Labelling Figures in TEX Documents

Alan Hoenig
Department of Mathematics, John Jay College/City University of New York
Mailing address: 17 Bay Avenue, Huntington, NY 11743 USA; (516) 385-0736.
Bitnet: ajhjj@cunyvm

Abstract

With practice, it becomes a relatively quick job to use METAFONT to generate figures that TEX can typeset in a document. How, though, may one use TEX to typeset labels that may be needed at various points in the figure? This presentation describes one approach. METAFONT is instructed to record coordinates in the font file which TEX may access and use as kerns for positioning label boxes.

Introduction

Many people perceive a problem in including graphics within TEX documents, but that problem no longer really exists. Thanks to the `\special` command in TEX, it's possible to include a wide variety of graphics within the printed document. The general strategy is as follows.

Ask TEX to leave the proper amount of white space in the document for the figure. Prepare a separate file by some other means which contains instructions for generating the figure. "Pull in" this separate file to the `dvi` file by means of the `\special` command. Finally, use a device driver which is alive to the possibility of external graphics files. Many such device drivers now exist.

There are, though, disadvantages to this approach. Primarily, the graphics file is not really a part of the TEX source file. This means that it will be impossible to preview this file unless you use a particularly smart previewer (one that also responds to `\special` commands and is intelligent enough to know how to display this material on the screen).

Need for Labels

Another disadvantage makes itself felt whenever a figure needs labels. How may they be included so that they are properly positioned? Some workers include the label as part of the graphic, but this leads to a visual inconsistency since the label font is most likely not any of the Computer Modern fonts that TEX is apt to be using. Ideally, we'd like TEX to typeset the labels, so they will be typeset as handsomely as the rest of the document.

In this article I'd like to describe one labelling approach that has worked well for me. I use METAFONT to generate the graphic as a font. While in this process, I decide which points in the figure need labels (although I need not decide on the labels themselves), and METAFONT incorporates the coordinates of these points in the font. Then, when TEX includes the figure, it may access these coordinates which it uses as amounts by which to move right or left and raise up or down a box containing the text of the label.

Other Work in this Area

Little work has apparently been done in this area. Rick Simpson was apparently the first to suggest (at least in print) that METAFONT be used as a drawing engine. His article [1990] is worth searching out.

One other effort in this field is the Metaplot macro package created by Patricia Wilcox [1989]. She showed how it was possible to enlist an interface between the user and METAFONT. In her case, it was a species of commonly used plotter. Various plotter commands were made to correspond to METAFONT commands, and she developed a package whereby it was possible to create sophisticated and pleasing graphics. (She is an individual blessed with considerably above average artistic skills.) These are amply illustrated in the article cited above. (Certain of my strategies were inspired by suggestions from Ms. Wilcox and Tom Rokicki, and I am happy to acknowledge their unwitting help.)

John Hobby [1989] is adopting a different approach. He is tailoring METAFONT so that it directly produces PostScript output rather than the

generic font format METAFONT normally produces. Although it will be possible to generate Computer Modern fonts in PostScript format, this MetaPost program is primarily a vehicle for generating figures and logos. His program will use a different approach for incorporating labels within the picture.

The METAFONT Part

We begin by seeing how the METAFONT portion of this project works. I imagine each figure to be the "A" character of a special-purpose METAFONT font that contains no other characters. TEX contains no requirement that an "A" look anything like a *real* "A", and does not have any preconceived notions of the proper size of such a character, so no one complains if the "A" we ask METAFONT to draw looks nothing like a real "A" but like the figure we need in our document. Suppose the METAFONT file containing the figure-drawing instructions is `figfont.mf`. Then, assuming you've done all the METAFONT things properly, you would typeset this figure in your TEX document by declaring the font:

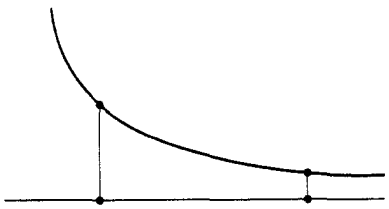
```
\font\figfont=figfont
```

and including the statement

```
{\figfont A}
```

where the figure is to appear. (Don't forget those enclosing braces!)

Here is a figure typical of one that might appear in a math text. (In fact, it appears somewhere in the first volume of Don Knuth's *Art of Computer Programming*.) The curve represents a portion of the graph of the function $f(x) = 1/x$. (Therefore, its METAFONT file is `1onx.mf`.)



It may be helpful to examine the METAFONT code to produce this drawing.

```
mode_setup;
u#=12pt#; nib#=.5pt#;
if (mode=smoke) or (mode=proof):
 u#:=.5pt#; nib#:=.1pt#; fi
define_pixels(u, nib);
beginchar("A", 12u#, 6u#, 0);
pickup pencircle scaled 2nib;
z1=(w/8,h); z2=(w/4, h/2);
z3=(w/2, h/4); z4=(w,h/8);
 % points on the curve
path p; p=z1..z2..z3..z4;
draw p; % draw the $1/x$ curve
z.x=point 2.6 of p; % another point
z5=(x2,0); z6=(x.x,0);
 % points on the $x$-axis
pickup pencircle scaled nib;
draw origin--(w,0); % bottom axis
draw z2--z5; draw z.x--z6;
 % vertical struts
pickup pencircle scaled 6nib;
drawdot z2; drawdot z.x;
drawdot z5; drawdot z6;
endchar; bye.
```

Most of these statements are reasonably self-explanatory, provided you are familiar with the rudiments of METAFONT syntax. The variable `nib#` governs the diameter of the pen we use for drawing, and `u#` is an ad hoc measure of length that's convenient for scaling the entire drawing. Of the other variables in this program, `w` and `h` are the width and height of the figure, and the quantities `z1`, `z2`, and so on (including `z.x`) refer to special points in the figure.

The lines

```
u#=12pt#; nib#=.5pt#;
if (mode=smoke) or (mode=proof):
 u#:=.5pt#; nib#:=.1pt#; fi
```

must be troubling to an ardent METAFONTer; they imply `u#` and `nib#` take on device dependent values, contrary to the METAFONT spirit. This device dependent code is dictated by limitations within my monitor. METAFONT is accustomed to working with pieces of type that are (roughly) ten points square. During on the screen development (when `mode` is either `smoke` or `proof`), METAFONT uses the entire screen to display the character. When the characteristic length of the character is several picas, METAFONT will only display small portions of the letter, unless `u#` is sufficiently small so that there will be room enough to display the character on the video monitor. The lines above attempt to implement this strategy. The characteristic dimensions `u#` and `nib#` have the values we'd like for any font making activity. However, during

the development mode, they are much smaller to facilitate reviewing the work on the video screen.

METAFONT Talks to TEX

We could use TEX to provide an identifying caption for this picture, but it would be more helpful to include labels. Knuth provided labels at each of the four circle like dots — can we?

Hackers are used to referring to both TEX and METAFONT as full-featured programming languages, but METAFONT doesn't completely fill that role. For one thing, METAFONT has no ability to create files except for `log` and `gf` files. Therefore, we cannot write the coordinates of the label points to files for use by TEX.

The key idea is to recognize the usefulness of the `fontdimen` parameter. Normally, METAFONT uses them to record universal constants for a particular font, such as the width of a quad or the interword spacing. There appears to be no upper bound on the number of allowable `fontdimen` parameters, nor constraints on the uses to which they be put. I therefore felt free to use them to store the coordinates of each label point. Each point uses two fontdimen parameters — one for the x coordinate and one for the y coordinate.

The file `convert.mf` contains the definitions of macros such as `convertz_` which transform the coordinates into a useful format. Therefore, we need the statement

```
input convert;
```

right after the `mode_setup` command. Before the `endchar` command, we need an additional line

```
fontdimen10: convertz_(z2,z.x,z5,z6);
```

the macro `convertz_` converts the list of pairs to numbers which follow the proper `fontdimen` syntax.

The TEX Part

Once the figures have been properly "METAFONT ed," it's easy to include them in a TEX document, as we have already discussed. But it's also easy to apply labels. For that, we use a set of `\pointing` macros, similar to those in Appendix D of *The TEXbook*. If `\x` and `\y` are TEX dimen registers, it's easy to get their values from the proper `\fontdimen`:

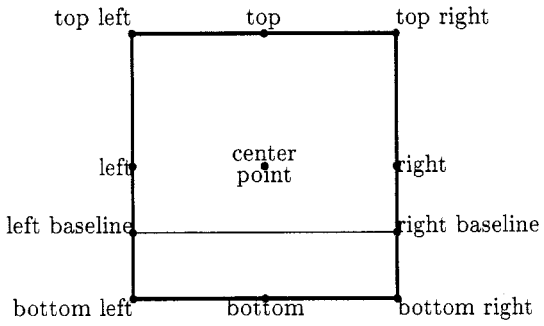
```
\advance\fontdimencount by 1
\x=\expandafter \the\fontdimen
 \the\fontdimencount \figfont
```

(and similarly for `\y`). Then, if `\box\labelbox` contains the text of the label, a construction like

```
\rlap{\kern\x \raise\y \box\labelbox}
```

will put the label in the proper position.

Actually, that's not quite true. There are eleven reference points associated with any label box, as the following diagram makes clear.



It's easy to create a set of macros which position a particular reference point at the point of the label. Thus, for example, the label "top right" was typeset using the command

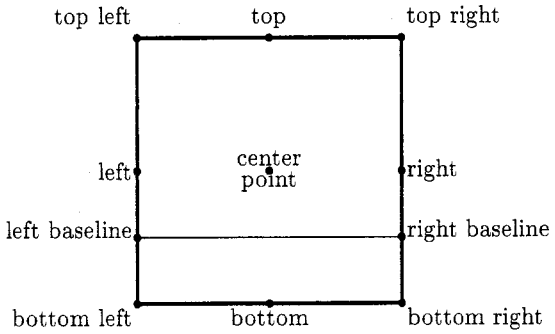
```
\blpoint{top right}
```

After all, the bottom left corner of the hbox containing "top right" must be positioned at the label point.

Still, this is not yet the full story. The reader who examines this figure closely will note that the labels seem somewhat cramped onto the label point. It may be necessary to leave some extra space. Commands like `\hskip2pt` or `\hskip-3pt` will move the label to the left or right, and it is easy to define macros `\up` and `\down` so that (for example)

```
\up3pt
```

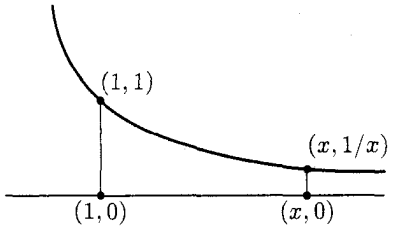
will help adjust vertical placement. We can use these considerations to produce a better reference point diagram



The following table lists all the pointing macros and their associated orientation. Some reference points have two macros. The user needing to position the top left point of a label box may use `\tlpoint` or `\ltpoint` so the question of remembering which of these two is correct does not arise.

Orientation	Macro Name(s)
top left	`\tlpoint`
	`\ltpoint`
top	`\tpoint`
top right	`trpoint`
	`\rtpoint`
right	`\rpoint`
bottom right	`\brpoint`
	`\rbpoint`
bottom	`\bpoint`
bottom left	`\blpoint`
	`\lbpoint`
left	`\lpoint`
left baseline	`\point`
	`\lBpoint`
	`\Blpoint`
right baseline	`\rBpoint`
	`\Brpoint`
central point	`\cpoint`

Using these tools, we can redo the Knuth's figure from *ACP*, this time with labels:

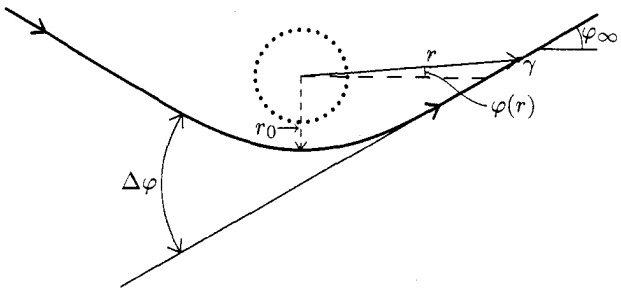


Here are the macro calls that I needed.

```
\font\figfont=lonx
\beginfig
\lbpoint{$(1,1)$}%
\lbpoint{\up2pt $(x, 1/x)$}%
\tpoint{\down 2pt $(1,0)$}%
\tpoint{\down 2pt$(x,0)$}%
\endfig
```

An Example

As a final example, here is a figure drawn from a general relativity text. The actual context or meaning of the figure is unimportant (it relates to the deflection of light due to relativistic effects), but it illustrate a number of interesting effects that are possible with METAFONT.



This example shows that METAFONT can create many important visual aids, such as arrow heads at arbitrary orientations, dotted lines, dashed lines, and so on.

Call to Arms!

One aim of this work is to strike fire into the heart of some ardent METAFONT artists. The TEX and METAFONT community could benefit from the creation of a macro package that stands in the same to METAFONT as does LATEX or AMS-TEX to TEX. Any takers?

Bibliography

Hobby, John D., "A METAFONT-like System with PostScript Output," *TUGboat* 10(4), pages 505–512, 1989.

Simpson, Richard O., "Nontraditional Uses of METAFONT." Pages 259–272 in *TEX: Applications, Uses, Methods*, Malcolm Clark, ed. (London: Ellis Horwood Ltd., 1990).

Wilcox, Patricia, "Metaplot: Machine-independent line graphics in TEX," *TUGboat*, 10(2), pp 179–187, 1989.