

Using Graphics in L^AT_EX Documents

Eric Behr, NIU Mathematical Sciences

February 1994 (rev. 3/1995, 6/2004)

1 Generating the picture

Graphs and plots of various kind can be created with a multi-purpose package such as Maple or Mathematica, but this is often an overkill. *Gnuplot* is a simpler and more convenient way of doing this. See section 6 for details.

Complicated or irregular figures which cannot be easily obtained as graphs of functions are a different matter. Perhaps the best way to produce them is to use `xfig`, which is available within the X-Windows environment on our system. Becoming skilled in using `xfig` takes quite a bit of time and effort, but it's a must if you plan to use illustrations in L^AT_EX – we simply don't have any better alternative for hand drawing.

After the picture has been created in `xfig`, it has to be saved as a PostScript™ file. Hit the “Export...” button; default settings of ‘Portrait’ orientation and ‘Encapsulated PostScript’ format are fine. The figure can also be scaled by entering a factor (in percent) in the appropriate field – that lets you work comfortably in `xfig` even though the resulting picture is meant to be small. The scaling can also be done later, within the L^AT_EX document. Type a unique name of the graphics file in the ‘Output filename’ box, and press ‘Export’. The picture will be saved in the correct format. It is customary to use filenames ending with `.eps` for EPS files.

Remember that `xfig` cannot read PostScript files back in, so if you want to edit your figures later, you will have to keep the `.fig` version and generate PostScript again after the picture has been altered. Save your `.fig` files often! In general, we have no facilities for editing PostScript files in any user-friendly way.

2 Including a PostScript file in a L^AT_EX document

Suppose that you have stored a figure in a PostScript file `Fig1.eps` in the same directory as the L^AT_EX source file. Make sure that the T_EX file uses the `graphicx` style (this is done by saying `\usepackage{graphicx}` in the preamble.

Type the following T_EX commands where you want the picture to appear:

```
\setlength{\unitlength}{1in}
\begin{center}
\begin{picture}(3.5,1.5)
```

```

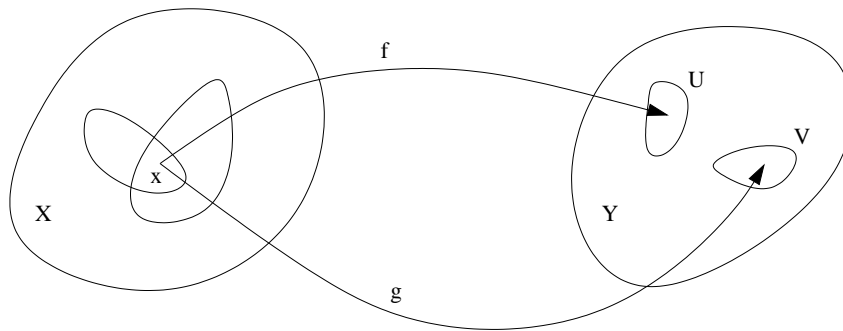
\put(-0.4,0){\includegraphics{Fig1.eps}}
\end{picture}
\end{center}

```

The numbers 3.5 and 1.5 specify the width and height of the picture (in inches), while the `put` command gives coordinates of the bottom left corner of the picture. Note that these dimensions only tell \LaTeX what kind of empty box it should use in typesetting the document. It is perfectly legal to specify a box of height 0, in which case \TeX will not leave any space for the image, but the picture will still be printed – only it’s likely to overlap with the text. Similarly, the horizontal size can very well be set to 0 - if the figure is displayed rather than in the middle of the text, as is usually the case, then it doesn’t matter how much horizontal blank space \TeX reserves for it.

The picture dimensions mentioned above *do not* scale the drawing. To make the picture itself smaller or larger, you should use `\includegraphics[scale=0.8]{file.eps}`.

It will take a few tries to get it right. See the \LaTeX manual and the *\LaTeX Graphics Companion* for more details about pictures. Here’s what I got from the above example:



3 Labels

The text generated by `xfig` uses typefaces which are different than the \LaTeX fonts; such a mixture doesn’t look elegant. You can make \LaTeX substitute its own text as follows.

While still in `xfig`, use the typing tool to put some unique strings (keywords) wherever you want your labels to appear; I used ‘f’, ‘g’, etc. in the example below. Make sure that the *left hand side* of the label on the picture is where you want your \LaTeX text to *begin*; this, again, is largely a matter of trial and error. Also remember that the substituted \LaTeX text will *not* have the same scaling as you specified for the figure – you have to control the type size using \LaTeX commands such as `\scriptstyle`.

Now modify your \LaTeX source slightly. First, in addition to `graphicx`, use the `psfrag` package in the document heading, e.g. `\usepackage{psfrag}`. At the point where you want to include the picture, type this:

```

\begin{center}
\setlength{\unitlength}{1in}
\begin{picture}(3.5,1.7)

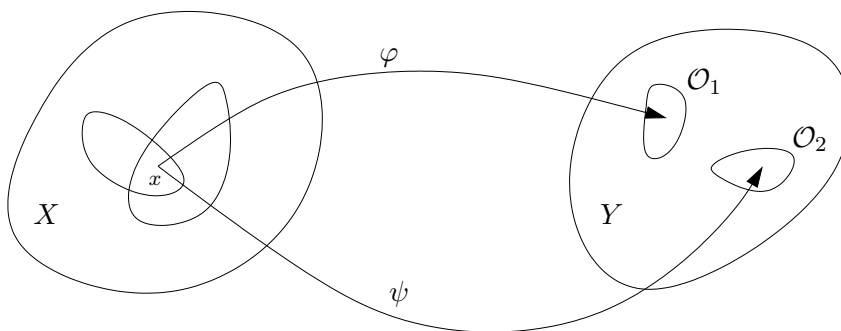
```

```

\psfrag{f}{$\varphi$}
\psfrag{g}{$\psi$}
\psfrag{x}{$\scriptstyle x$}
\psfrag{X}{$X$}
\psfrag{Y}{$Y$}
\psfrag{U}{${\cal O}_1$}
\psfrag{V}{${\cal O}_2$}
\put(-0.4,0){\includegraphics[scale=0.8]{Fig1.eps}}
\end{picture}
\end{center}

```

In this example we have also centered the picture, just for the fun of it. When you run this through \LaTeX and then print the .dvi file, you will see a figure with the new text substituted in place of the labels:



Note that `psfrag` substitutions have local scope if they appear inside some environment (e.g. ‘`picture`’ in our example), while if you put them outside of any environment, they will apply globally. If such ‘global’ substitutions are redefined in the document, you’ll get a lot of warnings, and \LaTeX may get confused. In summary, if you want to use `psfrag{gamma}{γ}` *everywhere* in the document, insert it only once in the top-level environment; but if there is a chance that the keyword `gamma` will later be reused to mean something else, put it after `\begin{picture}`.

4 Viewing the document with figures

Since positioning figures in \LaTeX is a hit-and-miss affair, you need to learn how to preview the finished result before printing the final version, or else you’ll waste tons of paper. Unfortunately the usual previewer (`xdvi`) cannot completely handle PostScript graphics.

The way to view the finished result most accurately is as follows. After you’ve run \LaTeX (preferably twice, so all references get resolved), you have a .dvi file, e.g. `mydoc.dvi`. Next, you need to generate a PostScript file containing the entire output. At the Unix prompt type

```
rainier.behr % dvips mydoc.dvi -o
```

Be sure that you don't have some other important file called `mydoc.ps` – it will be overwritten! Now view it with the command

```
rainier.behr % ghostview mydoc.ps
```

When you are satisfied with what you see, you can print it from within ghostview, or from the Unix prompt using `mprint mydoc.ps` or with some other printing command. To print just a few pages of a long document from within ghostview, click the middle mouse button next to the page number you want to go to, pull down the “Page” menu, and select “Mark”. An asterisk will show up next to the page number. Repeat this with all the pages you want to print. Then pull down the “File” menu, and choose “Print marked pages” – that's all there is to it.

5 Summary

Let's summarize the process described in this note:

- Create the graphics figure using `xfig` and save the `.fig` version; then export the figure as Encapsulated PostScript (or generate a PostScript illustration in some other way)
- Use `\begin{picture}...`, `\put(0,0){\includegraphics{figure.eps}}`, etc. in the \TeX file, along with the `psfrag` substitutions
- `latex mydoc.tex` (twice, as usual)
- `dvips mydoc.dvi -o`
- `ghostview mydoc.ps`

Naturally, whenever the figure itself is modified you have to repeat the entire cycle. If you only change the \LaTeX file, only the last three steps need to be repeated.

6 Gnuplot

To view graphs generated with `gnuplot` you should be in X-Windows. Start up the program by typing `gnuplot`. There is extensive built-in documentation which you can access by typing `help`.

`Gnuplot` can be used to graph functions of one or two variables, and to plot 2-D and 3-D data sets. You can graph two or more functions at once. Ranges and labels of axes, tickmarks, graph styles, titles etc. can all be modified with little effort.

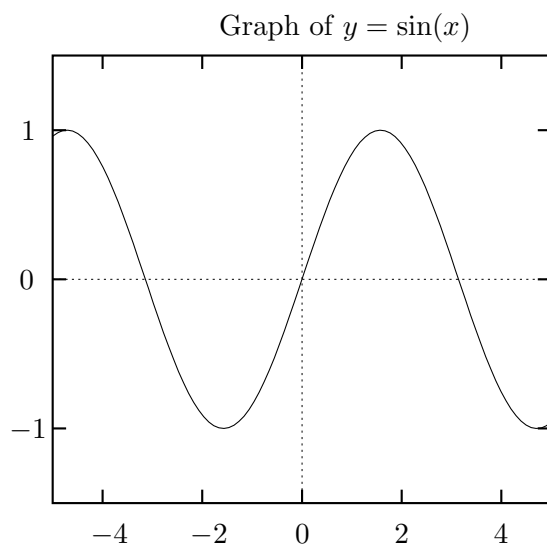
The following sample session produces a graph and stores it in a file (which you can inspect using `ghostview`, and later include in the \LaTeX document):

```

gnuplot> set term postscript portrait
gnuplot> set out "Sine.ps"
gnuplot> set title "sine function"
gnuplot> set size 0.45,0.5
gnuplot> set ytics -1,1,1
gnuplot> plot [-5:5] [-1.5:1.5] sin (x) notitle
gnuplot> quit

```

Obviously, we can use `psfrag` again to substitute \LaTeX text for labels as described in previous sections. Here's the result:



Let's end with one more example, obtained with

```

gnuplot> set nozeroaxis ; set nolabel ; set noborder
gnuplot> set noxtics ; set noytics ; set noztics
gnuplot> splot x**2 -2*x*y -2*y*y notitle

```

$$z = x^2 - 2xy - 2y^2$$

