A Simplified Introduction to LaTeX

Harvey J. Greenberg
University of Colorado at Denver
Mathematics Department
PO Box 173364
Denver, CO 80217-3364
hgreenbe@carbon.cudenver.edu
http://www.cudenver.edu/~hgreenbe/

October 23, 1999

Table of Contents

List of Figures

List of Tables

Preface

The majority of this book is about using LaTeX 2_ε [2, 10], a descendant of LaTeX, designed by Leslie Lamport [9], based on TeX, originated by Donald E. Knuth [8]. This is a *typesetting* program, not a word processor. You enter some editor that saves plain text files. Then, you type text freely until you need something special, such as *italic* font or a complex mathematical expression, like

$$\lim_{\varepsilon \to 0_+} \frac{\int_{a_i}^{a_i+\varepsilon} \sqrt{1+(x-\mu)^2}\, dx}{\Phi(\varepsilon)}.$$

It was the desire to have high quality, low cost publications in mathematics and related disciplines that caused Knuth (pronounced Kah-nooth) to invent TeX (pronounced Tek) in the late 1970's. Originally believing that he could write a program in less than a year that could typeset documents, he actually ended up defining an entire branch of research in computer science. It was 10 years later that he published his seminal book [8], but he published articles along the way, and he permanently changed the way mathematical documents are prepared. LaTeX (pronounced Lah-tek or Lay-tek) is a collection of *macros* built on top of TeX that "represents a balance between functionality and ease of use" [9, p. xiii]. LaTeX 2_ε is the current version, developed by a team of volunteers: Johannes L. Braams, David P. Carlisle, Alan Jeffrey, Frank Mittelbach, Chris Rowley, and Rainer Schöpf [2].

A comprehensive coverage of LaTeX and the many enhancements to it is given by the *The LaTeX Companion* [5]. By contrast, this book is designed to be a succinct introduction, omitting many of the things LaTeX 2_ε can do. My goal is to offer enough of an introduction that someone not acquainted with LaTeX (or with TeX) can write a term paper, thesis, or article, using LaTeX 2_ε to produce high quality results. Exercises are provided for guided instruction, which should be just a few classes. For one who is well acquainted with computers, particularly unix, the basics that are covered should take less than 10 hours, and one could do all of the exercises. For one who is just learning how to use a computer, it will take longer, especially getting used to functioning at the command line. In any case, the finer points require more study.

Happy TeXing.
— Harvey J. Greenberg

Acknowledgements

The author thanks the many contributors in the `comp.text.tex` newsgroup, particularly Donald Arseneau, Herman Bruyninckx, David Carlisle, Robin Fairbairns, Jonathan Fine, Denis Girou, David Haller, Dan Luecking, Timothy Murphy, Sebastian Rahtz, Axel Reichert, Thomas Ruedas, Bernd Schandl, Anton Schwaighofer, Mårten Svantesson, and Matt Swift, who were very generous with taking time to answer so many questions on a regular basis. I also received useful comments from people who read an earlier draft that I made available on the web: William Briggs and Kasper B. Graversen. Last, but not least, I thank Allen G. Holder, who taught me LaTeX in the first place.

Sources of LaTeX Software

The basic LaTeX software system is available free of charge for unix systems, and MiKTeX [13] is available free of charge for DOS systems. The best source of these, and additional packages that extend the LaTeX capabilities (and to which we refer in this book), is at the Çomprehensive TeX Archive Network (CTAN) [4], at three host sites (and many mirrors):

1. http://ctan.tug.org/ in Boston, MA, USA,
2. http://www.tex.ac.uk/ in Cambridge, UK, and
3. http://www.dante.de/ in Mainz, Germany (in German).

These all describe how to search and browse the FTP sites for software and documents.

1 Overview

You will create a file, called the LaTeX *source*, which is plain text. To keep things simple, its suffix is `.tex`, so for example we refer to `myfile.tex` as a plain text source file that you create. Figure 1 shows the structure of this file, which we shall be describing in greater detail throughout this book.

```
% This is myfile.tex
% notes to yourself can go here
```
Anything following % is ignored (used for comments).

```
\documentclass[options]{style}
 optional specifications
 — e.g., declaring use of packages
```
Preamble
(blank lines do not matter)

```
\begin{document}
        ⋮
\end{document}
```
Body
This is the document *environment.*

All that follows is ignored
(could be used for comments).

Figure 1: The Structure of a LaTeX Document.

In the preamble, there are many options, depending upon the style; the intrinsic document styles are: `article`, `book`, `letter`, `report`, and `slides`. Most publishers have their own style, which you can obtain free of charge. Among these are professional societies, notably the American Mathematical Society (amsmath style) and the Society for Industrial and Applied Mathematics (siam style).

Our focus throughout this book is on the article style. Further, we shall be using defaults for almost everything, concentrating on getting started with using LaTeX as quickly as possible. Later, some of the options, like margin settings and other preamble specifications are covered, as well as more advanced topics for customizing your document.

Are you ready to write your first LaTeX document? Copy the source file shown in Figure 2 and name it `myfile.tex`. Then, at the command line, enter:

```
latex myfile
```

(In an MS Windows system, the *command line* is the DOS command line, which you enter by running Start⟶Programs⟶MS-DOS Prompt.)

```
\documentclass{article}
\begin{document}

   Hello world.

\end{document}
```

Figure 2: Your First LaTeX Source File

This is called *compiling* your source, which creates several output files. The only one you need to be concerned with now is the *dvi file*, which the latex program (called a "compiler") names `myfile.dvi`. One of three things will have occurred:

Case 1. You got messages, but they were not fatal errors.

Among the non-fatal messages you will generally see are warnings like:

```
Overfull \hbox ...
Overfull \vbox ...
Underfull \hbox ...
Underfull \vbox ...
```

Do not worry about these.

Case 2. You got a fatal error message.

You must find and correct it. This is called *debugging* your source. Sometimes the error message tells you what went wrong, such as missing a brace (characters `{` and `}`, which you will come to know and love), or some command was not recognized due to being misspelled. Many times the message is not very informative, so you are advised to compile often. That way you will know that what you did in the last few minutes contains the error.

Case 3. You got no messages.

Something went wrong and you need to ask for help.

If all went well, the first thing to do is **SAVE A BACKUP** by copying your source file to another subdirectory, or to a different name. In unix do this by entering:

```
cp -p myfile.tex myfile-1.tex
```

(The `-p` is to keep the date and time of the source file.) Change `-1` to another qualifier each time (e.g., `-2` ...), so you have a collection of backups. If you are running under DOS, use `copy myfile.tex` *destination*, where the *destination* is either `a:` or some backup file name. (If you are familiar with DOS, nothing more need be said; if not, you need to learn how to create, edit and save plain text files.)

Next you want to view the result. If you are in a unix environment, you can view the result with the *dvi viewer*, `xdvi`. At the command line enter:

```
xdvi myfile
```

and it will come on your screen. (There is more to do if you are working remotely, in which case you must ask someone for help.)

If you are using DOS, the viewer that comes with MiKTeX [13], a free software system by Christian Schenk, is called YAP. At the DOS command line you enter:

```
YAP myfile
```

You will see various options for viewing and printing.

Under unix, `xdvi` does not have a print option, so you first need to convert the dvi file to postscript. This is done with the program, `dvips`. At the command line enter:

```
dvips myfile -o
```

(The `-o` tells the system you want the output to go to a file, rather than just print; your installation might already have file output as the default, in which case the `-o` is not needed.) This will result in the creation of the postscript file `myfile.ps`. You can print it in any number of ways, including the unix command, `lpr myfile.ps`.

The same conversion program can be run under DOS (and comes with MiKTeX), and you might want to obtain `myfile.ps` for a variety of reasons, including posting it on the web. To view or print a postscript file under DOS, you can run a program called `ghostview`. You will need to find out more about viewing and printing postscript files that suit your particular needs.

Summarizing, you begin by entering a plain text editor. In unix this could be pico, emacs, vi or vim. In DOS you can use EDIT at the command line, or you can use Notepad,Wordpad or MS Word. If you use a word processor, however, you must take absolutely no advantage of its formatting. You should even put in hard return characters (i.e., press ↵ Enter at the end of a line instead of letting the word processor do it for you), and **never use tabs**. In MS Word, when saving the file, be sure to specify plain text, and you must continue to specify the suffix as `.tex` (otherwise, it will use `.doc` as its suffix). If you want to check that the file is really plain text, enter

```
EDIT myfile.tex
```

at the DOS command line and see how the file appears. Once you have your source ready to compile, enter `latex myfile`, and if all is well, enter your dvi viewer. Under unix or DOS use `dvips` to convert the dvi file to a postscript (ps) file, which can be printed. These steps are given in Figure 3. Execute these commands for the source file shown in Figure 2. The result should be one line of output: Hello World. Congratulations!

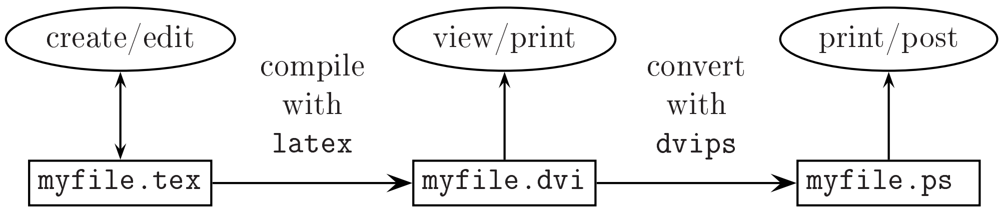


Figure 3: Command Sequence from Source to Postscript

Now change your document to specify a font size of 12pt (default is 10pt) by changing your first line to the following:

```
\documentclass[12pt]{article}
```

The "pt" (abbreviation for "point") is one of the units of measurement, about $\frac{1}{72}$in; other units used in many parts of LaTeX are `in` (inches), `cm` (centimeters), and `em` (like the letter m, which is a printer measure equal to the width of M in the current font).

2 Text

We begin by illustrating the most common text formatting, much like you would want in a word processor. (The power of LaTeX will be evident when we get to mathematical expressions, but even some text, like tables, will demonstrate the superior quality of the LaTeX results.) First, consider how to make sections and subsections in your article style. Figure 4 is the source that produces the result in Figure 5, showing how sections and subsections are defined. Note the automatic numbering, and how extra spaces and blank lines have no effect.

```
\documentclass[12pt]{article}

% We have defined the document to be an article using 12 point font.
% Blank lines mean nothing here, in the preamble.

\begin{document}               % Begin document "environment".

\section{This is a Section}
\subsection{This is a subsection}
This is the body of the subsection.
I can move to a new line anytime, and I can put in     lots
of       blanks     with       no        effect.



Skipping four lines is the same as skipping one line
--- it starts a new paragraph.

\subsection{Here is another subsection}
\section{Here is another section}
\end{document}
```

Figure 4: An Introductory Document Source (Result in Figure 5)

2.1 Fonts and Paragraphs

Figure 6 shows the source to produce different paragraph positions: centered, flush left, flush right, and justified (the default). Note that these are *envi-*

1 This is a Section

1.1 This is a subsection

This is the body of the subsection. I can move to a new line anytime, and I can put in lots of blanks with no effect.

Skipping four lines is the same as skipping one line — it starts a new paragraph.

1.2 Here is another subsection

2 Here is another section

Figure 5: An Introductory Document Result (Source in Figure 4)

ronments, a concept you need to understand about LaTeX. The general form of an environment uses the following syntax:

\begin{*environment*}
⋮
\end{*environment*}

```
\begin{center}
  The text is centered because I have entered the center environment.
  Text remains centered as long as we remain in this environment.
\end{center}
\begin{flushleft}
  Now we are out of the centering environment, and have begun the
  flushleft environment.
\end{flushleft}
\begin{flushright}
  This is another paragraph, but in the flushright environment.
  You will have occasion to use all four paragraph positions.
\end{flushright}

I am back to normal justification.  The added space you see between
the above paragraphs is due to entering those environments.
```

Figure 6: Positioning Paragraphs Source (Result in Figure 7)

The text is centered because I have entered the center environment. Text remains centered as long as we remain in this environment.

Now we are out of the centering environment, and have begun the flushleft environment.

This is another paragraph, but in the flushright environment. You will have occasion to use all four paragraph positions.

I am back to normal justification. The added space you see between the above paragraphs is due to entering those environments.

Figure 7: Positioning Paragraphs Result (Source in Figure 6)

Instead of the `center` environment, you can use the `\centerline` command; they differ in that the environment skips a line before and after the paragraph, shown in figures 8 and 9.

```
This precedes center environment.
\begin{center} This line is centered. \end{center}
This continues after centering.

This precedes centerline.

\centerline{This line is centered.}
This continues after centering.
```

Figure 8: Centering Source (Result in Figure 9)

This precedes center environment.

This line is centered.

This continues after centering.

This precedes centerline.
This line is centered.

This continues after centering.

Figure 9: Centering Result (Source in Figure 8)

You can also suppress indentation of the first line of a paragraph with the `\noindent` command. Here is an example:

`\noindent This paragraph is not indented.` produces:
This paragraph is not indented.

Table 1 lists the fonts that are intrinsic in a basic latex installation. (More fonts are available in packages, usually free of charge.) In technical writing, you will have particular use for the italic font, as it is used when introducing a new term. For example,

```
A \textit{group} is defined on a set of elements \dots
```
$\Rightarrow$ A *group* is defined on a set of elements ...

(The $\Rightarrow$ symbol can be read as "produces.") Note the use of the `\dots` command, which produces the ellipsis.

What you write		How it appears
`This is \textbf{boldface}.`	⇒	This is **boldface**.
`This is \textit{italic}.`	⇒	This is *italic*.
`This is \textrm{roman}.`	⇒	This is roman.
`This is \textsc{small caps}.`	⇒	This is SMALL CAPS.
`This is \textsf{sans serif}.`	⇒	This is sans serif.
`This is \textsl{slanted}.`	⇒	This is *slanted*.
`This is \texttt{typewriter}.`	⇒	This is `typewriter`.

Table 1: Intrinsic Font Styles

Some combinations of font styles can be produced. For example,

`\textbf{\textit{bolditalic}}` ⇒ ***bolditalic***.

The *argument* of `\textbf` is `\textit{bolditalic}`. The general form is `\text`*font*`{`*text*`}`, where *font* is one of {`bf, it, rm, sc, sf, sl, tt`}, as seen in Table 1.

Not all combinations are in the basic LaTeX 2_ε installation. In particular, you must put `\usepackage[T1]{fontenc}` in your preamble to obtain:

`\textbf{\textsc{bold small caps}}` ⇒ **BOLD SMALL CAPS**.

Font size can also be varied at will. Figures 10 and 11 give the source and result for common variations. Notice how the paragraph spacing changes to accommodate the variation in font size. These size variations can be combined with font styles, such as using `{\Large\textbf{heading}}` for some heading.

```
You can make the text {\large large} or {\Large larger} or
even {\LARGE larger still}.  You can also make it {\huge huge}.
You might want to make something {\small small} or
{\footnotesize smaller} or even {\scriptsize smaller still}.
You can make it really {\tiny tiny}.
```

Figure 10: Some Font Sizes Source (Result in Figure 11)

The use of braces to enclose a font size specification is like an environment. Optionally, we can explicitly use the environment syntax: `\begin{`*size*`}` ... `\end{`*size*`}`. For example, `\begin{large} This is large.\end{large}` produces the same result as `{\large This is large.}`. The environment syn-

You can make the text large or larger or even larger still. You can also make it huge. You might want to make something small or smaller or even smaller still. You can make it really tiny.

Figure 11: Some Font Sizes Result (Source in Figure 10)

tax is useful when you want to keep the size for a large block of text, and the braces format is useful for short phrases. (There is no intrinsic environment for font styles.)

It is straightforward to underline text — just write `\underline{`*text*`}`. We can also frame text just by writing `\frame{`*text*`}`. We can give frame some room around the edges by using `\fbox` instead. (More on framing in §6, p. 76.) To overline is as straightforward, but learning it must wait until we enter *math mode*.

Now consider ways to indent a block of text. Here is an example using the *quote environment*, which was generated by putting `\begin{quote}` before the text and `\end{quote}` after it:

> The construction of the real number system, notably by Dedekind cuts, was motivated by the need to fix calculus, which ran into trouble due to insufficient rigor in dealing with limits.

The quote environment is intended for short quotes, generally one short paragraph (as above), or a sequence of one line quotes, separated by blank lines. The `quotation` environment is used for long quotations, having more than one paragraph (separated by blank lines). The indentation is the same as the quote, except the first line of each new paragraph is indented. (Just as in the regular text, this can be overridden by the `\noindent` command.) Here is an example that was created by putting `\begin{quotation}` before the text and `\end{quotation}` after it.

> “Computers do not dream, any more than they play. We are far from certain what dreams are good for, but we know what they indicate: a great deal of information processing goes on far

> beneath the surface of man's purposive behavior, in ways and for reasons that are only very indirectly reflected in his overt activity."
>
> — Alan M. Turing
>
> "There are reports that many executives make their decisions by flipping a coin or by throwing darts, etc. It is also rumored that some college professors prepare their grades on such a basis. Sometimes it is important to make a completely 'unbiased' decision; this ability is occasionally useful in computer algorithms, for example in situations where a fixed decision made each time would cause the algorithm to run more slowly."
>
> — Donald E. Knuth

The quotes are by two pioneers of algorithms, Alan M. Turing and Donald E. Knuth. Their names appear on the right, after their quote, by skipping a line and entering `\hfill` (which means *horizontal fill*), to make the line flush right. Here are some other things to notice about this example:

- There are left and right quotation marks. I used ``` `` '' ```, not `" "`, to create this more stylistic quotation punctuation.
- The dash that appears before each name is created by three minus signs, `---`. The more minus signs you use, the longer the dash. The convention is that one dash is for hyphenation, two are for ranges, such as page numbers, and three are for punctuation — i.e., use `---` preceding "i.e."
- There is extra space between the two quotations. This was done with the `\bigskip` command.

Figures 12 and 13 illustrate three levels of skipping: small, medium and big. Later, we shall look at a way to have a much finer range of vertical spacing.

The `verse` environment will indent oppositely, indenting lines after the first. The following was generated by putting `\begin{verse}` before the text and `\end{verse}` after it:

```
This is a first line. \bigskip

The space you see is a big skip.  \medskip

The space you now see is a medium skip.  \smallskip

The space you now see is a small skip.

This is just an ordinary line space.
```

Figure 12: Skipping Line Spaces Source (Result in Figure 13)

This is a first line.

The space you see is the big skip.

The space you now see is a medium skip.

The space you now see is a small skip.
This is just an ordinary line space.

Figure 13: Skipping Line Spaces Result (Source in Figure 12)

> *Neglect of mathematics works injury to all knowledge, since he who is ignorant of it cannot know the other sciences or the things of this world. And what is worse, men who are thus ignorant are unable to perceive their own ignorance and so do not seek a remedy.* — Roger Bacon

The italics were specified in the usual way, by enclosing Bacon's verse with `\textit{ ... }`. (Designed for poetry, each line is a stanza in the verse, and if a stanza runs long, this form of indentation makes it clear.) Bacon's name appears flush right, again from the `\hfill` command, but this time it is on the last line of the verse, rather than a new line. This is achieved simply by not skipping a line after the verse:

```
\begin{verse}
\textit{
  Neglect of mathematics ...
} \hfill --- Roger Bacon
\end{verse}
```

2.2 Lists

There are three intrinsic list environments, distinguished by what appears at the beginning of each item: number, bullet, or your description (perhaps nothing). To illustrate, here is the use of a *description list environment* to itemize steps involved in learning LaTeX, whose source is indicated by Figure 14.

Basic Document Preparation. Knowing how to setup the latex source file, make paragraphs, vary fonts, and list items are enough to prepare a basic document without mathematics or tables (like a resume).

Making Tables. LaTeX provides a means to make tables with the *tabular environment*, and its versatility puts it far ahead of word processors.

Bibliography. Knowing how to create a bibliography, in particular with BibTeX.

Mathematics. This is a power of LaTeX and one reason why it has become standard in writing mathematical papers. I will show you how to do virtually any mathematical expression in line with the text, or in *math display mode*.

Graphics. This has progressed a great deal in the past few years thanks to many people who have provided packages free of charge.

Other. There are a great many things to learn beyond the simple introduction when using LaTeX to prepare a thesis, report or article.

Two new things appear in the example: the use of `\LaTeX` to produce LaTeX, and the use of `~` (called "tilde") to enter a space. Without the tilde, the result would be LaTeXprovides, even with a space after `\LaTeX`

```
\begin{description}
  \item [Basic Document Preparation.]  Knowing how to setup ...
  \item [Making Tables.] \LaTeX~ provides a means ...
  \item [Bibliography.]  Knowing how to create a bibliography ...
  \item [Mathematics.] This is the power of \LaTeX~ and one ...
  \item [Graphics.] This has progressed a great deal in the ...
  \item [Other.]  There are a great many things to learn ...
\end{description}
```

Figure 14: Description List Environment

in the source. (The reason is that a space is needed anyway after `\LaTeX` (or any keyword) in order to distinguish it completely, and one might want a punctuation mark, like a comma, following `\LaTeX`, which requires no space.)

The text within the square brackets is an option. If present, as in this example, it is printed in boldface. With no option, the description list is one way to have text indented the opposite of a normal paragraph: the first line is at the left and subsequent lines are indented. For example,

```
\begin{description}
   \item This is how one item in a description list environment
         looks with no optional text at the beginning.
 \end{description}
```

produces the following result:

This is how one item in a description list environment looks with no optional text at the beginning.

Unlike the verse environment, the first line goes almost to the left margin, and the lines extend all the way to the right margin.

Next, Figures 15 and 16 illustrate the *itemize list environment*, which prints bullets. Note the indentation of each item and the spacing between items. You see the nesting of two itemize lists, but any type of list can be nested within any other type.

```
\begin{itemize}
   \item This is item 1 and our task has just begun.  Blank lines
         before an item have no effect.

   \item This is item 2 and we shall limit to just this few.

         A blank line within an item does create a new paragraph,
         using the indentation of the itemize environment.

      \begin{itemize}
         \item A second (nested) itemized list changes the bullet
               and indents another level.
      \end{itemize}
\end{itemize}
```

Figure 15: Itemize List Environment Source (Result in Figure 16)

- This is item 1 and our task has just begun. Blank lines before an item have no effect.
- This is item 2 and we shall limit to just this few.

  A blank line within an item does create a new paragraph, using the indentation of the itemize environment.

  – A second (nested) itemized list changes the bullet and indents another level.

Figure 16: Itemize List Environment Result (Source in Figure 15)

Finally, we consider the enumerate list environment, where the default numbering is with Arabic numerals. With nested enumeration, the numbering changes at each level. Figures 17 and 18 illustrate with three levels of nesting.

```
\begin{enumerate}
   \item This is item 1, and we are having fun.
   \item This is item 2, and it's time to number anew.
      \begin{enumerate}
         \item Back to item 1, but we are not yet done.
         \item Two is new.
            \begin{enumerate}
               \item One again!
               \item Two (b) or knot 2b?
            \end{enumerate}
      \end{enumerate}
\end{enumerate}
```

Figure 17: Enumerate List Environment Source (Result in Figure 18)

1. This is item 1, and we are having fun.
2. This is item 2, and it's time to number anew.
   (a) Back to item 1, but we are not yet done.
   (b) Two is new.
      i. One again!
      ii. Two (b) or knot 2b?

Figure 18: Enumerate List Environment Result (Source in Figure 17)

2.3 Making Tables

A table is made with the tabular environment, which has the following syntax:

\begin{tabular}{*column specs*} *options*
first row spec \\\\
⋮
last row spec [\\\\ *options*]
\end{tabular}

As indicated, each row ends with two backslashes, \\\\. Each column specification can be left, center or right, abbreviated by just one character:

`l`, `c` or `r`, respectively. In the body of the table, each column is separated by `&`. Figure 19 shows an example of a 2×3 table.

How it appears			What you write
			`\begin{tabular}{lcr}`
left	center	right	` left & center & right \\`
1	2	3	`   1  &  2     &  3`
			`\end{tabular}`

Figure 19: A 2×3 Table

We can draw a horizontal line before any new row by specifying `\hline`. To draw a line after the last row, we enter `\\ \hline` (the `\\` is simply part of the syntax and does not add an extra row to the table). The column specifications can have `|` on either side to indicate a vertical line. Figure 20 illustrates a combined use of these options.

How it appears

-110	-120.12	-130
210	220.	230

What you write

```
\begin{tabular}{|l|c|r|} \hline
  -110 &  120 & -130 \\ \hline
   210 & -220 &  230 \\ \hline
\end{tabular}
```

Figure 20: A 2×3 Table with Horizontal and Vertical Lines

We could draw lines that span some rows and/or columns. The way to vary vertical line drawing is with the column specifications: put `|` only where you want a vertical line. The way to vary horizontal line drawing is by using `\cline{`*first col-last col*`}` instead of `\hline`. This is illustrated in Figure 21.

How it appears

Name	Test 1	Test 2
Bob	67	72
Sue	72	67

What you write

```
\begin{tabular}{l|cc|}
 Name  & Test 1 & Test 2 \\ \cline{1-1}
 Bob   &  67    &  72 \\
 Sue   &  72    &  67 \\ \cline{2-3}
\end{tabular}
```

Figure 21: A Table with Partially Spanning Horizontal and Vertical Lines

We can have tables nested within tables. Figures 22 and 23 illustrate this, while showing more variation with lines and using various fonts. Here are some things to note:

- The entire table uses sans serif font style. This is done by specifying `\textsf{` *before* entering the tabular environment (and closing with `}` just after it).
- Within the tables, fonts are varied: Roman is in the Roman font, specified by `\textrm{Roman}`, Greek is in italic, specified by `\textit{Greek}`, and upper case is in small caps, specified by `\textsc{upper case}`.
- A new column specification is introduced: `p{`*length*`}`, where any unit of measure can be used as the length of the spacing. In this example .3 inches is specified. Note that this counts as a column, so you see `&&` to separate the two tables, each being a column of the main table.
- The `\underline` command is used to underline Table 1, which is column 1 of the main table, whereas `\cline{3-3}` is used to underline all of column 3 of the main table, headed by Table 2.

```
\textsf{
 \begin{tabular}{lp{.3in}l} \\
   \underline{Table 1}  &&    Table 2 \\ \cline{3-3}
  \\
   \begin{tabular}{|lc|} \hline
      Object    & Symbols used \\ \hline
      variable  & lower case \textrm{Roman} \\
      parameter & \textit{Greek}            \\
      constant  & \textsc{upper case} \textrm{Roman} \\
   \end{tabular}
 && % Begin Table 2
   \begin{tabular}{|rcc|} \hline
     * & 1 & 2 \\ \cline{2-2}
       & 3 & 4 \\ \cline{1-1}\cline{3-3}
   \end{tabular}
 \end{tabular}
} % end sf
```

Figure 22: Nested Tables Source (Result in Figure 23)

Table 1

Object	Symbols used
variable	lower case Roman
parameter	*Greek*
constant	UPPER CASE Roman

Table 2

*	1	2
	3	4

Figure 23: Nested Tables Result (Source in Figure 22)

There are times when we want to put a good bit of text into some columns of a table. Suppose, for example, we write the following:

```
\begin{tabular}{|l|l|} \hline
    This amount of text is too long to fit on one line of
    the page.  & This is column 2. \\ \hline
\end{tabular}
```

The result will be to run off the edge of the paper:

This amount of text is too long to fit on one line of the page.	This is column 2.

One solution is to insert new rows and break up the text manually:

```
\begin{tabular}{|l|l|} \hline
    This amount of text is too long to fit on one
          & This is column 2. \\
    line of the page.  & \\ \hline
\end{tabular}
```

⇒

This amount of text is too long to fit on one	This is column 2.
line of the page.	

Instead, one can assign a fixed width to a column by specifying `p{`*length*`}`.

For example,

```
\begin{tabular}{|p{2in}|l|} \hline
This amount of text is too long to fit on one line of the page.
         & This is column 2. \\ \hline
\end{tabular}
```

⇒

This amount of text is too long to fit on one line of the page.	This is column 2.

Another is solution is to use the `\parbox` command (short for "paragraph box"). This has the form `\parbox[`*option*`]{`*width*`}{`*text*`}`, where the *option* is the placement: `t` = top and `b` = bottom (default is center). Here are two examples:

```
\begin{tabular}{|l|l|} \hline
    \parbox{2in}{This amount of text is too long to fit on
       one line of the page.} & This is column 2. \\ \hline
\end{tabular}
```

⇒

This amount of text is too long to fit on one line of the page.	This is column 2.

```
\begin{tabular}{|l|l|} \hline
    \parbox[t]{2in}{This amount of text is too long to fit on
       one line of the page.} & This is column 2. \\ \hline
\end{tabular}
```

⇒

This amount of text is too long to fit on one line of the page.	This is column 2.

They differ only in the placement of the paragraph box, the latter being at the top to align it with column 2 in the manner shown.

When making a column or parbox small, the spacing can become unsightly due to being justified. This is overcome with the flushleft environment.

Figures 24 and 25 illustrate this, and note that it contains other commands that can be in any paragraph.

```
\begin{center}
\begin{tabular}{ll}
   \parbox[t]{3in}{\begin{flushleft}
      This is column 1, and I might want to display something:

      \medskip\centerline{\fbox{How sweet it is.}}\medskip

      This is not the same as

      \medskip\fbox{\centerline{How sweet it is.}}
                  \end{flushleft} }
  & \parbox[t]{1in}{\begin{flushleft}\textsf{
      This is column 2, which I have put in sans serif font.}
                  \end{flushleft} }
\end{tabular}
\end{center}
```

Figure 24: `\parbox` Source (Result in Figure 25)

This is column 1, and I might want to display something: $\boxed{\text{How sweet it is.}}$ This is not the same as $\boxed{\text{How sweet it is.}}$	This is column 2, which I have put in sans serif font.

Figure 25: `\parbox` Result (Source in Figure 24)

Any measurement, such as the width of a paragraph box, can be determined by some length parameter, rather than a fixed constant. For example, see exercise 9 at the end of this chapter and consider `\parbox{.3\linewidth}{...}`.

If we want some heading to span several columns, this is done by the command, `\multicolumn{`*number*`}{`*col spec*`}{`*entry*`}`, . The first argument is the number of columns to span (starting where `\multicolumn` is specified).

This must be in the range 1 to however many columns remain from the current position. The second argument is any valid column specification, such as `l`, `c`, `r`, with, or without, a vertical line specification, `|`, on either side. Finally, the third argument is the text.

The `\multicolumn` command can also serve to override some column specification. Suppose, for example, we want the columns to be left justified, but we want the headers centered. Figures 26 and 27 illustrate these uses of `\multicolumn`. The first is used to center 'Test number' over columns 2 and 3. The line in the source begins with `&` to skip column 1, then the `\multicolumn` specifies 2 columns, centered with vertical lines before and after. The second use simply centers the 'Student' header. The last use of `\multicolumn` centers 'Taken in class' over columns 2 and 3. Unlike the first use, the vertical line at the end is missing because `c` was specified instead of `c|`.

```
\begin{center}
\begin{tabular}{l|cc|c}
             & \multicolumn{2}{|c|}{Test number} \\
  \multicolumn{1}{c|}{Student} &    1   &    2   & Average \\ \hline
                  Bill         &   67   &   72   &   70.5  \\
                  Charleetah   &   72   &   67   &   70.5  \\ \hline
             & \multicolumn{2}{c}{Taken in class}     \\ \cline{2-3}
\end{tabular}
\end{center}
```

Figure 26: Multicolumn Source (Result in Figure 27)

	Test number		
Student	1	2	Average
Bill	67	72	70.5
Charleetah	72	67	70.5
	Taken in class		

Figure 27: Multicolumn Result (Source in Figure 26)

2.4 Special Characters

We have already seen that some characters are special, in that they *delimit* something. In particular, \ delimits every LaTeX command, and `%` ends a line (to allow for comments). How do we print such characters? One way is with the symbol, itself, like `\%`. Other times a keyword, like `\textbackslash`, is needed. The Appendix contains complete tables of these (including those I do not cover in the chapters). Of particular importance are the braces, written as `\{ \}` to obtain { }. (Recall that the braces by themselves create a local environment, like `\textit{` ... `}`.)

When using a keyword to specify a special character, it appears with whatever font is active. Table 2 illustrates this with commonly used special characters. The brackets, [ ], are different because they can be entered directly, except when they are used to delimit an option in the syntax, in which case they can be obtained by enclosing them in braces. One example is the description list environment, illustrated in Figure 28. (Another is in the tabular environment (page 16), but we omitted a discussion of position options that are specified by brackets.)

How it appears	What you write

This is option for item.	`\begin{description}`
	`\item [This is option] for item.`
	`\end{description}`

[This is not option] for item.	`\begin{description}`
	`\item {[This is not option]} for item.`
	`\end{description}`

Figure 28: Obtaining Brackets in a Description List Environment

Another way to print the unprintable is with the verbatim environment or the `\verb` command (short for verbatim). Unlike all other commands, `\verb` does not use braces to delimit its argument. It uses any other character to delimit a string, which can contain any character except itself. For example, we can write `\verb@{}%$#\@` to generate the string `{}%$#\`, which is printed in typewriter font style. The verbatim environment uses the usual syntax:

```
\begin{verbatim}
        ⋮
\end{verbatim}
```

Character (Roman)	How you write it	Other fonts: italic	Other fonts: large
{ }	\{ \}	*{ }*	{ }
% $ # & _	\% \$ \# \& _	*% $ # & _*	% $ # & _
\	\textbackslash	**	\
^	\textasciicircum	*^*	^
~	\textasciitilde	*~*	~
[ ]	{[ ]}	*[ ]*	[ ]

Table 2: Writing Special Characters

This is how the source code was created for the figures, like Figure 26 (p. 22).

Another class of special characters are letters with accents. Table 3 shows some common examples; a complete table is in the Appendix. For example, write Poincar\'{e} to have Poincaré and G\"{o}del to have Gödel. An accent could be applied to any letter, even if it does not relate to some language.

LaTeX has a basic library of accents and special characters for writing in languages other than English — e.g., write ?‘\ae\aa\OE!‘ to obtain ¿æåŒ¡. However, these are not sufficient in some cases, particularly if the entire document is to be in a non-English language. For that purpose there are some packages, available free of charge, such as Babel [1] (also see [5, chapter 9]).

What you write		What you see
\"{a}	$\Rightarrow$	ä
\‘{e}	$\Rightarrow$	è
\'{i}	$\Rightarrow$	í
\~{o}	$\Rightarrow$	õ
\^{u}	$\Rightarrow$	û

Table 3: Some Accents for Letters

2.5 Tabbing

The tabbing environment provides an alternative to the tabular environment by letting you set your own column tabs. Table 4 shows a simple case with two basic tabbing commands, `\=` to define a tab setting, and `\>` to move to a tab setting. In addition, `\\` ends each row, but unlike the tabular environment, the first sentence continues normally, without extra spaces, so that the position of the tab is not equivalent to that of a table's column.

What you see	What you write

Begin: set tab 1 . . . set tab 2
skip to 1 then to 2
skip to 2
	`\begin{tabbing}`
	`Begin: \=set tab 1\dots \=set tab 2\\`
	`\>skip to 1 \>then to 2\\`
	`\> \>skip to 2`
	`\end{tabbing}`

Table 4: The Tabbing Environment

Sometimes we do not want to have the longest portion of text first, yet it is needed to define the tab. Table 5 illustrates how this is solved with the `\kill` command. In the first tabbing, the lines are in the order we want, but the tab is set by the shorter string '1-3', making '4-6-8' extend past the tab. The second tabbing puts the longer field first, in order to set the tab correctly, then specifies `\kill` instead of `\\` to suppress (or "kill") the printing of the line.

What you see	What you write

1-3 sting like a bee
4-6-8don't be late
	`\begin{tabbing}`
	`1-3 \= sting like a bee \\`
	`4-6-8 \> don't be late \\`
	`\end{tabbing}`

1-3 sting like a bee
4-6-8 don't be late
	`\begin{tabbing}`
	`4-6-8 \= don't be late \kill`
	`1-3 \> sting like a bee \\`
	`4-6-8 \> don't be late \\`
	`\end{tabbing}`

Table 5: The `\kill` Tabbing Command

Figures 29 and 30 illustrate the tabbing environment with fixed field widths. It first uses the `\hspace*` command for horizontal spacing, then it uses the name of the last field to set what follows.

```
\begin{tabbing}
  \= \hspace*{.5in} \= \hspace*{2in} \= Last field: \= \kill
  \> Field 1 (following tab 1)
  \\ \> \> Field 2 on new line \> Last field
  \\ \> \>                    \> Last field on new line
\end{tabbing}
```

Figure 29: Tabbing Source (Result in Figure 30)

Field 1 (following tab 1)		
	Field 2 on new line	Last field
		Last field on new line

Figure 30: Tabbing Result (Source in Figure 29)

2.6 Line and Page Breaks

You can cause a new line by entering `\linebreak`. When text is justified (the default), this could result in an undesirable appearance, like the following:

`\textsf{This example is \linebreak extreme.}`
⇒This example is
extreme.

The `\newline` command forces a new line without justifying it.

`\textsf{Here is the extreme \newline example.}`
⇒Here is the extreme
example.

The `\nolinebreak` command works analogously, preventing a line break, even if it means extending into the right margin. Also, it is better style to keep certain 'words' together, such as 'figure 1' or 'p. 10'. To prevent a line break where you want a blank, use the space character `~`. We would thus write `figure~1` or `p.~10`.

There are two commands to force a page break: `\pagebreak` and `\newpage`. The `\newpage` command follows the analogy with `\newline` in forcing a page break precisely at the point it is specified, rather than completing the line as

`\pagebreak` does. The `\nopagebreak` command disallows a page break immediately following the next blank line. The `\samepage` command prevents a page break within its scope. Here is an example that keeps line 1 on the same page as line 2.

```
{\samepage
  line 1
 \nopagebreak

  line 2
}
```

2.7 Spacing

We have already seen the use of `~` to insert one space and `\hfill` to put the remaining text flush right. The most versatile method to insert horizontal spaces is with `\hspace` and `\hspace*`. These have one argument: the amount of space to be inserted. For example,

`I insert .3~in \hspace{.3in} here.` ⇒ I insert .3 in here.

The `\hspace` command has no effect at a line boundary, but the `\hspace*`
inserts the space no matter what. For example, the previous sentence is written as:

```
The \verb|\hspace| command has no effect at a line boundary, but
the \verb|\hspace*| \hspace*{1in} inserts the space no matter what.
```

That is why you see the 1 inch space at the beginning of the second line. `\hspace` would not insert the 1 inch, but `\hspace*` does.

Two variations of `\hfill` are:

`\dotfill` ....
`\hrulefill` ____

Analogously, vertical spacing is controlled by `\vspace`, `\vspace*` and `\vfill`. The height of one line of normal text is in the keyword `\baselineskip`, so `vspace{\baselineskip}` skips one line at the next new line. The vertical space is not added if this goes to the top of a new page; that is what `\vspace*` does. In particular, at the very beginning of your document, if you want to make your own title page, you use `\vspace*{2in}` to put a 2 inch margin at the top (`\vspace` would not insert the space).

The easiest way to control line spacing throughout your document is to specify `usepackage{setspace}` in your preamble. This gives you three commands:

`\singlespacing` `\onehalfspacing` `\doublespacing`

Right after you specify one of these, that spacing will commence.

Exercises. Submit a printed copy of both the LaTeX source (tex file) and the associated postscript result (ps file). Be sure your name is on each.

1. Write a paragraph in article [and letter] style with the following properties:

   (a) Each font style in Table 1 is used as one letter in a word that has more than one letter.

   (b) Each font style in Table 1 is used for one complete word.

   (c) Each font style in Table 1 is used for two consecutive complete words.

2. Write two paragraphs in article [and letter] style with each of the following properties:

   (a) Default indentation on both paragraphs.

   (b) No paragraph is indented.

   (c) Both paragraphs are indented.

   (d) There is added space between paragraphs.

3. Write a paragraph in article style and make a cover page with the following properties (like the cover page of this document):

   - All lines are centered.
   - The title appears first in very large letters.
   - Your name appears second in letters that are not as large as the title, but larger than normal size, and it is preceded by extra space.
   - Your e-mail address appears third.
   - Your web address appears fourth.

- Course number and title appears next, with extra space preceding it.
- Date appears last, with extra space preceding it.

4. Give an enumeration of at least three things you like about mathematics. Give the same list without numbers.

5. Produce the following table:

Colors	
Primary	Secondary
Red	Green
Blue	Orange
Yellow	Purple

6. Produce the following table (including the accents and alignments).

Mathematician	Birth	Death
Gabrielle Emilie Le Tonnelier de Breteuil Marquise du Châtelet	1706	1749
Benjamin Banneker	1731	1806
Sophie Germain	1776	1831
Julius König	1849	1913
Rózsa Péter	1905	1977

7. Produce the following table.

<table>
<tr><th colspan="2">Payoffs ($)</th></tr>
<tr><th>Player A</th><th>Player B</th></tr>
<tr>
<td><table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr></table></td>
<td><table><tr><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td></tr><tr><td>5</td><td>6</td></tr></table></td>
</tr>
</table>

8. How can you have an entire table whose columns are of fixed width?

9. Create a 3-column text such that each column is a paragraph of arbitrary length using about $\frac{1}{3}$ of the page width each.

10. Use the tabbing environment to produce the following:

apples	integral	derivative
grapefruit	sum	difference
	variables	constants

11. Use the tabbing environment to produce what you see on page 33.

12. Produce the following:

rate of mass accumulation in the compartment	=	net rate of mass entering the compartment by convection	+	net rate of mass entering by diffusion.

3 Bibliography with BibTeX

3.1 Overview

It might seem strange to have this section so early, instead of with §7. That is because I require students to produce an annotated bibliography early in the semester, and I want them done with BibTeX. So here we are.

BibTeX [11] was developed by Oren Ptashnik and is available free of charge. It reads a plain text file, called a *bib file* (plus one of the files created by the latex compiler, about which you need not be concerned). The bib file contains the *bibliographic database*, which could extend beyond one document. The bibtex program that you apply to your source creates another file (which you need not examine), from which a second latex compilation causes the bibliography to be created. The execution looks like this (same under unix and DOS):

```
latex myfile
bibtex myfile
latex myfile
```

You might have to compile with latex more times, until you do not have any "unresolved" bibliography citations. Once this is successful, you do not have to `bibtex myfile` again until you change your bib file or add a citation. This added loop is illustrated in Figure 31.

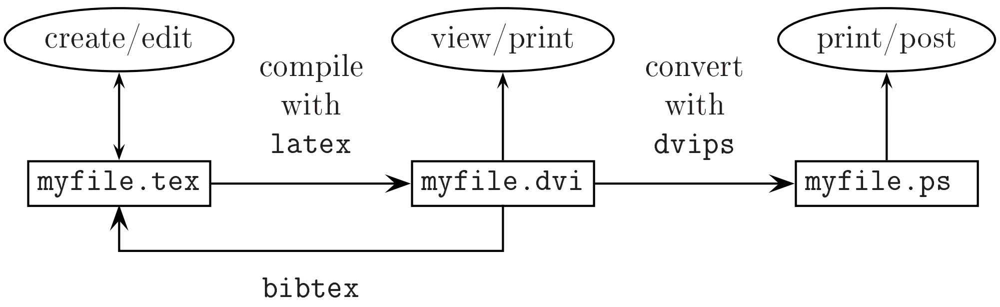


Figure 31: Adding `bibtex` to the Command Sequence

3.2 The bib File

3.2.1 Main body

For purpose of this introduction, we suppose your bibliography is in a file called `mybiblio.bib`, but that name is arbitrary as long as it ends with `.bib`. We begin with the most important part of your bib file, which are the entries you want to include in its database. Each entry has the following form:

```
@type {label,
  field = "value",
     ⋮
}
```

For example, Knuth's book [8] would be entered as follows:

```
@article{tex,
 author  = "Donald E. Knuth",
 title   = "The {\TeX} Book",
 publisher = "Addison-Wesley Publishing Company",
 address = "Reading, MA",
 year    = "1989",
 edition = "15th",
}
```

Most authors devleop a style to labelling bibliographic entries. The use of one keyword is somewhat simplistic and could cause problems with a great number of entries because the labels must be unique. We cannot, for example, have two entries with `tex` as their label. Here are two styles I have seen, which you might consider:

Form	Example
author . year	`knuth.89`
author : first_keyword_in_title	`knuth:tex`

With two authors, you can put both of their names; with more than 2, you can add et al. (Linguistically, the use of the Latin et al. in formal writing follows this rule, though some require more than three authors to use it.) In the first form, if there are two publications by the same authors in the same year, some people add `a`, `b`, ... after the year (no blank). In the second form, if there are two publications by the same authors in the same year, some people add another keyword. You must discover what style works best for you.

Before listing each *style* (`article` is one style) and the *fields* they can or must have (`author` is one field), here are a few things to note.

- The label is arbitrary, but do not use any LaTeX special characters or blanks. In the example, the label is specified as `tex` and it must be followed by a comma. Also, labels are case-sensitive, so `tex` is not the same as `TeX`.
- Each bib entry must have a unique label, so it can be cited without ambiguity in the source file.

- The order of the fields is arbitrary, and fields are separated by commas (hence the comma *after* the terminal quote). The last field does not require a comma at the end, but it will not hurt anything, and it gives flexibility if you want to add a field or change the order.
- Fields do not have to be on separate lines, but it is more readable that way.
- The *field value* can be anything recognized by LaTeX, even mathematical symbols in math mode.
- There is a final `}` to close the entry — `@`*type*`{ ... }`.

Here is a list of standard entry types with their required fields. What are "optional fields" in BibTeX are not necessarily optional as far as having a complete bibliography citation. For example, the volume and page numbers of an article are necessary to include even though they are optional to satisfy BibTeX syntax. (What is "necessary" depends upon the standard one applies, but most journals require the volume of the journal and the page numbers for the cited article.) Fields that are neither required nor optional are ignored, even if they are valid fields in other types of entries.

`article` refers to an article from a journal or magazine.
  Required fields: `author`, `title`, `journal`, `year`.
  Optional fields: `volume`, `number`, `pages`, `month`.
`book` refers to a book with an explicit publisher.
  Required fields: `author` or `editor`, `title`, `publisher`, `year`.
  Optional fields: `volume` or `number`, `series`, `address`, `edition`, `month`.
`booklet` refers to a bound, printed document, but without an explicit publisher.
  Required fields: `author` or `key`, `title`.
  Optional fields: `author`, `howpublished`, `address`, `month`, `year`.
`inbook` is a part of a book, such as a chapter or just some range of pages.
  Required fields: `author` or `editor`, `title`, `chapter` and/or `pages`, `publisher`, `year`.
  Optional fields: `volume` or `number`, `series`, `type`, `address`, `edition`, `month`.
`incollection` is a part of a book having its own title.
  Required fields: `author`, `title`, `booktitle`, `publisher`, `year`.

Optional fields: `editor`, `volume` or `number`, `chapter series`, `type`, `pages`. `address`, `edition`, `month`.

`inproceedings` is an article in a conference proceedings.
Required fields: `author`, `title`, `booktitle`, `year`.
Optional fields: `editor`, `volume` or `number`, `series`, `pages`, `month`, `organization`, `publisher`, `address`.

`manual` is some technical documentation.
Required fields: `author` or `key` (see note below). `title`.
Optional fields: `author`, `organization`, `address`, `edition`, `month`, `year`.

`mastersthesis` is a Master's thesis.
Required fields: `author`, `title`, `school`, `year`.
Optional fields: `type`, `address`, `month`.

`misc` is when nothing else fits.
Required fields: `author` or `key` (see note below).
Optional fields: `author`, `title`, `month`, `howpublished`, `year`.

`phdthesis` is a PhD thesis.
Required fields: `author`, `title`, `school`, `year`.
Optional fields: `type`, `address`, `month`.

`proceedings`
Required fields: `title`, `year`.
Optional fields: `editor`, `volume` or `number`, `series`, `publisher`, `organization`, `address`, `month`.

`techreport` is a report published by some institution.
Required fields: `author`, `title`, `institution`, `year`.
Optional fields: `type`, `number`, `address`, `month`.

`unpublished` is a document with an author and title, but not formally published, even as a technical report. (Some note of explanation is required.)
Required fields: `author`, `title`, `note`.
Optional fields: `month`, `year`.

In addition to the optional fields listed, which vary by the type of entry, the `note` field is always an option. This lets you enter a note that will appear at the end of the citation. To have a comment that is not printed, enter an unrecognized field, such as `comment = "..."`, (this is ignored with no error message given).

If a required field is missing when you compile, you will get an error message. Possibly there will be some standard fixup, but it is best if you provide the missing field value. If a document has no author, you must provide a `key` for sorting. For example, consider the following entry for LaTeX 2$_\varepsilon$ [10], which has no person as an author.

```
@manual{usrguide,
  key = "LaTeX2e",
title = "{\LaTeXe} for authors",
 type = "World Wide Web site",
 institution = "Comprehensive {\TeX} Archive",
 address = "{CTAN\url{/macros/latex/doc/usrguide.ps}
              (see~\cite{CTAN} for replacing CTAN)}",
 year = "1995--99",
}
```

The bibliography will be sorted with the key, `LaTeX2e`, used to order this entry relative to author names. The key will not be printed.

When there are multiple authors, we separate them with `and` (no commas). For example, [5] in this document has the following BibTeX entry:

```
@Book{companion,
  author    = "Michel Goosens and Frank Mittelbach and
               Alexander Samarin",
  title     = "The {\LaTeX} Companion",
  publisher = "Addison-Wesley Publishing Company",
  address   = "Reading, MA",
  year      = "1994",
}
```

The use of the braces in `{\LaTeX}` is to tell the bibtex program to take everything inside just as it is written (for the latex program to process). Otherwise, the bibtex program might try to process it itself and produce an unintended result. This applies to accents too. In ordinary LaTeX, we write `G\"{o}del` to produce Gödel, but this will not work in BibTeX. Instead, we write `G{\"{o}}del` (or simply `G{\"o}del`).

The use of braces to force a particular result is necessary in other instances, such as writing `{F}ourier analysis` to force the capital F; otherwise, the bibtex program will produce 'fourier analysis' (the plain style produces article titles in lower case, except the first letter of the first word). Some authors, however, use this feature inappropriately by putting braces around everything. That defeats one of the primary advantages of using LaTeX and BibTeX in the first place: we want to let the style files determine the final form, so we can switch styles and use the same source (tex and bib files).

3.2.2 Web citations

When BibTeX was developed, the World Wide Web did not exist. Now it is a major source of information. There is no universally accepted standard for how to reference web documents, but here is one way.

If it is a book, use the `book` type and specify:

```
publisher = "World Wide Web",
address   = "url ",
```

Here is an example:

```
@book{Strunk,
   author = "William Strunk{, Jr.}",
   title  = "Elements of Style",
   publisher = "World Wide Web",
   address   = "http://www.columbia.edu/acis/bartleby/strunk/",
   year      = "1999",
   note   = "This is the web version of the classic book by
             Strunk and White~\cite{StrunkWhite}",
}
```

The reference `\cite{StrunkWhite}` presumes there is the entry for the original publication. The use of the braces in the name is to be sure that the author appears as intended: William Strunk, Jr. Otherwise, without the braces, the comma would signal the bibtex program that 'Jr.' is the first name of the author, and it would appear as 'Jr. William Strunk'.

If the document is a technical report, use that style but include the url as a note or in the address field. Eventually, you will run into some difficulty with writing urls. For one thing, the url could contain special chararcters; in particular, the ~ is in many urls, and writing it will produce a space. Also, a url could become very long, and with latex having no place to break, you will see a line with lots of spaces (for justification), followed by the url. (The unsightly line with spaces could appear after the url.) The next example shows how to overcome this.

There are now occasions when we want to reference an entire web site. One example is the LaTeX 2_ε reference [2], given by:

```
@misc{latex2e,
 author = "Johannes L. Braams and David P. Carlisle and
           Alan Jeffrey and Frank Mittelbach and Chris Rowley
           and Rainer Sch{\" o}pf",
 title  = "{\LaTeXe} and the {LaTeX}3 Project",
 howpublished = "World Wide Web,
               \url{http://www.latex-project/org/latex3.html}",
 year   = "1994",
}
```

The `\url` specification is not actually an intrinsic command in LaTeX; it is defined in a *package*. Its main use is to determine where the url can be broken in order to put it on two lines, if needed. Another feature of the url package is that `\url` prints special characters, like `~`. To have the `\url` command active in your document, put the following declaration into your preamble: `\usepackage{url}`. The default font it uses is `tt`, but you can change this to another font with the specification:

```
\renewcommand\url{\begingroup\urlstyle{font}\Url}
```

For example, this book specifies the sans serif font:

```
\usepackage{url}
\renewcommand\url{\begingroup\urlstyle{sf}\Url}
```

We have seen several packages so far, and we shall learn more about packages in §6, where we describe enhancements for having graphics in LaTeX. However, this is the first use of `\renewcommand`, about which I shall say more when I describe ways to customize your document in §8.

3.2.3 Additional features

One element of good style is to be consistent in your terms, including abbreviations and names of publishers. One sometimes sees "Kluwer," other times "Kluwer Academic Publishers," and still other times "Kluwer Pub." To help be consistent and to save some work in the long run when we write many different documents and produce more bib files, we can define *strings* with the entry:

```
@string{name = "string"}
```

Then, we can refer to the string anywhere in the value of a field by excluding the quotes. (That is why we needed the quotes before, when we wrote *literals*.)

For example, suppose we define:

```
@string{kluwer = "Kluwer Academic Publishers"}
```

Then, we can enter:

```
publisher = kluwer,
```

to produce the publisher value = "Kluwer Academic Publishers." Besides consistency, an advantage is that if some name changes, we merely change the one string value and recompile.

We can concatenate strings and/or literals with `#`. For example, suppose we write

```
@string( mom = "My Mother" )
@string( dad = "My Father" )

author = mom,
title  = mom # dad,
editor = dad,
```

Then, the three field values are equivalent to:

```
author = "My Mother",
title  = "My MotherMy Father",
editor = "My Father",
```

Note the absence of a space between the string values in the title. To ensure a space, use the space character, ~, as a literal:

```
title  = mom # "~" # dad,
```

The same title as the above is obtained by any of the following:

```
title  = "My Mother " # dad,
title  = mom # " My Father",
```

Another useful feature of BibTeX is the `crossref` field for cross referencing. For example, suppose we have the following entry (kluwer is a string; the other values are literals):

```
@Proceedings{Byrnes:FAA-89,
 editor   =  "J.S. Byrnes and Jennifer L. Byrnes",
 title    = "Recent Advances in {F}ourier Analysis and its
             Applications: Proceedings of the {NATO}
             {A}dvanced {S}tudy {I}nstitute",
 publisher = kluwer,
 year   = 1990,
}
```

Then, we can have the following entry:

```
@InProceedings{Artemiadis:FAA-89-311,
   crossref = "Byrnes:FAA-89",
   author   = "N.K. Art{\'e}miadis",
   title    = "Results on the Absolutely Convergent Series
                of Functions and of Distributions",
   pages    = "311--316",
}
```

If these were the only references, the result would appear as follows:

[1] N.K. Artémiadis. Results on the absolutely convergent series of functions and of distributions. In Byrnes and Byrnes [2], pages 311–316.

[2] J.S. Byrnes and Jennifer L. Byrnes, editors. *Recent Advances in Fourier Analysis and its Applications: Proceedings of the NATO Advanced Study Institute*. Kluwer Academic Press, 1990.

BibTeX also recognizes a preamble in our bib files to enable us to define some LaTeX commands. The general form is

`@Preamble{` *string* `}`

where *string* is any concatenation of literals and strings. Here is an example [11] that is useful for guiding the sorting of references in a special circumstance:

```
@Preamble{ "\newcommand{\noopsort}[1]{}" }
```

The `\newcommand` is something we shall describe more fully in §8. For now, it is used to define a command, `\noopsort`, requiring one argument. Command `\noopsort` ignores the argument that it receives, producing nothing (indicated by `{}`). Here is how this can be used.

Suppose there is a 2-volume work by the same authors, originally published 1971, but a second edition of volume 1 is printed in 1973. The bib entries would have the years in the opposite order than we want because sorting is first by the authors, which are the same, then by year. To force the first volume to sort before the second, we fool the bibtex program with the following specifications:

Volume 1	Volume 2
`year = "{\noopsort{a}}1973",`	`year = "{\noopsort{b}}1971",`

This fools the bibtex program into thinking the years are `a1973` and `b1971`, thus putting volume 1 first. The definition of `\noopsort`, however, does not actually print the letters, so just the years appear.

3.3 Declaration and Citation

At the end of your source file (where you want the bibliography to appear), before `\end{document}`, put the following commands (in either order):

```
\bibliography{mybiblio}
\bibliographystyle{plain}
```

The first declares the bibliography to be in the bib file, `mybiblio.bib`.

The second command defines the format style of the bibliography to be `plain`, which comes with every installation of latex. There are other bibliography format styles, including some provided by publishers. Here is a list of the most basic ones (included in every installation):

`plain` is the most common because it formats entries according to accepted standards. Entries are sorted by the alphabetical order of author names, breaking ties with the year of publication, and they are labeled with numbers.

`abbrv` differs from `plain` by abbreviating names of journals, among other things (to give a more compact bibliography).

`alpha` differs from `plain` by citing by labels, rather than numbers.

`unsrt` differs from `plain` by sorting entries by the order in which they are cited, rather than by the author names.

We shall use only the `plain` style here, but know that many other styles have been written and are available free of charge.

To cite particular references, the LaTeX command is `\cite{`*label* `[, ... ]}`, where *label* is what we put in our bib file entry. For example, [8] is produced by specifying `\cite{tex}`. You can put more than one citation, separated by commas. For example, `\cite{tex,latex}` produces [8, 9] for this document.

You can insert some further citation information as an optional input argument to the `\cite` command. For example, `\cite[p.~46]{latex}` produces [9, p. 46] in this document. (In the option, delimited by `[ ]`, the `~` is used to ensure that there is a space but no line break when giving the page number as "p. 46" in the citation.)

The rule is that only those bib entries that are cited appear in the final document. There are times when we want to be sure a particular bib entry appears, but we do not want to cite it in the text. This is done with the `\nocite` command. In particular, if we want to have every entry in our bib file appear, we specify `\nocite{*}`. If we want only some particular list of entries to appear, we use `\nocite` with their labels, such as `\nocite{tex}` to be sure Knuth's TeX book appears, even if it is not cited explicitly. Figure 32 shows a complete source file for having all entries in `mybiblio.bib` appear, and that is the entire document!

```
\documentclass[12pt]{article}
\begin{document}
   \nocite{*}
\bibliographystyle{plain}
\bibliography{mybiblio}
\end{document}
```

Figure 32: A Document to Print the Bibliographic Database

We can specify more than one bib file, such as:

```
\bibliography{mybiblio,another}
```

The bibtex program will search them sequentially for any citation. If we have the same label in both bib files, the entries must be identical; otherwise, we will get a fatal error message, `Repeated entry-` telling us which label is repeated. If we have the same entry with different labels, they will appear twice if both labels are used (or if we used `\nocite{*}`).

Exercises. Submit a printed copy of the LaTeX source (tex file), the BibTeX data (bib file), and the associated postscript result (ps file). Be sure your name is on each.

1. Produce a document with one paragraph that cites three bibliographic items, one for each of the following types:
   (a) An article in a journal.
   (b) An entire book with at least three authors.
   (c) A chapter in a book.
   (d) A technical report.
2. Produce a document that lists your entire database, which consists of at least one entry for each of six different document types. Further, at least one entry must have more than two authors.
3. Produce a document with one paragraph that cites three bibliographic items, one for each of the following types:
   (a) A technical report on the web.
   (b) A book on the web.
   (c) An entire web site.
4. Produce a document that has only a bibliography composed of the following three entries (in the order shown).

[1] I.M. Rich, editor. *Impossible Dreams*, volume I. MacTaco, second edition, 1999.

[2] I.M. Rich, editor. *Impossible Dreams*, volume II. MacTaco, 1990.

[3] I.M. Smart, U.R. Tu, and V.F. Money. *How to Square a Circle*, chapter 1. Volume II of Rich [2], 1990.

5. Produce an annotated bibliography of the following form (note the indentations on left and right margins):

[1] P.R. Halmos, *Naive Set Theory*, Van Nostrand, Princeton, NJ, 1960.

This is a good book, which I assign to my Ph.D. students. The first 100 pages seem simple. The next 100 reveal lack of understanding the first 100.

[2] G. Polya, *How To Solve It*, Princeton University Press, Princeton, NJ, 1945.

This is a seminal book that articulates the problem-solving — i.e., theorem-proving — process. There are many editions, and there are modern descendants, such as ...

4 Counters, Labels, and References

4.1 Basic Concepts

A *counter* is a numerical value that refers to something that is being numbered, such as pages, sections, figures, and equations. A *label* is the identification of a particular value, and a *reference* is a citation to a label. The LaTeX syntax for labeling a counter is `\label{`*label*`}`, placed where the counter's value is set, where *label* is unique in the document. The LaTeX syntax for referencing a label is `\ref{`*label*`}`. For example, in this book I defined:

```
\section{Bibliography with \Bibtex} \label{sec:Bibliography}
```

Now I can refer to §3 by `\S\ref{sec:Bibliography}`. The choice of label is arbitrary, except do not use LaTeX special characters or blanks, just as the labels in the bib file entries.

There are times when you just want to produce the counter value, without a label. This is done by `\the`*counter*. For example, `\thepage` produces the page number. On the other hand, if you want to use the counter's numerical value as an argument in a command, specify `\value{`*counter*`}`.

In the next section we consider intrinsic counters and illustrate how to label and reference them. Then, we introduce the figure and table environments, which have intrinsic counters associated with them.

4.2 Intrinsic Counters

Anything to which LaTeX assigns a number has a counter associated with it. Here we illustrate some of those that are in all document styles. Counters that depend upon the style, like a chapter in a book, can be labelled and referenced in the same manner.

You are looking at page 44, which I was able to print by writing `\thepage`. Similarly, you are reading subsection 4.2 of section 4, whose numbers I could write by `\thesubsection\` and `\thesection`, respectively.

To illustrate how I can reference other parts of this document, the following labels were defined (when the subsection and subsubsection were first written):

```
\subsection{The bib File}  \label{subsec:bibfile}
\subsubsection{Web citations} \label{subsubsec:webcite}
```

Then, I can refer to these as follows:

`\S\ref{subsec:bibfile}`	⇒ §3.2
`\S\ref{subsubsec:webcite}`	⇒ §3.2.2

I can also refer to their page numbers:

`p.~\pageref{subsubsec:webcite}`	⇒ p. 36
`p.~\pageref{subsec:bibfile}`	⇒ p. 31

For any counter, `\pageref{`*counter*`}`, gives the page number where its label is defined, just as `\ref{`*counter*`}` gives its value. (Recall from p. 26 that `~` is used to have a space without a linebreak, which is an element of good style.)

Equation (6), page 65, was labelled `\label{eqn:hessian}`, so in this sentence I wrote its number by `\ref{eqn:hessian}` (with parenthesis added) and its page number by `\pageref{eqn:hessian}`. In the exercise to list what you like about mathematics, I entered the label `\label{exer:likeaboutmath}` (page 29), which I can now reference as exercise #4 by writing `\#\ref{exer:likeaboutmath}`.

The choice of label, such as `subsubsec:webcite`, is any string you want to use that does not contain embedded blanks or special characters used by LaTeX, such as `\`. In my choice of label, I used the structure *prefix*:*name*. That is a matter of style, and I used the prefix `subsec` here. This helps me to distinguish labels for different things. For equations, I use the prefix `eqn`. Some people use this same form but with different prefixes, such as `ss` for subsection and `e` for equation. You can choose any labeling convention that is meaningful to you. (If you have a lot of labels and need to keep track of them by printing each label and citation in your drafts, see the showkeys package at CTAN [4].)

4.3 Figures and Tables

In this section we consider figure and table environments, which have the same syntax:

```
\begin{figure}[options]                  \begin{table}[options]
[\caption{caption[\label{label}]}]       [\caption{caption[\label{label}]}]
  ⋮                                         ⋮
[\caption{caption[\label{label}]}]       [\caption{caption[\label{label}]}]
\end{figure}                             \end{table}
```

The caption, if present, can go at the top or bottom; where you put it is where it will appear. The label to reference a figure or table is put inside the caption. (If you put it outside the caption, as given in [9], it will not be understood, even though you will get no error message.)

Because figures and tables are not split, their exact location depends upon how much room there is. For that reason they are called "floating objects," or "floats." The *options* define where the float is to be located. The four choices are shown in Table 6. In this document most tables and figures are specified with `[ht]`, which means they are to be placed "here," the place where it is specified in the source, if possible. If there is not enough room, it is to be located at the top of the following page.

Option	Meaning
`h`	Locate here (where the environment is declared).
`t`	Locate at the top of the next page.
`b`	Locate at the bottom of the page (or the next page, if this page does not have enough room).
`p`	Locate on a separate page, called a *float page*, which has no text, only figures and tables.

Table 6: Figure and Table Location Options

The placement of a float is sometimes a source of frustration. We might specify `[ht]` and find the float in an unexpected place, perhaps on a page by itself. One cause could be an accumulation of floats that should be cleared at some point before continuing. This is done with the `\clearpage` command. This does the same as `\newpage`, except that it also prints all remaining floating objects. It is also advisable to specify `\usepackage{float}` in the preamble. One of the enhancements is the placement option: `[H]`, which insists that the float be placed here (note the capital H and no other option specified). This option is used in many places in this book, which is why you sometimes see pages with some blank space in the lower portion, followed by a figure or table. I did this to avoid confusion by having some float appear pages after it is cited and discussed.

The table environment is not to be confused with the tabular environment. The latter makes tables, but the table environment does not have to contain a table; it differs from a figure only in its name, and they have separate counters. If you see the figures and tables in this document, you note that they appear as

Figure *number* : *caption* vs. Table *number* : *caption*

That's it.

As a matter of style, we generally use the figure environment to present what we usually think of as figures, notably pictures, and we generally use the table environment to present information in tabular form. However, neither of these conditions is necessary for their LaTeX environments.

Figures can be framed, using the `\fbox` command. For example, what you see in Figure 33 is obtained by the following code:

```
\begin{figure}[ht]
 \setlength{\fboxrule}{3pt}  % make border lines thick
 \setlength{\fboxsep}{.2in}  % increase distance to border of box
 \begin{center} \fbox{ This is a framed figure. }
 \end{center}
 \caption{A Framed Figure with Caption at Bottom \label{youcanlabelthis}}
\end{figure}
```

This is a framed figure.

Figure 33: A Framed Figure with Caption at Bottom

The parameter settings have returned to their default values, upon leaving the figure environment. (This is called a *local setting.*) Thus, the frame in Figure 34 has thin lines and no extra padding around the border. Note how the caption is put at the top from the following specifications:

```
\begin{figure}[ht]
 \caption{A Framed Figure with Caption at Top \label{fig:fboxtop}}
 \begin{center} \fbox{This is framed with default parameter values.}
 \end{center}
\end{figure}
```

Figure 34: A Framed Figure with Caption at Top

This is framed with default parameter values.

4.4 Defining Your Own

In the preamble you can define your own counter with the `\newcounter` command:

`\newcounter{`*name*`}[`*within*`]`

where *name* is the (unique) name of the counter (cannot be the same as one of the intrinsic counter names). The initial value of the counter is 0. For example, `\newcounter{mycounter}` defines a counter whose name is `mycounter`. You can also define the counter to be *within* another counter. For example,

```
\newcounter{mycounter}[section]
```

defines `mycounter` to be within the section counter. This will cause the value of `mycounter` to be reset to 0 when entering a new section.

The counter values are printed in Arabic numerals, but you can specify the type of numeral, shown in Table 7.

What you see	What you write
a, b, c, d, ...	`\alph{mycounter}`
A, B, C, D, ...	`\Alph{mycounter}`
1, 2, 3, 4, ...	`\arabic{mycounter}`
i, ii, iii, iv, ...	`\roman{mycounter}`
I, II, III, IV, ...	`\Roman{mycounter}`

Table 7: Numerals to Print Counters

Counter values can be incremented with the `\addtocounter` command. For example, `\addtocounter{mycounter}{1}` adds 1 to the value of `mycounter`. If we just want to increment the counter by 1, we can specify `\stepcounter{mycounter}`. Counter values can be set to some absolute value with the `\setcounter` command. For example, `\setcounter{mycounter}{5}` sets the value of `mycounter` to 5. This can also be used to transfer the value of one counter to another. For example,

```
\setcounter{mycounter}{\value{page}}
```

sets the value of `mycounter` to the current page number (value of the intrinsic counter, `page`).

When using a counter for some non-intrinsic sequence, we want to be able to label it for future reference. This is done with the `\refstepcounter` command, which also increments its value. For example, to increment `mycounter` by 1 and establish a label to its value at the place this is done, write

```
\refstepcounter{mycounter} \label{mylabel}
```

Then, we can use `\ref{mylabel}` and `\pageref{mylabel}` wherever we like.

The default numeral type is `arabic`, but you can change the appearance to be any of those listed in Table 7 by applying the `\renewcommand` to `\the`*counter*. For example,

```
\setcounter{mycounter}{0}
\renewcommand{\themycounter}{\roman{mycounter}}
\stepcounter{mycounter} (\themycounter),
\stepcounter{mycounter} (\themycounter), \dots
```

⇒ (i), (ii), . . .

This can be used for intrinsic counters too. For example, consider the enumerate list environment, where the types of numerals for the four levels are: `arabic`, `alph`, `roman` and `Alph`. We can change these to be any type we want, such as illustrated in Figures 35 and 36.

```
\renewcommand{\theenumi}{\Roman{enumi}}
\renewcommand{\theenumii}{\Alph{enumii}}  % changes numeral type
\renewcommand{\labelenumii}{\theenumii.}  % changes appearance
\begin{enumerate}
  \item Introduction
  \item Terms and Concepts
      \begin{enumerate}
         \item Groups and fields
         \item Picnics and frolic
      \end{enumerate}
\end{enumerate}
```

Figure 35: Source (Result in Figure 36)

The second level, whose counter is `enumii`, had its *label* changed to what is specified in the source: `\renewcommand{\labelenumii}{\theenumii.}` (the "appearance" parameter is `\labelenumii`). These changes remain in effect, so we must change them back if we want to restore the defaults, shown in Table 8.

I. Introduction

II. Terms and Concepts

A. Groups and fields

B. Picnics and frolic

Figure 36: Result (Source in Figure 35)

Counter	What changes	Command
`enumi`	numeral	`\renewcommand{\theenumi}{\arabic{enumi}}`
	label	`\renewcommand{\labelenumi}{(\theenumi)}`
`enumii`	numeral	`\renewcommand{\theenumii}{\alph{enumii}}`
	label	`\renewcommand{\labelenumii}{(\theenumii)}`
`enumiii`	numeral	`\renewcommand{\theenumiii}{\roman{enumiii}}`
	label	`\renewcommand{\labelenumiii}{(\theenumiii)}`
`enumiv`	numeral	`\renewcommand{\theenumiv}{\Alph{enumiv}}`
	label	`\renewcommand{\labelenumiv}{(\theenumiv)}`

Table 8: Default Settings for Enumerate Counters

Exercises. Submit a printed copy of the LaTeX source (tex file) and printed copy of the associated postscript result (ps file). Be sure your name is on each.

1. Write a document with at least two pages and two sections. Put an enumerated list of items near the beginning of your document, and use the `\ref` or `\pageref` command to reference each of the following.
   (a) Reference §2 by a label that you assign to section 2 (make whatever label name you like).
   (b) Somewhere near the end of your document reference the page number of the first section.
   (c) Reference item #2 of your enumerated list.
2. Include two tables and figures in your document, and reference them by label. Also reference the page that they appear.
3. Produce lists using the enumerate environment with the following appearance:

1. ...
   1.1 ...
   1.2 ...
2. ...
   2.1 ...
   2.2 ...

5 Math Mode

One can write mathematical expressions by entering *math mode*, signified by delimiters `$ ... $` or `\[ ... \]`. The `$` delimiter keeps the mathematical expression in the text, like this:

```
A consequence of Einstein's postulates is that $E = mc^2$.
```

$\Rightarrow$ A consequence of Einstein's postulates is that $E = mc^2$.

The other form is *math display mode*, like this:

```
A consequence of Einstein's postulates is that \[E = mc^2.\]
```

$\Rightarrow$ A consequence of Einstein's postulates is that

$$E = mc^2.$$

5.1 Mathematical Symbols

The example also illustrates the use of the *superscript* operator, `^`. Table 9 shows other common operations in math mode. (Each of the tables in this section applies only to math mode.)

		Example	
Operation	Symbol	How it appears	What you write
subscript	`_`	x_3	`x_3`
superscript	`^`	x^3	`x^3`
multiply	`\times`	$a \times b$	`a\times b`
divide	`\divide`	$a \div b$	`a\div b`

Table 9: Some Mathematical Operations

The braces enclose an expression that can be used to define a more complex operand. For example, x_{a+b} is written as `$x_{a+b}$` and x^{a^2} is written as `$x^{a^2}$`. The order of subscripts and superscripts does not matter:

`x_{a+b}^{c+d}` ⇒ x_{a+b}^{c+d}

`x^{c+d}_{a+b}` ⇒ x^{c+d}_{a+b}

Table 10 shows some set notation. Preceding any symbol by `\not` puts the line through the symbol, as in `\not\subset` ⇒ $\not\subset$. Here are some examples:

`A\cap B=\emptyset`	⇒	$A \cap B = \emptyset$
`A\subseteq B`	⇒	$A \subseteq B$
`x\in A\cup B`	⇒	$x \in A \cup B$
`x\not\in A\setminus B`	⇒	$x \not\in A \setminus B$

What it is	How it appears	What you write
empty set	$\emptyset$	`\emptyset`
intersection	$\cap$	`\cap`
union	$\cup$	`\cup`
set minus	$\setminus$	`\setminus`
element in	$\in$	`\in`
subset (proper)	$\subset$	`\subset`
subset or equal	$\subseteq$	`\subseteq`
superset (proper)	$\supset$	`\supset`
superset or equal	$\supseteq$	`\supseteq`

Table 10: Set Notation

You can control the size of the font by using the usual specification before entering math mode. For example, to have $(x \div y) + z$, write `{\Large $(x\div y) + z$}`. This does not apply to font style because math mode has its own, separate from text mode. You can make math fonts boldface by specifying `\boldmath` before entering math mode. For example, to have $\boldsymbol{x^n + y^n = z^n}$, write `{\boldmath $x^n+y^n=z^n$}`. Note that `\boldmath` is surrounded by the braces; otherwise, math fonts would remain bold, even when leaving and re-entering. The following illustrates this, where $B \cup C$ is boldface in the first case, and returns to normal style in the second case.

`\boldmath$A\supset B$  text $B\cup C$` ⇒ $\boldsymbol{A \supset B}$ text $\boldsymbol{B \cup C}$

`{\boldmath$A\supset B$} text $B\cup C$` ⇒ $\boldsymbol{A \supset B}$ text $B \cup C$

Within math mode, we can control the font style of letters with the command, `\math`*font*`{`*expression*`}`, where *font* is one of: `{bf, cal, it, normal, rm, sf, tt}` (analogous to the `\text`*font* command, p. 9). Unlike `\boldmath`, this applies only to letters, digits and accents, but not to special mathematical symbols. For example,

`{\boldmath$\tilde A\times\vec{1}\otimes\overline{2}$}` ⇒ $\boldsymbol{\tilde{A} \times \vec{1} \otimes \overline{2}}$
`$\mathbf{\tilde A\times\vec{1}\otimes\overline{2} }$` ⇒ $\mathbf{\tilde{A}} \times \vec{\mathbf{1}} \otimes \overline{\mathbf{2}}$

Table 11 illustrates the outcome of each font for this expression:
`\math`*font*`{\tilde A\times\vec{1}\otimes\overline{2}}`

Font Style	Command	Example Result
boldface	`\mathbf`	$\mathbf{\tilde{A}} \times \vec{\mathbf{1}} \otimes \overline{\mathbf{2}}$
calligraphic	`\mathcal`	$\tilde{\mathcal{A}} \times \vec{\infty} \otimes \overline{\in}$
italic	`\mathit`	$\mathit{\tilde{A}} \times \vec{\mathit{1}} \otimes \overline{\mathit{2}}$
normal	`\mathnormal`	$\mathnormal{\tilde{A}} \times \vec{\mathnormal{1}} \otimes \overline{\mathnormal{2}}$
roman	`\mathrm`	$\mathrm{\tilde{A}} \times \vec{\mathrm{1}} \otimes \overline{\mathrm{2}}$
sans serif	`\mathsf`	$\mathsf{\tilde{A}} \times \vec{\mathsf{1}} \otimes \overline{\mathsf{2}}$
typewriter	`\mathtt`	$\mathtt{\tilde{A}} \times \vec{\mathtt{1}} \otimes \overline{\mathtt{2}}$

Table 11: The `\math`*font* Commands

The calligraphic style applies only to capital letters, causing unintended results when applied to other symbols, as shown in Table 11. The calligraphic alphabet looks like this (and it is available only in math mode):

$$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}.$$

Write `${\cal P} = A + B$` to produce $\mathcal{P} = A + B$; without the braces, the calligraphic fonts remain in effect: `$\cal P = A + B$` ⇒ $\mathcal{P} = \mathcal{A} + \mathcal{B}$.

Greek letters are defined only in math mode, and they are specified by spelling them as keywords. For example, to produce

$$\alpha - \beta = \Delta - \delta$$

write `\[ \alpha - \beta = \Delta - \delta \]`. As Lamport [9, p. 43] says, "Making Greek letters is as easy as π (or Π)" (written `$\pi$` or `$\Pi$`). (Not every Greek letter is included — see Appendix Table 34.)

5.2 Fractions and Variable Size Functionality

To make fractions, we could write `$(x+y)/4$` to make $(x + y)/4$, but if we want $\frac{x+y}{4}$, we use the `\frac` command: `$\frac{x+y}{4}$`. We can make this appear larger, as $\frac{x+y}{4}$, by preceding the math mode with `\large`.

The general form is `\frac{`*numerator*`}{`*denominator*`}`, where the numerator and denominator can be any mathematical expression. Here is a more complex equation in display mode:

$$A = \frac{x^2 + y_\alpha}{1 + \frac{\eta}{x^2+1}},$$

written as `\[ A = \frac{x^2+y_\alpha}{1+\frac{\eta}{x^2+1}}, \]`. Note how the sizes of the fractions are adjusted automatically.

Some mathematical symbols adjust their size to fit the expression. Table 12 shows some of the most common of these, and we shall examine more examples below. In the case of the integrals, note the use of `\,` between the integrand and dx. This inserts a *thin space* (compare the results by writing the expression with and without the `\,`).

In LaTeX, symbols whose size you would want to adapt to expressions are generally designed to do so. Figures 37 and 38 illustrate this with another example, which uses the `\sqrt` and `\prod` functions:

```
\[ \sqrt{\frac{\prod_{n=1}^N \left( \sum_{i\in I_n} x_i^n\right)}
                {\sqrt[3]{\sum_{i\in I_\infty} x_i}}
        }
\]
```

Figure 37: Variable Sizes Source (Result in Figure 38)

$$\sqrt{\frac{\prod_{n=1}^N \left(\sum_{i\in I_n} x_i^n\right)}{\sqrt[3]{\sum_{i\in I_\infty} x_i}}}$$

Figure 38: Variable Sizes Result (Source in Figure 37)

Operation	How it appears	What you write
sum	$\sum$	`\sum`
	$\sum_{i=1}^n x_i$	`\sum_{i=1}^n x_i`
integral	$\int$	`\int`
	$\int_a^b f(x)\,dx$	`\int_a^b f(x)\,dx`
parentheses	$()$	`\left( \right)`
	$\left(\frac{x}{1+y} \right)$	`\left(\frac{x}{1+y} \right)`
braces	$\{\}$	`\left\{ \right\}`
	$\left\{\sum_i x_i\right\}$	`\left\{\sum_i x_i\right\}`
brackets	$[]$	`\left[ \right]`
	$\left[\int_0^\infty f(x)\,dx\right]$	`\left[\int_0^\infty f(x)\,dx\right]`

Table 12: Variable Size Mathematical Operation Symbols

Notice that even though it is written in math display mode, the indices on the sums and product appear as they would in line. LaTeX compilers make judgments about the layout, but you can force either of the two styles with the `\displaystyle` and `\textstyle` commands. Figures 39 and 40 illustrate this.

$$\sqrt{\frac{\displaystyle\prod_{n=1}^N \left(\sum_{i\in I_n} x_i^n\right)}{\displaystyle\sqrt[3]{\sum_{i\in I_\infty} x_i}}}$$

Figure 40: `\displaystyle` Result (Source in Figure 39)

```
\[ \sqrt{\frac{\displaystyle
          \prod_{n=1}^N \left( \sum_{i\in I_n} x_i^n\right)}
         {\sqrt[3]{\displaystyle\sum_{i\in I_\infty} x_i}}
       }
\]
```

Figure 39: `\displaystyle` Source (Result in Figure 40)

In text mode you can force the display style of placing these subscripts and superscripts on functions, as well as sizing the expression, as though it were in display mode. Figure 41 gives more examples to compare in line text and display mode, using `\textstyle` and `\displaystyle` to override the default form for the mode. The "default" is not always predictable; in particular, math display mode does not always use `displaystyle`.

Appearance	What to write in text mode	What to write in display mode
$\frac{x}{2}$	`\frac{x}{2}`	`\textstyle\frac{x}{2}`
$\displaystyle\frac{x}{2}$	`\displaystyle\frac{x}{2}`	`\frac{x}{2}`
$\max_{x\in X}$	`\max_{x\in X}`	`\textstyle\max_{x\in X}`
$\displaystyle\max_{x\in X}$	`\displaystyle\max_{x\in X}`	`\max_{x\in X}`

Figure 41: Examples to Compare Text and Display Modes

Table 13 shows symbols used in logical expressions. For example, to have

$$(x \in A \Rightarrow x \in B) \Leftrightarrow (A \subseteq B).$$

write `\[ (x\in A\Rightarrow x\in B) \Leftrightarrow (A\subseteq B). \]` To have

$$\forall x \exists y \ni [P(x) \wedge Q(y)].$$

write `\[ \forall x\exists y\ni [P(x)\wedge Q(y)]. \]`

Logical Term	How it appears	What you write
existential quantifier	$\exists$	`\exists`
universal quantifier	$\forall$	`\forall`
negation	$\neg$	`\neg`
disjunction	$\vee$	`\vee`
conjunction	$\wedge$	`\wedge`
implication	$\rightarrow$	`\rightarrow`
	$\Rightarrow$	`\Rightarrow`
equivalence	$\Leftrightarrow$	`\Leftrightarrow`
	$\equiv$	`\equiv`
such that	$\ni$	`\ni`

Table 13: Some Symbols in Logic

The quantifiers in this last example seem a bit crowded, so we might want to add some spacing between terms. In math mode a full space is obtained by specifying `\;` and a half space by `\,`. Here is how each looks:

`\forall x     \exists y`	$\Rightarrow$	$\forall x \exists y$
`\forall x\,   \exists y`	$\Rightarrow$	$\forall x\, \exists y$
`\forall x\;   \exists y`	$\Rightarrow$	$\forall x\; \exists y$

There are other spacing commands, including negative spacing, shown in Table 35.

Table 14 shows some relations for ordered sets (besides those on the keyboard: `< = >`). Here are some examples:

`(-\infty,0) = \{x\ni x \le 0\}`	$\Rightarrow$	$(-\infty,0) = \{x\ni x \le 0\}$
`a_j\prec b_i \equiv b_i \succ a_j`	$\Rightarrow$	$a_j\prec b_i \equiv b_i \succ a_j$
`\forall y\,\{x\ni x\not\prec y\}` `\not\subset {\cal A}`	$\Rightarrow$	$\forall y\,\{x\ni x\not\prec y\} \not\subset {\cal A}$

We have seen how we can embed math mode into text, but we can also do the reverse with the `\mbox` command. Compare each of the following:

$x_i < 0 for all i=1,\dots$	written as `$x_i < 0 for all i=1,\dots$`
$x_i < 0$ for all $i=1,\dots$	written as `$x_i < 0$ for all $i=1,\dots$`
$x_i < 0 \mbox{ for all } i=1,\dots$	written as `$x_i < 0 \mbox{ for all } i=1,\dots$`

Relation	How it appears	What you write
less than or equal	$\le$	`\le`
greater than or equal	$\ge$	`\ge`
not equal	$\ne$	`\ne`
precedes	$\prec$	`\prec`
precedes or equals	$\preceq$	`\preceq`
succeeds	$\succ$	`\succ`
succeeds or equals	$\succeq$	`\succeq`

Table 14: Order Relations

The first line points out that blanks mean nothing in math mode, and all letters are in the math form of italic (not quite the same as the italic in text mode). The use of `\mbox` is particularly convenient in math display mode, which we shall illustrate in the next section.

5.3 Equations and Arrays

The `array` environment is to math mode what tabular environment is to text mode, and more. It has the form:

`\begin{array}{`*column specs*`}`*options*
first row spec `\\`
⋮
last row spec [`\\` *options*]
`\end{array}`

The column specifications and options are the same as in the tabular environment, but the body is in math mode. The following table has text headers and math body, so it can be generated in either of two ways: with the tabular environment, using the math mode designation for each body entry: `$ ... $`, or with the array environment, using `\mbox` for each header entry.

Variable	Current Value	Limit
x	1.234567	1
y	-9.87	-12.2

This can be generated by either of the following two ways:

```
$\begin{array}{ccc}
  \mbox{Variable} & \mbox{Current Value} & \mbox{Limit} \\ \hline
          x       &    1.234567          &    1      \\
          y       &   -9.87              &   -12.2   \\  \hline
 \end{array}
$
```

or

```
\begin{tabular}{ccc}
     Variable & Current Value & Limit \\ \hline
       $x$    &  $ 1.234567$  &  $ 1    $ \\
       $y$    &  $-9.87    $  &  $-12.2 $ \\  \hline
\end{tabular}
```

You can align a series of equations to appear this way:

$$\begin{array}{lcl} x &=& 5.2 \\ y &=& 2.5 \\ z &=& 7.7 \; (= x+y) \end{array}$$

The above was produced by the following use of math display mode (which is always centered):

```
\[ \begin{array}{lcl}
      x &=& 5.2 \\
      y &=& 2.5 \\
      z &=& 7.7 \; (= x+y)
   \end{array}
\]
```

The `\;` specifies a space; otherwise, `7.7 (= x+y)` ⇒ $7.7(= x + y)$.

Another environment is the `eqnarray`. This is like a 3-column array with specifications `{lcl}`, as above, but each row is numbered:

$$x = y \tag{1}$$
$$y = z \tag{2}$$

(Another difference is that the eqnarray environment uses `displaystyle`.) We use the eqnarray environment directly (without entering math display mode), so the above is produced by the following:

```
\begin{eqnarray}
    x &=& y   \label{eqn:xy} \\
    y &=& z   \label{eqn:yz}
\end{eqnarray}
```

The `\label` statements are to illustrate that we can reference these by writing `(\ref{eqn:xy})` to produce (1) and `(\ref{eqn:yz})` to produce (2). (Note that `\ref` gives just the number; parentheses are added.)

The relation need not literally be an equation; anything could be used for the middle column. Further, there are times when we need to use more than one line for an 'equation,' in which case we need to suppress the numbering of all but one of the rows. Figures 42 and 43 give an example. The `\nonumber` command causes no number to be assigned to the first part of the second equation.

```
\begin{eqnarray}
    x &\mbox{is equal to}& y \\
    y & \preceq & \frac{a+b+c+d}{\Psi} + \frac{e+f+g+h}{\Phi}
                          + \nonumber \\
      &         & I+K+J+L
\end{eqnarray}
```

Figure 42: eqnarray Environment Source (Result in Figure 43)

$$\begin{array}{rcl} x & \text{is equal to} & y \qquad (3) \\ y & \preceq & \dfrac{a+b+c+d}{\Psi} + \dfrac{e+f+g+h}{\Phi} + \\ & & I+K+J+L \qquad (4) \end{array}$$

Figure 43: eqnarray Environment Result (Source in Figure 42)

There is also an `eqnarray*` environment, which is the same as eqnarray, but without the equation numbers. There is no apparent advantage to this since the same result can be produced by the ordinary array environment, specialized to this column specification. It does, however, let us change our mind easily as to whether or not to include equation numbers by simply adding or removing the `*` from the environment definition.

For a single, numbered equation, there is the `equation` environment. This poses no particular advantage over specifying `eqnarray` and merely entering one row (except that column separators (`&`) are not used in the equation environment). Analogous to `eqnarray*`, there is the `equation*` environment, which suppresses the equation numbering. To illustrate, Figures 44 and 45 show how to present a matrix equation.

```
\begin{equation*}
  Ax = \left[ \begin{array}{rrr}
                     1.1 &  1.2 &  1.3 \\
                    21.0 & 22.0 & -2.1 \\
              \end{array} \right]
           \left( \begin{array}{cc} x_1 \\ x_2 \\ x_3
                  \end{array} \right).
\end{equation*}
```

Figure 44: Matrix Equation Source (Result in Figure 45)

$$Ax = \left[\begin{array}{rrr} 1.1 & 1.2 & 1.3 \\ 21.0 & 22.0 & -2.1 \\ \end{array} \right] \left(\begin{array}{cc} x_1 \\ x_2 \\ x_3 \end{array} \right).$$

Figure 45: Matrix Equation Result (Source in Figure 44)

Array environments can be nested, as illustrated in Figures 46 and 47. Notice how the vertical line was drawn by the column specification, `{c|c}`, and the horizontal line separating the blocks is obtained by specifying `\hline` before the second row of the outer array.

```
\[ \left[ \begin{array}{c|c}
     \begin{array}{ccc} A_{11} & A_{12} & A_{13} \\
                        A_{21} & A_{22} & A_{23}
     \end{array}
     & 0 \\ \hline
   0 & \begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22}
       \end{array}
   \end{array} \right]
\]
```

Figure 46: Nested Arrays Source (Result in Figure 47)

$$\left[\begin{array}{c|c} \begin{array}{ccc} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{array} & 0 \\ \hline 0 & \begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array} \end{array} \right]$$

Figure 47: Nested Arrays Result (Source in Figure 46)

We can enclose mathematical expressions in a box, sometimes used for emphasis. For example,

```
\fbox{$ \begin{array}{lcl}
  \displaystyle\int_0^\infty xe^{-\tau x}\,dx
          &=& \displaystyle\frac{1}{\tau} \\ \\
          &=& \displaystyle\oint_a^{b+c} \Psi(x)\,dx
        \end{array}
      $}
```

$$\Rightarrow \boxed{\begin{array}{lcl} \displaystyle\int_0^\infty xe^{-\tau x}\,dx &=& \displaystyle\frac{1}{\tau} \\ \\ &=& \displaystyle\oint_a^{b+c} \Psi(x)\,dx \end{array}}$$

We can use `\fbox` within math mode, such as writing `$x = \fbox{y} + z$` to produce $x = \boxed{\mathrm{y}} + z$. Note how the line height does not adjust to the frame, causing an undesirable clash. This could be overcome by putting a vertical space command just after the expression. In particular, putting `\vspace{.2\baselineskip}` after $x = \boxed{\mathrm{y}} + z$ causes extra vertical space equal to 20% of the value of `\baselineskip`, which is the height of one line of normal text. (In the longrun, it is better to use parameters, like `\baselineskip`, rather than absolute measurements for spacing, because the former takes into account the font size, which you might choose to change.)

Now consider the following conditional assignment:

$$f(x) = \left\{ \begin{array}{rll} -1 & \mbox{if} & x < 0; \\ 0 & \mbox{if} & x = 0; \\ 1 & \mbox{if} & x > 0. \end{array}\right.$$

produced by the following LaTeX code:

```
\[ f(x) = \left\{ \begin{array}{rll}
                    -1 & \mbox{if} & x < 0; \\
                     0 & \mbox{if} & x = 0; \\
                     1 & \mbox{if} & x > 0.
                  \end{array}\right.
\]
```

Note the use of `\right.` after the array. This is because `\left` and `\right` must balance — i.e., there must be an equal number of each. It is not necessary that the left symbol be related to the right one — i.e., `\left\{` does not require `\right\}` to balance; any right symbol will do. The period is not printed in this case, used specifically for this purpose of balance.

We have seen the use of `\left` and `\right` for brackets around a matrix. Now the use of the `\left` LaTeX command for conditional assignment raises related uses of the *underbrace* and *overbrace*. Figures 48 and 49 illustrate these, along with `\overline`, `\underline`, `\widehat` and `\widetilde`.

```
\[ \begin{array}{cc}
       \mbox{This sum has} \\ \mbox{an overbrace} \\
      \overbrace{\overline{i\dots j} + \underline{k\cdots l}}
    & \underbrace{\widehat{xy} - \widetilde{ab}}
  \\ & \mbox{This difference}
  \\ & \mbox{has an underbrace}
   \end{array} \]
```

Figure 48: Horizontal Braces Source (Result in Figure 49)

$$\overbrace{\overline{i\ldots j}+\underline{k\cdots l}}^{\substack{\text{This sum has}\\ \text{an overbrace}}} \qquad \underbrace{\widehat{xy}-\widetilde{ab}}_{\substack{\text{This difference}\\ \text{has an underbrace}}}$$

Figure 49: Horizontal Braces Result (Source in Figure 48)

5.4 Special Functions and Alphabets

Math mode recognizes a collection of special functions. Table 15 shows some common ones. These special functions are used to make the source clearer, rather than using `\mbox` to achieve the same result.

Function	How it appears	What you write
limit	$\lim$	`\lim`
lim inf	$\liminf$	`\liminf`
log	$\log$	`\log`
maximum	$\max$	`\max`
tangent	$\tan$	`\tan`

Table 15: Some Common Mathematical Functions

Among the special functions are the complete set of trigonometric functions. For example, we write `$\tan\theta = \frac{\sin\theta}{\cos\theta}$` to produce: $\tan\theta = \frac{\sin\theta}{\cos\theta}$. The Appendix (Table 39, p. 120) has a much longer list of special functions, as well as the arrows used in some of the examples shown in Table 16.

There is also a package of AMS symbols, which you declare in your preamble with `\usepackage{amssymb}`. This gives the following alphabet with the `mathbb` font:

`$\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$`

$\Rightarrow \mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

How it appears in text style	in display mode	What you write
$\lim_{n\rightarrow\infty}x_n$	$\displaystyle\lim_{n\rightarrow\infty}x_n$	`\lim_{n\rightarrow\infty}x_n`
$\liminf_{n\downarrow 0}\log\,x_n$	$\displaystyle\liminf_{n\downarrow 0}\log\,x_n$	`\liminf_{n\downarrow 0}\log\,x_n`
$\max_{x\in X}f(x)$	$\displaystyle\max_{x\in X}f(x)$	`\max_{x\in X}f(x)`
$\frac{\tan(\theta + \pi)}{\ln\,x}$	$\dfrac{\tan(\theta + \pi)}{\ln\,x}$	`\frac{\tan(\theta + \pi)}{\ln\,x}`

Table 16: Examples of Mathematical Functions

For example, the real line is sometimes denoted by $\mathbb{R}$, rather than $\Re$, which is the LaTeX special symbol, `$\Re$`. Table 17 shows how `\mathbb` can be used for specifying other numerical spaces.

What you write†		How it appears	What it means
`\mathbb{R}`	$\Rightarrow$	$\mathbb{R}$	Real values
`\mathbb{C}`	$\Rightarrow$	$\mathbb{C}$	Complex values
`\mathbb{Z}`	$\Rightarrow$	$\mathbb{Z}$	Integer values
`\mathbb{Q}`	$\Rightarrow$	$\mathbb{Q}$	Rational values

†In math mode.

Table 17: Some Notation Using mathbb Fonts from amssymb Package

Another alphabet is `\mathscr`, for which you specify `\usepackage{mathrsfs}` in the preamble. This gives the following alphabet:

`$\mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$`

$\Rightarrow \mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

In particular, $\mathscr{L}$ is often used to denote the Laplace transform or the Lagrangian, and $\mathscr{H}$ is sometimes used to denote the Hamiltonian. (Compare with `\mathcal`, $\mathcal{H}$, which is also used by some authors.)

5.5 Derivatives and Integrals

We can express a total derivative, $df(x)/dx$, by writing `$df(x)/dx$`; or, we can use the `\frac` command to produce $\frac{df(x)}{dx}$. The partial derivative symbol, ∂, is written `\partial`, so you can write `\partial f(x)/\partial x` to produce $\partial f(x)/\partial x$, and `{\large$\frac{\partial f(x)}{\partial x}$}` to produce $\dfrac{\partial f(x)}{\partial x}$.

The usual notation for the gradient of a function is the *nabla*, denoted by the symbol ∇, (also called "del"), which is an upside down delta (introduced by Hamilton in 1853). In LaTeX it is produced by `\nabla`, and its mathematical definition is the vector of first partial derivatives:

$$\nabla f(x) = (\partial f(x)/\partial x_1, \ldots, \partial f(x)/\partial x_n). \tag{5}$$

I leave it as an exercise to show the LaTeX code that produced equation (5).

The Hessian is the matrix of second partial derivatives:

$$\begin{array}{lll} \nabla^2 f(x) &=& \left[\displaystyle\frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right]. \end{array}$$

This was produced by the following code:

```
\[ \begin{array}{lll}
   \nabla^2f(x) &=& \left[ \displaystyle\frac{\partial^2f(x)}
                                    {\partial x_i \partial x_j}
                    \right].
   \end{array} \]
```

There seems to be some crowding in this direct specification. Compare with the following and see if you can produce it:

$$\nabla^{\,2} f(x) = \left[\frac{\partial^{\,2} f(x)}{\partial x_i \, \partial x_j} \right]. \tag{6}$$

There are two integral signs: `\int`$\Rightarrow \int$ and `\oint`$\Rightarrow \oint$, which are both variable size symbols. For example, note how the outer integral is large in the following expression:

$$\int_a^b \lim_{\lambda \to \infty} \frac{\oint_{X(v)} x e^{\lambda f(x)} \, dx}{\oint_{X(v)} e^{\lambda f(x)} \, dx} \, \Phi(v) \, dv.$$

This was obtained by the following code:

```
\[ \int_a^b \lim_{\lambda\rightarrow\infty}
               \frac{\oint_{X(v)} xe^{\lambda f(x)}\,dx}
                    {\oint_{X(v)}  e^{\lambda f(x)}\,dx}
             \,\Phi(v)\,dv .
\]
```

(Note the use of the thin space, `\,`.)

Definite multiple integrals are no problem. To have

$$\int_0^\infty \int_0^{x_n} \int_0^{x_{n-1}} \cdots \int_0^{x_2} H(x_1,\dots,x_n)\,dx_1\cdots dx_n$$

write

```
\[ \int_0^\infty \int_0^{x_n} \int_0^{x_{n-1}} \cdots
   \int_0^{x_2} H(x_1,\dots,x_n)\,dx_1\cdots dx_n
\]
```

However, consider the following:

$$\int\int_S (u\nabla v - v\nabla u)\cdot dS = \int\int\int_\tau (u\nabla\cdot\nabla v - v\nabla\cdot\nabla u)\,d\tau.$$

The domains of integration, and the spacing of the integral signs, are better with the following, which is not produced by standard LaTeX 2_ε, but by specifying `\usepackage{amsmath}` in the preamble (see *The LaTeX Companion* [5, p. 223]):

$$\iint\limits_S (u\nabla v - v\nabla u)\cdot dS = \iiint\limits_\tau (u\nabla\cdot\nabla v - v\nabla\cdot\nabla u)\,d\tau.$$

The source uses the amsmath `gather*` environment:

```
\begin{gather*}
  \iint\limits_S (u\nabla v - v\nabla u)\cdot dS =
  \iiint\limits_\tau (u\nabla\cdot\nabla v -
                     v\nabla\cdot \nabla u)\,d\tau
\end{gather*}
```

Note how the domains are centered on the multiple integrals and the spacing of the integral signs.

5.6 Theorems and Definitions

The foundations of mathematics are *axioms* and *rules of inference*. The rules create *theorems*, which are statements whose truths are established relative to the underlying *logic*. This is so fundamental that LaTeX has the facility to define a special environment that includes a keyword, like "Theorem," and a name, which is not only the name of the environment, but is also the name of the associated counter. Consider the following example:

Theorem 5.1 *For $n > 2$, there is no solution to $x^n + y^n = z^n$ for*
$x, y, z \in \mathbb{Z}_{++}$.

Notice how "**Theorem 5.1**" appears, all text is in italic, and we have the counter value: `\thetheorem`=5.1. This was defined in the preamble by:

```
\newtheorem{theorem}{Theorem}[section]
```

Then, the theorem was produced by the following LaTeX code:

```
\begin{theorem}
   For $n > 2$, there is no solution to $x^n + y^n = z^n$ for
   \newline $x,y,z\in \LZ_{++}$.
\end{theorem}
```

Other theorem-like environments can be defined to have the same properties. This requires both a keyword, like `Theorem`, and a unique name for the environment, like `theorem`, also used as a counter. Here is the syntax:

`\newtheorem{`*name*`}{`*keyword*`}`[*within*]

The *name* defines the environment name, and it defines a counter, so it must be different from all other environment and counter names. The *within* option defines the counter to be within some other, which can be intrinsic or some other counter defined by the `\newcounter` command (p. 47) or by some other `\newtheorem`. In this document, I defined the theorem environment to be numbered within the section, so you see **Theorem 5.1**, rather than **Theorem 1**. To further illustrate, here is a corollary environment:

Corollary 5.1.1 *The sum of cubes cannot be a cube.*

It was defined in the preamble as follows:

```
\newtheorem{corollary}{Corollary}[theorem]
```

Note that this is within the theorem counter, which is valid by having been defined by its own `\newtheorem`. Then, the above corollary was written as:

```
\begin{corollary} The sum of cubes cannot be a cube.
\end{corollary}
```

The following creates an axiom environment that is not within any other counter.

```
\newtheorem{axiom}{Axiom}
```

The "Axiom of Choice" can then be stated thusly:

Axiom 1 *From any (infinite) family of sets a new set can be created that contains exactly one element from each set in the family.*

This was created by the following code:

```
\begin{axiom} \label{axm:choice}
   From any (infinite) family of sets a new set can be created
   that contains exactly one element from each set in the family.
\end{axiom}
```

The label allows us to refer to the Axiom of Choice as 'Axiom 1 on page 68' by writing `Axiom~\ref{axm:choice} on page~\pageref{axm:choice}`.

5.7 Refinements

Mathematical delimiters, shown in Table 43 (p. 122) must be varied to enclose some expressions, like arrays and fractions. Whereas `\left` and `\right` commands adjust the size of a mathematical delimiter to fit the enclosed expression, we can also enlarge these delimiters ourselves. One way is to precede math mode with a size command — for example,

`{\large(}$E=mc^2${\large)}` $\Rightarrow$ $\big(E = mc^2\big)$.

There are, however, delimiter size control commands: `\big`, `\Big`, `\bigg`, and `\Bigg`. For example,

`$\big(E=mc^2\big)$` $\Rightarrow$ $\big(E = mc^2\big)$.

The use of text font environments comes close to the corresponding math size, (`large`$\leftrightarrow$`big`, ... , `Huge`$\leftrightarrow$`Bigg`), but they are different, especially the thicknesses. This is more evident with the square and angular brackets:

`$\Big[E=mc^2\Big)$` $\Rightarrow \Big[E = mc^2\Big].$

`{\Large[}$E=mc^2${\Large]}` $\Rightarrow \text{\Large[}E = mc^2\text{\Large]}.$

`$\bigg\langleE=mc^2\bigg\rangle$` $\Rightarrow \bigg\langle E = mc^2 \bigg\rangle.$

`{\LARGE$\langle$}$E=mc^2${\LARGE$\rangle$}` $\Rightarrow \Big\langle E = mc^2 \Big\rangle.$

The remaining refinements use the `amsmath` package. The `gather*` environment was illustrated with multiple integrals (p. 66). More generally, the `gather` and `gather*` environments allow the new line specification, `\\`, in math mode. They behave like the `eqnarray` and `eqnarray*` environments, except the equations are not aligned. For example,

```
\begin{gather}
  (a+b)^2 = a^2 + 2ab + b^2 \\
  {\cal L} \oplus M^\varepsilon - V = H_0 \\
  A(x) = \{y: \phi(y) = \cup_{a\in \cal A} \Psi(x)\}
\end{gather}
```

produces the following:

$$(a+b)^2 = a^2 + 2ab + b^2 \tag{7}$$
$$\mathcal{L} \oplus M^\varepsilon - V = H_0 \tag{8}$$
$$A(x) = \{y : \phi(y) = \cup_{a\in\mathcal{A}} \Psi(x)\} \tag{9}$$

The same result without the equation numbers is obtained by using the `gather*` environment.

When writing a matrix within text, we could produce $\left(\begin{array}{cc} a&b \\ c&d \end{array}\right)$ by specifying `$\left(\begin{array}{cc} a&b \\ c&d \end{array}\right)$`. An alternative is with the amsmath `smallmatrix` environment: $\left(\begin{smallmatrix} a&b \\ c&d \end{smallmatrix}\right)$ is obtained by `$\left(\begin{smallmatrix} a&b \\ c&d \end{smallmatrix}\right)$`. (Note that there are no column specifications.) This is not equivalent to preceding the array specification with a text size environment; in particular, `\scriptsize` produces $\left(\begin{array}{cc} \scriptstyle a & \scriptstyle b \\ \scriptstyle c & \scriptstyle d \end{array}\right)$. While the letters inside the matrix are approximately the smallmatrix size, the spacing and parentheses are not the same.

The amsmath package has a command to put dots across any number of columns in an array. Its syntax is `\hdotsfor[`*spacing*`]{`*n*`}`, where *spacing* determines the spacing between the dots, and n is the number of columns it spans. For example,

```
\left|\begin{array}{ccccc}
       1 & 2 & 3 & 4 & 5 \\
        \hdotsfor{3}     \\
       &  \hdotsfor{3}   \\
       \hdotsfor[2]{5}   \\
       \hdotsfor[.5]{5}  \\
    \end{array}\right|
```

$$\left|\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \cdots & \cdots & \cdots & & \\ & \cdots & \cdots & \cdots & \\ \cdot\ \cdot & \cdot\ \cdot & \cdot\ \cdot & \cdot\ \cdot & \cdot\ \cdot \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{array}\right|$$

The `\stackrel` command lets us put characters over a relation, but the `\overset` and `\underset` amsmath commands enable us to put any characters over or under any character. For example,

`$\overset{a}{X}$` $\Rightarrow$ $\overset{a}{X}$

`$\underset{b}{Y}$` $\Rightarrow$ $\underset{b}{Y}$

`$\overset{a}{\underset{b}{Z}}$` $\Rightarrow$ $\overset{a}{\underset{b}{Z}}$

This can be used to stack subscripts:

```
$\displaystyle{\sum_{ \stackrel{ \mbox{\scriptsize$i\in I$} }
                  {j\in J}
               }     }  A_{ij}
  = \underset{j\in J}
             { \underset{i\in I}{\sum} } A_{ij} $
```

$$\Rightarrow \sum_{\stackrel{i \in I}{j\in J}} A_{ij} = \underset{j\in J}{\underset{i\in I}{\sum}} A_{ij}$$

Nesting the `\underset` command can be unwieldy; an alternative is the `\substack` command:

```
$\displaystyle\sum_{\substack{i\in I\\j\in J\\k\in K}} A_{ij}$
```

$$\Rightarrow \sum_{\substack{i\in I\\j\in J\\k\in K}} A_{ij}$$

There are many more refinements, and more packages to make things nicer. Many of these are described in *The LaTeX Companion* [5, Chapter 8].

5.8 Grammar

When writing mathematical expressions, people make some common errors. The general guide is to treat a mathematical expression linguistically. In English this means that every sentence has a subject and predicate, clauses are separated by commas, and phrases are appropriately punctuated. Here are some of the most common elements of grammar to consider.

1. **Punctuate math display mode.** The expression usually needs a comma or period. For example, note the colon before the display and the comma at its end, which is incorrect to omit.

   A symmetric rearrangement of a matrix has the following form:

   $$R = P^t M P,$$

   where P is a permutation matrix.

2. **Define before use.** As you read articles notice that those that are among the most confusing are when the authors used a term that is not defined until pages later. For example, we might see "The distinguishing property of an abelian group is the commutivity ... " But a *group* had not yet been defined.

3. **Reference object is located after the reference.** For example, a figure appears after its first reference. LaTeX does this automatically, but you might want to take control over locating figures.

4. **An object has only one definition.** For example, if we write $\Phi = au + bv$, we cannot later refer to $\Phi(u, v)$. Sometimes we define the complete object, $\Phi(u, v) = au + bv$, then tell the reader something like, "We shall use Φ^k, instead of $\Phi(u^k, v^k)$, when there is no risk of confusion." The overriding principle is clarity, and it is important that the reader be told of this.

5. **If ... then ... is not correct.** Suppose A and C are expressions. We can write either 'If A, C.' or 'Suppose A. Then, C.' The first form is preferred if A and C are simple expressions. If either A or C are compound, the second form is clearer. The form, 'If A, then C.' seems like it ought to be all right, and the comma is used to clarify where the antecedent (A) ends and the consequent (C) begins. In English, however, this is not correct.

6. **Equivalence needs commas.** The expression, 'A if and only if B.' should be written as 'A if, and only if, B.'

Exercises. Submit a printed copy of the LaTeX source (tex file) and of the associated postscript result (ps file). Be sure your name is on each. (Lookup special symbols in the Appendix.)

1. Produce each of the following in math display mode.

   (a) $x^2 = B^2 - 4AC$ implies $x = \pm\sqrt{B^2 - 4AC}$.

   (b) If $\Delta F_{n+1} = F_n$, it follows that $\Delta^2 F_{n+1} = \Delta F_n$.

   (c) $x^+ = \left\{ \begin{array}{ll} x & \text{if } x \geq 0; \\ 0 & \text{otherwise.} \end{array} \right.$

2. Produce the following in math display mode with the `array` environment and/or with the `eqnarray*` environment.

$$\begin{array}{rcl} \Delta^2 F_n & = & F_{n+2} - 2F_{n+1} + F_n \\ & = & 2F_n - F_{n+1}. \end{array}$$

3. Produce each of the following formulas in line with text (construct your own sentences that contain them, and include proper punctuation).

   (a) $\ln e^x = x$

   (b) $\sin\{\theta + 2\pi\} = \sin\theta$

   (c) $y_n = \sum_{i=i_0}^{n-1} x_i \Rightarrow y_{n+1} - y_n = x_n - x_{i_0}$

   (d) $\displaystyle f(x) = \sum_{n=0}^{\infty} f^{(n)}(0)\, \frac{x^n}{n!}$

   (e) $\displaystyle \frac{\partial}{\partial x} \int_a^{x^2} f(y) dy = 2x f(x^2)$

4. Produce the following equation in math display mode.

$$\left[\begin{array}{rrr} 1.1 & -1.2 & -1.3 \\ -2.1 & 2.2 & 2.3 \end{array} \right] + \left[\begin{array}{ccc} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{array} \right] = \left[\begin{array}{rrr} \alpha & -\beta & \gamma \\ -\delta & \lambda & \theta \end{array} \right].$$

5. Produce the expression in the Preface.

6. Produce equations (5) and (6).

7. Produce each of the following expressions:

   (a) $x = y \bmod n \stackrel{\text{def}}{=} x - y = kn$ for some $k = 0, 1, \ldots$

   (b) $\overbrace{\vec{\boldsymbol{\alpha}}_1 + \vec{\boldsymbol{\beta}}_2} - \underbrace{\overline{\boldsymbol{x}}^2 + \overline{\boldsymbol{y}}^3}$

8. Produce each of the following in line with text (that you compose) and in math display mode.

   (a) $\mathcal{A} \stackrel{?}{=} \{S \in \mathcal{S} : S \notin S\}$

   (b) $\sqrt{|\mathcal{F} \times \mathcal{P}|} \leq \pi$

   (c) $\left(\begin{array}{c} \left|\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array}\right| \\ \mathcal{CB} \\ BC \end{array}\right)$

9. Produce each of the following formulas in math display mode (with punctuation):

   (a) $$q^*(G) \geq \max\left\{\Delta(G), \frac{2m(G_A)}{\sqrt{A-1}}\right\} \text{ if } G \neq \emptyset.$$

   (b) $$(0,\ x^T)T = (0,\ x^T)\left(\begin{array}{cc} A & B \\ 0 & C \end{array}\right) = \left(\begin{array}{c} 0 \\ x^T C \end{array}\right) = (0,\ C^T x)^T,$$

   (c) $$V = \frac{3\sqrt{3}}{2}\int_0^a (-x^{\frac{1}{3}} + a)^2\, dx;$$

10. Combine your knowledge of derivatives, conditional assignment (with array environment), and mathematical symbols to produce the following (called the *truncated gradient*):

$$\nabla^+ f(x)_j = \left\{\begin{array}{ll} \max\{0, \partial f(x)/\partial x_j\} & \text{if } x_j = a_j \\ \\ \partial f(x)/\partial x_j & \text{if } a_j < x_j < b_j \\ \\ \min\{0, \partial f(x)/\partial x_j\} & \text{if } x_j = b_j \end{array}\right.$$

11. Produce the following symbols:

    (a) Extended reals: $\mathbb{R}_\infty$.

    (b) Strictly positive integers: $\mathbb{Z}_{++}$.

    (c) Complex n-vectors: $\mathbb{C}^n$.

    (d) Non-negative rational n-vectors: $\mathbb{Q}_+^n$.

12. What is grammatically wrong with each of the following segments.

    (a) A key is how to add velocities the formula is

    $$\frac{(u+v)}{1+\frac{uv}{c^2}}$$

    where c is the velocity of light.

    (b) A result of these assumptions is the following equation

    $$E = mc^2$$

    Einstein first noticed this equivalence between energy (E) and mass (m).

    (c) Let x be an n-vector and ω a scalar, and define

    $$y = Ax - \omega b,$$

    where A is an $m \times n$ matrix and b is an m-vector. Now suppose $y(\omega)$ is specified and we want to find x.

    (d) Now we consider adding velocities.

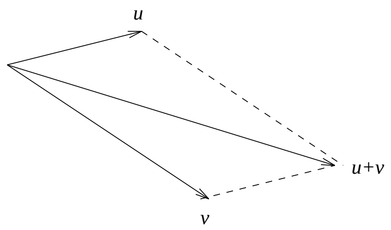
    

    Figure 1. Adding velocity vectors: $u + v$.

    Figure 1 (above) shows how to add velocities simply as vectors.

    (e) **Theorem** *If $x, y, z \in \mathbb{Z}^+$ and $x^n + y^n = z^n$, then $n < 3$.*

The remaining exercises are more difficult. You are to produce the mathematical expressions shown in math display mode.

13. The following is tricky to get the evaluation expression, $t = \frac{1}{2}$ to be the right size and location.

$$\frac{d}{dt} f(x + t\nabla^{+} f(x)) \bigg|_{t=\frac{1}{2}} = -\frac{2v}{(1/2)^{1/4}} + 1.$$

14. Note the row and column labels outside the matrix.

$$A = \begin{array}{c c} & \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

15. Row pointers:

$$A = \begin{bmatrix} 11 & 12 \\ 21 & 22 \end{bmatrix} \begin{array}{l} \quad \leftarrow \text{rows in 1} \\ \leftarrow \text{rows in 2 (this arrow is closer to matrix)} \end{array}$$

16. Column pointers:

$$A = \begin{array}{c} \begin{bmatrix} 11 & \qquad & 12 \\ 21 & & 22 \end{bmatrix} \\ \begin{array}{cc} \uparrow & \uparrow \\ \text{columns} & \text{columns} \\ \text{in 1} & \text{in 2} \end{array} \end{array}$$

17. Row and column pointers:

$$A = \begin{array}{c} \begin{bmatrix} 11 & \qquad & 12 \\ 21 & & 22 \end{bmatrix} \\ \begin{array}{cc} \uparrow & \uparrow \\ \text{columns} & \text{columns} \\ \text{in 1} & \text{in 2} \end{array} \end{array} \begin{array}{l} \leftarrow \text{rows in 1} \\ \leftarrow \text{rows in 2} \\ \\ \\ \\ \end{array}$$

6 Graphics

Graphics may be part of a LaTeX document by one of three ways:

1. Use standard LaTeX 2_ε commands, notably the picture environment;
2. Use a graphics package to draw within the document;
3. Use a package to import some standard graphics file.

We illustrate each, but we do not provide a complete list of the relevant packages (see CTAN [4] and *The LaTeX Companion* [5]). Each package we use must be defined in the preamble as follows:

`\usepackage{`*name*`}`,

where *name* is the name of the package. We have seen `\usepackage{amsmath}` (p. 66), and the use of packages is something we shall consider more in §8.

6.1 Picture Environment

If all we want is a series of boxes and arrows, we can do this simply with `\fbox` and a long arrow in math mode, as follows:

```
\fbox{left}$\longrightarrow$\fbox{center}$\longrightarrow$\fbox{right}
```

$\Rightarrow$ $\boxed{\text{left}}\longrightarrow\boxed{\text{center}}\longrightarrow\boxed{\text{right}}$

The `\framebox` command can be used instead of `\fbox` to produce the same result. However, `\framebox` also has two optional arguments to control the length of the box and the position of the text within it. For example,

```
\framebox[2cm][l]{left}$\longrightarrow$\framebox[2cm][c]{center}$%
$\longrightarrow$\framebox[2cm][r]{right}
```

$\Rightarrow$ $\boxed{\text{left}\qquad\quad}\longrightarrow\boxed{\quad\text{center}\quad}\longrightarrow\boxed{\qquad\quad\text{right}}$

The `%` at the end of the first line is to avoid having a blank between the "center" box and the `$\longrightarrow$` that follows it. The first optional argument of this `\framebox` command is the width of the box, given as 2 cm for each box. The second optional argument is the position of the inscribed text: `l` = left, `c` = center, and `r` = right.

We can make the contents of a box obey all paragraph controls in text mode by the `\parbox` command. By itself, it lets us stack short phrases, like top
middle
bottom (note how the paragraph spacing adjusts). Combined with `\framebox`, we can create vertical diagrams easily, as illustrated in Figures 50 and 51.

```
\parbox{2cm}{
  \framebox[2cm]{top}    \\ \centerline{$\downarrow$} \\
  \framebox[2cm]{middle} \\ \centerline{$\downarrow$} \\
  \framebox[2cm]{bottom}
            }
```

Figure 50: Vertical Diagram Source (Result in Figure 51)

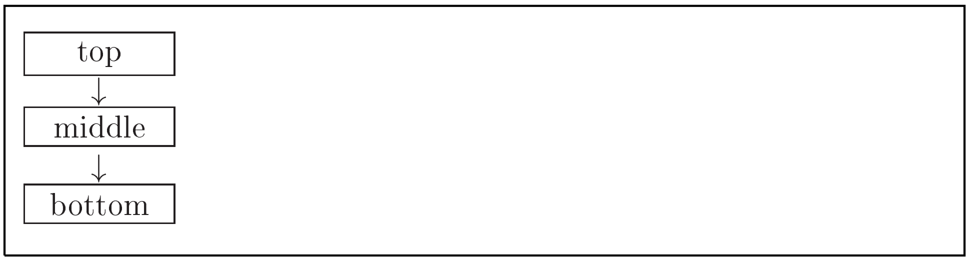


Figure 51: Vertical Diagram Result (Source in Figure 50)

The *box* created by `\parbox` has its center aligned with the text, but it has an optional argument to align its top or bottom with the text. This is done by specifying `\parbox[t]{`*width*`}{`*text*`}` or `\parbox[b]{`*width*`}{`*text*`}`, respectively.

These commands can be combined, along with other box commands (listed in the Appendix), but there is a need for more versatility, like ovals and diagonal arrows, and more control over positioning. We also want to be able to draw curves that approximate given points. A basis for this is the `picture` environment. To begin, Figure 52 shows a more elaborate chart, which was created by the picture environment, whose source is shown in Figure 53. Going through its parts will serve to explain the various commands.

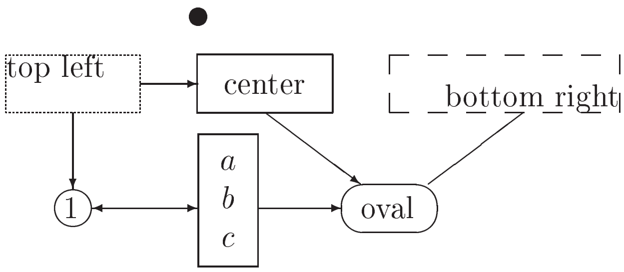


Figure 52: Variety of Objects in Picture Environment

The first command begins a center environment, and we use the `\setlength` command to set the units of measurement to be 1 inch. This means that when we specify some length = 5, we are specifying 5 inches. The parameter that determines this is `\unitlength`, and the default for the picture environment is 1pt. Then, we enter the picture environment stating that the point of entry is the origin, indicated by the *coordinates* (0,0). (There is an alternative way to begin the picture environment, which is not described here.) The filled circle shows where (0,0) is in this picture.

Every picture command begins with `\put`, which is exclusively for the picture environment. The complete syntax is: `\put(`x`,`y`){`*stuff*`}`, where *stuff* can be text or some picture object. The (x, y) coordinates are relative to where the position is when the picture environment is entered. This could be at the left margin, as in beginning a paragraph with `\noindent`; it could be a column in a table defined within the tabular environment; or ☺ it could be in the middle of a sentence, just as the smiley face appears here (see Exercise 1).

The first `\put` in Figure 53 specifies the position at the origin, and the *stuff* is a filled circle with diameter .1 inches (centered at the origin):

`\put(0,0){\circle*{.1}}` $\Rightarrow$●

The next three commands put three different kinds of boxes, each beginning at .5 inches below the origin (i.e., $y = -.5$). The first is similar to `\framebox` in text mode, but its syntax is different. In picture mode it enables control over not only the width, but also the height, and this extends the position options to a second character: `t` = top; `b` = bottom. The general form of the `\framebox` command in the picture environment is as follows:

`\framebox(`*width*`,`*height*`)[`*posn*`]{`*text*`}`

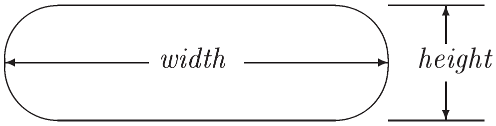
In the example shown in Figure 53, the specifications are *width* = .7 inches and *height* = .3 inches; the position is centered because that is the default.

```
\begin{center} \setlength{\unitlength}{1in}
\begin{picture}(0,0)
  \put( 0,  0){\circle*{.1}}
  \put( 0,-.5){\framebox(.7,.3){center} }
  \put(-1,-.5){\dashbox{.01}(.7,.3)[tl]{top left} }
  \put( 1,-.5){\dashbox{.1}(1.2,.3)[br]{bottom right} }
  \put(-.65, -1){\circle{.2}} \put(-.7,-1.05){1}
  \put( 1, -1){\oval(.5,.25)} \put(.85,-1.05){oval}
  \put(0,-1){\fbox{$\begin{array}{c}a\\b\\c\end{array}$}}
  \put(-.3,-.35){\vector(1,0){.3}}
  \put(-.65,-.5){\vector(0,-1){.4}}
  \put( .35,-.5){\vector(4,-3){.5}}
  \put(-.55,-1){\vector(1,0){.55}}
  \put( 0,-1){\vector(-1,0){.55}}
  \put(.32,-1){\vector(1,0){.43}}
  \put(1.2,-.895){\line(1,1){.3975}}
\end{picture}
\end{center} \vspace{1in}
```

Figure 53: Source for Figure 52

The next `\put` puts a dashed box, having the same dimensions as the framed box, with the length of the dash set to .01 inches. The next dashed box has the dash length set equal to .1 inches, resulting in fewer dashes to compose the box. The box length is set to 1.2 inches, and the text is at the bottom right because of the optional specification, `[br]`.

Now we come to the `\circle` specification, located at coordinates $(-.65, -1)$ (from `\put`), with diameter = .2 inches. The "1" inside the circle required another `\put`, and some *trial and error* was needed to establish its position. We know the center of the circle is at $(-.65, -1)$, but that is not where we want to put the inscribed text to be centered. Unlike the box family, we cannot include the centering of text within the circle command. The same applies to the `\oval` specification, followed by putting text that required some trial and error to locate. The oval, itself, has dimensions $.5 \times .25$ (inches), where .5 measures the entire width:



After the `\oval` specification, we use the `\fbox` command. This is the same as we used in text mode, except here we use it to frame an array, defined as usual in math mode: the array has three rows and one column, which is centered.

Now we begin to draw the vectors, which are lines with arrow heads. Both `\vector` and `\line` have the same syntax:

`\line(`Δx`,`Δy`){`*len*`}`   `\vector(`Δx`,`Δy`){`*len*`}`

If $\Delta x = 0$, the line is vertical, and *len* is the amount of change above or below the original point (it does not matter what the magnitude of Δy is; only its sign matters). If $\Delta y = 0$, the line is horizontal, and *len* is the amount of change to the right or left of the original point (it does not matter what the magnitude of Δx is; only its sign matters). Otherwise, if $\Delta x \neq 0$, the actual change in x is still len, and the slope of the line is $\frac{\Delta y}{\Delta x}$. This is undoubtedly confusing, so consider Figure 54. The new point is determined by moving from (x_0, y_0) along the line with slope $\frac{\Delta y}{\Delta x}$ until the new x-coordinate is $x_0 + len$. Then, the new y-coordinate is $y_0 + len\frac{\Delta y}{\Delta x}$. The actual length of the line segment is $len\sqrt{1 + \left(\frac{\Delta y}{\Delta x}\right)^2}$.

As if this unnatural definition of the line segment were not enough, there is an important restriction: $\Delta x, \Delta y$ must be integer-valued and within -6 to 6. Suppose our original point is (x_0, y_0), and we want our destination point to be (x_t, y_t). If $x_t = x_0$, the calculation is simple: set $\Delta x = 0$, $len = |y_t - y_0|$, and

$$\Delta y = \begin{cases} 1 & \text{if } y_t > y_0; \\ -1 & \text{otherwise.} \end{cases}$$

If $x_t \neq x_0$, we could have problems with approximating the results. Suppose, for example, we want $(x_t, y_t) = (x_0 + 1.3, y_0 + 1.5)$. If we we set $len = 1.3$ to obtain the correct x-coordinate, how should we set the slope parameters? Ideally, we would set $\frac{\Delta x}{\Delta y} = \frac{13}{15}$, but the restrictions do not permit this. The closest we could come is $\frac{4}{5}$.

Fixing $len = x_t - x_0$, then searching for a nearest slope approximation, is not necessarily the best overall approximation. We could setup a least-squares estimation problem, but trial and error in selecting the parameters tends to be just as efficient. Either way, we have some work to do.

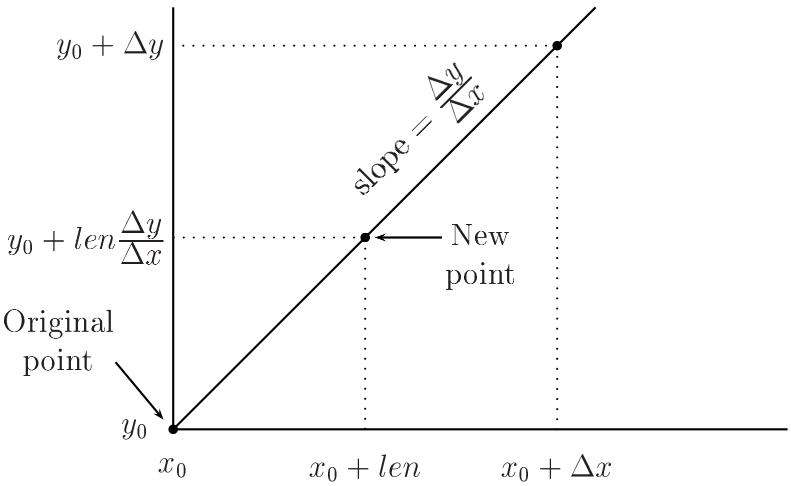
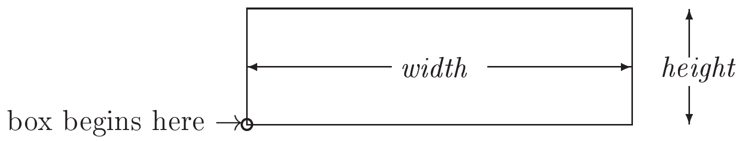


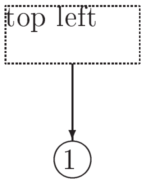
Figure 54: Line Parameters

The first \vector command in Figure 53 starts at $(-.65, -1)$, which I calculated to be from the "top left" box to the "center" box.



In Figure 52 the "top left" box starts at $(-1, .5)$, and its width is .7, so the right edge of that box is at $x = -.3$. Starting at $y = -.35$, the vertical position changes by moving up half of the height, so the coordinate where the arrow begins (called its *tail*) is $(-.3, -.5 + \frac{1}{2}.3) = (-.3, -.35)$. That accounts for the initial position given by `\put(-.3,-.35)`. The arrow is to be horizontal, drawn left to right, so $\Delta x = 1$ and $\Delta y = 0$, as specified with `\vector(1,0)`. Finally, we need to determine the length, specified as `{.3}`. We want the coordinate of the end of the arrow (called the *head*) to be flush to the left side of the "center" box. That box begins at $(0, -.5)$, so $\Delta x = 0 - (-.3) = .3$. It required these computations to determine the complete picture command: `\put(-.3,-.35){\vector(1,0){.3}}`.

Now consider the next \vector, which is a vertical arrow from the same box to the circle below it. The initial position is calculated simply as the midpoint of the bottom edge of the box: $(x, y) = (x_0 + \frac{1}{2}h, y_0)$, where the box starts at (x_0, y_0) and $h = height$. In this case, $(x_0, y_0) = (-1., -.5)$ and $h = .3$, so we obtain the coordinates of the arrow's tail: $(x_t, y_t) = (-1 + \frac{1}{2}.3, -.5) = (-.65, -5)$, as specified. Since the arrow is downward, $\Delta x = 0$ and $\Delta y < 0$, given by `\vector(0,-1)`. The length is determined by where we want the arrowhead: at the top of the circle. The circles y coordinate is $-.65$, which is its center. We must add the radius, which is $\frac{1}{2}(.2)$ since .2 is the diameter specified by `\circle{.2}`. Thus, the position of the arrowhead is $(x_h, y_h) = (x_t, y_c - y_t + r) = (-.65, -1 - (-.5) + .1) = (-.65, -.4)$, and we set $len = |\Delta y| = .4$, which is what is specified:



⇐ `\vector(0,-1){.4}`

The next arrow is double-headed, so we use two \vector commands to draw one arrow left to right, then an arrow at the same end points, but drawn right to left. Further, this involves more calculations because the arrow is not simply horizontal or vertical. We begin the same way, by computing the coordinates of the tail and head. The left end point is at the y-coordinate of the center of the circle, and its x-coordinate is to the right by the length of the radius: $(x_1, y_1) = (x_c + r, y_c) = (-.65 + .1, -1) = (-.55, -1)$, so that is where we \put the first arrow. The head is to be flush with the left edge of the \fbox, and this needs some trial and error. The uncertainty is the width of the box; we know only that the center of the box was put at $(0, -1)$, but we do not know the width of the box. With just a few iterations, the end point was determined to be $x = 0$, so $len = \Delta x = .55$. The reverse arrow begins at $(0, -1)$, and its slope is $(-1, 0)$, which is why we have `\vector(-1,0){.55}`.

The last vector also required trial and error, due to not having the corner of the oval coordinates. In this case, the end points were determined to be from $(1.2, -.895)$ to $(1.6, -.5)$. The former was found by trial and error, but the latter was computed by knowing that the "bottom right" box starts at $(1, -.5)$ and has a width of 1.2, so the midpoint of the bottom edge is at $(1.6, -.5)$. Now the true slope of the line we want is $\frac{.395}{.4}$, but the restrictions do not allow this. The closest slope we can have is with $(\Delta x, \Delta y) = (1, 1)$, which is what is specified. Given this slope, the best choice of len can be found as the average of the deviations:

$$len = \frac{1}{2}(.4 + .395) = .3975.$$

Thus, we specify `\line(1,1){.3975}` to obtain the line shown in Figure 52.

There are packages to extend the picture environment, and we can plot curves, called *Bezier approximations*, to a set of points. However, we shall we shall cover these in the next section with a powerful package called PSTricks.

Table 45 (in the Appendix, p. 122) gives the commands in the picture environment, but here are some things to note:

- Only boxes can have inscribed text; the circles and ovals require separate `\put` commands, which can take some trial and error to position.
- Some calculations and some trial and error are needed to align objects and lines.
- Moving a portion of the picture can be tedious, requiring re-calculations and more effort for the new positions.
- There is no direct way to control the size or style of the arrow heads, and there is very limited control over line thicknesses.

These can make using the picture environment time consuming and rather unpleasant. There is a better way!

6.2 PSTricks

PSTricks [14] was written by Timothy Van Zandt, and is provided free of charge. (It is not standard with MiKTeX, but it does come with unix installations, and you can obtain it at CTAN [4].) In the preamble specify `\usepackage{pst-all}` for the entire system. (You can use parts, in which case you specify the parts you use instead of `pst-all` — see [14] for loading individual portions.)

One thing you need to know is that not all of the pst results can be seen with a dvi viewer. Some require converting to postscript and viewing the ps file. This is especially true of commands that involve scales and rotations.

PSTricks (pst, for short) is designed to overcome difficulties with using the picture environment, some of which were listed above. Here are some of the features of PSTricks that we shall illustrate.

- Circles and ovals, in addition to boxes, can have inscribed text.
- Lines and arrows have the same command, identifying any of a great variety of arrowheads simply.
- Only one command is needed to put lines through a sequence of points, and slopes need not be calculated.
- Objects can be named (as *nodes*) and lines and arrows can be drawn between them by naming the tail and head, thereby eliminating the need for calculation or trial and error.
- Arrow heads are adjustable.
- Shapes are highly variable.
- Drawing curves is simple, including plots of points that can come from a data file, and Bezier approximations of four points are available.

Another widely distributed picture-drawing system is MetaPost [6, 7], written by John D. Hobby, also provided free of charge. It is more difficult to learn than PSTricks, but MetaPost is more open-ended in its design, which makes it potentially more versatile, especially on varying the types of file outputs (PSTricks is tied to postscript). In particular, `pdflatex` (not covered here; see [4]) does not work with PSTricks, but it does with MetaPost.

There are many packages [4], typically available free of charge, that do many of the things done by PSTricks (and some additional things). Many of these are described in *The LaTeX Companion* [5].

All of the pst commands have options to override default settings for relevant parameters. The defaults, themselves, can be set with the `\psset` command: `\psset{`*parameter* = *value*[, …]`}`. For example, the default unit of measurement is 1 cm, and the default fill color is white, but we can change them by specifying:

```
\psset{unit=1in,fillcolor=gray}
```

A fundamental command in pst is `\rput`, but unlike the `\put` command in the picture environment, this is not the only way to put objects. The commands, themselves, can specify where to put them. Table 18 gives some of the common commands to draw objects and lines. For those examples, the unit of measurement was set to 1 mm. For each command, we can specify relevant options as `[`*parameter* = *value*`]`. For example, to produce a solid circle with radius .1 cm, centered at the origin, we write `\pscircle[fillstyle=solid](0,0){.1}` (having already set `fillcolor=gray`).

The origin is determined by where you are when issuing a pst command; no environment is entered. Thus, I can put that circle right here: All commands use the `linewidth` parameter to control the thickness of the lines used in the drawing, and objects that could be made solid, like boxes and circles, use the `fillstyle` parameter. I shall illustrate the commands in Table 18 first, showing the ease and versatility of PSTricks, then I shall show some additional shapes and commands. This is meant to be an introduction, so many features are not presented here. The *User's Guide* [14] is freely available and clearly written.

`psframe`$(x_0, y_0)(x_1, y_1)$ `\psframe(0,1)(10,-2)`	Draws rectangle with a corner at (x_0, y_0) and opposite corner at (x_1, y_1).
`pscircle`$(x, y)\{r\}$ `\pscircle(5,0){2}`	Draws circle centered at (x, y) with radius $= r$.
`psellipse`$(x, y)(r_x, r_y)$ `\psellipse(3,0)(5,2)`	Draws ellipse centered at (x, y) with horizontal radius $= r_x$ and vertical radius $= r_y$

`psline`$\{a\}(x_0, y_0) \ldots (x_n, y_n)$
`\psline{-}(0,0)(10,0)`
`\psline{<->}`
`(0,0)(5,-2)(1,0)`
`\psline{|-*}`
`(0,0)(10,-2)`
	Draws line or arrow, determined by a:
	`-` no arrow; `->` forward arrow;
	`<->` double arrow' `<-` backward arrow;
	(there are more!), along path given by coordinates.

`pspolygon`$(x_0, y_0) \ldots (x_n, y_n)$
`\pspolygon(0,0)`
`(0,-3)(6,-3)`
	Draws closed polygon with given coordinates; same as `\psline{-}` ... , except figure is closed by drawing line from (x_n, y_n) to (x_0, y_0).

Table 18: Some Basic Drawing Commands in PSTricks

In using these commands, we do want the `\rput` command in order to put text into various objects. The idea of a *box* is to have some shape enclose text. PSTricks extends the rectangle in `\framebox` by having a variety of shapes, shown in Table 19. A parameter used by these commands is the distance between the border and the text inside, called `framesep=`*len*, where the default value of *len* is 3 pt. (As usual, other parameters include `linewidth`, `linestyle`, `linecolor`, and `fillcolor`.) The pst figures are drawn after specifying

`psframebox{`*stuff*`}`	Draws rectangle but could have rounded corners
framebox	`\psframebox{framebox}`
framebox	`\psframebox[framearc=.4]{framebox}`
`psshadowbox{`*stuff*`}`	Adds shadow to `psframebox`
shadow added	`\psshadowbox{shadow added}`
`psdblframebox{`*stuff*`}`	Draws double frame
double frame	`\psdblframebox{double frame}`
`pscirclebox{`*stuff*`}`	Draws circle around *stuff*
circle	`\pscirclebox[linewidth=2pt]{circle}`
`psovalbox{`*stuff*`}`	Draws oval around *stuff*
oval	`\psovalbox[linestyle=dotted]{oval}`

Table 19: Boxes in PSTricks

`\psset{unit=1mm,fillcolor=white}`.

These commands can be used in the text. For example, we obtain this oval by writing ... `we obtain \psovalbox{this oval}` ...

Boxes need not be enclosed (like `\makebox`), and they can be scaled by specifying one of the following:

`\scalebox{`*size*`}{`*stuff*`}`	scales *stuff* keeping the same aspect ratio
`\scalebox{`*width*`,`*height*`}{`*stuff*`}`	scales the width and height individually

Here are some examples:

Halving the circle

```
\scalebox{.5}{\pscirclebox{
        \begin{tabular}{c}
    Halving \\ the \\ circle
        \end{tabular} } }
```

Doubling

```
\scalebox{2}{\psframebox{
  \textsl{Doubling} }}
```

Tall

```
\scalebox{1 3}{Tall}
```

Wide

```
\scalebox{3 1}{Wide}
```

There are times when we want to rotate *stuff*. Here is how:

Left Down Right

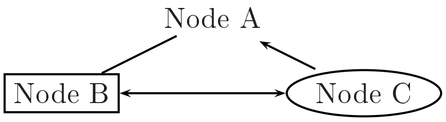
```
\rotateleft{Left}\rotatedown{Down}
\rotateright{Right}
```

One application is given by the following:

Who is the founder of TEX?
Answer: Donald E. Knuth

```
Who is the founder of \TeX?
\rotatedown{Answer: Donald E. Knuth}
```

So far we have described a variety of shapes, by themselves and as enclosures for boxes. These can be connected by `\psline`, with a great variety of styles, including variations of arrowhead shape. To avoid the tedious calculations in locating the coordinates of the tail and head, the objects being joined can be referenced by name. In PSTricks, the named objects are called *nodes*. Consider the following example:



The source code is shown in Figure 55. After entering the centering environment and setting the default units of measurement, the `\rput` command puts a node, with the `\rnode` command. The name is set to `A`, and the text `Node A` is put there (with no frame). The syntax for `\rnode` is:

`\rnode{`*name*`}{`*stuff*`}`

The next two commands put nodes named `B` and `C`, each enclosed with a frame. The `\ncline` command has the same arrow options as `\psline`, but with the following syntax:

`\ncline{`*a*`}{`*name of node A*`}{`*name of node B*`}`

The first `\ncline` in Figure 55 draws a plain line from node A to node B. The `[nodesepA=3pt]` option gives 3 pt separation between the end of the line and node A. Otherwise, the line would touch `Node A` text, which is not what we want. The separation is exaggerated to 5 pt in the arrow from node C to node A. The default value is `nodesep=0pt`, which is what we want when the nodes are enclosed boxes, like B and C. In general, node separation can be specified for either end point, or for both end points, by specifying `nodesepA=`n, `nodesepB=`n, or `nodesep=`n, respectively. (`nodesepA` and `nodesepB` are keywords and have nothing to do with the names we assign to our nodes.)

```
 \begin{center}
   \psset{unit=1cm}
   \rput( 0, 0){\rnode{A}{Node A}}
   \rput(-2,-1){\rnode{B}{\psframebox{Node B}}}
   \rput( 2,-1){\rnode{C}{\psovalbox{Node C}}}
   \ncline[nodesepA=3pt]{A}{B}
   \ncline[nodesepA=5pt]{<-}{A}{C}
   \ncline{<->}{B}{C}
 \end{center}
\vspace*{.5in}
```

Figure 55: PSTricks Source for Connecting Nodes

Now we consider curves that go through, perhaps approximately, given points. The examples that follow use the following pst settings:

`\psset{unit=.5cm,showpoints=true}`

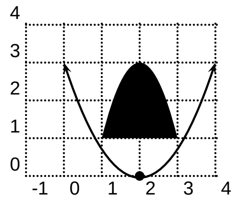
(The `showpoints=true` setting is what causes the points to be included in the picture you see.)

We begin with the parabola, whose command syntax is:

`\parabola{`*a*`]`$(x_0, y_0)(x_1, y_1)$|

where (x_0, y_0) is one point on the parabola, and (x_1, y_1) is the (unique) point having $dy/dx = 0$. `\parabola*` specifies filling the parabola. For example,

```
\psgrid[subgriddiv=1,griddots=10,gridlabels=7pt](-1,0)(4,4)
\parabola{<->}(4,3)(2,0)
\parabola*[fillcolor=black,showpoints=false](1,1)(2,3)
```

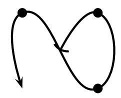
⇒ 

Question: What is the pst command to draw the parabola given by

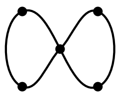
$y = ax^2 + bx + c$, where $a \neq 0$?

Answer: `\parabola`$(0,c)(-\frac{b}{2a}, -\frac{b^2}{4a} + c)$

The following shows two commands: `pscurve` and `psccurve`, the latter being a *closed* curve that joins the last point with the first.



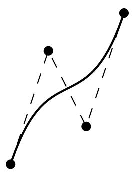
```
\pscurve{(->}(0,0)(1,1)(1,-1)(-1,1)(-1,-1)
```



```
\psccurve(0,0)(1,1)(1,-1)(-1,1)(-1,-1)
```

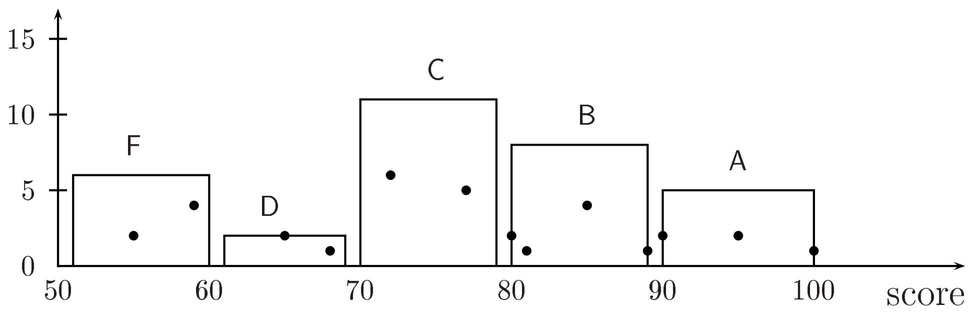
The Bezier curve joins two end points and comes as close as possible to two intermediate points. The command syntax is:

`\psbezier[`*parameters*`]{`a`}`$(x_0, y_0)(x_1, y_1)(x_2, y_2)(x_3, y_3)$



```
\psbezier(0,0)(1,3)(2,1)(3,4)
```

We can read data from a file, perhaps produced by mathematical software like gnuplot©, Octave©, Maple©, Mathematica©, MATLAB©, and S-PLUS©. The data file just needs pairs of coordinates, which can be separated by a comma or just blank and can have parenthesis, braces, or nothing around each pair. The following histogram was plotted by the source code in Figure 56, which we shall explain. The data file had y = number of students with test score $= x + 50$. (The offset of 50 was used in establishing the origin in the plot.)




```
\psset{unit=2mm, showpoints=false}
\fileplot[plotstyle=dots]{mydata.dat}
\psaxes[Ox=50,Oy=0,Dx=10,Dy=5,dx=10,dy=5,ticks=y]{<->}(60,17)
\rput[r](60,-2){\large score}
\psline(1,0)(1,6)(10,6)(10,0)     \rput( 5, 8){\textsf{F}}
\psline(11,0)(11,2)(19,2)(19,0)   \rput(14, 4){\textsf{D}}
\psline(20,0)(20,11)(29,11)(29,0) \rput(25,13){\textsf{C}}
\psline(30,0)(30,8)(39,8)(39,0)   \rput(35,10){\textsf{B}}
\psline(40,0)(40,5)(50,5)(50,0)   \rput(45, 7){\textsf{A}}
```

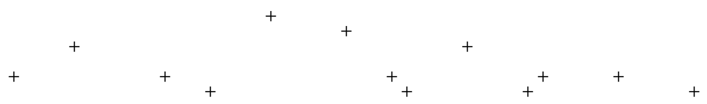
Figure 56: Source Code for Drawing Histogram of Test Scores

After setting the units of measurement to 2 mm, the data file is read and its points plotted with the `\fileplot` command. (Setting `showpoints=false` suppresses plotting the points in the `\psline` commands.) The data file is plain text and has the following entries:

```
% This is mydata.dat
  5 2  9 4              % F = [0,65)
 15 2 18 1              % D = [65,70)
 22 6 27 4              % C = [70,80)
 30 2 31 1 35 4 39 1    % B = [80,90)
 40 2 45 2 50 1         % A = [90,100]
```

The plot, itself, is just the points, specified by `plotstyle=dots`. There are other plot styles, such as `plotstyle=line`, and there are 11 dot styles. Here is one of the alternatives:

`\fileplot[dotstyle=+,plotstyle=dots]{mydata.dat}` $\Rightarrow$

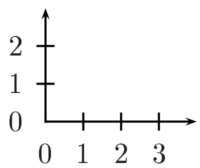


Next, axes are superimposed with the `\psaxes` command:

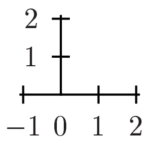
`psaxes[`*params*`]{`a`}`$(x_0,y_0)(x_1,y_1)(x_2,y_2)$

where (x_0,y_0) is the origin, (x_1,y_1) is the Southeast corner, and (x_2,y_2) is the Northwest corner. As in `\psline`, if (x_0,y_0) is absent, the origin is assumed to be at $(0,0)$. If (x_1,y_1) is absent, it is assumed to be equal to the origin. Here are some examples:

`\psaxes[unit=.5cm]{->}(4,0)(8,3)` $\Rightarrow$



`\psaxes[unit=.5cm]{->}(0,0)(-1.1,0)(2.1,2.1)` $\Rightarrow$



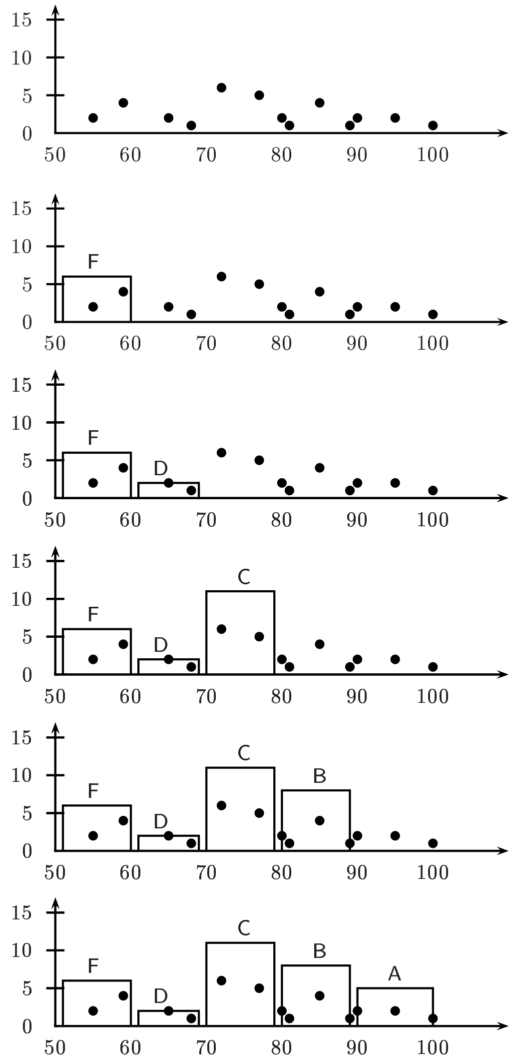
Horizontal	Vertical	Default	Meaning
`Ox=`n	`Oy=`n	0	Label at origin
`Dx=`n	`Dy=`n	1	Label increment
`dx=`n	`dy=`n	0	Label spacing

Table 20: Parameters for `\psaxes`

Note that *ticks* are uniformly spaced on each axes. This is suppressed for the x-axis in Figure 56 by specifying the option, `ticks=y`. The other parameter settings are described in Table 20. (The default values, `dx`=`dy`=0, cause the spacing to be equal (approximately) by using `Dx`÷`\psxunit` and `Dy`÷`\psyunit`, respectively.)

The next command, `rput[r](60,-2){\large score}` puts "score" in large font, flush right (indicated by `[r]`) at the coordinates $(60, -2)$. Thus, when we superimpose the commands `\fileplot`, `\psaxes` and `\rput`, we obtain the data plot. The remaining commands draw the histogram boxes and put the letter grade above each box in sans serif font. Leaving off the "score," Figure 57 shows the sequence of how each `\psline` and `\rput` adds to the picture. To fit the picture and the code next to it, this is scaled (simply, by specifying `\psset{unit=1mm}`):

We shall stop here, but this does not exhaust the PSTricks commands. See [14] for lots more, including many examples.




```
\fileplot[plotstyle=dots]{mydata.dat}
    \psaxes[Ox=50,Oy=0,Dx=10,Dy=5,
     dx=10,dy=5,ticks=y]{<->}(60,17)
```

```
\psline(1,0)(1,6)(10,6)(10,0)
     backslashrput(5,8){\textsf{F}}
```

```
\psline(11,0)(11,2)(19,2)(19,0)
    \rput(14,4){\textsf{D}}
```

```
\psline(20,0)(20,11)(29,11)(29,0)
    \rput(25,13){\textsf{C}}
```

```
\psline(30,0)(30,8)(39,8)(39,0)
    \rput(35,10){\textsf{B}}
```

```
\psline(40,0)(40,5)(50,5)(50,0)
    \rput(45,7){\textsf{A}}
```

Figure 57: Sequence of PSTricks Commands to Draw Histogram

6.3 Importing pictures

The way to import a picture into LaTeX is to convert it to encapsulated postscript (eps). An exceptionally clear description of this, including historical context, is given by Keith Reckdahl [12]. (He also goes deeper into customizing placements of pictures in figures.) Many systems that let us draw figures, and those that plot mathematical functions or data, have an option to export an eps file. (If you can get a ps file, you could use `\psfig`, or there is a conversion utility, `\ps2epsi`.) On unix, `xfig` is an excellent system to draw figures, and the export options include the eps file format. A basic plotting system for functions and data, for both unix and DOS that produces eps files, is `gnuplot`. This is available free of charge at FTP://ftp.dartmouth.edu/pub/gnuplot/. Octave extends the capabilities of gnuplot and is also available free of charge, at http://www.che.wisc.edu/octave/. There are also commercial systems, like Maple©, Mathematica©, MATLAB©, and S-PLUS©, which can produce eps files of plots.

Another way to obtain an eps file is with conversion. The unix systems `xv` and `Image Magick` can do this for a large variety of graphic file formats, including bitmap (xbm), gif and jpeg files. There are commercial and free conversion systems on MS Windows. For example, `wmf2eps` is available at http://www.lake.de/home/lake/p60/wmf2eps/.

Once the file is in eps format, we can import it using the *Graphics Bundle* [3], written by David P. Carlisle, provided free of charge. It comes with MiKTeX and basic unix installations. There are two packages that provide essentially the same capabilities but with different syntax. One is called `graphics`, the other is `graphicx`. Here we shall use `graphicx`; in the preamble we specify `\usepackage{graphicx}`.

To include an eps file, simply specify `\includegraphcs[`*options*`]{`*filename*`}`. For example, Figure 58 shows a figure that was imported with the following statement:

```
\begin{center}\includegraphics[scale=.5]{sin.eps}\end{center}
```

In this case we specified the option, `scale=.5`, which prints the figure half the size it was produced (in this case by MATLAB, by specifying `print sin -deps` after plotting the sin function over the indicated grid). Figure 59 shows the same eps file, but with the width and height set as follows:

```
\begin{center} \includegraphics[width=2in,height=1in]{sin.eps}
\end{center}
```

For a very large picture, we might want to specify `width=\textwidth,height=!`, and let it fill the entire width of the page. The height specification (`!`) says to maintain the aspect ratio.

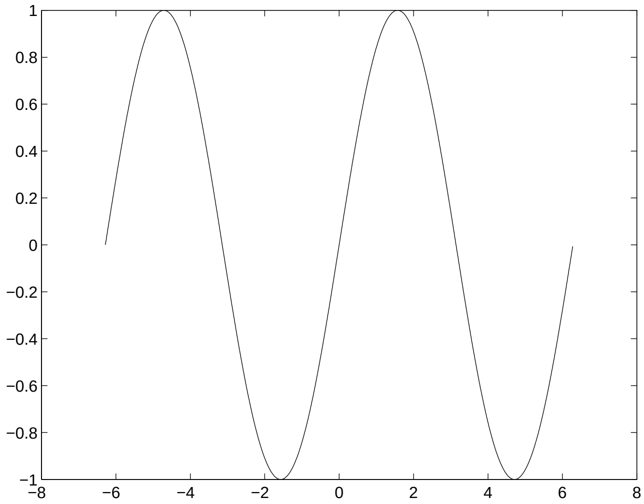


Figure 58: Applying `\includegraphics` to Import an eps File

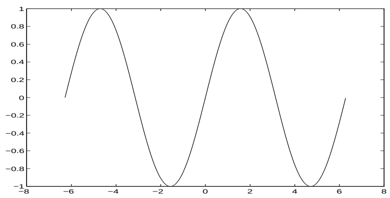


Figure 59: Specifying Dimensions in `\includegraphics`

If you find yourself importing eps files but would like to make some changes in LaTeX, read about the PSfrag system. It is a package, by Michael C. Grant and David Carlisle, that comes with a basic installation (including MiKTeX), whose documentation is at CTAN [4]. It has two basic operations: (1) edit some string or position in the figure (i.e., the eps file), and (2) translate LaTeX commands that you put in the figure in the first place. The documentation gives examples, with eps files produced by MATLAB and xfig.

Importing graphics is only one of the functions of `graphicx`. It can also perform scaling, rotation, and sizing of an arbitrary box. The box could contain text, pictures, or almost any *stuff*. Here are examples:

Double your fun

```
\scalebox{2}{Double your fun}
```

Open wide

```
\resizebox{1in}{!}{\fbox{Open wide}}
```

Reflect on this

```
\reflectbox{Reflect on this}
```

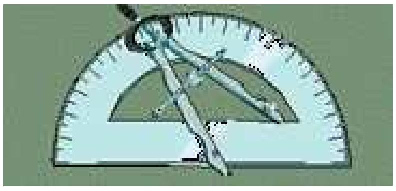
Landscape

```
\rotatebox[origin=c]{90}{Landscape}
```

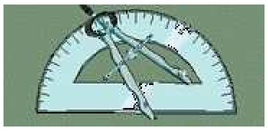
Was
Pythagoras
a square?

```
\rotatebox[origin=rt]{45}
 {\psframebox{
    \begin{tabular}{c}
  Was\\Pythagoras\\a square?
    \end{tabular}
 }           }
```

These operations are available because the programs that perform them are used in the `\includegraphics` command. Although it is feasible to perform the operation after importing a graphic, it is more efficient to specify that option in the `\includegraphics`. Here are some examples:



```
\includegraphics
  {protractor.eps}
```



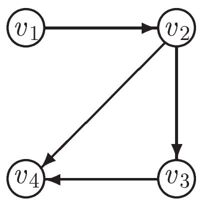
```
\includegraphics[width=.25\textwidth,
  height=!]{protractor.eps}
```

```
\includegraphics[height=.5in,width=!,
  angle=90,origin=c]{protractor.eps}
```

Exercises. Submit a printed copy of the LaTeX source (tex file) and printed copy of the associated postscript result (ps file). Be sure your name is on each.

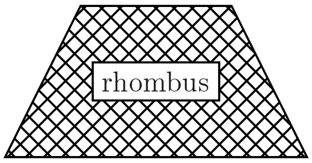
1. Use the picture environment to draw the smiley face on page 78.

2. Draw the following graph with the picture environment, where `\thicklines` is specified and `\unitlength` = 1mm

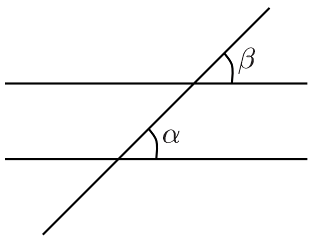


3. Use PSTricks to draw Figure 3.

4. Use PSTricks or the picture environment to draw the following.



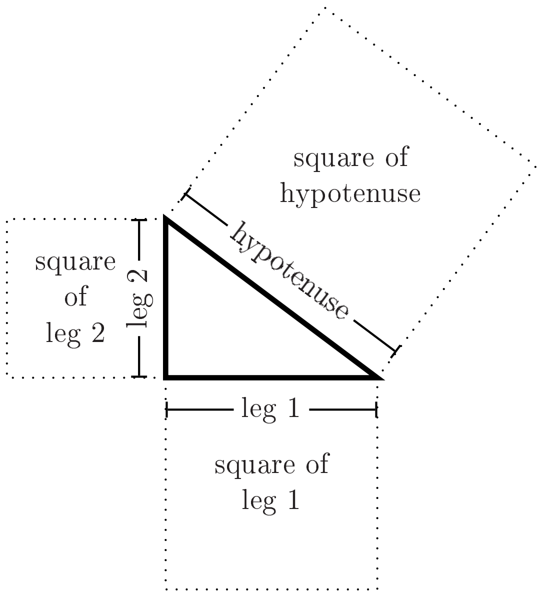
5. Use PSTricks or the picture environment to draw the following.



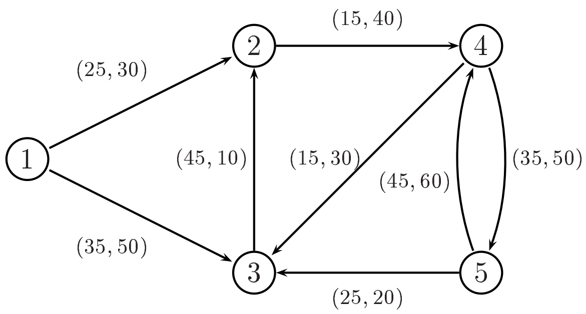
6. Make a figure in some system that lets you save it as an eps file (or use some conversion program). Then, include it in your document.

7. Use whatever means you prefer (or that your instructor requires) to include each of the following figures in your document. (They were drawn here with PSTricks, but this section did not describe all that is needed, so you must obtain the *PSTricks User's Guide* [14].)

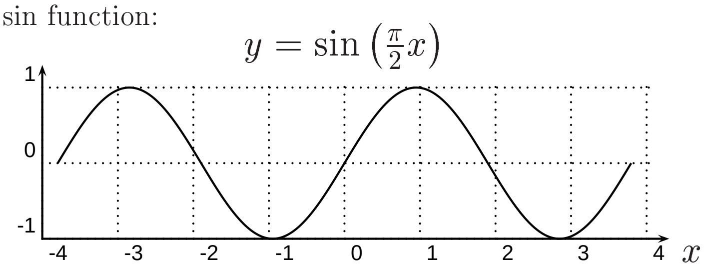
   (a) Graphic view of Pythagorean Theorem:



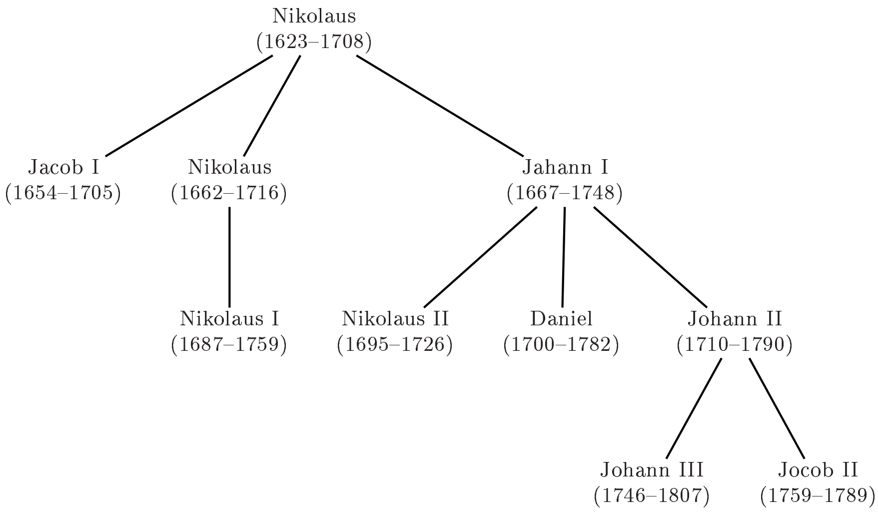
   (b) Network with arc data:



(c) The sin function:



(d) Bernoulli family tree:



7 Making Special Parts

7.1 Cover Page

The easiest way to make a cover page is with the `\maketitle` command. This is done in the document environment, generally just following `\begin{document}`. The necessary parameters are `\author` and `\title`, which can be defined anyplace before the `\maketitle`. Typically these are put into the preamble, or right after `\begin{document}` followed immediately by `\maketitle`; it depends upon your management style. Multiple authors are separated by `\and`, such as in the example shown in Figures 60 and 61. (The jagged edges in Figure 61 mean that there is more space between the title and the top of the paper.)

```
\title{The \LaTeX\ Companion}
\author{Michel Goosens \and Frank Mittelbach \and Alexander Samarin}
\date{1994}
\maketitle
```

Figure 60: Title Page Source (Result in Figure 61)

Specifying `\date` is optional (`\maketitle` puts in the current date if the date is not defined). The cover page is by itself and is not numbered.

The LaTeX Companion

Michel Goosens Frank Mittelbach Alexander Samarin

1994

Figure 61: Title Page Result (Source in Figure 60)

Since articles often have this information on the first page of the article (rather than a separate page), `titlepage` must be specified as an option in the `\documentclass` command. For example, the following does this while specifying 12pt font as another option:

```
\documentclass[12pt,titlepage]{article}
```

Addresses, affiliations, and other information about each author can be added, using `\\` to create new lines. For example, Figure 62 shows how the authors appear when the `\author` definition in Figure 60 is changed to the following:

```
\author{Michel Goosens    \\ Geneva, Switzerland
   \and Frank Mittelbach  \\ Mainz, Germany
   \and Alexander Samarin \\ Geneva Switzerland}
```

As illustrated in Figure 62, `\maketitle` puts the third author on a separate line. This is because the added width of author information would make it too long to fit on one line. All three authors would be put on separate lines if the address information were extended further, or if the names were very long.

The LaTeX Companion

Michel Goosens
Geneva, Switzerland

Frank Mittelbach
Mainz, Germany

Alexander Samarin
Geneva, Switzerland

1994

Figure 62: Adding Addresses to Authors

There are times when we want to acknowledge support for one or more of the authors. The `\thanks` command does this by creating a footnote, using different footnote marks for each one. Figures 63 and 64 illustrate this along with some variation in the date.

```
\title{Pieces of $\pi$\thanks{Renamed.}}
\author{Archimedes\thanks{Supported by the army.}\\ Syracuse, Sicily
   \and Pythagoras \\ Samos, Ionia }
\date{210 {\sc bc} (revision of earlier version, 510 {\sc bc})}
```

Figure 63: Footnotes in the Cover Page Source (Result in Figure 64)

Pieces of π[1]

Archimedes[2]
Syracuse, Sicily

Pythagoras
Samos, Ionia

210 BC (revision of earlier version, 510 BC)

[1]Renamed.
[2]Supported by the army.

Figure 64: Footnotes in the Cover Page Result (Source in Figure 63)

7.2 Abstract

The `abstract` environment is in all document styles, except `article`. To have it, specify `titlepage` as an option in `\documentclass` (even if you do not intend to use `\maketitle`). This environment is defined to produce an abstract on a separate page (placed wherever you put the environment specification), with the header: **Abstract**, in boldface and centered. The abstract, itself, is one paragraph and is printed without indentation. Figures 65 and 66 illustrate this. (Like the cover page, the abstract is placed far from the top of the paper, which is not shown in Figure 66.)

```
\begin{abstract}
 This shows that the ratio of the circumference to the diameter
 of any circle is the same constant value, denoted $\pi$.
 We further prove that this constant is bounded by
 $\frac{223}{71} < \pi < \frac{22}{7}$.
\end{abstract}
```

Figure 65: Making an Abstract Source (Result in Figure 66)

Abstract

This shows that the ratio of the circumference to the diameter of any circle is the same constant value, denoted π. We further prove that this constant is bounded by $\frac{223}{71} < \pi < \frac{22}{7}$.

Figure 66: Making an Abstract Result (Source in Figure 65)

7.3 Other Front Matter

The `\tableofcontents` command makes a table of contents; it is placed wherever you put the command, which should be right after the cover page. Then, you can include lists of figures and tables with the `\listoffigures` and `\listoftables` commands, respectively.

The table of contents generally includes numbered parts, like sections and subsections. To include other front matter, LaTeX provides the `\addcontentsline` command. For example, the table of contents in this document was obtained with the specifications given in Figure 67.

The `\pagenumber` specification causes the page numbers for the front matter to be put into Roman numerals. That is why you see the Table of Contents on page i (first numbered page, just after the cover). Then, I declare `\listoffigures`, which is on page iv, followed by the list of tables. Each of these are put on a new page. Just above each declaration, I use the `\addcontentsline` to add it to the table of contents, indicated by the `toc` specification. The `section` parameter tells the latex program to format it like a section — flush left.

```
\newpage \pagenumbering{roman} \pagestyle{myheadings}
\tableofcontents \newpage
\addcontentsline{toc}{section}{List of Figures}
\listoffigures \newpage
\addcontentsline{toc}{section}{List of Tables}
\listoftables \newpage
```

Figure 67: Some Front Matter Specifications for This Document

The page numbering is reset when we finish the front matter by specifying

```
\newpage \pagenumbering{arabic} \pagestyle{headings}
```

This switches to the Arabic numerals and initializes the page counter.

The same format as the abstract can be used for other front matter that we want to format the same way. The only change we require is another header name. This is done by re-defining the `\abstractname` parameter used by the abstract environment. The `\renewcommand` enables us to do this. §8 has more to say about using this command to customize many things. For now, consider the following example that illustrates how we can have an Acknowledgements page:

```
\renewcommand\abstractname{Acknowledgements}
\begin{abstract}
   I thank my family and friends for all of their support.
   I also thank the contributors to the Comprehensive \TeX\
   Archive Network (CTAN).
\end{abstract}
```

Alternatively, we might want something to look like a section (and automatically added to the table of contents), but we do not want it to have a section number. This is achieved by the `\section*` command, where the `*` suppresses the numbering. For example, `\section*{Preface}` puts "Preface" in the same style as any section, but with no number (and the section counter remains unchanged).

7.4 Back Matter

After the main part of the document is finished, we put the bibliography (see §3 and §8.5). We might first want to have appendices that follow the main text. This could be done with the appendix environment: `\begin{appendix}` ... `\end{appendix}`.

The last portion in the back of any book is its index. This could also be desirable in a long report. To make an index, we have three things to put into our source file:

1. Put `\usepackage{makeidx}` in the preamble.
2. Put `\makeindex` at the end of the preamble.
3. Put `\printindex` just before `\end{document}`.

After a successful compilation, with all references resolved, enter at the command line:

```
makeindex myfile
```

Then, compile again. This is analogous to the use of `bibtex` (p. 31), and is illustrated in Figure 68.

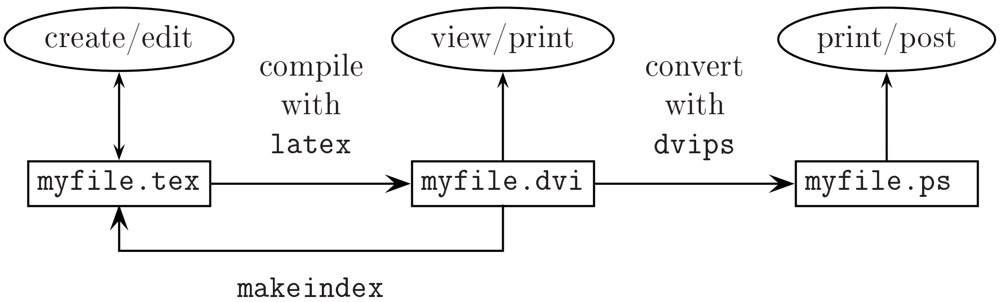


Figure 68: Adding `makeindex` to the Command Sequence

Exercises. Submit a printed copy of both the LaTeX source (tex file) and the associated postscript result (ps file). Be sure your name is on each.

1. Write an article with a title page and abstract. Make the main body have at least three sections: Introduction, Main Results, and Conclusions.
2. Extend exercise 1 to have acknowledgements and references (using BibTeX).
3. Combine exercises 1 and 2 and add a table of contents showing not only all sections and subsections, but also the abstract, acknowledgements and references.

8 Taking Control

This section introduces you to fundamentals of customizing your document. It is still in the context of an introduction, choosing only a few of the things you can change. A key to these changes are the `\newcommand` and `\renewcommand` commands, which enable you to define your own commands and change parameter values of existing commands.

8.1 Your Own Abbreviations and Commands

The command that gives us the ability to make our own has the following form: `\newcommand{`*name*`}[`*n*`]{`*whatever*`}`, where n is the number of arguments, and *whatever* is whatever you want the command to do. Here are two examples simply to abbreviate commands with long names:

```
\newcommand{\ul}{\underline}
\newcommand{\mc}{\multicolumn}
```

The first lets us write `\ul{something}` to underline something. The second lets us write `\mc{3}{c}{stuff}` to enter a multicolumn, in either a tabular or an array environment, spanning 3 columns and centered.

A related use is when the command requires some lines of code. Consider the following example:

```
\newcommand{\Box}{\mbox{\begin{picture}(0,0)
                          \put( 2,0){\framebox(7,7)}
                        \end{picture} }}
```

(`\mbox` is used to ensure text mode). Now `\Box` $\Rightarrow \Box$ and, having defined the `\Box` command, we can use it in other new commands. For example,

```
\newcommand{\chkbox}{\mbox{$\Box\hspace{-.2em}^\surd\;$}}
```

`\chkbox` $\Rightarrow \Box\hspace{-.2em}^\surd$

Some commands are specifically for math mode, but we want them to work in any mode. This is achieved by the command: `\ensuremath{`*math stuff*`}`. For example, consider `\newcommand{\Gs}[1]{G_{#1}}`. If we are already in math mode, `\Gs{i+j}` is replaced by `G_{i+j}` to produce G_{i+j}; otherwise, if `\Gs{i+j}` is specified in text mode, it is replaced by `$G_{i+j}$` to put it into math mode first. Thus, we can specify `\Gs{`*subscript*`}`, no matter which mode we are in, and obtain the correct result.

Another reason to have our own commands is for consistency, particularly of notation. Suppose we have a key term, like the *null space* of a matrix. Some authors write $\mathcal{N}(A)$, some write $\mathsf{nul}\ A$, and there are still more symbols people use. We can choose one and define `\newcommand{\nul}[1]{\ensuremath{{\cal N}(#1)}}`. Then, we can write `\nul{A}` to obtain $\mathcal{N}(A)$ (and we can be in text or math mode when we write this). Some publishers have their own notation, so we must be careful not to override them with ours. A way to do this is to choose a different name, like `mynul`, then add to the preamble:

```
\newcommand{\usenul}{\mynul}
```

and specify `\usenul` in the document. If you need to use the publisher's, simply change the one line to:

```
\newcommand{\usenul}{\nul}
```

(where the publisher's command name is `\nul`).

The preamble can become very long as we add our commands, so it is useful to put them in a separate file, say `mycommands.tex` (note the `.tex` suffix). Then, we use the `\input` command to have the latex compiler read it wherever it is placed. In particular, the preamble can contain the command:

```
\input{mycommands}
```

(the suffix `.tex` is assumed and need not be written).

8.2 Your Own Names, Titles and Numbers

There are times when we prefer some name other than the default. Table 21 shows the common names we might want to change. For example, in this document the "Table of Contents" was obtained by specifying the following in the preamble:

`\renewcommand{\contentsname}{Table of Contents}`.

You might want to change the numbering of some intrinsic counter. We saw an example of this in changing the counters for enumeration lists (p. 48). The general form is

`\renewcommand{\the`*counter*`}{`*something*`}`

What it is	How it is called (keyword)
Abstract	`\abstractname`
Appendix	`\appendixname`
Chapter	`\chaptername`
Contents	`\contentsname`
Index	`\indexname`
List of Figures	`\listfigurename`
List of Tables	`\listtablename`
Part	`\partname`
References	`\refname` for article style
	`\bibname` for book and report styles

Table 21: Intrinsic Name Parameters

Another example is to change section numbering in a report document style. The first level of division is assumed to be a chapter, so the numbering will be *chapter.section*[*.subsection*] ... If you have no chapters, it will number the first section as 0.1. Making the first level division chapters will overcome the numbering problem, but the format of chapters is different, similar to a book. Making the document class an article will also solve the problem, since the section is the first level division. This might not be appropriate due to other considerations, such as entering the document into a database using BibTeX, where you want it to be counted as a report, not as an article. The way to do this is as follows:

```
\makeatletter
\renewcommand\thesection{\@arabic\c@section}
\makeatother
```

The preceding command, `\makeatletter`, is to make the @ character a letter. The succeeding command, `\makeatother`, restores @ to its special meaning (`\@` is for certain spacing, equal to about 2 spaces).

8.3 Your Own Environments

The `\newenvironment` command enables us to define our own environments, and the `\renewenvironment` command enables us to revise an existing environment. They have the same syntax:

`newenvironment{`*name*`}{`[*n*]`{`*begin*`}{`*end*`}`
`renewenvironment{`*name*`}{`[*n*]`{`*begin*`}{`*end*`}`

where *name* is the name of the environment, n = number of arguments (omit `[0]` for $n = 0$), *begin* is what is executed upon entering the environment, and *end* is what is executed upon leaving the environment. For example, suppose we want to create an environment where a quote is put into small, italic font. Then, define

```
\newenvironment{myquote}{\begin{quote}\small\it}{\end{quote}}
```

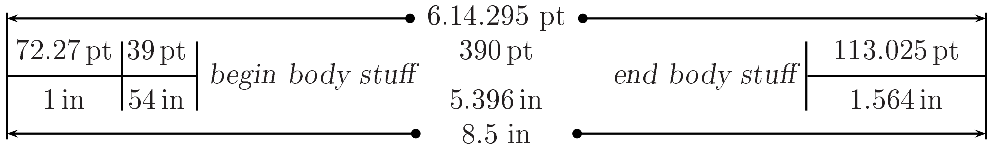
For example, `\begin{myquote} This is my quote. \end{myquote}` produces:

> *This is my quote.*

8.4 Your Own Margins and Spacing

The default margins and spacing are set with purposeful values, and you will usually not need to change them. When you do, however, they can be changed by setting certain parameters in the preamble. The margins of the document are controlled by the parameters shown in Figure 69 and described in Table 22. (See Table 24 for conversion factors; in particular, 1 pt = 72.27 in.)

For example, if we are using $8\frac{1}{2} \times 11$ paper, the current settings (shown in Table 22) break down the horizontal parts as follows:



Parameter	Current Setting[†]	Meaning
`\footskip`	30.0pt	space between bottom of body and top of footer
`\headsep`	25.0pt	space between bottom of header and top of body
`\headheight`	12.0pt	height of header
`\hoffset`	0.0pt	horizontal offset to add to indentation of body
`\oddsidemargin`	17.0pt	extra space added at left (applies only to odd numbered pages if the style is two-sided, in which case there is also an `\evensidemargin` parameter)
`\paperheight`	794.96999pt	height of the paper
`\paperwidth`	614.295pt	width of the paper
`\textheight`	548.5pt	height of the body
`\textwidth`	390.0pt	width of the body
`\topmargin`	17.0pt	space added before the top of the header
`\voffset`	0.0pt	vertical offset to add to indentation of body

[†]Printed using `\the`*parameter*.

Table 22: Margin Parameters

We can increase the text width by setting `\textwidth=`*number* in the preamble. For example, `\textwidth=6in` increases the text width to 6 inches. Note that the body expands to the right unless we also change `\oddsidemargin`. Margin settings can be negative; for example, we raise the body by 1 inch by specifying `\topmargin=-1in` in the preamble. This might be accompanied by increasing the text length.

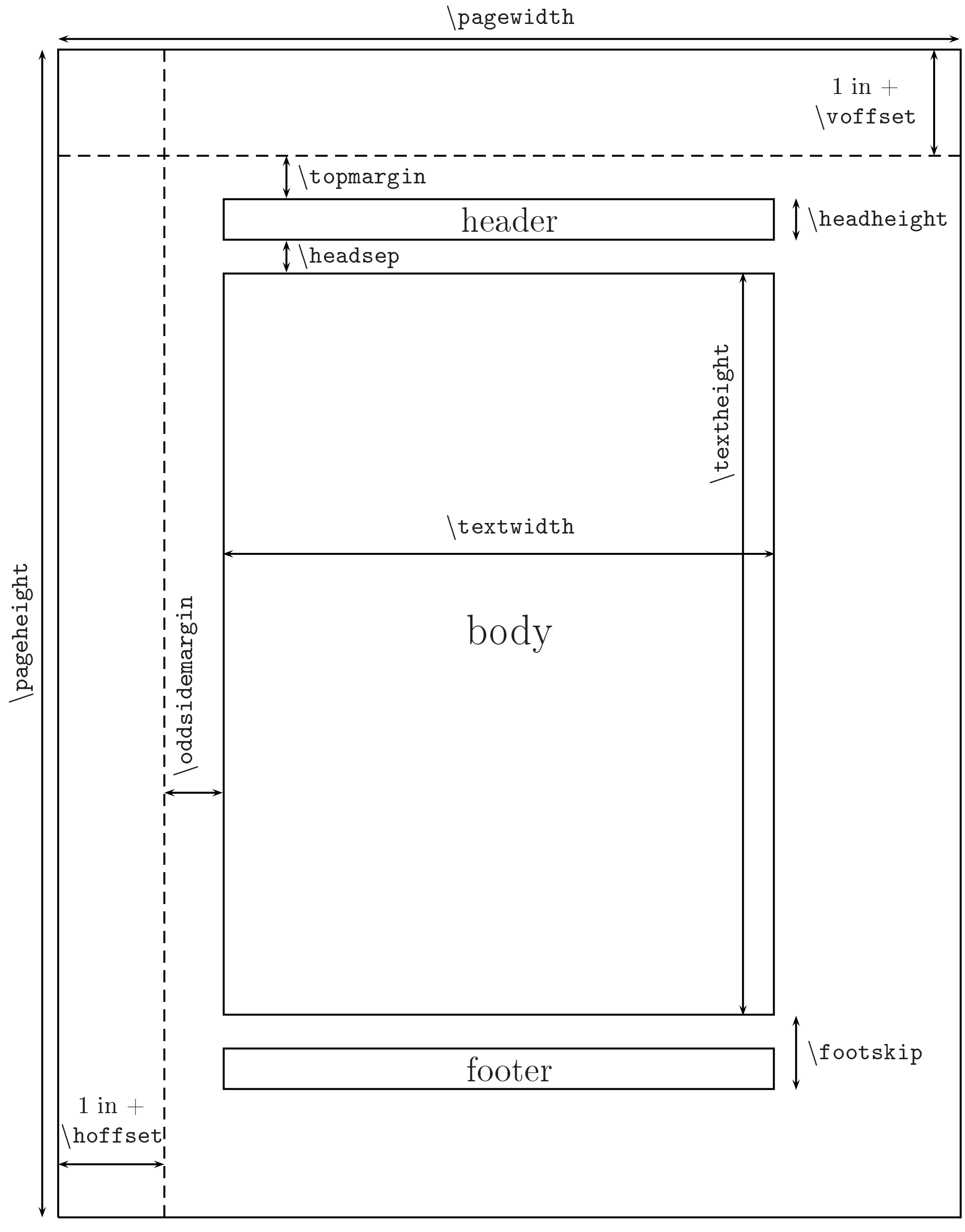


Figure 69: Document Margins

The `\hspace*` and `\vspace*` commands provide a great deal of control over horizontal and vertical spacing, respectively. We might want some global settings to make repeated use of these unnecessary. Table 23 lists some you can set with the `\setlength` command, showing their default values (used in this document).

Parameter	Meaning
`\itemsep`	space added to `\parsep` between items in a list.
`\parindent`	indentation at beginning of paragraph.
`\parsep`	space between paragraphs in the same item of a list.
`\parskip`	space between paragraphs.

Table 23: Spacing Parameters

In the case of list parameters, they must be set after entering the list environment. (Defaults are restored after leaving.) For example, the lists in §2.2 (p. 13) are spaced by default values. Here is what happens when we change `\itemsep`:

- The default value of `\itemsep` is 4.5pt plus 2.0pt minus 1.0pt, and I have saved it by: `\setlength{\mylength}{\itemsep}`.

- See the above spacing between items. What you see next is with `\setlength{\itemsep}{0pt}`.
- What you see next is with `\setlength{\itemsep}{10pt}`.

- Next is back to normal by `\setlength{\itemsep}{\mylength}`.

- We are back to normal with `\itemsep` = 4.5pt plus 2.0pt minus 1.0pt.

Figures 70 and 71 show the presentation of an array with a `p`-column to put horizontal space between the other two columns. Note how congested it is, so we want to increase its vertical spacing.

```
\[ \begin{array}{lp{.3in}l}
    \,B\,x_B = b_N + \frac{1}{2}\theta\delta b_N
        && \pi_N B = c_B + \frac{1}{2}\theta\delta c_B \\
      B^*x_B > b_B + \frac{1}{2}\theta\delta b_B
        && \pi_N N < c_N + \frac{1}{2}\theta\delta c_N
   \end{array}
\]
```

Figure 70: Array with Fixed Width Column Source (Result in Figure 71)

$$\begin{array}{ll} B\,x_B = b_N + \frac{1}{2}\theta\delta b_N & \pi_N B = c_B + \frac{1}{2}\theta\delta c_B \\ B^* x_B > b_B + \frac{1}{2}\theta\delta b_B & \pi_N N < c_N + \frac{1}{2}\theta\delta c_N \end{array}$$

Figure 71: Array with Fixed Width Column Result (Source in Figure 70)

Here is 1.3 line spacing: `\renewcommand{\arraystretch}{1.3}`

$$\begin{array}{ll} B\,x_B = b_N + \frac{1}{2}\theta\delta b_N & \pi_N B = c_B + \frac{1}{2}\theta\delta c_B \\ B^* x_B > b_B + \frac{1}{2}\theta\delta b_B & \pi_N N < c_N + \frac{1}{2}\theta\delta c_N \end{array}$$

Here is 1.6 line spacing: `\renewcommand{\arraystretch}{1.6}`

$$\begin{array}{ll} B\,x_B = b_N + \frac{1}{2}\theta\delta b_N & \pi_N B = c_B + \frac{1}{2}\theta\delta c_B \\ B^* x_B > b_B + \frac{1}{2}\theta\delta b_B & \pi_N N < c_N + \frac{1}{2}\theta\delta c_N \end{array}$$

Back to default: `\renewcommand{\arraystretch}{1}`

$$\begin{array}{ll} B\,x_B = b_N + \frac{1}{2}\theta\delta b_N & \pi_N B = c_B + \frac{1}{2}\theta\delta c_B \\ B^* x_B > b_B + \frac{1}{2}\theta\delta b_B & \pi_N N < c_N + \frac{1}{2}\theta\delta c_N \end{array}$$

8.5 Your Own Bibliography

You can choose not to use BibTeX, and use thebibliography environment instead. You will have complete control over the formatting, and there will be no sorting — the list of references will appear in the order you put them. Instead of the BibTeX commands, `\bibliography{mybiblio}` and `\bibliographystyle{plain}`, specify the following:

```
\begin{thebibliography}{n}
  \bibitem[what appears]{label (that you cite)} entry
     ⋮
\end{thebibliography}
```

where n is the width of the widest label you want to allow. (It works if you specify 99.) Each `\bibitem` is an entry, as described for BibTeX in §3 (p. 30), with *label* the unique identifier used by the `\cite` command. The option is an alternative to having the references numbered, and you can enter whatever you like.

Here is a complete example with two references, which I formatted to agree with the plain style of BibTeX:

```
\begin{thebibliography}{99}
\bibitem{companion} Michel Goosens, Frank Mittelbach and Alexander
            Samarin, \textit{The \LaTeX\ Companion}, Addison-Wesley
            Publishing Company, Reading, MA, 1994.
\bibitem{tex} Donald E. Knuth, \textit{The \TeX\ Book},
            Addison-Wesley Publishing Company, Reading, MA,
            15th edition, 1989.
\end{thebibliography}
```

These will appear in the document's list of References even if they are not cited. They can be cited in the same way described in §3: by `\cite{companion}` and `\cite{tex}`, respectively. When citing Knuth's book, for example, we obtain [2] in the text. Alternatively, we can exercise the option:

```
\bibitem[Knuth, 1989]{tex} Donald E. Knuth, ...
```

in which case `\cite{tex}` ⇒ [Knuth, 1989].

Some publishers give you no choice, but if you are writing a report and have control over the formatting, it generally helps the reader to know something about the citation. Thus, [Knuth, 1989] is preferred to [2] because it immediately gives the reader information about the document without having to flip to the bibliography section.

With you in control, there is no format monitoring, so each entry appears however you put it, even if there are inconsistencies in style. This is one reason it is usually better to use BibTeX, even though you lose control over what appears (i.e., they will be numbers).

If you want to have several bibliographic units in one document, such as at the end of each chapter of a book, use the `bibunits` package, which you obtain from CTAN [4].

Closing Remarks

Now you know how to write a mathematical document in $\LaTeX\,2_\varepsilon$ and you know there is much more you can learn to gain refinements. Besides what you can do yourself to elevate the quality of the results, there are many packages, available from CTAN [4]. Figure 72 shows the preamble used for this document. As you begin to use packages, it is necessary to become aware of updates. You learn about these at CTAN [4].

```
\usepackage{graphicx,pst-all}   % graphics
\usepackage{makeidx}            % index
\usepackage{url}                % \url{...}
\usepackage[T1]{fontenc}        % ...to write \textbf{\textsc{..}}
\usepackage{float}              % enable float [H] option
\usepackage{fancyvrb,moreverb}  % verbatim
\usepackage{multirow}           % like multicolumn
\usepackage{amsmath}            % formerly amstex
\usepackage{amssymb}            % ams symbols (\mathbb fonts)
\usepackage{mathrsfs}           % more math symbols (like \mathscr)

\renewcommand\contentsname{Table of Contents}   % Change ‘Contents’
\renewcommand\url{\begingroup\urlstyle{sf}\Url} % put url in sf font

\input{mydefs} % My commands and environments
\makeindex     % make myfile.idx (input to makeindex at command line)
```

Figure 72: Most of the Preamble for this Document

Appendix

This contains complete tables of font information and basic LaTeX commands. It is designed like a reference manual for easy lookup, beginning with Table 24, which gives conversion among three common units of measurement.

	pt	in	cm
pt	1	.01384	.03515
in	72.27	1	2.54
cm	28.45	.3937	1

Table 24: Conversions of Common Units of Measurement

Table 25 is a guide to how most of the remaining tables are organized. Afterwards, Table 44 gives special symbols that can be used in either text or math mode, and Table 45 gives the commands for the picture environment.

	Table	Contents
Text mode	26	Commands/Environments for Font Appearance
	27	Text Accents and Symbols
	28	Commands/Environments for Controlling Position
	29	Commands for Counters
	30	Commands/Environments to Organize Document
	31	Commands to Control Document Style
Math mode	32	Commands to Control Fonts in Math Mode
	33	Mathematical Accents and Symbols
	34	Greek and Special Letters
	35	Spacing Commands in Math Mode
	36	Frequently Used Mathematical Symbols
	37	Binary Operations
	38	Operators and Quantifiers
	39	Special Functions
	40	Relation Symbols
	41	Arrows
	42	Dots Circles, Triangles and Lines
	43	Variable Size Symbols

Table 25: Reference Tables

textbf	textit	textrm	textsc	textsf	texttt
tiny	scriptsize	footnotesize	small	normalsize	large
Large	LARGE	huge	Huge	underline	verb
verbatim					

Table 26: Commands/Environments for Text Font Appearance

á	\'{a}	ŭ	\u{u}	ç	\c{c}	ẋ	\.{x}
è	\`{e}	ñ	\~{n}	ḍ	\d{d}	z̄	\={z}
î	\^{i}	H̋	\H{H}	ḇ	\b{b}	v̌	\v{v}
ö	\"{o}	o͡o	\t{oo}	...	\dots		
æ	\ae	œ	\oe	å	\aa	ø	\o
Æ	\AE	Œ	\OE	Å	\AA	Ø	\O
Ł	\L	ß	\ss	¿	?`	¡	!`

Table 27: Text Accents and Special Symbols

bigskip	center	centerline	clearpage	flushleft
flushright	hfill	hspace	hspace*	linebreak
medskip	newpage	noindent	nolinebreak	nopagebreak
pagebreak	quotation	quote	raisebox	samepage
smallskip	tabbing	tabular	verse	vfill
vspace	vspace*			

Table 28: Commands/Environments for Controlling Text Position

addtocounter	label	newcounter	pageref
ref	refstepcounter	setcounter	stepcounter
the*counter*	value		

Table 29: Commands for Counters

abstract	addcontentsline	addtocontents
appendix	bibliography	bibliographystyle
listoffigures	listoftables	makeindex
maketitle	printindex	section
subsection	subsubsection	subsubsubsection
tableofcontents	thanks	thebibliography

Table 30: Commands/Environments to Organize Document

markright	markboth	pagenumbering	pagestyle
renewcommand	setlength	thispagestyle	

Table 31: Commands to Control Document Style

left	boldmath (set in text mode)		
cal	displaystyle	mathbf	mathcal
mathit	mathnormal	mathrm	mathsf
mathtt	mbox	overbrace	overline
right	textstyle	underbrace	underline

Table 32: Commands to Control Fonts in Math Mode

$\check{a}$	`\check{a}`	$\breve{e}$	`\breve{e}`	$\acute{i}$	`\acute{i}`	$\grave{o}$	`\grave{o}`
$\dot{x}$	`\dot{x}`	$\ddot{y}$	`\ddot{y}`	$\bar{z}$	`\bar{z}`	$\vec{v}$	`\vec{v}`
$\hat{\imath}$	`\hat{\imath}`	$\tilde{\jmath}$	`\tilde{\jmath}`			$\hbar$	`\hbar`
$\widehat{xyz}$	`\widehat{xyz}`	$\widetilde{abc}$	`\widetilde{abc}`				

(Note that it is better style to use `\imath`, rather than i, and `\jmath`, rather than j, to avoid the clash between the accent and dot.)

Table 33: Accents in Math Mode

α	`\alpha`	θ	`\theta`	o	`o`	τ	`\tau`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	υ	`\upsilon`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	ϕ	`\phi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	φ	`\varphi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	χ	`\chi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ψ	`\psi`
ζ	`\zeta`	ν	`\nu$`	ς	`\varsigm`	ω	`\omega`
η	`\eta`	ξ	`\xi`				
Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		
$\aleph$	`\aleph`	ℓ	`\ell`	$\Re$	`\Re`	$\Im$	`\Im`
$\mathcal{A} \ldots \mathcal{Z}$	`{\mathcal A ... Z}`						

Table 34: Greek and Special Letters

What you write		What you see	
`x y`	$\Rightarrow$	xy	no space
`x\,y`	$\Rightarrow$	$x\,y$	thin space
`x\;y`	$\Rightarrow$	$x\;y$	medium space
`x\quad y`	$\Rightarrow$	$x\quad y$	space = 1em
`x\qquad y`	$\Rightarrow$	$x\qquad y$	space = 2em
`x\!y`	$\Rightarrow$	$x\!y$	negative thin space
`x\negmedspace y`	$\Rightarrow$	$x\negmedspace y$	negative medium space
`x\negthickspace y`	$\Rightarrow$	$x\negthickspace y$	negative thick space

Table 35: Spacing Commands in Math Mode

{superscript}	`^{}`	$\prime$	`\prime`	∞	`\infty`	$\emptyset$	`\emptyset`
{subscript}	`_{}`						

Table 36: Frequently Used Mathematical Symbols

$\pm$	\pm	$\cap$	\cap	$\cup$	\cup	$\odot$	\odot
$\mp$	\mp	$\sqcap$	\sqcap	$\sqcup$	\sqcup	$\otimes$	\otimes
$\times$	\times	$\wedge$	\wedge	$\uplus$	\uplus	$\oslash$	\oslash
$\div$	\div	$\vee$	\vee	$\oplus$	\oplus	$\ominus$	\ominus
$\setminus$	\setminus	$\bigcap$	\bigcap	$\bigcup$	\bigcup	$\bigodot$	\bigodot
$\backslash$	\backslash	$\bigvee$	\bigvee	$\bigoplus$	\bigoplus	$\bigotimes$	\bigotimes
$\biguplus$	\biguplus	$\bigwedge$	\bigwedge	$\bigsqcup$	\bigsqcup		

Table 37: Binary Operations

∇	\nabla	∂	\partial	$\surd$	\surd	$\wp$	\wp
$\forall$	\forall	$\exists$	\exists	$\neg$	\neg		

Table 38: Operators and Quantifiers

arccos	arcsin	arctan	arg	cos	cosh	cot	coth
csc	det	dim	exp	gcd	hom	inf	ker
lg	lim	liminf	limsup	ln	log	max	min
Pr	sec	sin	sinh	sup	tan	tanh	

Table 39: Special Functions

$\leq$	\leq	$\geq$	\geq	$\neq$	\neq	$\equiv$	\equiv
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim	$\doteq$	\doteq
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq	$\models$	\models
$\ll$	\ll	$\gg$	\gg	$\cong$	\cong	$\propto$	\propto
$\subset$	\subset	$\supset$	\supset	$\asymp$	\asymp	$\in$	\in
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\approx$	\approx	$\ni$	\ni
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq				

Table 40: Relation Symbols

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow$
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow
$\rightleftharpoons$	\rightleftharpoons				

Table 41: Arrows

$\circ$	\circ	$\bigcirc$	\bigcirc
$\cdots$	\cdots	$\ddots$	\ddots
$\vdots$	\vdots	$\bullet$	\bullet
$\frown$	\frown	$\smile$	\smile
$\triangle$	\triangle	$\diamond$	\diamond
$\triangleright$	\triangleright	$\triangleleft$	\triangleleft
$\bigtriangleup$	\bigtriangleup	$\bigtriangledown$	\bigtriangledown
$\bowtie$	\bowtie	$\perp$	\perp
$\top$	\top	$\bot$	\bot
$\dashv$	\dashv	$\vdash$	\vdash
$\angle$	\angle	$\Vert$	\\|
$\mid$	\mid	$\parallel$	\parallel

Table 42: Dots, Circles, Triangles and Lines

$\sum$	`\sum`	$\int$	`\int`	$\oint$	`\oint`
$\sqrt{\cdot}$	`\sqrt{ . }`	$\overbrace{\cdot}$	`\overbrace{ . }`	$\underbrace{\cdot}$	`\underbrace{ . }`
$\frac{n}{d}$	`\frac{n}{d}`	$\overline{\cdot}$	`\overline{ . }`	$\underline{\cdot}$	`\underline{ . }`
$\prod$	`\prod`	$\lfloor$	`\lfloor`	$\rfloor$	`\rfloor`
$\coprod$	`\coprod`	$\langle$	`\langle`	$\rangle$	`\rangle`

Table 43: Variable Size Symbols

†	`\dag`	§	`\S`	©	`\copyright`
‡	`\ddag`	¶	`\P`	£	`\pounds`
...	`\ldots`				

Table 44: Special Symbols in Both Text and Math Modes

`put`(x,y){*stuff*}	`multiput`$(x,y)(\Delta x,\Delta y)${*number*}{*stuff*}
`line`(x,y){*length*}	`framebox`(*width*, *height*)[*p*]{*text*}
`vector`(x,y){*length*}	`dashbox`{$dash_size$}(*width*, *height*)[*p*]{*text*}
`circle`{*radius*)	`makebox`(*width*, *height*)[*p*]{*text*}
`circle*`{*radius*)	`oval`(*width*, *height*)[*p*]
`linethickness`{*dimension*}	

$p \in \{$`l,r,t,b,lt,lb,rt,rb`$\}$. For `oval`, it is the portion selected; for boxes, p is where the text goes.

Table 45: Commands and Parameters in Picture Environment

References

[1] Johannes L. Braams. Babel, a multilingual style-option system for use with LaTeX's standard document styles. *TUGboat*, 12(2):291–301, 1991. Available at CTAN [4].

[2] Johannes L. Braams, David P. Carlisle, Alan Jeffrey, Frank Mittelbach, Chris Rowley, and Rainer Schöpf. LaTeX 2ε and the LaTeX3 project. World Wide Web, http://www.latex-project/org/latex3.html, 1994–99.

[3] David P. Carlisle. *Packages in the 'graphics' bundle.* World Wide Web, CTAN/macros/latex/required/graphics/ (see [4] for replacing CTAN), 1994–99.

[4] Comprehensive TeX archive (CTAN). UK: ftp.tex.ac.uk/tex-archive/; Germany: ftp.dante.de/tex-archive/; USA: ftp.tug2.cs.umb.edu/tex-archive/ . These are host sites, which contain a list of mirror sites.

[5] Michel Goosens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Publishing Company, Reading, MA, 1994.

[6] John D. Hobby. A User's Manual for MetaPost. Computing Science Technical Report no. 162, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Available at http://cm.bell-labs.com/who/hobby/MetaPost.html/.

[7] John D. Hobby. Drawing Graphs with MetaPost. Computing Science Technical Report no. 164, AT&T Bell Laboratories, Murray Hill, New Jersey, 1993. Available at http://cm.bell-labs.com/cs/cstr/164.ps.gz.

[8] Donald E. Knuth. *The TeX Book.* Addison-Wesley Publishing Company, Reading, MA, 15th edition, 1989.

[9] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley Publishing Company, Reading, MA, 1986 (also see 2nd edition, 1994).

[10] *LaTeX 2ε for authors.* CTAN/macros/latex/doc/usrguide.ps (see [4] for replacing CTAN), 1995–99.

[11] Oren Patashnik. BibTeX*ing.* World Wide Web, http://www.uic.edu/depts/adn/infwww/ps/btxdoc.ps, 1988.

[12] Keith Reckdahl. Using imported graphics in LaTeX 2ε. World Wide Web site Version 2.0, Comprehensive TeX Archive, CTAN/info/epslatex.ps (see [4] for replacing CTAN), 1995–97.

[13] Christian Schenk. *MiKTeX Local Guide.* World Wide Web, http://www.miktex.de/, 1998–99 (version 1.2).

[14] Timothy Van Zandt. *PSTricks: PostScript macros for Generic TeX.* World Wide Web, http://www.tug.org/applications/PSTricks/, 1993–98.

Index