Making posters with LaTeX
A guide to doing things the simple way.

Hugh C. Pumphrey
School of GeoSciences
The University of Edinburgh

LaTeX can be a useful tool for producing conference posters, assuming that you like it already. I don't imagine that fans of PowerPoint or Illustrator are going to take it up in droves. But they don't usually care how beautifully their equations are typeset. This poster explains how I have gone about typesetting posters with LaTeX and is itself an example. I certainly don't claim that it is the only way, or the best way of doing this.

1 Introduction

Time was when people constructed conference posters by pinning individual pages to the board provided. You were jolly sophisticated if you used a coloured backing card for each page. The poster printer has become a common item in large university departments and computing centres and no self-respecting conference-goer nowadays wants to be seen producing their poster with anything else. The poster printer is still an expensive, temperamental and very slow device, however, often minded by a technician who does not suffer fools gladly. The running costs are high, too, at £10 per poster for plain paper and £30 for glossy. So everyone wants their poster to print right first time.

You can lay out a poster with a vector drawing package (anything from the free and cheerful xfig to the nastily expensive Illustrator) or a specialist layout package (free scribus, expensive Quark/InDesign). You can even use Microsoft PowerPoint to do the job. If you know and love one of those packages it is probably your best bet. If, however, you know and love LaTeX and you care deeply about how your mathematics are typeset, you may want to use it to make posters. My aim in this poster is to show you how it can be done, allowing you to concentrate a lot on your material and a little on your layout and to worry hardly at all about the undeniable vileness that LaTeX sometimes seems to impose on you. To that end, this poster requires nothing that does not come with an ordinary *modern* installation of LaTeXand ghostscript.

2 Paper and font sizes

Traditionally, LaTeX has been wedded to a small number of paper sizes and fonts. For making posters we need better control over both of these. In particular, we need to be able to use both the 36 inch or 48 inch paper on the poster printer, but also to produce A4, A3 (or US letter) size copies for handing out to interested parties. We may require fonts in a wide variety of sizes, depending on how far from the poster the audience are likely to be standing.

2.1 Paper sizes

To control these without giving yourself a headache, you need the geometry package. The geometry package is your friend. Most TeX distributions provide it, along with a very helpful manual. (The manual is likely to be somewhere like `/usr/share/doc/texmf/latex/geometry/geometry.pdf`) Make sure that you have version 3 of the geometry package (which came with teTeX 2.0 or later) and not version 2 or earlier (which came with teTeX 1.x). (Local note: This means you need to run LaTeX on a Solaris 9 machine). The geometry package allows you to control both the paper size that LaTeX thinks you want as it is setting the type and, quite independently, the size that the final poster is produced at. For example, you can say

```
\usepackage[dvips,a3paper,landscape,hmargin=1.2cm,tmargin=.8cm,
    bmargin=1.2cm,mag=707]{geometry}
```

and get your typesetting done for A3 paper, and then shrunk to A4 size for output. If you set `mag=3075` instead, the poster will be typeset exactly the same but will be scaled for 36 inch paper. This is the only reliable way to get output for different sizes of paper. The various post-processing programs that are supposed to re-size your postscript file (such as `poster` and `psresize`) tend always to do the wrong thing with LaTeX+ dvips output. I'm not sure what stage of the process is at fault, but they are best avoided. You can specify quite arbitrary paper sizes and aspect ratios if you like:

```
\usepackage[dvips,papersize={300mm,350mm},margin=0.8cm]{geometry}
```

Note that it is important to use the `dvips` argument so that the geometry package knows that the output will be processed by dvips.

2.2 Font Sizes

LaTeX has traditionally provided only a small selection of font sizes: tiny, scriptsize, footnotesize, small, normalsize, large, Large, LARGE, huge, and Huge. By default, these correspond to 5, 7, 8, 9, 10, 12, 14.4, 17.28, 20.74 and 24.88 point respectively. For an A4 document that's about all you would need. However these may not match up to your requirements for your poster. You can work around this to a certain extent by choosing a paper size that fits with the available fonts for your purposes and then scaling the whole result with the `mag=` argument of the geometry package. This is *definitely* the best way to get the fonts the size you want. If that isn't enough, you need to use the `type1cm` package, which allows you to scale LaTeX's fonts to any size. For example `\fontsize{1cm}{1.4cm}\selectfont` allows you to use

letters that are 1 cm tall with 1.4 cm line spacing.

(Note that the line spacing value doesn't have an affect in the above example. I think it only kicks in at the start of a section. As a guide the second argument should be 1.2 times as big as the first.) You can also use something like

```
\newcommand{\HUGE}{\fontsize{40}{45}\selectfont}
```

to define a new font size or

```
\renewcommand{\Large}{\fontsize{30}{35}\selectfont}
```

to re-size one of the existing standard sizes. This poster was typeset as if it were on an A2 sheet. This box is made with the fonts at their default sizes, for the rest of the poster they are scaled up by $\sqrt{2}$. There is a serious limitation to the business of re-scaling the fonts as the small LaTeX fonts are *not* simply shrunk-down versions of the large ones — they are wider so that they are more readable at a small size. This means that magnifying or shrinking the fonts by more than about 1.5 times is *not* a good idea.

3 This example

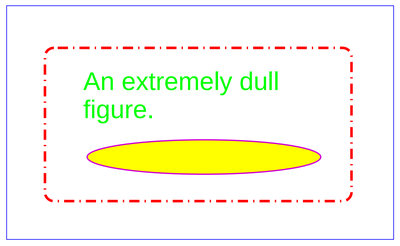
Here I have made extensive use of minipages to lay the text out in an assortment of rectangles. (I also have a more formal example which uses multiple columns to lay out the poster — it is called `exampleposter.tex`.) To delimit the rectangles of material with boxes there are two possibilities: the `fancybox` package (used for this box) and the `pstricks` package (used for the beige box to the right). We can set out one box of each type like this:

```
\hspace{0.08\textwidth}{\Inred \shadowbox{
\begin{minipage}{0.35\textwidth}
  {\Inblue Your text goes here}
\end{minipage}}}\vspace{0.1\textwidth}

\hspace{0.26\textwidth}\psframebox[linecolor=red,
fillcolor=yellow,fillstyle=solid]{
\begin{minipage}{0.23\textwidth}
Your other text goes here
\end{minipage}}
```

The box will be automatically sized so that your contents, whatever they are, fit inside it. That's the easy bit. The hard bit is arranging your rectangles on the page. LaTeXwill not help you much here. The commands that move your box about are the `\hspace` and `\vspace` and LaTeXwillfully ignores them unless they are positioned as shown, with the horizontal positioning done just before the box starts, and the vertical positioning done just after the previous box finishes.

4 Figures



Postscript figures may be included with `\includegraphics` in the usual manner.

This box is made by the pstricks command `psframebox`. This means that we can do extra magic with it, such as give it a coloured background. There may be things that don't work: the main one I have found is the `verb` command and the `begin{verbatim}` and `end{verbatim}` commands. (That is why there are no backslashes in this box.) We can do inline maths like $x^2 + y^2 = z^2$. Equations like

$$y = \int_0^1 x^2 dx$$

also seem to work. Finally, you can include figures with `includegraphics`

5 Printing it

The result of typesetting this poster with `latex myfile ; dvips myfile -o` should be a postscript file which is ready to go to an a4 postscript printer (or via ghostscript, to any other a4 printer). Alter the `mag=` argument and re-process to make a file printable on a different size of paper. You can preview the poster with any postscript previewer: I prefer `gv`.

Whether your poster prints correctly will depend on the details of your poster printer. I can only report on our own HP2500CP printer (known as vc23 to University of Edinburgh folk). To print to this from a Unix machine using the old BSD lpr/lpd print system (e.g. `ginger.geos.ed.ac.uk`) do:

```
lpr -Pvc23 -Ga0u -b myfile.ps
```

Unfortunately, machines running the newer CUPS printing system cannot talk to vc23 yet. This printer has 5/8 inch non-printable margins down the edge of the 36 inch paper. This poster should just fit inside them. Your printer probably has such margins too and they may be even wider.

One issue of concern is the orientation of your poster. If you use `mag=1537` so that the long edge is 36 inches long, the poster comes out with its long edge across the roll. If, however, you use `mag=2174` so that the short side is 36 inches long, it still comes out with the long edge across the roll. You therefore lose 40% of your poster. (I suspect that newer poster printers are less stupid about fitting the poster onto the paper.) One possible fix for vc23 is to forcibly rotate the PostScript file before you print it. One way to do this is to convert to PDF and back to postscript like this:

```
ps2pdf myfile.ps myfile.pdf
pdftops -paperw 2602 -paperh 3679 myfile.pdf rotated.ps
```

In the second command, paperw and paperh are width and height in *points*, that is, 1/72.27 inches — this assumes that you have mag=2174. There must be a better way to rotate the poster, however.

Although `gv` is the nicest way to preview your poster as you prepare it, it is not helpful for checking the orientation before printing. This is because `gv` seems to make an attempt to ensure that it shows you the text horizontally. Instead, you need to use `gs` directly:

```
gs -r10 myfile.ps
```

(The `-r10` option shrinks the preview to a sensible size.) The direction running across the roll of paper is horizontal on your screen. With the above examples, `gs` shows `myfile.ps` with the text running horizontally and `rotated.ps` with it running vertically. In my experience, this means that `rotated.ps` is going to print correctly.