Finally, it should once more be emphasized that — if you have a good knowledge of the PostScript language — it is generally much more efficient to create the picture PostScript file by directly editing the picture rather than using the automated but necessarily clumsy version which is provided by the program TEX2PS.

7 The picture implementation

The method described above yields the following steps of procedure:

1. Start your main LaTeX text file with the line

   `\immediate\openout\bilder=`⟨*name of parameter file*⟩

   and close it by the line

   `\closeout\bilder.`
2. Introduce the option `neubild` into your document style.
3. Write your text file and introduce the line

   `\varpsbild...`

   with the appropriate parameters as explained above at any place where you want to insert a picture.
4. Compile the text file to the corresponding `dvi` file. This also produces the parameter file.
5. Create all the pictures in a PostScript setting either manually or by help of the program TEX2PS, taking into account the correct size and name of every picture (the parameter file contains these parameters for each picture involved).
6. Convert the main `dvi` file by help of the `PTIPS` driver, using all of the PostScript picture files, to the final PostScript file.
7. Print the final PostScript file either on a PostScript printer or via a soft interpreter like `FREEDOM OF PRESS`.

8 Conclusion

The two procedures described above are certainly not the most elegant ones for implementing graphics in TeX. As has been shown, the first method, however, has the advantage of not using any graphics input apart from that which is admissible in the LaTeX `picture` environment, and it is completely driver-independent. A somewhat similar approach to this problem is, for instance, given by M. Ballantyne and collaborators[8]; their method, however, is at the moment not applicable within a LaTeX environment and, moreover, does not seem to work very well if the pictures are to be surrounded by text passages. On the other hand, that method can also be used to include complex tables into a TeX file.

We have not discussed the various methods which use graphics input from different drawing programs to be included into TeX source files. These methods depend heavily on the output format of the drawing programs (e.g. whether or not they are pixel oriented) as well as the ability of TeX drivers to implement the different graphic formats (usually by means of `\special` commands).

The second procedure allows the insertion of much more complex pictures into LaTeX text files at the price of using part of the PostScript machinery. We feel that it might be a good compromise if the time factor does not have first priority and the pictures to be inserted into the text are of moderate complexity.

A diskette containing the files used in these approaches can be ordered from the author. Please, enclose an empty diskette and DM 5,- for postage.

◇ Gerhard Berendt
Institut für Mathematik I
Freie Universität Berlin
Arnimallee 2-6
1000 Berlin 33
Germany
`berendt@fubinf.uucp`

[8] M. Ballantyne et al., *TUGboat* 10, no. 2, p. 164, 1989

Including Macintosh Graphics in LaTeX Documents

Len Schwer

Abstract

The basics of including Macintosh graphics in LaTeX documents are discussed for the person who is inexperienced at doing so. Because there is no universal way to incorporate such graphics, other than with scissors and glue, this article tries to be as general as possible, but ultimately references specific software and hardware, *e.g.* ArborText's DVIPS, Trevor Darrell's `psfig` macros, and an Apple LaserWriter+. The reader is assumed to have some knowledge of

the Macintosh interface, PostScript programming, and LaTeX document preparation.

1 Background

The method of including Macintosh graphics in LaTeX documents is very simple:

- create the graphic with the user's favorite Macintosh application;
- convert the graphic into its PostScript representation;
- transfer the PostScript file to the LaTeX host machine;
- include the PostScript file in the LaTeX document via the DVI-PostScript driver's `\special` command.

While this sounds like a relatively straightforward procedure, it gets complicated, sometimes very complicated. The complications arise from three elements:

1. There are several DVI-PostScript drivers available and they all treat the `\special` command differently. Many of the differences are simply syntactical, but some are more subtle and involve differences in the PostScript prologue which precedes the material from the LaTeX document. The good news here is that there is an organized movement within the TeX community to develop standards for DVI drivers [1] and someday users may benefit from these standardization efforts.
2. Not all PostScript devices are the same. Macintosh QuickDraw, a PostScript language shorthand, in combination with various PostScript implementations of DVI drivers produces different results on different PostScript devices. For example, the ArborText DVI driver, DVIPS, and LaserWriter+ apparently cannot be coaxed into including Macintosh figures according to ArborText's instructions, while the same DVIPS generated file works flawlessly with an NEC PostScript printer.
3. Lastly, but of equal importance, the individuals who decide to dabble in this topic need a working knowledge of the Macintosh interface, PostScript programming, and TeX or LaTeX document preparation.

The original *TUGboat* article on this topic by Hal Varian and Jim Sterken [2] appeared in March 1986. Since that time, there has been a dramatic increase in the number[1] of users of both LaTeX and the Macintosh. New and 'old' LaTeX users are becoming Macintosh users for many reasons, not the least of which is the ability to easily produce high quality graphics for inclusion in their high quality typeset documents. Many of these new users are seeking ways to include Macintosh graphics in LaTeX documents by means other than scissors and glue.

This article updates the information presented in the original *TUGboat* article and complements the information provided in a more recent article by J.T. Renfrow [3]. The present article is intended to serve as a general overview for the person who is new to including Macintosh graphics in LaTeX documents. It tries to be general, when possible, by indicating how things are supposed to work, but in many places it is very specific. Where appropriate, mention will be made of available public domain software and possible sources for obtaining it.

There are two major sections in this article:

Macintosh graphics. This section describes capture or conversion of Macintosh graphics to an equivalent PostScript representation and changes needed for the LaserPrep file.

Using BBFIG and `psfig` to include graphics. This section describes a very useful set of macros that simplify inclusion of Macintosh graphics in LaTeX documents.

2 Macintosh Graphics

The Macintosh was designed to be used with an Apple LaserWriter printer, a PostScript device. Thus all Macintosh applications need to support PostScript if they are to allow printing. We would like to capture the PostScript representation of a graphic in a file, transfer the file to the host machine where LaTeX is used, and include the PostScript graphic file in a LaTeX document. This section describes a 'universal' method for capturing the PostScript representation of a graphic in a file.

2.1 Capturing Macintosh graphics in a PostScript file

Unfortunately, few Macintosh applications, other than Cricket Draw and Adobe Illustrator, provide a menu option for generating a PostScript file. This is probably due to the existence of a universal technique for capturing any printer-directed application output into a PostScript file. The technique is quite simple:

> After completing the graphic, select the *Print* option under the **File** menu. This produces

[1] The growing number of these users is evident to those who monitor electronic information groups such as `comp.text.tex`, `comp.lang.postscript`, and `comp.sys.mac` on Usenet and `TeXhax` on the Internet or Bitnet.

the 'Print Dialog Box' which allows the user to select various options before sending a graphic to the printer. In the upper right-hand corner of the 'Print Dialog Box' is an OK button. Click the mouse down, but **do not release**, on the OK button. With the mouse still clicked down, depress and hold down the F key and then the Command (Apple or Flower) key (this combined key stroke is usually refered to as *Command-F*), then release the mouse button. A dialog box should appear stating that a PostScript file is now being created. Note: In *most* applications the Command key need not be depressed.

Recently, a very handy Macintosh application named **myPageSetup** by D.G. Gilbert has appeared[2] which activates a previously hidden selection box in the Print Dialog. The activated box is called **Disk File** and selecting it will cause the LaserWriter application to create a PostScript file rather than sending the file to the printer when the OK button is clicked. This eliminates the need for the somewhat clumsy Command-F keystroke sequence just described.

The above techniques will cause a file named 'PostScript0' (or, more generally, 'PostScript*n*' where *n* is incremented by one for each PostScript file generated) to be created in one of several places:

- the folder where the generating application resides, *e.g.* where you keep MacDraw;
- the folder where the graphic was *launched*, *e.g.* where you have stored the graphic file;
- the DeskTop level of your startup disk;
- the System folder.

The exact location is application dependent[3], but the Macintosh **FINDER** may be used to locate these PostScript files in any case.

Although these files are labeled as PostScript files, the files generated using the Command-F technique are not quite PostScript files, but are more correctly referred to as *QuickDraw* files. QuickDraw is a special PostScript shorthand created by Apple Computer.[4] NOTE: QuickDraw files will not produce a graphic image when sent to a PostScript printer unless a special initialization file has previously been sent to the printer.

The QuickDraw initialization file is commonly called a *LaserPrep* file and various versions of it are known as AppleDict Version #*nn*, where *nn* is the version number, *e.g.* AppleDict Version #70 (a.k.a. LaserPrep 70). The LaserPrep is a dictionary that translates QuickDraw into PostScript. The resulting translation does produce a printable graphic image on PostScript printers. Usually the LaserPrep file is downloaded only once to a PostScript printer connected to a Macintosh. This PostScript dictionary, called 'md' (for Macintosh Dictionary?), remains resident in the printer's volatile memory until the printer is powered down. A copy of the Macintosh's current LaserPrep file may be generated by following the Command-F procedure described above, but substituting *Command-K* before releasing the mouse clicked down on the OK button in the 'Print Dialog Box'. Using the Command-K key sequence generates a file named 'PostScript*n*' that contains both the LaserPrep file and the contents generated by the Command-F key sequence; *i.e.* the LaserPrep file is inserted as a prologue to the QuickDraw file. The boundary between the LaserPrep and QuickDraw is located at the first occurrence of the string `%%EOF`, which is the last line of the LaserPrep file.

One more note about the LaserPrep file: near the bottom of the LaserPrep file are lines of hexadecimal numbers followed by several lines of zeros. The lines of hexadecimal numbers are very long and will break most file transfer programs. These lines may be shortened by inserting carriage returns at appropriate distances along the string. Alternatively, for the less faint-of-heart, these lines may be deleted from the LaserPrep file. More specifically, the lines between and including:

```
currentfile ok userdict/ ...
...
cleartomark
```

may be deleted. According to Bill Woodruff [4]

> The dictionary [LaserPrep] includes some special encrypted assembly-language procedures that are proprietary to Apple Computer. If you check the "Faster BitMap Printing" option in the generic Macintosh Page Setup dialog, for example, you activate an Apple bitmap smoothing routine that will

[2] This freeware application is available via anonymous FTP from `sumex-aim.stanford.edu` in the directory `/info-mac/util` and probably from many other such Macintosh archives.

[3] An init named **LaserFix** by David P. Sumner modifies the print dialog box in the same manner as **myPageSetup**, but invokes a standard file location dialog box after the OK button is clicked. This allows a user to specify a folder name where all such PostScript files will be created. This init is also available from `sumex-aim.stanford.edu` in the directory `/info-mac/init`.

[4] Some very interesting comments on the development of QuickDraw are provided in a brief article by Bill Woodruff [4].

not work if the installation of 'md' recognizes the printer is not from Apple.

2.2 Modifying the LaserPrep File

Woodruff [4] also comments:

> LaserPrep was patched and fixed and extended and patched and fixed until it has reached its current state where even within Apple it is considered an embarrassing morass. Yet, it remains the world's most used PostScript program and the fundamental bottleneck through which almost all Macintosh printing is done. But to change it, even slightly, is to move a gigantic tectonic plate on which the entire superstructure of civilized Macintosh printing hangs in fragile balance.

In order to include Macintosh PostScript files generated with the Command-F technique described above with any DVI to PostScript driver, the LaserPrep file must be modified. There are two major types of changes:

1. Changes to keep the LaserPrep from altering the printer's status, *i.e.* changes to `statusdict`.
2. A change to prevent the included PostScript file from issuing a `showpage` or `copypage` command.

Listings of the modified Macintosh LaserPrep files, even 'differences' listings, are too lengthy for this article. Suitably modified versions of Macintosh LaserPrep #65 and #68 files[5] are available via anonymous FTP from `ymir.claremont.edu`. The modified version of LaserPrep #65 is derived mostly from the instructions issued by ArborText for their DVIPS driver.

The modified version of LaserPrep #68 was created by Trevor Darrell who claims that it should be compatible with Tony Li's public domain PostScript device driver DVI2PS when used with the appropriate TEX prologue. DVI2PS, the TEX prologue file, modified LaserPrep #68 file, and associated information are available via anonymous FTP from `linc.cis.upenn.edu` [130.91.6.8] in files `dvi2ps.tar.Z` and `lprep68.tar` of the subdirectory `dist/psfig`. These files are in Unix TAR format and, in the case of the `dvi2ps.tar.Z` file, Unix TAR and compressed binary format. However, after uncompressing the `dvi2ps.tar.Z` file and extracting the files (unTARing), you will find it also contains modifications for implementation on VMS systems; the unTARed `lprep68.tar` file contains plain text files. Another source for these files is the recent Digital Equipment Corporation Users Society (DECUS) TEX/LATEX tape collection, prepared by Ted Nieland; for availability call DECUS at (505)480-3418 or contact your Local User's Group (LUG).

The modified LaserPrep file should be given an appropriate name, *e.g.* `LaserPrep68.ps`, and placed in a directory where DVIPS or DVI2PS can find it to include in DVI files. A suggested directory location is one of the directories searched by the logical `TeX$Inputs`. Also, the modified LaserPrep file must be included in LATEX documents before the occurrence of the first Macintosh PostScript file. The LaserPrep file is usually included prior to the `\begin{document}` statement *e.g.* by using ArborText's `\special` command,

```
\special{ps: plotfile LaserPrep68.ps global}
```

It is important to use the modified version of the LaserPrep file that corresponds to the LaserPrep used to generate the Macintosh graphic with the Command-F procedure. To determine which version is appropriate, search for a line like the following near the top of the file:

```
%%IncludeProcSet: "(AppleDict md-figure)" 68 0
```

The number 68 in this example indicates that modified LaserPrep #68 should be used.

To determine the current version number of the LaserPrep file on your Macintosh, either (1) select the LaserPrep icon in the System folder and use *Get Info* under the **File** menu, or (2) when attempting to print a document from the Macintosh, look just to the left of the 'OK' button in the Print Dialog Box. In either case the number should be 4.0 or 5.2. 4.0 is equivalent to LaserPrep #65 and 5.2 is equivalent to LaserPrep #68. Basically, Macintosh system software 5.x, LaserPrep Version 4.0, and 'md' Version 65 all go together, as do Macintosh system software 6.x, LaserPrep Version 5.x, and 'md' Version 68. Simple, huh? This means that, technically, LaserPrep files such as LaserPrep #68 are referring to the version of the Apple dictionary and not the actual LaserPrep version number. Hopefully, someday soon, Apple will get all these numbers in sync.

The most recent version of the Macintosh LaserPrep file is #70, which is associated with LaserWriter version 6.0 and accompanies the latest release of the Macintosh operating system. LaserPrep #70 is considerably different from previous versions in that it contains information for Apple's implementation of Color QuickDraw. Requests to various electronic forums, and Trevor Darrell, for a suitably modified LaserPrep #70 indicate that knowledgeable LaserPrep hackers have not modified version #70. An ef-

[5] The hexadecimal strings at the bottom of the LaserPrep files have been deleted.

fective work-around is to install LaserPrep #68, *viz.* LaserWriter 5.2, on your Macintosh and rename it, for example to `LaserWriter68`. This version of the LaserWriter can then be selected via the *Chooser* before creating PostScript files.

3 Using BBFIG and `psfig` to Include Macintosh Graphics

Acknowledgement: BBFIG was authored by Ned Batchelder. `psfig` was developed and placed in the public domain[6] by Trevor J. Darrell. This subsection borrows quite liberally from Mr. Darrell's very nice documentation, *Incorporating PostScript and Macintosh Figures in TEX*. Current versions of the software and documentation are available via anonymous FTP from `linc.cis.upenn.edu` [130.91.6.8] in the sub-directory `dist/psfig`. This author, and undoubtly many other `psfig` users, are indebted to Mr. Darrell for his outstanding programming skills and his unselfish willingness to share this software and knowledge with others. The author hopes that this document continues Mr. Darrell's spirit of freely sharing knowledge.

`psfig` is a TEX macro package that facilitates the inclusion of arbitrary PostScript figures in LATEX documents. The real advantage of using the `psfig` macros is that they work within the Adobe constructs for Encapsulated PostScript Files (EPSF) [5]. For the `psfig` user, that means that graphics can both be easily placed within a LATEX document and be printed on their own directly to a PostScript device; other systems for including PostScript graphics require a `currentpoint` to be set and the `showpage` to be explicitly disabled, thus disabling direct printing of the file.

To properly locate a PostScript figure, the DVI driver must know the size of the figure and its relative position on the printed page. This information is implicitly available within the various graphic constructs used in the PostScript file. The information *should* also be available explicitly in the BoundingBox comment [6] of the PostScript file's prologue.

The bounding box encloses all the marks made on a page as a result of executing (printing) a PostScript program. The BoundingBox comment has four parameters:

`%%BoundingBox:` ll_x ll_y ur_x ur_y

The four integer parameters, in units of points, represent the coordinates of the lower left (ll_x,ll_y) and upper right (ur_x,ur_y) corners of the bounding box in the default user (creator) coordinate system.

Although good PostScript programming practice dictates that the BoundingBox comment be provided in PostScript files, few Macintosh programs provide this information. Either the BoundingBox parameters are not specified, *e.g.* as with

```
%%BoundingBox:  ?  ?  ?  ?
```

which is the case for all Command-F generated PostScript files, or an entire 8.5 × 11 page is specified, *e.g.* with

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

To supply the proper bounding box information, one can measure the required dimensions or use the PostScript utility BBFIG, which calculates the bounding box parameters from the graphic information implicit in the PostScript file. The BBFIG PostScript file is prepended to the PostScript file[7] for which the bounding box is to be determined, and the combined file is sent to a printer. The result is a printed image of the PostScript graphic with the bounding box drawn around the graphic and bounding box parameters listed below the bounding box; the bounding box parameters are also returned via the print job's log file, if the log file feature is implemented for the host machine's print queue.

The accuracy of BBFIG in determining the bounding box information is not as good as may be needed in some circumstances. If you notice your figures missing parts or wandering around the page, check the bounding box information. Of course, the exact bounding box information can be obtained by simply printing the figure and using a ruler, in points if possible, to measure the four coordinates. Measuring the bounding box coordinates is also necessary for graphics from some Macintosh applications such as Cricket Software's Cricket Graph. For some unknown reason, Cricket Graph figures contain an invisible (un-stroked) path around the entire edge of the paper.

Once the proper bounding box information has been added to the BoundingBox comment of a PostScript file, the file may easily be included in a LATEX document by using the TEX macro `\psfig`. Simply

[6] Copyright notice from the `psfig` source: "All software, documentation, and related files in this distribution of psfig/tex are Copyright (©) 1987 Trevor J. Darrell. Permission is granted for use and non-profit distribution of psfig/tex providing that this notice be clearly maintained, but the right to distribute any portion of psfig/tex for profit or as part of any commercial product is specifically reserved for the author."

[7] If the PostScript file was generated using Command-F (QuickDraw), then a *non-modified* LaserPrep must also be prepended to the QuickDraw file.

load the `psfig` macros at the beginning of your document with

```
\input{psfig}
```

then invoke the macro

\psfig{figure=*{input}*}

where *input* is the name of a PostScript file. `psfig` will automatically position the figure at the current place on the page, and reserve the proper amount of space in LaTeX so that it does not conflict with any other objects.

For example, if we have a file called `piechart.ps` that contains the PostScript code to draw a pie chart, we would use the command

```
\centerline{\psfig{figure=piechart.ps}}
```

Since no mention of size is made in the above example, `psfig` would draw the figure at its natural size (as if it were printed directly by a PostScript printer.) If the pie's natural size is several inches across, which is a little large, the pie could be reduced with:

```
\centerline{%
\psfig{figure=piechart.ps,height=1.5in}}
```

The `height` option specifies how tall the figure should be on the page. Since no `width` is specified, the figure would be scaled equally in both dimensions. By specifying both a `height` and a `width`, figures can be scaled disproportionately, with interesting results.

There are a few caveats associated with using `psfig`:

- For `psfig` to find the natural size of a figure, the figure must have a proper bounding box comment; see previous bounding box discussion.
- Some versions of LaTeX will fail to center a lone figure properly in a center environment; a good work-around is to precede the figure with a hard space, *e.g.*

  ```
  \begin{center}
  \ \psfig{figure=...}
  \end{center}
  ```
- On very large documents with many figures, the printer memory allocated to DVIPS may have to be limited; refer to ArborText documentation for setting LaserWriter memory.
- The `\psfig` macro will be confused by extra white space or new lines in its argument, *e.g.*

  ```
  \psfig{figure=piechart.ps, height=1.5in}
  ```

  causes `psfig`'s parsing routine to terminate at the space; LaTeX will interpret the `height=1.5in` as text. Long `psfig` command lines may be split using `%` line terminators, *e.g.*

  ```
  \centerline\psfig{figure=piechart.ps,%
  height=1.5in,width=2.3in,clip=}}
  ```

Certain PostScript figures (such as large bitmap images being transmitted at 9600 baud) can tie up a slower PostScript device such as an Apple LaserWriter for quite some time. To circumvent this, a figure may be printed in draft mode, which will reserve the same space on the page, but will print just the name of the file[8] from which the figure is derived and not actually include it. The macro `\psdraft` will switch into draft mode, and all subsequent `psfig` macros will produce draft figures. The macro `\psfull` will switch out of draft mode.

The preceding discussion of `psfig` is appropriate for PostScript graphics generated by Cricket Draw and *any* other type of EPS file. The only exception to this broad statement is a QuickDraw file generated by the Command-F option. QuickDraw files require a suitably modified LaserPrep file to be prepended to the QuickDraw file. `psfig` provides an easy mechanism for including the LaserPrep file as described below.

There is a `psfig` macro `prolog` option for specifying a file that should be prepended to the figure. The name of the prolog is, of course, site dependent; we have used `lprep68.pro`. For example, if you had a file `frog.mac` that contained the QuickDraw to draw Kermit (The Frog), he could be included with:

```
\psfig{figure=frog.mac,prolog=lprep68.pro}
```

If there are many such figures, it is probable that the repeated inclusion of the `prolog` file will cause a significant increase in the size of the print file and its transmission time. An alternative method is to load the `prolog` file once globally, so that it will be available throughout the rest of the document. Use

```
\psglobal{lprep68.pro}
```

at the beginning of your document to achieve this effect. For this to work properly, the `\psglobal` must appear before any Macintosh figures, and the final output must not be page reversed.[9]

Recent experience has shown that the use of `\psglobal` with the ArborText DVIPS driver conflicts with Macintosh graphics included inside a fig-

[8] The current implementation of `psfig` does not support underscores (_) in file names for draft mode. Since `psfig` places the file name on the page using TeX commands, reserved LaTeX characters cannot be used in the file name.

[9] A page reversed document prints the last page first and first page last. It is possible to use `\psglobal` in a page reversed document; place it just before the last figure in your document. This is living dangerously, and you do so at your own risk.

ure environment; *i.e.* the figure prints by itself on a page separate from the LaTeX document. Two solutions are: always include a prolog along with each Macintosh graphics (brute force) or include the `\psglobal` inside the first figure environment (lucky hacque).

4 Conclusion

The combination of a Macintosh for producing high quality graphics and LaTeX for producing high quality typeset documents is becoming very popular as a 'total' document preparation system in many working environments. The development of tools, such as the `psfig` macros, that make integrating Macintosh graphics in LaTeX documents easier, will undoubtedly grow in popularity. It is hoped that the information collected in this article helps more users produce better documents.

References

[1] Hosek, D., *Report from the* DVI *Driver Standards Committee*, *TUGboat*, Vol. 10, No. 1, p. 56, April 1989.

[2] Varian, H. and J. Sterkin, *MacDraw Pictures in TeX Documents, TUGboat*, Vol. 7, No. 1, pp. 37–40, March 1986.

[3] Renfrow, J.T., *Methodologies for Preparing and Integrating PostScript Graphics, TUGboat*, Vol. 10, No. 4 — 1989 Conference Proceedings, pp. 607–626, December 1989.

[4] Woodruff, B., *PostScript and the Macintosh: A History*, **MacTech Quarterly**, Volume 1, Number 2, Summer 1989, pp. 119–120.

[5] *Encapsulated PostScript Files Specification Version 2.0*, Adobe Systems Inc., 1585 Charleston Road, P.O. Box 7900, Mountain View, CA 94039-7900, (415)961-4400, 16 January 1989.

[6] *PostScript Language Reference Manual*, Appendix C: Structuring Conventions, Adobe System Inc., Addison-Wesley Publishing Co., Inc., 1985, p. 268.

◇ Len Schwer
APTEK, Inc.
4320 Stevens Creek Blvd.
Suite 195
San Jose, CA 95129
`micro2.schwer@sri.com`

Combining Graphics with TeX on PC Systems with Laser Printers, Part II

Lee S. Pickrell

Abstract

In this article we will extend our premise that TeX affords an excellent mechanism for combining graphics in TeX documents. We propose a method for including graphics that brings to bear the full power and versatility of TeX for positioning the graphics as well as the text. The technology for implementing this feature will be discussed, including certain limitations. We will also consider possible benefits of file conversion utilities, particularly the potential advantage of converting graphics to the PK/TFM file format of TeX fonts. One application of this feature is that the captured graphics can be used with PostScript drivers. This technique can significantly increase the number of graphics sources available to PostScript-based TeX by accessing applications that support the LaserJet PCL language. Finally screen capture will be examined as an adjunct to printer capture in the case that printer capture is not practical.

1 Introduction and review

In our first article [1] we made several assertions, in particular, that TeX provided a natural platform for mixing graphics with typeset text. Several graphic plots were included that were obtained from different application programs (several more will be included in this article), which we hope substantiated our assertions:

- TeX provides a natural platform for graphics insertion, certainly comparable to any other word processing system.
- TeX has suffered from a *perception* that it does not handle graphics well, probably grounded more in psychology than technical reality, and possibly due to the broad spectrum of computing systems and distinct device driver programs over which TeX is implemented.
- The IBM PC and LaserJet printer are the logical *starting* place for demonstrating the graphics capabilities of TeX because graphics applications for the PC/LaserJet combination have become ubiquitous.
- Printer output capture is the best method for obtaining graphics images because the available resolution is much higher than screen capture and the number of graphics sources is much larger than file conversion.