problems, the best solution is to encapsulate the figures in the LaTeX source:

1. It is much easier to remember to update an illustration if one doesn't have to switch files in the editor.
2. One does not have to invent illustrative filenames, if the computer keeps track of them.

This concept of integrating the image generating commands into the LaTeX source was implemented for METAFONT by Thorsten Ohl[1] in the EMP-package. The egplot package now allows the encapsulation of gnuplot [5] into LaTeX [1, 2, 3]. Although gnuplot provides several output formats that are suitable for the inclusion into LaTeX the egplot package is only intended for use with the Postscript terminal of gnuplot so far.
In addition to the image inclusion commands egplot provides the user with commands to let gnuplot do calculations and include the results into the document. Unfortunately these features are implemented with the UN*X text utils and so they are only usable if these are installed on the system. If the user does not provide a name for the gnuplot file the names for the PostScript and the result values files are built by appending the number of the gnuplot file, the figure/calculation number and a three letter extension (`.eps` or `.val`) to `\jobname`. So the user has to choose a `\jobname` that is short enough so that the generated filenames fit into the conventions of certain operating systems.

2 Usage

2.1 Options

`Options` Besides the options of the `graphicx` package egplot recognizes the following

---

[1] e-mail:`Thorsten.Ohl@Physik.TU-Darmstadt.de`

options:

`german`: If `german` is specified the calculated values and the tic labels of the diagrams are changed to show a ',' as decimal point character. The default is a '.'. This feature is also implemented with UN*X text utils and is only available if they are installed on the system.

`gnuplot35`: If `gnuplot35` (default) is specified the gnuplot commands generated by `egplot` will be compatible with the syntax of the official gnuplot version 3.5. Of course the user has to look for the right syntax in his gnuplot code himself. Special care has to be taken for the `\egpprelude{...}` and the `\egpfigepilog{...}` commands since these are used to implement the missing `reset` command of gnuplot-3.5.

`gnuplot36beta`: If `gnuplot36beta` is specified the gnuplot commands generated by `egplot` will use the features of the beta version gnuplot 3.6beta. As mentioned above the user has to look for the right syntax in his gnuplot code himself.

2.2 Commands and Environments

2.2.1 Miscellaneous

`egpfile`

All descriptions that should go into one gnuplot file are placed inside a `egpfile` environment which takes the name of the gnuplot file as an optional argument:

```
\begin{egpfile}[⟨gnuplot-file⟩]
   ...
\end{egpfile}
```

The default gnuplot-filename is `\jobname.gp`.

`egpcmds`
`\egpwrite`

Write gnuplot commands to the current file outside of a figure. The `\egpwrite` command is intended for short one line commands.

```
\begin{egpcmds}
   ⟨gnuplot-commands⟩
\end{egpcmds}
```

`\egpprelude`
`\egpaddtoprelude`

Define and add to the set of commands that are prepended to the top of every gnuplot file. It is intended for the global definition of variables or functions. The default is empty.

2.2.2 Figures

`egp`
`egpx`
`egpdef`

The `egp` as the `egpx` environment contains the description of a single figure that will be placed at the location of the environment. The `egpdef` environment only defines a figure but does not include it into the document. This is useful, because these environments use the `verbatim` package and can therefore *not* be used as an argument to other macros. The ⟨*name*⟩ that is assigned to the figure is used for later inclusion with the `\egpuse{`⟨*name*⟩`}` command. For the `egp` and `egpx` environment the assignment of the ⟨*name*⟩ is optional. The required argument of the `egpx` environment accepts any set of keys accepted by the `\includegraphics` command of the `graphicx` package.

```
\begin{egp}[⟨name⟩]
    ⟨gnuplot-commands⟩
\end{egp}
```

```
\begin{egpx}[⟨name⟩]{⟨key val list⟩}
    ⟨gnuplot-commands⟩
\end{egpx}
```

```
\begin{egpdef}{⟨name⟩}
    ⟨gnuplot-commands⟩
\end{egpdef}
```

`\egpuse` Reuse a previously defined figure. The optional argument of the `\egpuse` command accepts any set of the keys that is accepted by the `\includegraphics` command of the `graphicx` package.

```
\egpuse[⟨key val list⟩]{⟨name⟩}
```

`\egpfigprelude`
`\egpaddtofigprelude` Define and add to a gnuplot prelude that is prepended to the output of every `egp`, `egpx` or `egpdef` environment. The default is:

```
set terminal postscript eps monochrome dashed "Helvetica" 17
```

In fact this is the command where the terminal for the gnuplot-`plot` command is set. So the user has to take care that (Encapsulated) PostScript output is generated.

`\egpfigepilog`
`\egpaddtofigepilog` Define and add to a gnuplot epilog that is appended to the output of every `egp`, `egpx` or `egpdef` environment. This command can be used for e.g. `replot`ting the figure to the screen or `reset`ing to the defaults after every figure.
The defaults are as follows:

Option:	none, `gnuplot35`	`gnuplot36beta`
	`load "reset.gp"`	`reset`

2.2.3 Calculating

In addition to the commands and environments to generate and include gnuplot figures the egplot-package provides commands to use gnuplot for the calculation of arbitrary arithmetic expressions. Since the gnuplot-`plot` command is used for this feature every expression that is accepted by this command is possible. But this may also lead to unexpected results if the expression contains the variable x which is used as the independent variable of the gnuplot-`plot` command. As stated above (cf. p. 2) the UN*X text utils are used for the implementation and so the calculation commands can only be used on systems where these are installed.

`\egpcalc` Let gnuplot calculate the value of a ⟨*gnuplot-expression*⟩. The result is written to a file. The optional argument assigns a name to be used with `\egpuseval{`⟨*name*⟩`}`.

```
\egpcalc[⟨name⟩]{⟨gnuplot-expression⟩}
```

`\egpuseval` Insert a previously defined calculation result.

`\egpshowval` Does the same as the `\egpcalc`-command but additionally the result is placed in the output at the position of the `\egpshowval`-command.

`\egpassign` The first argument is the name of a ⟨*gnuplot-variable*⟩ or ⟨*gnuplot-user function*⟩ which is assigned the second argument which is a ⟨*gnuplot-expression*⟩. The result is placed in the output as for the `\egpshowval` command.

2.3 Procedure

After LaTeX has done it's job for the first time you have to invoke gnuplot on the generated file (default: `\jobnameX.gp`, where `X` is a number). Then another LaTeX run is necessary to include the figures and the results into the output.

2.4 Examples

For a simple example, let's draw the function $f(x) = \sin(\sqrt{x^2 + y^2})/\sqrt{x^2 + y^2}$.

```
⟨*sample⟩
\begin{egpfile}
\begin{center}
\begin{egpx}[sombrero]{width=0.8\linewidth}
    set hidden3d
    set nogrid
    set samples 1000
    set isosamples 35
    splot [-10:10] [-10:10] sin(sqrt(x*x+y*y))/sqrt(x*x+y*y)
\end{egpx}
\end{center}
```

Additionally we define a figure that will not be shown here but at the place of the appropriate `\egpuse` command.

```
\begin{egpdef}{kleinbottle}
    set hidden3d
    set parametric
    set nokey
    set nogrid
    set noborder
    set noxtics
    set noytics
    set noztics
    set xrange [-10:10]
    set yrange [-10:10]
    set zrange [-3:3]
    set urange [0:2*pi]
    set vrange [0:2*pi]
    set isosamples 39,60
    set view 60,120
    set title "Klein bottle"
    splot  (2*sin(u)*cos(v/2)-sin(2*u)*sin(v/2)+8)*cos(v), \
           (2*sin(u)*cos(v/2)-sin(2*u)*sin(v/2)+8)*sin(v), \
           2*sin(u)*sin(v/2)+sin(2*u)*cos(v/2)
\end{egpdef}
```

Since we have given a name to each diagram, we can now use them with

```
\begin{figure}
  \begin{center}
    \fbox{\egpuse[scale=0.4]{sombrero}}
    \fbox{\egpuse[scale=0.4]{kleinbottle}}
  \caption{Two examples taken from the \GP{} demo}\label{fig:demo}
  \end{center}
```

□□

Figure 1: Two examples taken from the gnuplot demo

```
\end{figure}
```

and the result is shown in figure 1.
To calculate the value of $f(\pi/4)$ we issue the command
$f(\pi/4) =$

```
$f(\pi/4)=\egpshowval[sin_quarter_pi]{sin(pi/4)}$
```

and get $\frac{\sqrt{2}}{2} = \square$[2]

```
and get $\frac{\sqrt{2}}{2}=\fbox{\egpuseval{sin_quarter_pi}}$.
\end{egpfile}
⟨/sample⟩
```

3 Acknowledgements

I would like to thank Thorsten Ohl for submitting the EMP package to CTAN. By using it as a template I managed it to adapt the idea of integrating the image generating commands into LaTeX for gnuplot. A lot of code of the EMP package was reused with only marginal changes. This is also caused by the fact that I am far away from understanding all of the code of EMP.

4 Todo

In additon to optimising egplot it would be nice if the features that are provided through the use of UN*X text utils were implemented in TeX/LaTeX. Another interesting feature to implement in following versions of egplot is the possibility to use other output formats provided by gnuplot, especially the `pslatex` and `pstricks` terminals but also the `png` terminal for inclusion into PDF could be useful.

References

[1] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach, *The LaTeX Graphics Companion*, Addison-Wesley, Reading MA, 1997.

[2] Leslie Lamport, *LaTeX — A Documentation Preparation System*, Addison-Wesley, Reading MA, 1985.

[3] Michel Goossens, Frank Mittelbach, and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, Reading MA, 1994.

[4] Thorsten Ohl, `emp`, available from CTAN (cf. p. 7), in the `macros/latex/contrib/supported/emp` directory.

[5] Thomas Williams and Colin Kelley, *gnuplot*, available from `ftp.dartmouth.edu` in the `/pub/gnuplot` directory.

---

[2] I couldn't figure out how to remove the trailing space, yet. Any hints ?

Distribution

egplot is available by anonymous internet ftp from any of the Comprehensive TEX Archive Network (CTAN) hosts

`ftp.tex.ac.uk`, `ftp.dante.de`

in the directory

`macros/latex/contrib/supported/egplot`

5 Implementation

```
⟨*style⟩
\def\fileversion{v1.02a}
\NeedsTeXFormat{LaTeX2e}
\gdef\filename{egplot.sty}%
\gdef\filedate{1998/07/08}%
\gdef\filemaintainer{Axel Probst}%
```

And now the standard procedure:

```
\ProvidesPackage{egplot}[\filedate\space\fileversion\space
  Encapsulated gnuplot LaTeX Package (\filemaintainer)]
```

Load the required packages:

```
\RequirePackage{verbatim}
\RequirePackage{ifthen}
```

Now the options are specified:

```
\newboolean{egp@german}
\setboolean{egp@german}{false}
\DeclareOption{german}{%
        \setboolean{egp@german}{true}}
\newboolean{egp@oldgp}
\setboolean{egp@oldgp}{true}
\DeclareOption{gnuplot35}{%
        \setboolean{egp@oldgp}{true}}
\DeclareOption{gnuplot36beta}{%
        \setboolean{egp@oldgp}{false}}
```

Every option we don't understand is sent down to graphicx:

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{graphicx}}
\ProcessOptions
\RequirePackage{graphicx}[1994/12/15]
```

`\egpwrite` Write out the argument to the gnuplot file.

```
{\catcode`\#=11\gdef\egpcomment{#}}
\def\egpwrite#1{%
  \if@egpio
    \immediate\write\@outegp{#1}%
  \fi
  \ignorespaces}
\newif\if@egpio
\@egpiotrue
\newwrite\@outegp
```

`\egpfile`

This environment encloses each gnuplot input file. The single optional argument gives the name of the file.

```
\newcounter{egpfilenum}           % 1998-03-07
\setcounter{egpfilenum}{0}        % 1998-03-07
\newcommand{\egpfile}[1][\jobname\theegpfilenum_]{%
  \def\theegpfile{#1}%
  \ifthenelse{\equal{\theegpfile}{\jobname\theegpfilenum_}}{%
     \stepcounter{egpfilenum}%
     \def\theegpfilename{\jobname\theegpfilenum.gp}}{%
     \def\theegpfilename{\theegpfile.gp}}
```

Open the gnuplot file.

```
  \if@egpio
    \immediate\openout\@outegp=\theegpfilename\relax
    \egpwrite{\egpcomment\space \theegpfilename -- %
              do not edit, generated automatically by \jobname.tex^^J}
```

append the defined prelude and write it out:

```
    \expandafter\ifx\expandafter*\the\egp@prelude*\else
      \egpwrite{\the\egp@prelude;}%
    \fi
  \fi
```

Count the figures and the calculations

```
  \setcounter{egpfig}{0}
  \setcounter{egpcalc}{0}}
\let\theegpfile\relax
\newcounter{egpfig}
\newcounter{egpcalc}
```

Standard preludes for the whole file and for every figure and the per figure epilog:

```
\newtoks\egp@prelude
\newtoks\egp@figprelude
\newtoks\egp@figepilog
%
```

`\egpprelude`
`\egpfigprelude`
`\egpfigepilog`
`\egpaddtoprelude`
`\egpaddtofigprelude`
`\egpaddtofigepilog`

Define and add to the file or figure prelude and the figure epilog.

```
\def\egpprelude#1{\egp@prelude={#1}}
\def\egpfigprelude#1{\egp@figprelude={#1}}
\def\egpfigepilog#1{\egp@figepilog={#1}}
\def\egpaddtoprelude#1{\egp@prelude=\expandafter{\the\egp@prelude^^J#1}}
\def\egpaddtofigprelude#1{\egp@figprelude=\expandafter{\the\egp@figprelude^^J#1}}
\def\egpaddtofigepilog#1{\egp@figepilog=\expandafter{\the\egp@figepilog^^J#1}}
```

`\endegpfile`

And here is how we close the `egpfile` environment:

```
\def\endegpfile{%
  \egpwrite{\egpcomment\space the end.}%
  \let\theegpfile\relax
  \if@egpio
    \immediate\closeout\@outegp
  \fi}
```

`\egp`
`\egpx`
`\egpdef`

Here are the environments to define and to define and include the gnuplot diagrams.

```
\newcommand{\egp}[1][*]{%
  \def\egp@@name{#1}%
  \egp@}
\newcommand{\egpx}[2][*]{%
   \def\egp@@name{#1}%
   \egp@x{#2}}
\newcommand{\egpdef}[1]{%
  \def\egp@@name{#1}%
  \egp@def}
```

`\egp@` `\egp@x` `\egp@def`

And here the real work is done.

```
\def\egp@{%
  \egp@start%
  \ifthenelse{\boolean{egp@oldgp}}
     {\egpwrite{\egpcomment\space --- \theegpfile\theegpfig.eps ---}}
     {\egpwrite{print 'generating picture ---- \theegpfile\theegpfig.eps'}}
  \egpwrite{set output '\theegpfile\theegpfig.eps'}
  \egp@includegraphics{\theegpfile}{\theegpfig}%
  \egpcmds}
\def\egp@x#1{%
  \egp@start%
  \ifthenelse{\boolean{egp@oldgp}}
     {\egpwrite{\egpcomment\space --- \theegpfile\theegpfig.eps ---}}
     {\egpwrite{print 'generating picture ---- \theegpfile\theegpfig.eps'}}
  \egpwrite{set output '\theegpfile\theegpfig.eps'}
  \egp@includegraphicx[#1]{\theegpfile}{\theegpfig}%
  \egpcmds}
\def\egp@def{%
  \egp@start%
  \ifthenelse{\boolean{egp@oldgp}}
     {\egpwrite{\egpcomment\space --- \theegpfile\theegpfig.eps ---}}
     {\egpwrite{print 'generating picture ---- \theegpfile\theegpfig.eps'}}
  \egpwrite{set output '\theegpfile\theegpfig.eps'}
  \egpcmds}
```

`\egp@start`

```
\def\egp@start{%
  \egp@checkfile
```

We can't use `\stepcounter` because of the `amstext` option of AMS-LaTeX disables it sometimes.

```
  \global\expandafter\advance\csname c@egpfig\endcsname \@ne
  \egp@@def{\egp@@name}%
```

Start the gnuplot figure:

```
    \expandafter\ifx\expandafter*\the\egp@figprelude*\else
     \egpwrite{\the\egp@figprelude}%
  \fi}
```

`\egp@checkfile` Make sure that a gnuplot file is open, otherwise *really* obscure error messages are possible:

```
\def\egp@checkfile{%
  \ifx\theegpfile\relax
    \errhelp={Outside an egpfile environment, I have no clue as to where^^J%
```

```
              the gnuplot commands should go.   I will use egpdefault.gp^^J%
              for this graph, but you’d better fix your code!}%
  \errmessage{I detected a egp environment outside of egpfile}%
  \egpfile[egpdefault]
  \fi}
```

`\egp@includegraphics`
`\egp@includegraphicx`

Include the Postscript files that were generated by gnuplot

```
\def\egp@includegraphics#1#2{%
  \leavevmode
  \IfFileExists{#1#2.eps}%
    {\includegraphics{#1#2.eps}}%
    {\typeout{%
      egp: File #1#2.eps\space not found:^^J%
      egp: Process \theegpfilename\space with gnuplot and then %
           reprocess this file.}}}
\newcommand{\egp@includegraphicx}[3][scale=1]{%
  \leavevmode
  \IfFileExists{#2#3.eps}%
    {\includegraphics[#1]{#2#3.eps}}%
    {\typeout{%
      egp: File #2#3.eps\space not found:^^J%
      egp: Process \theegpfilename\space with gnuplot and then %
           reprocess this file.}}}
```

`\egpcmds`

Write to the file:

```
\def\egpcmds{%
  \begingroup
    \@bsphack
    \let\do\@makeother\dospecials
    \catcode`\^^M\active
    \def\verbatim@processline{\egpwrite{\the\verbatim@line}}%
    \verbatim@start}%
```

`\endegpcmds`

```
\def\endegpcmds{%
    \@esphack
  \endgroup}
```

`\endegp`
`\endegpx`
`\endegpdef`

If the german option is used the decimal point character is changed to be `{,}`. This is done to avoid the additional space LaTeX inserts after the ‘,’ in math mode. This is implemented by using some of the UN*X text utils and therefore these have to be availlable on the system to benefit from this feature.

```
\def\endegp{%
  \endegpcmds
  \ifthenelse{\boolean{egp@german}}{%
       \egpwrite{!sed -e ’/[0-9]*[.][0-9]*)\ .show/s/[.]/,/’ %
                \theegpfile\theegpfig.eps >\theegpfile\theegpfig.tmp}
       \egpwrite{!cp \theegpfile\theegpfig.tmp \theegpfile\theegpfig.eps}
       \egpwrite{!rm -f \theegpfile\theegpfig.tmp}}
       {}
   \expandafter\ifx\expandafter*\the\egp@figepilog*\else
    \egpwrite{\the\egp@figepilog}%
  \fi
  \egpwrite{}}
\def\endegpx{\endegp}
\def\endegpdef{\endegp}
```

\egp@@def

```
\def\egp@@def#1{%
  \global\e@namedef{egp@k:f:#1}{\theegpfile}%
  \global\e@namedef{egp@k:c:#1}{\theegpfig}}
```

\e@namedef

```
\def\e@namedef#1{\expandafter\edef\csname #1\endcsname}
```

\egpuse Reuse a previously defined figure. The figure is referred to by the name given on the `egp`, `egpx` or `egpdef` environment.

```
\newcommand{\egpuse}[2][scale=1]{%
  \@ifundefined{egp@k:f:#2}%
   {\typeout{egp: \string\egpuse: `#2' undefined!}}%
   {\egp@includegraphicx[#1]{\@nameuse{egp@k:f:#2}}{\@nameuse{egp@k:c:#2}}}}
```

\egpcalc Calculate the expression in the required argument.

```
\newcommand{\egpcalc}[2][*]{%
   \def\egp@@name{#1}%
   \def\egp@expression{#2}
   \egp@calc}
```

\egp@calc Write the commands to the `gnuplot` file. To get the calculated results in a file the `gnuplot` `table` terminal is used. The number of samples is set to the lowest possible value and the `zero` tolerance is set to 0.

```
\def\egp@calc{%
  \egp@checkfile
  \global\expandafter\advance\csname c@egpcalc\endcsname \@ne
  \egpc@@def{\egp@@name}%
  \ifthenelse{\boolean{egp@oldgp}}
      {\egpwrite{\egpcomment\space --- \theegpfile\theegpcalc.val ---}}
      {\egpwrite{print 'calculating value ----- \theegpfile\theegpcalc.val'}}
  \egpwrite{set term table; set output '\theegpfile\theegpcalc.tmp'^^J%
             set samples 2^^J%
             set zero 0^^J%
             plot [0:0] \egp@expression}
```

Here intensive usage of UN*X text utils is made to extract the calculated value out of the file `gnuplot` generated.
If the `german` option is used the decimal point character is changed to be ‘`{,}`’. This is done to avoid the additional space LaTeX inserts after the ‘,’ in math mode.
Maybe someone is able to implement all this in TeX what would make this package much more portable.

```
   \ifthenelse{\boolean{egp@german}}{%
       \egpwrite{!tail -3 \theegpfile\theegpcalc.tmp | head -1 |%
                 cut -f 2 -d' ' | sed -e 's/[.]/{,}/' %
                 >\theegpfile\theegpcalc.val}}
       {\egpwrite{!tail -3 \theegpfile\theegpcalc.tmp | head -1 |%
                  cut -f 2 -d' ' >\theegpfile\theegpcalc.val}}
```

```
  \egpwrite{!rm -f \theegpfile\theegpcalc.tmp}
%   \ifthenelse{\boolean{egp@oldgp}}{%
%       \egpwrite{load "reset.gp"}}{%
%       \egpwrite{reset}}
  \egpwrite{}}
```

`\egpc@@def`

```
\def\egpc@@def#1{%
  \global\e@namedef{egp@k:f:#1}{\theegpfile}%
  \global\e@namedef{egp@k:v:#1}{\theegpcalc}}
```

`\egp@includevalue` With this command the generated result is read into the LaTeX file. Unfortunately a trailing `\␣` is shown after the included value what is caused — as I think — by the `\input` command. There should be a way to avoid this but I don't know how. Any wizards out there?

```
\newcommand{\egp@includevalue}[2]{%
% \InputIfFileExists{#1#2.val}{\ignorespaces}%
  \IfFileExists{#1#2.val}%
               {\input{#1#2.val}}%
               {\typeout{%
                  egp: File #1#2.val\space not found:^^J%
                  egp: Process \theegpfilename\space with gnuplot and then %
                       reprocess this file.}}}
```

`\egpshowval` Calculate and include the result during the LaTeX run.

```
\newcommand{\egpshowval}[2][*]{%
  \def\egp@@name{#1}%
  \def\egp@expression{#2}%
  \egp@calc%
  \egp@includevalue{\theegpfile}{\theegpcalc}}
```

`\egpuseval` Include a previously defined value.

```
\newcommand{\egpuseval}[1]{%
  \@ifundefined{egp@k:f:#1}%
  {\typeout{egp: \string\egpuseval: '#1' undefined!}}%
  {\egp@includevalue{\@nameuse{egp@k:f:#1}}{\@nameuse{egp@k:v:#1}}}}
```

`\egpassign`

```
\newcommand{\egpassign}[2]{%
  \egpwrite{#1=#2}\egpshowval{#1}}
```

Define the file prelude: If the user specifies that the official version gnuplot-3.5 is used a file with the name `reset.gp` is generated at the start of the gnuplot run. Wherever a `reset` is done in the gnuplot-3.6 file this file is `load`ed instead.

```
\ifthenelse{\boolean{egp@oldgp}}
  {\egpprelude{save "reset.gp"}}
  {\relax}
```

Define the default prelude for the figures:

```
\egpfigprelude{set terminal postscript eps monochrome dashed "Helvetica" 17}
```

To get e.g. Computer Modern as font for the axis tics you can specify the name of a CM-Type-1 font file as fontname option of the gnuplot postscript terminal. For example:

```
\egpfigprelude{set terminal postscript eps monochrome dashed "CMSS17" 20}
```

To see the correct font in the Postscript file you have to use the appropriate fontmap when calling dvips or you have to download the file `cmss17.pfb` as header file. The error message of dvips can then be ignored.

Reset all options to their default values after every egp, egpx and egpdef environment. As mentioned above the file `reset.gp` that is generated at the start of the gnuplot run is `load`ed to implement the new `reset` command of gnuplot-3.6beta if the user didn't specify gnuplot36beta as package option.

```
\ifthenelse{\boolean{egp@oldgp}}{%
  \egpfigepilog{load "reset.gp"}}{%
  \egpfigepilog{reset}}
```

You can configure egplot by putting the appropriate commands in the file egplot.cfg that has to be located where TEX can find it.

```
\InputIfFileExists{egplot.cfg}
                  {\typeout{egp: Using configuration file 'egplot.cfg'}}
                  {}
⟨/style⟩
```

A Driver File

```
⟨*driver⟩
\documentclass[a4paper]{article}
\usepackage{doc}
\usepackage{multicol}

\IfFileExists{mflogo.sty}%
  {\usepackage{mflogo}%
   \def\GP{\textsf{gnuplot}}%
   \def\EGP{\textsf{egplot}}%
   \def\EMP{\textlogo{EMP}}}%
  {\def\GP{\textsf{gnuplot}}%
   \def\EMP{\textsf{EMP}}%
   \def\EGP{\textsf{egplot}}}

\usepackage[gnuplot35]{egplot}
%\usepackage[gnuplot36beta]{egplot}

\setlength{\parindent}{0pt}
\def\manindex#1{\SortIndex{#1}{#1}}
⟨manual⟩\OnlyDescription
\EnableCrossrefs
\RecordChanges
\CodelineIndex
\DoNotIndex{\def,\gdef,\long,\let,\begin,\end,\if,\ifx,\else,\fi}
\DoNotIndex{\immediate,\write,\newwrite,\openout,\closeout,\typeout}
\DoNotIndex{\font,\jobname,\documentclass,\char,\catcode,\ }
\DoNotIndex{\CodelineIndex,\DocInput,\DoNotIndex,\EnableCrossrefs}
\DoNotIndex{\filedate,\filename,\fileversion,\logo,\manfnt}
\DoNotIndex{\NeedsTeXFormat,\ProvidesPackage,\RecordChanges,\space}
\DoNotIndex{\begingroup,\csname,\edef,\endcsname,\expandafter}
\DoNotIndex{\usepackage,\@ifundefined,\ignorespaces,\item,\leavevmode}
```

```
\DoNotIndex{\newcounter,\newif,\par,\parindent}
\DoNotIndex{\relax,\setcounter,\stepcounter,\the,\advance}
\DoNotIndex{\CurrentOption,\DeclareOption,\documentstyle}
\DoNotIndex{\endgroup,\global,\hfuzz,\LaTeX,\LaTeXe}
\DoNotIndex{\macrocode,\@makeother,\OnlyDescription,\PassOptionsToPackage}
\DoNotIndex{\ProcessOptions,\RequirePackage,\string,\textsf,\unitlength}
\DoNotIndex{\@bsphack,\@esphack,\@nameuse,\@ne,\active,\do,\dospecials}
\DoNotIndex{\errhelp,\errmessage,\ifcase,\IfFileExists,\includegraphics}
\DoNotIndex{\manindex,\SortIndex,\newcommand,\newtoks,\or,\origmacrocode}
\DoNotIndex{\alpha,\displaystyle,\frac,\sin,\texttt}
```

Cut the line breaking some slack for macro code which might contain long lines (it doesn't really hurt if they stick out a bit).

```
\let\origmacrocode\macrocode
\def\macrocode{\hfuzz 5em\origmacrocode}
\begin{document}
  \DocInput{egplot.dtx}
\end{document}
⟨/driver⟩
```