Using LaTeX at the Nora Eccles Harrison Cardiovascular Research and Training Institute CVRTI

Rob MacLeod

February 6, 2001
For LaTeX 2_ϵ

Contents

About this Manual

This document is meant to acquaint the user of the UNIX computers at the Nora Eccles Harrison Cardiovascular Research and Training Institute with the local installation of the LaTeX document preparation system. The manual is arranged in sections which run parallel to chapters of the same name in the actual LaTeX manual by Lamport. Hence, for example, information on BibTeX is found in section 4, just as it appears in chapter 4 of the LaTeX manual. The Appendices contain additional information related to the use of LaTeX.

The reader is assumed to have some previous knowledge of basic UNIX commands, to be able to run one of the available text editors, and to have access to the Silicon Graphics computers installed at the CVRTI.

1 Getting Acquainted

LaTeX runs on a variety of computers at many different sites all over the world. LaTeX is not a text processor but a document preparation system which is entirely portable and is a widely used standard for user-written documentation. We have seen LaTeX documents increasingly used to document software developments in the public domain; in fact, your first exposure to LaTeX may be in a file, or set of files, with the `.tex` extension that arrives with the distribution of a software package from a distant ftp site.

This document tells you how to use LaTeX on the Silicon Graphics (SGI) computers at the CVRTI. We have a complete version of LaTeX installed, but at the version 2ϵ, *i.e.,* pretty much the newest generation. This manual is not about LaTeX itself, which is described by any number of books, for example, *LaTeX: A Document Preparation System*, published by Addison-Wesley. In section 1.1 of this document are other references to information about LaTeX and its associated programs.

If you have a question that you can't answer by reading the manual and this document, ask Rob MacLeod (macleod@cvrti.utah.edu). The person should also be informed of any possible LaTeX bugs or undocumented anomalies.

1.1 Other Documentation

The original for LaTeX is the book by Leslie Lamport, which is now in a second edition. All the usual online book vendors and even the University bookstore usually carry this, as well as the original TeXBook by Knuth. A more recent book called *A Guide to LaTeX* Kopka and Daly is actually a more extensive book and highly recommended. Another nice supplementary text is *The LaTeX companion* by Goosens, Mittelbach, and Samarin, by Addison Wesley Press. These are sometimes available in the CVRTI Graphics Lab and many fine bookstores.

Locally, we have supplementary guides for different aspects of using LaTeX. For BibTeX we have the manual by Oren Patashnick (the author of BibTeX), called BibTeX*ing and Designing* BibTeX *Styles* and for the DVI to Postscript conversion process, there is a lengthy man page as well as a separate manual. There are also bibliography files for BibTeX available from the server at `ftp.math.utah.edu` via anonymous ftp. These include a list of books on or about TeX and LaTeX, and several collections of article references from ACM Transactions and other mathematical journals. We also have a printed bibliography of books related to computer programming, systems and graphics, kept in the Graphics Lab. Other items will be added to this collection as they are found and collated.

1.2 Utility programs and scripts

1.2.1 *texit*

The simplest way to use LaTeX is by running the script called *texit*, which manages all stages of the document conversion phase, from a LaTeX file to the finished printed copy. Because it has been developed locally, *texit* reflects the actual and current configuration of programs, printers, and dvi-to-postscript software at the CVRTI. To call this script, simple type:

```
texit [filename]
```

where `filename` is the root name of the file to be processed, enter without any extension. All needed filenames are built from this one, under the assumption of the source text file having the `.tex` extension. The `filename` argument is optional and you will be prompted for a filename if you do not supply it at invocation.

Once started, *texit* offers all the different options required to take a file through the different steps in processing with LaTeX:

1. Convert .tex file(s) to dvi with LaTeX (or TeX).
2. Run BibTeX on the .tex to resolve bibliography references
3. View the dvi file using the xdvi viewer.
4. Convert the dvi file to a printable postscript file.
5. View the postscript file on a terminal capable of running X-windows.
6. Print the postscript file to the printer of choice, after checking to make sure that printer is really available.
7. Stripping away the LaTeX commands to make a plain text file with *detex* (See Section 1.2.2
8. Converting the LaTeX to HTML using the *latex2html* program.
9. Starting emacs on the LaTeX source file.
10. Cleaning up the extra files that are generated during the above steps.

Each of these steps can be carried out independently and repeatedly so that *texit* can even be used just to print a postscript file to any printer in the building, including a check to ensure that the printer is online and powered.

When you launch *texit* on a file, you should see something like:

```
bianchi:macleod/tex> texit sample

  Welcome to the Rob's Latex Processing Script
    Version: Dec 11, 1999
 The input filename is: sample.tex

 Choose your processing wish:

      LaTeX to DVI file ...................... l
      TeX to DVI file ........................ t
      View dvi file with xdvi ................ v
      DVI to postscript ...................... d
      View postscript file with gv ........... g
      Print an existing .dvips file........... p
      LaTeX to printed copy (the works) ....... w
      BibTeX ................................. b
      Edit the .tex file with emacs ...........e
      Strip out LaTeX commands with detex ......s
      Convert LaTeX to HTML ...................h
      Clean up extra files from LaTeX ......... c
      To quit here ........................... q
      [default = l]
```

1.2.2 *detex*

This utility program converts LaTeX files to plain ASCII text. This is useful if you have to get text out of LaTeX and into another text editor. The markup commands for formatting are obviously lost, but the text is at least intact and clean. *detex*, which is available on the SGI, and has a man page describing its use. *detex* is especially useful when you want to know how many words of text are in a LaTeX file. This is accomplished using the following Unix pipe:

```
detex myfile | wc -l
```

Note there will be extra words included in the output of *detex* which are not part of the written text. If the word count has to be exact, you will have to generate a *detex*-ed file and edit out the extra text, then pass it to `wc` for counting.

2 Getting Started

2.1 Running a Sample File

Before preparing your own documents, you may want to get acquainted with LaTeX by running it on a sample input file. First make your own copy of the file `sample.tex` by setting up a LaTeX directory and typing the following UNIX command:

```
cp /usr/local/share/texmf/localguide/sample.tex .
```

(You must type the space followed by the period at the end. This and all other UNIX commands are ended by typing *return*.) A copy of the file `sample.tex` is now in your current directory; you can edit it just like any other file. If you destroy or mess up your copy, typing the above command again gets you a fresh one.

Next, run LaTeX on the file `sample.tex` by typing:

```
latex sample
```

This will start LaTeX and produce output which looks at least something like:

```
This is TeX, Version 3.14159 (Web2C 7.2)
(sample.tex
LaTeX2e <1999/06/01> patch level 1
(/usr/local/share/texmf/tex/latex/base/article.cls
Document Class: article 1999/01/07 v1.4a Standard LaTeX document class
(/usr/local/share/texmf/tex/latex/base/size10.clo)) (sample.aux)
(/usr/local/share/texmf/tex/latex/base/omscmr.fd) [1] [2] [3] (sample.aux) )
Output written on sample.dvi (3 pages, 7112 bytes).
Transcript written on sample.log.
```

When LaTeX has finished, it will have produced the file `sample.dvi` in your directory. You can print this file by first converting the `.dvi` file to its Postscript equivalent, which will have the extension `.dvips`, and then sending this file to a Postscript printer. This is most easily accomplished by typing the command:

```
dvips sample
printit sample.dvips
```

The command `printit` is really a script program that manages printing at the CVRTI. It provides access to all printers in the building and is usually kept up to date as hardware changes.

After your output has been printed, you should delete the extra files produced by LaTeX by typing:

```
rm sample.dvi
rm sample.ps
rm sample.aux
rm sample.log
```

or running the script *texclean*, which will take care of cleaning up the extra files. None of these files are required to re-process the document and the `.dvi` and `.ps` files especially can be quite large. The contents and purpose of these supplementary files are discussed in section C.10.1 in the LaTeX manual. See section 3.1 in this document for some information on how UNIX affects the way LaTeX handles these files.

Note that all the above steps are included in the script *texit*, which is described in more detail in section 1.2. A page of information in `/usr/news` also describes the *texit* script and will be more up to date than this document. Use the command `news -a | more` to view the news files, or just read the files themselves, which are all stored in `/usr/news` as plain ASCII files.

2.2 Previewing the Document

In order to see what the final output from LaTeX will look like before printing, you must use a special previewing program. This program reads the contents of the `.dvi` file and formats it for display on a graphics screen. At the CVRTI, we have a previewer for the SGI called *xdvi* that runs on any terminal that supports X-windows, *i.e.,* the console in the Graphics Lab or any Macintosh or PC running X-windows server software. To run *xdvi* on the sample file, type:

```
xdvi sample
```

The right hand side of the screen will then fill with the first page of the sample document, as it would appear on paper. There are a number of options and controls for *xdvi*, some of which are available as mouse-operated buttons on the screen and all of which are documented in the man page.

2.3 Preparing and Running LaTeX on Your Own Files

You must use a text editor to prepare an input file for LaTeX. The editors available on the UNIX systems at the CVRTI are the UNIX standard *vi* and GNU Emacs. A copy of the documentation on Emacs should be available in the Graphics Lab. The easiest way to start learning about LaTeX is by examining the file `sample.tex` with your text editor.

After you have prepared your own file, whose name should have the extension `tex`, you must run it through LaTeX and print the output. Follow the instructions in Section 2.1, except substitute the first name of your file for "`sample`". Remember to save disk space by deleting the `dvi, dvips, log, aux` file after printing the output.

If you want to stop LaTeX in the middle of its execution, perhaps because it is printing a seemingly unending string of uninformative error messages, type *Control-C* (press "C" while holding down the key labeled *CTRL*). This will make LaTeX stop as if it had encountered an ordinary error, and you can return to UNIX command level by typing `X`, as described in the manual. If typing *Control-C* doesn't work, typing *Control-Z* will get you immediately to UNIX command level, but this will leave a stopped job hanging around. A stopped job won't hurt anything and will disappear when you log out, but it forces you to type two successive `logout` commands to log out.

2.4 LaTeX support in Emacs

There is an excellent collection of support commands for LaTeX which are included in our installation of Emacs. Theses were written by Nelson Beebe and are powerful and well documented (see Graphics Lab binders) in the form of 2 papers. All commands start with `^C` and do things like set up environments, apply different fonts or emphasize to words, even set up a skeleton document when you first begin to write. If you use Emacs to write LaTeX documents, you **need** to know about these features. For most users at the CVRTI, these commands should load automatically when you open a document with a `.tex` extension, if not, see Rob MacLeod (macleod@cvrti.utah.edu).

2.5 Using the Spell Checker with LaTeX

To use the *ispell* program (a much improved version of the standard UNIX *spell*) for finding spelling errors in a LaTeX input file named `myfile.tex`, edit the document using Emacs and start the spell checker with the command:

```
M-x ispell-buffer
```

ispell is smart enough to recognize LaTeX commands and skip over them, but allows you to interactively replace words, add new ones to your personal dictionary, *etc,*. The *ispell* program can also be used from the command line `ispell filename` but using it within *emacs* is generally the preferred option. To run *ispell*, you need some lines of code in your `.emacs` file, which are described in the news item dedicated to *ispell*. Most users will have this automatically if they use the standard local distribution `.emacs` file, is not, see Rob MacLeod (macleod@cvrti.utah.edu).

3 Carrying On (more of the details)

3.1 LaTeX on UNIX

The only special problems in using LaTeX caused by the UNIX operating system involve the way UNIX handles files. The first problem arises because, when a program starts to write a file, UNIX destroys the previous version of that file. Thus, if an error forces you to stop LaTeX prematurely (by typing *Control-C* or *Control-Z*), then the files that LaTeX was writing are incomplete, and the previous complete versions have been destroyed. You probably don't care about the output on the `dvi` file, but, if you are making a table of contents or using cross-referencing commands, then LaTeX also writes one or more *auxiliary files* that it reads the next time it processes the same input file. If the auxiliary files are incomplete because LaTeX was stopped before reaching the end of its input file, then the table of contents and cross-references will be incorrect the next time LaTeX is run on the same input file. You will have to run LaTeX a second time to get them right. If you want to avoid having to run LaTeX twice after making an error—for example, if your input is very long—then you should save copies of these auxiliary files before running LaTeX. An input file named `myfile.tex` and all the auxiliary files produced by LaTeX from it are included in the UNIX file specifier `myfile.*`. Use the UNIX `cp` command to save copies of these files.

The second problem in using LaTeX on UNIX involves the files that LaTeX reads. The file whose name you type with UNIX's `latex` command is called the *root file*. In addition to reading the root file, LaTeX also reads the files specified by `\input` and `\include` commands. With the UNIX directory system, LaTeX must know not only the names of these file but also on what directories they are. It will have no problem finding the correct files if you follow two simple rules:

1. Run LaTeX from the directory containing the root file.

2. Keep all files specified by `\input` and `\include`commands in the same directory as the root file.

If you follow these rules, you never have to type a UNIX path specifier when using LaTeX.

You should never break the first rule, otherwise LaTeX will have trouble finding auxiliary files. (To run LaTeX on someone else's file, copy the file to your directory.) If you break the second rule, specifying a file from another directory in an `\input` or `\include` command, you must use a complete path name. For example, to include the file `hisfile.tex` from Jones' directory `/foo/bar`, you can type

```
\include{/u/jones/foo/bar/hisfile}
```

A ˜ character may not appear in the argument of an `\input` or `\include` command, so you *can't* use a file name such as `~jones/foo/bar/hisfile`.

For people who don't like to obey rules, here is exactly how LaTeX finds its files. The root file is found by UNIX according to its usual rules. LaTeX's auxiliary files are read and written in the directory from which it is run. All file names specified in the LaTeX input, including the names of document-style (`sty`) files specified by the `\documentstyle` command, are interpreted relative to the directory from which LaTeX is run. If LaTeX does not find a file starting in this directory, it looks in the system directory `/usr/local/share/texmf/tex/latex`. You can change the directories in which LaTeX looks for its input files by setting the environment variable `TEXINPUTS`. The defaults for this environmental variable are set in the file /etc/profile for all users and you should seldom need to change anything. If you do, it is best to include all the defaults. This is done either interactively form the shell prompt or by inserting text in your `.profile` file, *e.g.*,

```
TEXINPUTS=:.:/u/macleod/tex/latex:${TEXINPUTS}
export TEXINPUTS
```

in your `.login` file causes LaTeX to look for files first in the current directory, then in macleod's `/tex/latex` directory, and then in the current default path.

3.2 LaTeX on the Mac

TeX and LaTeX are free and open software and thus have versions for pretty much all operating systems, including the MacOS. The program OzTeX is the public domain implementation that is freely available for the Mac and you can download the latest version at www.trevorrow.com/oztex/index.html. The text editor of choice for the Mac is Alpha, which is also available as shareware from alpha.olm.net To find all these links and many more, go to the definitive source for LaTeX on the Mac at www.esm.psu.edu/mac-tex

3.3 Making fonts on demand

One phenomenon that you may experience is that in the processing making and viewing the .dvi file or converting it to postscript, there is a slight delay and then some other program starts dumping lots of lines of output to your screen. This is the Metafont program at work creating the actually fonts you need for the display or the postscript output. Don't panic, just let it run; you have the pleasure of knowing that you have called for a font in a size or style that no one else has so far required. Once the system creates this font, it keeps it and makes it available to all users. So you should never see it make the same font twice.

This is a feature of the particular installation of TEX and LATEX that we use at the CVRTI.

4 Moving Information Around

4.1 Using BIBTEX

BIBTEX is a program for compiling a reference list for a document from a bibliographic database. It is run by typing

```
bibtex myfile
```

where `myfile.tex` is the name of your LATEX input file. This reads the file `myfile.aux`, which was generated when you ran LATEX on `myfile.tex`, and produces the file `myfile.bbl`. BIBTEX sshould be run from the directory containing `myfile.tex` (which should be the same directory from which LATEX was run on that file).

If the `bib` file is not in the same directory as the LATEX input file—for example, if you're using someone else's `bib` file—then you must include a path as part of the file name specified by the `\bibliography` command. A `~` cannot appear in the argument of a `\bibliography` command, so you should use a complete path name. For example, the LATEX command

```
\bibliography{/u/jones/bibfiles/gnus}
```

specifies the file `gnus.bib` kept by Jones in his `/bibfiles` directory.

To simplify this process, you can also set the environmental variable `BIBINPUTS` with a command like

```
BIBINPUTS=.:${HOME}/litbase/data:/u/quan/bib:${BIBINPUTS}
export BIBINPUTS
```

There is now no formal provision for sharing bibliographic database information, but Emacs can assist in making your own `bib` files by providing pull down menus and key-command shortcuts for the different type of BIBTEX entries. There are also some utilities at the CVRTI for "cleaning" bib files and also selecting subsets of bib files based on search keys.

4.2 BIBTEX styles

Different journal require (or at least they claim to require but seldom really require) that you use specific formats for the bibliographics. This is possible in BIBTEX and we have developed a few specific styles for relevant journals, for example Circulation and IEEE Transactions. The best bet when you need this is to talk to Rob MacLeod (macleod@cvrti.utah.edu) for more details.

4.3 Using SliTex

This is a remnant of the old format of LaTeX and SliTex is just another feature of the standard release so see your favorite reference book for details. The most common use of SLiTeX anymore is to create decent looking overhead transparencies that include equations.

5 Designing it Yourself

5.1 Document Styles

The LaTeX manual describes the basic styles used in formatting documents: `article,` `report,` `book,` and `letter`. While these basic formats can be all a user ever needs or wants to know about document styles, LaTeX allows the user to customize and create styles by modifying the `.sty` files, as described in section 5.1 of the LaTeX manual. There are a large number of document styles and style options already available at the CVRTI that are not described in the manual. A few examples of these are:

- the `html` features for subsequent conversion to HTML by the program latex2html.
- The `fancyhdr` option for setting up fancy headers and foots, like the ones used here (at least in the paper version).
- The `ifthen` option for implementing **if-then-else** and **while-do** control structures.
- The `epsfig` for inserting postscript files as figures into the documents.

5.1.1 The `ifthen` Style Option

This option provides two programming language features that are useful only for people who already know how to program. It defines the two commands

`\ifthenelse{`*test*`}{`*then clause*`}{`*else clause*`}`
`\whiledo{`*test*`}{`*do clause*`}`

that implement the following two Pascal language structures

if *test* **then** *then clause*
  else *else clause*
while *test* **do** *do clause*

The *then*, *else*, and *do* clauses are ordinary LaTeX input; *test* is one of the following:

- A relation between two numbers formed with `<`, `>`, or `=`; for example, `\value{page}>3`.
- `\equal{`*string1*`}{`which evaluates to *true* if *string1* and *string2* are the same strings of characters after all commands have been replaced by their definitions. (Upper- and lowercase letters are unequal.)

- A logical combination of the above two kinds of tests using the operators `\or`, `\and`, and `\not` and the parentheses `\(` and `\)`—for example:

```
\not \( \value{section} = 1  \and  \equal{Jones}{\myname} \)
```

The *test* argument is a violently moving argument, which means that not only fragile commands but even some commands that are not normally fragile will break, causing TEX to enter an infinite loop. The `\protect` command works in these situations.

These commands, together with `\renewcommand` and the commands of Section C.7.4 for manipulating counters, open up a whole new world of hacking.

5.1.2 Letters

The `letter` document style, described in the manual, should be used for generating personal letters. For generating letters to be copied onto CVRTI letterhead, we do not yet have a specific format. Any suggestions or contributions in designing such a style are more than welcome.

5.2 Where the Files Are

All LATEX files mentioned in the manual, including the `sty` and `doc` files, are in the directory `/usr/local/share/texmf/tex`. Fonts are stored in a set of directories which are found under `/usr/local/share/texmf`. The environmental variable `TEXFONTS` determines where LATEX looks for the fonts. You will find other LATEX files in the `/usr/local/share/texmf` area so feel free to snoop around for packages that you read about. Good candidate directories to peruse are the subdirectories of `/usr/local/share/texmf/tex/latex/` .

You should not really need to worry about this level of detail as the programs should find all the packages and files automatically because of the way we organize both the directories and the associated environment variables.

5.3 Fonts

The default font family used in TEX and LATEX was designed by Donald Knuth and named Computer Modern. While LATEX generally takes care of managing the fonts for the user, enlarging type for headings, shrinking it for sub- and super-scripts, the user can manually select the font size using the commands described in section 5.8.1 of the LATEX manual.

There are other font families available for LATEX, perhaps the most useful of which is the Concrete Roman fonts developed by Don Knuth for the book *Concrete Mathematics*, by Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, Addison-Wesley, 1989. To use these fonts, simply change the document style to concrete as follows:

```
\documentstyle[concrete]{article}
```

Other fonts are also to be found on the CVRTI LATEX implementation, however, they have not been extensively tested and very little, if any, documentation exists for them.

They can be found in separate directories under `/usr/local/share/texmf/fonts` and to use them, make sure you add the name of the directory in which they are located to the TEXFONTS environmental variable. If there is a `.sty` file for these fonts, also make sure you copy the `.sty` file to your own directory, or add the name of the directory to the TEXINPUTS environmental variable. (If you don't know how to do this, you probably should not be playing around with alternate fonts in the first place).

5.3.1 Fonts in TEX — Primer

TEX ffonts are rather mysterious and since there is no tutorial article that I know of, here is a brief description provided by Nelson Beebe.

METAFONT produces two output files, the TFM (TEX Font Metric) file containing only the character measurements, and a character bitmap file that contains measurements and bitmaps for each character. TEX itself only uses the TFM files, while DVI drivers require the bitmap files, and only rare drivers ever consult a TFM file.

Old METAFONT (written in SAIL) produced character bitmap files in PXL format, which had scan lines padded to 32-bit intervals, and supported only 128-char fonts.

New METAFONT (written in WEB) produces character bitmap files in GF format, which use run-length encoding within scan lines, and no padding to 32-bit boundaries. GF fonts can have 256 chars.

Tom Rokicki at Texas A&M (now at Stanford) developed a new format, PK, which achieves additional compression by run-length encoding the complete set of rasters, treated as a continuous line of bits.

PXL format is absolutely obsolete and should not be used at all. The fonts created in that format were Knuth's first attempt at designing Computer Modern, and are known as the am (almost Computer Modern) series. The second, and final, version is the cm fonts that you are probably already using.

Conversion utilities are available for moving between all pairs of PXL, PK, and GF formats (standard Computer Modern fonts have only 128 chars, so the PXL limit is not a problem for them). These utilities are known as xxtoyy (pktogf, gftopk, etc). Most people store only PK formats, having converted them from the GF files from METAFONT. This conserves about half the disk space that GF fonts would use up.

Originally, there was a limited set of fonts, and they all resided under `/usr/local/share/texmf/fonts` (TFMs and GF/PK/PXL). Now, many of us have multiple font families, so we have split them into several families:

ams	cm	greek	pandora
ap	cm-misc	imagen	ps
aps	concrete	music	ransom
bin	euler	oe	vf

Thus `/usr/local/share/texmf/fonts/tfm` has all of the TFM fonts for the default Computer Modern family of fonts, while `/usr/local/share/texmf/fonts/pk` contains the set of pk fonts.

size	default (10pt)	11pt option	12pt option
`\tiny`	5pt	6pt	6pt
`\scriptsize`	7pt	8pt	8pt
`\footnotesize`	8pt	9pt	10pt
`\small`	9pt	10pt	11pt
`\normalsize`	10pt	11pt	12pt
`\large`	12pt	12pt	14pt
`\Large`	14pt	14pt	17pt
`\LARGE`	17pt	17pt	20pt
`\huge`	20pt	20pt	25pt
`\Huge`	25pt	25pt	25pt

Table 1: Type sizes for LaTeX size-changing commands.

TFM files are named by *fontname*.tfm. PXL files are named by *fontname*.5 $*$ *dpi*pxl (dpi = dots/inch), based on an old scheme for the Versatec 200dpi printer, so that 1000 == 200dpi. GF and PK files are named by *fontname.dpi*gf or *fontname.dpi*pk.

On systems that have limited file name lengths (e.g. old VAX VMS versions, MS DOS, TOPS-10, Atari TOS, ...), the magnification field is usually split off into a directory name. Thus cmr10.300pk (10pt Computer Modern in normal magnification for a 300dpi laser printer) might be known as 300/cmr10.pk on such systems.

Table 1 tells you what size of type is used for each LaTeX type-size command in the various document-style options. For example, with the `12pt` option, the `\large` declaration causes LaTeX to use 14pt type.

6 Including Graphics in LaTeX

There is no section in the original LaTeX manual dealing with figures other than those created inside LaTeX using the picture environment. There are, however, now whole books dedicated to this topic, one called *The LaTeX Graphics Companion* is especially good. To add graphics to the output, postscript is the most natural path and there has been considerable development toward a standard means of adding postscript to LaTeX.

At the CVRTI, we can add postscript files to LaTeX, using *dvips*. To include postscript files in LaTeX, we use the *epsfig* commands, which are described briefly in *A Guide to Latex* and more fully in the LaTeX Companion books. Note that there is no support for anything but postscript files directly into LaTeX. Briefly, the steps are as follows:

1. Include `\usepackageepsfig` command in the header of the .tex file.
2. Include the figure file using the command `\epsfigfile=figfile.eps` or perhaps more completely, in a figure environment as:

```
\begin{figure}[htb]
  \centerline{\epsfig{file=figfile.eps,height=4in}}
  \caption{\label{yourlabel}Your caption}
\end{figure}
```

Both height and width can be specified and if both are, the aspect ration of the output may be altered so use care. The value of `columnwidth` can be used to make the figure always fill the width of the page, as in

```
\begin{figure}[htb]
  \centerline{\epsfig{file=figfile.eps,width=\columnwidth}}
  \caption{\label{fig:yourlabel}Your caption}
\end{figure}
```

3. When the dvips program runs over the .dvi file, the postscript files will be incorporated and a complete printable file created.

4. You can refer to this figure with a line like

```
The figure~\ref{fig:yourlabel} shows an example of lead
and channels information layered one on top of the other.
See the figure caption for details.
```

6.1 Adjusting figure placement

To adjust the way LaTeX lays out the figures on the page, there are some controls, defined in any LaTeX book, that determine the constraints within which the placement algorithms work. Here are the settings I like to use that allow more space for figures on the page.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Setting to control figure placement
% These determine the rules used to place floating objects like figures
% They are only guides, but read the manual to see the effect of each.
\renewcommand{\topfraction}{.8}
\renewcommand{\bottomfraction}{.6}
\renewcommand{\textfraction}{.2}
```

Setting the larger numbers to 1.0 and the `\textfraction` to 0.0 will apply almost no constraints and allow figures to fill any page.

6.2 Wrapping text around graphics

Sometimes it is nice to be able wrap text around graphics objects on a page. LaTeX does not directly support this behavior and instead uses the full column width even for narrow figures. There are two packages I know of that will do wrapping of text around figures, `picinpar` and `wrapfigure`.

To use `picinpar`, you have to mark the block of text that should be wrapped with `\begin{figwindow}[2,r,%` and `\end{figwindow}`. Embedded in the text between these two must be the figure itself, defined in the standard way as a figure environment. The parameters in the `\begin{figwindow}[2,r,%` command define how many lines there should be above the figure (in this case, 2) and whether the figure is to be on the right (r), left, (l), or center (c) or the page.

To use `wrapfigure`, place before the regular figure environment `\begin{wrapfigure}{r}{3in}` and after the figure `\end{wrapfigure}`. There are optional additional commands for this environment:

```
%  \begin{wrapfigure}[12]{r}[34pt]{5cm} <figure> \end{wrapfigure}
%                     --  -  ----  ---
%  [number of narrow lines] {placement} [overhang] {width of figure}
%
%  Placement is one of   r, l, i, o, R, L, I, O,  for right, left,
%  inside, outside, (here / FLOAT).
%  The figure sticks into the margin by ‘overhang’, if given, or by the
%  length \wrapoverhang, which is normally zero.
%  The number of wrapped text lines is normally calculated from the height
%  of the figure, but may be specified manually ("12" above).
%
```

I recommend adding wrapping as part of a final pass through the document when you can see how it all fits together.

6.3 The problem with Matlab figures

Matlab is a wonderful program for doing all sorts of things with data and simulations. Many of us use it to make figures for papers and reports, and include those figures as postscript files in LaTeX documents. Unfortunately, it produces nasty postscript files that I believe are not compliant with the standard because the contain the following line at the end:

```
%%EOF
```

Now postscript geeks will recognize that `%%` should only mean a comment, but real postscript nerds will know that `%%` is also the symbol for lots of *de facto* commands in postscript. What it means to many postscript interpreters is that the file ends here (EOF = end of file). When such a postscript file becomes part of a larger postscript file, *e.g.,* the output of your LaTeX document, then it will abort the printing process as the first Matlab figure comes along.

The solution to this is brutally primitive. Just edit the postscript file that you get from Matlab; look at the end of the file and remove the `%%EOF`. While you are at it, send a nasty note to Mathworks about this annoying problem.

A Bugs

There are a few known bugs in LaTeX that occur very seldom and cause the user little trouble, but would be very difficult to fix. Moreover, given the nature of complex systems, it is not unlikely that the corrections would lead to even worse problems. Therefore, these bugs will probably not be fixed.

The bugs and ways to get around them are listed below. Do not worry about any of them until you are preparing the final draft, since changes to the text are very likely to cause the problem to disappear.

- In rare instances, a figure or table will be printed on the page preceding the text where the `figure` or `table` environment appears. This can be fixed by moving the environment further towards the end of the document.

- A marginal note at the top of a page may appear in the wrong margin. This can be fixed by inserting a redundant `\pagebreak` command to force a page break exactly where LaTeX started the new page anyway.

- A footnote can be broken across two pages when it should fit on a single page. This happens when there is one or more figures or tables on the page. The problem is corrected by moving, towards the end of the file, the last `figure` or `table` environment that produces a figure or table on the page where the footnote starts.