Inserting Postscript Figures Using the Nashboro Press Macros

Larry L. Schumaker

Abstract. The purpose of this note is to explain how to use the Nashboro Press macro files to insert postscript figures into a TEX document.

§1. Introduction

There are a number of public domain TEX macros which can be used to insert postscript files into a TEX document. I have found the macro file `epsf.tex` to be particularly easy to use. To use this macro file, you need three things:

1) a copy of `epsf.tex`
2) a dvi driver which is capable of working with `epsf.tex`. One which works on unix machines is `dvips`.
3) postscript files of the figures you wish to insert. These can be created using any convenient software.

The macro file `epsf.tex` should already be in your TEX installation. If it is not, you can download it from the conference site. You do not need to input it into your document as it already read in by the `nash03.tex` macro file.

Advances in Constructive Approximation
XXX (eds.), pp. 1–7.

ISBN 0-9728482-x-x

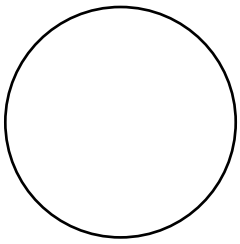

Fig. 1. A circle.

§2. Inserting a Single Figure

The basic macro for inserting a figure is called `\psone`. It has two parameters. The first parameter gives the name of the file containing the figure, and the second parameter sets the size of the figure in inches (1 inch = 2.5 cm).

To illustrate the use of `\psone`, I will now insert a figure called `schumaker1.fig`. This is a postscript file, but instead of calling it `schumaker1.ps` I have named it `schumaker1.fig`. This helps me remember that it is a figure, and prevents me from deleting it accidently when removing all postscript files in a directory. Here is the TEX code for inserting my figure:

```
\topinsert
\psone{schumaker1.fig}{1.0}
\figcap{1}{A circle}{0}
\endinsert
```

In inserting this figure, I have also made use of a macro in `at03.tex` called `\figcap` which is used to create a caption. It has three parameters. The first gives the figure number, the second gives the actual caption, and the third gives the number of inches of space which you wish to insert between the figure and the caption. When inserting postscript figures with proper bounding boxes, this would generally be 0. If you use `\figcap` to insert space for a figure to be pasted in, this parameter would be the amount of space to leave for the figure.

To get a smaller version of the same figure, I simply change the second parameter in `\psone` from 1.0 to .6:

```
\topinsert
\vskip .2 in
\psone{schumaker1.fig}{.6}
\figcap{2}{A smaller circle}{.2}
\endinsert
```

The size of the first figure is set to 1.0 inch while the size of the second one is set to .6 inch. Unfortunately, these numbers may not always give

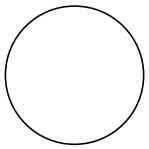

Fig. 2. A smaller circle.

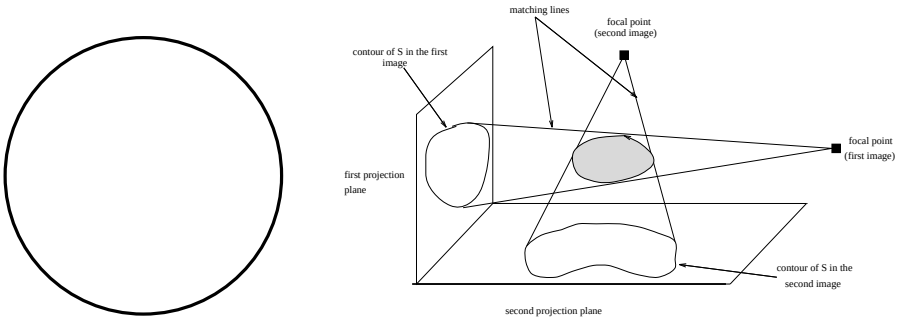

Fig. 3. Two figures side by side.

the actual size of the figure — it depends on how the postscript file was created. Thus, it may take some experimentation to find the setting for this parameter which will give a figure of the desired size.

In Fig. 2, I have used `\vskip .2 in` to insert an extra .2 in of space above the figure. In addition, by choosing .2 as the third parameter in `\figcap`, I have inserted .2 inches of space between the figure and the caption. It is also possible to skip by a negative number of inches to take space out.

§3. Inserting Figures Side by Side

To insert two figures side by side, I have written a macro called `\pstwo`. Its use is completely analogous to the use of `\psone`:

```
\topinsert
\pstwo{schumaker1.fig}{1.2}{schumaker3.fig}{2.4}
\figcap{3}{Two figures side by side}{0}
\endinsert
```

Because the sizes of these two figures in the postscript files were not the same, to get them to match in vertical size, I had to set the parameter controlling the size of `schumaker1.fig` to 1.2, and the parameter controlling the size of `schumaker3.fig` to 2.4. In practice some experimentation is usually required to size the figures correctly.

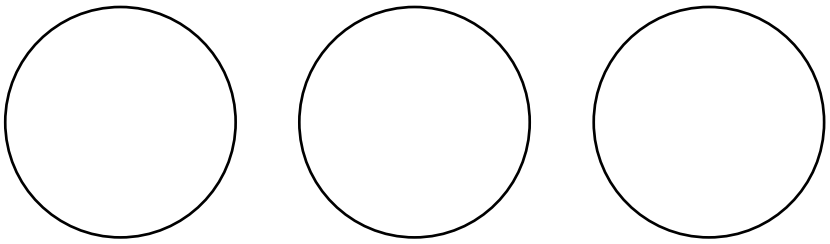

Fig. 4. Three figures side by side.

The right-hand figure in Fig. 3 includes some writing which is really too small to read. If you want to include words on your figure, be sure they are written large enough so that they can be read at the final size.

The same idea can be used to put three or more figures side by side. Here is an example with three figures:

```
\topinsert
\psthree{schumaker1.fig}{1}
{schumaker1.fig}{1}
{schumaker1.fig}{1}
\figcap{4}{Three figures side by side}{0}
\endinsert
```

§4. Creating the Postscript Files

A postscript file is identified by its first line which should begin with the two symbols `%!`. These symbols may be followed by a comment identifying the software which produced the file. If these two symbols are not present and you send the file to a laser printer, you will probably get a line-by-line print-out of the entire file (which can be huge). Here is the first line of the file `schumaker1.fig`:

```
%!PS-Adobe-2.0 EPSF-2.0
```

Postscript files describing figures can be created by a variety of software. These include

1) LaTeX
2) PicTeX
3) Mathematica
4) SGI ShowCase
5) MATLAB.

It is also possible to create your own postscript files directly, see the manuals [1,2]. In any case, if you do use postscript, make sure that you give us a complete file which prints correctly on a standard laser printer, and if possible verify that it can be inserted into your TeX file correctly.

§5. Bounding Boxes

The macros in `epsf.tex` will work with most postscript files, provided that the file includes a line which describes the size of a bounding box enclosing the figure. Here is the line for the file `schumaker1.fig`:

```
%%BoundingBox:  18 679 104 765
```

The four numbers here are in postscript units. There are exactly 72 postscript units per inch. The first pair of numbers describe the location of the lower-left corner of the bounding box relative to the *origin*, which is taken to be the lower-left corner of the page. In this case the lower-left corner of the bounding box is 18 units to the right and 679 units up from the origin. The second pair of numbers describes where the upper-right-hand corner of the bounding box lies. In this case it is 104 units to the right and 765 units up from the origin.

Some drawing packages will furnish a precise bounding box, but I have seen others which don't give any bounding box at all, and others which give an incorrect one (often the whole page). Our next figure illustrates what happens with a file where the bounding box is not properly set. It was inserted using the code

```
\topinsert
\psone {schumaker2.fig}{4}
\figcap {5}{A circle surrounded by too much space}{0}
\endinsert
```

If you compare the files `schumaker.fig1` and `schumaker2.fig`, you will see that the only difference is in their second lines where the size of the bounding box is described. In particular, the line setting the bounding box in `schumaker2.fig` is

```
%%BoundingBox:  0 0 612 792
```

This describes a full (8.5×11 inch) page. The circle only occupies a small part of the page. But now because of the bounding box, the entire page is inserted, explaining all the white space in Fig. 5.

If you have a file with no bounding box or an incorrectly sized one, you can fix it with your editor. First print the file on a laser printer, then take a ruler and physically measure the location of the lower-left and upper-right corners of an appropriate bounding box. If there is already a statement defining the bounding box, just edit the file to set the correct dimensions. Remember that 1 inch is equal to 72 units. If there is no such statement, insert one in the file. (I would put it as the second line in the file, after the first line which identifies the file as a postscript file).

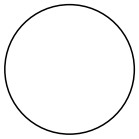

Fig. 5. A circle surrounded by too much space.

§6. Remarks

In case you are wondering, if you use `epsf` to insert figures and later decide to change the magnification of your document, everything (including the figures) is automatically rescaled.

Another set of macros which I have seen is called `psfig.tex`. They work in much the same way as `epsf.tex`. With some dvi drivers, it is possible to insert postscript figures using a `\special` command. It would be nice if there were a standard way to create and handle postscript files – but unfortunately (analogous to the LaTeX, AMSTeX mess), this is not

the case.

The postscript files for the figures included in this document are relatively small (on the order of 5K each). If you create postscript files by some bitmapping process (for example using "snapshot" to capture a screen on an SGI), you can get postscript files which are 10 or even 20 megabytes. It is better to avoid such huge figure files if possible since unless your laser printer has a lot of memory, it will take a long time to print.

Finally, I would like to mention that the macro file `gridbox.tex` contains macros created by Carl de Boor which are very handy for putting text into figures (and can also be used to arrange figures side-by-side, etc.) Instructions on how to use these macro are contained in the file. To see how they work, try the file `gridtest.tex`.

References

1. *Postscript Language, Reference Manual*, Addison-Wesley, Reading, MA, 1985.
2. *Postscript Language, Tutorial and Cookbook*, Addison-Wesley, Reading, MA, 1985.
3. Schumaker, L. L., Instructions for using the Nashboro Press Macros, available on the conference home page.

Larry L. Schumaker
Dept. of Mathematics
Vanderbilt University
Nashville, TN 37240
s@mars.cas.vanderbilt.edu
http://www.math.vanderbilt.edu/∼schumake/