13 February 2023

COMPUTATION AND PROBLEM SOLVING IN UNDERGRADUATE PHYSICS

Second Edition

	IDL	MATLAB	OCTAVE
	PYTHON	MAXIMA	MAPLE
	MATHEMATICA	FORTRAN	C
	NUMERICAL RECIPES	LSODE	MUDPACK
●	LaTeX	TGIF	

DAVID M. COOK

Department of Physics
Lawrence University
711 E Boldt Way, SPC 24
Appleton, Wisconsin 54911

This document is Appendix A from the book *Computation and Problem Solving in Undergraduate Physics* (*CPSUP*). The first edition of *CPSUP* was registered with the Library of Congress with the call number QC20.C66.2004.

The second edition was registered with the Library of Congress with the LCCN 2019947884. It has been assigned the ISBN 9780961342968 and carries the Library of Congress call number QC20.C66.2019.

Preface

This document contains a printing of the LaTeX appendix from the second edition of a much larger book titled *Computation and Problem Solving in Undergraduate Physics* (*CPSUP*). It is published separately so that it can be available to provide a stand-alone quick introduction to LaTeX for those who are interested in only that piece of software. This Appendix provides only a start. In Section A.18, it also points readers to a number of materials to which they can refer to expand their knowledge beyond the limited background here provided.

LaTeX is a very large share-ware collection of components for creating publication-quality documents containing text, figures, equations, tables, and many other components that often festoon scientific documents. It is freely available and runs successfully and reliably on numerous platforms and under a wide variety of operating systems. More detailed information about the spectrum of capabilities, including information about how to acquire and install LaTeX can be found in many places, including the following:

LaTeX – see TeX

MacTeX

MacTeX is a shareware implementation of the TeX/LaTeX system for Macintosh computers. Information is available at the URL `www.tug.org/mactex`.

MiKTeX

MiKTeX is an implementation of the TeX/LaTeX system for Windows computers. Information is available at the URL `www.miktex.org`. MiKTeX is also available from CTAN, the TeX/LaTeX distribution network accessible from links at `www.tug.org`.

TeX/LaTeX/dvips/graphicx/makeindex/xdvi/dvips/...

The primary site for information (history, current plans, downloads, ...) for TeX, LaTeX, and numerous other publicly available components of TeX and its derivatives is the web site of the TeX Users' Group (TUG), `www.tug.org`. This organization maintains CTAN (the Comprehensive TeX Archive Network), which has a handful of backbone machines around the world and a number of mirror sites, from any of which an enormous number of files associated with the TeX/LaTeX system can be downloaded.

TeXlive

TeXlive is among the newer implementations of the TeX/LaTeX system for all platforms. Information about this distribution and instructions for downloading and installing it are available from the URL `www.tug.org/texlive`.

The much larger work *CPSUP*, which has been under continuous development and refinement at Lawrence University since the mid 1980's, was first published in 2003 with a second edition somewhat expanded from the first edition coming available in 2017. *CPSUP* is a flexible, customizable text designed to

- support efforts to acquaint students with computational procedures and resources early enough so that they will be motivated and prepared to use these resources *on their own initiative* when circumstances warrant and so that later work need not be interrupted to deal with computational issues as an aside to its main purposes, and
- provide students with both the background and the confidence to support informed reading of vendor manuals, which usually do a splendid job of listing capabilities exhaustively but typically burden the beginner with initially irrelevant refinements and fail to illustrate adequately how even the rudimentary capabilities can be combined to perform useful tasks.

At Lawrence University, portions of *CPSUP* are used in a required sophomore course titled *Computational Mechanics* (CompMech) in which students develop an intermediate understanding of classical mechanics but also start the process of learning how to use available computational tools to pursue their studies in physics. The LaTeX Appendix fosters an acquaintance with what may well be the most versatile tool for publishing scientific results, and students in that course are expected to submit some of their problem solutions as polished LaTeX documents. The hope is that the materials treated in *CPSUP* and in CompMech will provide strong background so that instructors in subsequent courses can assign computational exercises and students can in subsequent courses have the confidence to exploit computational resources on their own initiative.

David M. Cook
Appleton, Wisconsin
13 February 2023

Table of Contents

Appendix A

Introduction to LaTeX

> *Note*: All LaTeX source files (`.tex`, `.template`), all Windows batch files (`.bat`) program (`.pro`) files, and all UNIX command files (no file type) referred to in this chapter are available in the directory `$HEAD/tex`, where (as defined in the *Local Guide*) `$HEAD` must be replaced by the appropriate path for your site.

During the 1970s, when Donald Knuth (a Stanford University computer scientist) was creating his monumental, several-volume series of books on all aspects of computer science, he recognized the need for a computer-based type-setting/word-processing system tailored to the needs of authors of technical manuscripts. Interrupting his main project, he developed TeX[1] as the essential engine to support computerized technical type setting. The first version of TeX was made available to the using public in 1978 and a revised (and final) version was published in 1984. Written by Leslie Lamport, LaTeX[2] is a versatile and extensive collection of macros that sit on top of TeX and facilitate exploitation of TeX's capabilities. Broadly, LaTeX is a document preparation system that formats equations, tables, and illustrations as easily as plain text. That the whole complex (TeX, LaTeX, and many other components) has been deliberately placed in the public domain,[3] that carefully tested versions exist for essentially every computing platform and operating system, and that many scientific journals and an increasing number of publishers will accept manuscripts submitted as LaTeX files together provide substantial incentive for learning to use this tool. Although it is not wysiwyg,[4] it has powerful and sophisticated capabilities, most of which are *not* described in this Appendix. We convey only the basics with, however, the hope that we will cause you to realize that essentially any desired formatting at all should be possible if only you can figure out how to achieve it. For further information, try surfing the web in your favorite browser and looking at the *LaTeX2ε User's Guide and Reference Manual* (which we will refer to as *The LaTeX Manual*), the *LaTeX Companion*, and/or to any of the several other TeX and LaTeX books.[5] The wisdom of taking fifteen minutes every now and then to browse in these publications cannot be overstressed; as you become more skilled, the fine print and other subtleties will gradually be more and more meaningful.[6]

[1] As per Knuth's instruction (see the first two paragraphs of Chapter 1 of *The TeXbook* as identified in Section A.18), TeX is pronounced 'tech' as in 'technology'.

[2] Here, individuals disagree on the pronunciation of the 'la', and Leslie Lamport offers no guidance. Some say lahTeX while others say layTeX, with the emphasis on the first syllable in both cases. We simply must become accustomed to each other's preferences.

[3] The primary site for information (history, current plans, downloads, ...) for TeX, LaTeX, and numerous other publicly available components of TeX and its derivatives is the web site of the TeX Users' Group (TUG), `www.tug.org`. This organization maintains CTAN (the Comprehensive TeX Archive Network), which has a handful of backbone machines around the world and a number of mirror sites, from any of which an enormous number of files associated with the TeX/LaTeX system can be downloaded.

[4] *w*hat *y*ou *s*ee *i*s *w*hat *y*ou *g*et.

[5] See Section A.18 for more detailed references.

[6] You may also find the web pages of TUG at the URL `http://www.tug.org` to be valuable. One link on that page points to an engaging description of the history of the development of TeX.

A.1 Creating a Simple Document

Producing a final document with LaTeX involves a number of steps, beginning with the creation of an ASCII text file containing the LaTeX "source code"—hereafter simply "code"—for the document. Appropriate conversion programs are then invoked to create either a PostScript or a PDF file containing the formatted document.[7] Finally, the PostScript or PDF file will be viewed on the screen or sent to a printer. In this section, we describe how to structure the code for a *very* simple document and then how to carry out the remaining steps to convert that code into a displayed or printed document. The remainder of this Appendix will explain numerous embellishments that might be invoked in the code. Processing that code to produce the final document is (more or less) independent of the complexity of the code.

To keep the discussion comparatively simple, we assume initially that your document incorporates no figures, does not have internal cross references, and includes neither a table of contents or an index. Those embellishments will be described in later sections.

A.1.1 Structuring a LaTeX Source File

As input, LaTeX requires an ASCII file, which can be created with any text editor (`xemacs`, `notepad`, `wordpad`, `winedt`, `gedit`, ...) and which contains both the text of the desired document and embedded formatting commands. All of LaTeX's commands *begin* with a backslash (\). LaTeX distinguishes upper and lower case letters; while most standard commands and parameters use exclusively lower case letters, we must nonetheless pay attention to case.

Three commands are sufficient to create the simplest document containing straight text formatted with LaTeX's defaults. The very first required command in the code for *any* LaTeX document specifies the document *class* and has the form[8]

`\documentclass{`*Class*`}` or `\documentclass[`*options*`]{`*Class*`}`

where

- *Class* is any one of the options shown in Table A.1 and
- *options* may be omitted altogether if the defaults are acceptable. The most commonly used options, one or more of which may be separated by commas in a string containing *no* spaces, are listed in Table A.2.

The document itself must be enclosed between the second and third of the essential commands, the second marking the beginning of the document proper and having the form

```
\begin{document}
```

and the third marking the end of the document and having the form

```
\end{document}
```

[7] Other formats (HTML, ePub, ...) may also be produced, though this Appendix will not address those alternatives.

[8] Be aware that some of the features described in this Appendix are specific to LaTeX2$_\varepsilon$ and will not work with the previous version (LaTeX 2.09). LaTeX2$_\varepsilon$, however, will automatically enter 2.09 emulation mode if the first line in the code is the now obsolete command `\documentstyle` instead of the new command `\documentclass`.

Table A.1: Document classes valid in standard LaTeX. Other available classes may be described in the *Local Guide*.

`article`	This class is by far the most common. It is used for most short documents. See Sections 2.2.2 and C.5.1 in *The LaTeX Manual*.
`report`	This class is generally used for longer documents. See Sections 2.2.2 and C.5.1 in *The LaTeX Manual*.
`book`	This class is meant for actual books. See Sections 5.1 and C.5.1 in *The LaTeX Manual*.
`slides`	This class can be used in creating originals from which overhead transparencies to be used in presentations can be made. The type size, including the size used for mathematical symbols, is quite large, and transparencies readable from some distance will be produced. See Sections 5.2 and C.5.1 in *The LaTeX Manual*.
`letter`	This class facilitates creating letters. See Sections 5.3 and C.5.1 in *The LaTeX Manual*.

Table A.2: The most common options for the `documentclass` command. Numerous other options are described in Section C.5.1 in *The LaTeX Manual* and in the other references listed in Section A.18.

`10pt` or `11pt` or `12pt`	(default = 10pt) Specifies default type size.
`letterpaper` or `legalpaper` or `a4paper` or ...	(default depends on site but is usually `letterpaper` in the US) Specifies size of paper to be used.
`landscape`	(default is portrait orientation) Specifies that text is to be formatted for landscape orientation on the specified paper size.
`oneside` or `twoside`	(default = `oneside`) Specifies whether output is to be formatted for single- or double-sided printing. The `twoside` option allows for left and right margins and running head positions to be different on odd and even numbered pages.
`onecolumn` or `twocolumn`	(default = `onecolumn`) Specifies one- or two-column formatting on each page.

While there can be material in the code beyond this final command, none of that material will be read or processed by LaTeX.

The three commands described in the previous paragraph are *mandatory* in all documents. Supplemented by the additional feature that a blank line in the code triggers a new paragraph, they are also *sufficient* for the creation of code for any document consisting of nothing but paragraphs of straight text to be formatted in accordance with all of LaTeX's defaults. Table A.3 shows a listing of a simple code that invokes only these commands.

Whether the code is simple (as in Table A.3) or much more elaborate, producing the final printed document involves additional steps. One first creates a printable file—PostScript and PDF are the most common—containing the formatted document and then displays that file on a screen and/or sends it to a printer.

Table A.3: A simple LaTeX code.

```
\documentclass{article}

\begin{document}

This sample illustrates the simplest code containing the
mandatory commands and a brief text. Since no
optional formatting commands have been included, the text will be
formatted using all of the built-in defaults (margins,
paragraph indent, type size and font, etc.).

The blank line preceding this line in
the code will trigger a new paragraph.
Each paragraph is indented, but note that there is no extra
space between paragraphs. Note also that the lines in the
code can be quite ragged; they will be filled and justified in
the processing that produces the final copy.

\end{document}
```

A.1.2 Creating a PostScript File

Converting the LaTeX source file into a PostScript file describing the desired document involves two steps:

1. First, we must "compile" the code by processing the file with LaTeX, a task accomplished either (1) by typing a command like

   `latex` *filename* (Default extension for the input file is `.tex`.)

   where *filename* specifies the input file to be processed, or (2) by selecting an item from a menu in a graphical user interface (GUI).[9] However LaTeX is invoked, it will create several output files, all of which will have the same name as the input file except for the extension: (1) a binary output file, with extension `.dvi`, which contains the translation of the original file into TeX's *dev*ice *i*ndependent language; (2) an ASCII log file, with extension `.log`, which contains an expanded log of the messages that appear on the screen as `latex` works its magic on the input file; and (3) an auxiliary ASCII file, with extension `.aux`, which contains information that is important only if the code makes use of LaTeX's capabilities for generating internal cross references.[10] The resulting `.dvi` file can be directly displayed on the screen (see Section A.1.4).[11]

2. Second, we must translate the `.dvi` file into a file that can be understood by the printer on which the document is to be printed. Most often, the `.dvi` file will be translated into

[9] See the *Local Guide* for the precise command at your site.

[10] These capabilities will be described in later sections of this Appendix. For now, we may ignore messages relating to the `.aux` file, noting only that, when we ultimately do make use of internal references, the *first* pass of the document through LaTeX writes the `.aux` file and a *second*—and occasionally a third—pass is necessary to incorporate that information in the final formatting of the pages.

[11] Remember that we have limited the present discussion to documents containing no figures. See Section A.7 for the adjustments to be made if PostScript or PDF figures are included.

Table A.4: The output from the code in Table A.3. The page number at the bottom of the page has been cut off in this display.

> This sample illustrates the simplest code containing the mandatory commands and a brief text. Since no optional formatting commands have been included, the text will be formatted using all of the built-in defaults (margins, paragraph indent, type size and font, etc.).
>
> The blank line preceding this line in the code will trigger a new paragraph. Each paragraph is indented, but note that there is no extra space between paragraphs. Note also that the lines in the code can be quite ragged; they will be filled and justified in the processing that produces the final copy.

PostScript, the most common program for effecting that translation being `dvips`. The necessary translation will be achieved either (1) by typing a command like[12,13]

`dvips -o` *filename*`.ps -t letter` *filename* (Default extension for the input file is `.dvi`.)

which translates the entire file into PostScript, stores it in a file, and (with the `-o` option here used) gives the stored file the same name as the input file but with extension `.ps` and with the `-t` option forces use of letter size paper[14] or (2) by selecting an item from a menu.[15]

To be more explicit, the file `texsample1.tex`, contains the LaTeX code in Table A.3. Once this file has been copied to the default directory, we would produce the `.dvi` and `.ps` files with statements like

```
latex texsample1
dvips -o texsample1.ps -t letter texsample1
```

or equivalent selections from a menu.

A.1.3 Creating a PDF File

PDF files can be created from the LaTeX code in at least two ways:

1. Starting with LaTeX source file created in Section A.1.1, one submits to the operating system the single statement[16]

   `pdflatex` *filename* (Default file type is `.tex`)

   or, for the sample above,

   `pdflatex texsample1`

[12] Additional information about `dvips` can be found by typing the command `dvips` with no arguments. Further, the option `-pp` allows selection of specific pages from the full document, the options `-p` (first page) and `-l` (last page) allow selection of a range of pages from the full document, and the option `-t` allows printing of the output in landscape orientation rather than in the default portrait orientation. Even more detailed information is available in the on-line help for `dvips`, accessed in many systems by typing the statement `texdoc dvips` at a *Shell* prompt or, in the UNIX and LINUX operating systems, the statement `man dvips` at a *Shell* prompt.

[13] If the explicit specification of the output file is omitted, `dvips` will attempt to send the file directly to the printer—and the whole operation will probably fail. Further, the PostScript file will then not be stored for subsequent use.

[14] This option is often the default. The option `-t landscape` will override the default portrait orientation.

[15] Check the *Local Guide* for more details. In particular, some sites may have implemented the shorthand command `dvip` *filename* for the command `dvips -o` *filename*`.ps` *filename*.

[16] The program `pdflatex` is normally installed automatically whenever LaTeX is installed.

which will produce directly a PDF file named *filename*`.pdf` or `texsample1.pdf` as well as two additional files, specifically the `.aux` (which we can ignore for now) and the `.log` file containing an expanded log of the messages that appear on the screen as `latex` works its magic.

2. Starting with the `.ps` file produced in Section A.1.2, one executes the single statement

   `ps2pdf` *filename*`.ps` (No default file type)

   or, for the sample above

   `ps2pdf texsample1.ps`

   which will produce a PDF file named *filename*`.pdf` or `texsample1.pdf` from the `.ps` file.

Exploration of other routes to a PDF file is left to the reader. In particular, the program `dvipdfm` can sometimes be used to convert the `.dvi` file created above directly to a PDF file without creating the PostScript file as an intermediary.

A.1.4 Displaying the Document on the Screen

Numerous programs for displaying documents produced with LaTeX on the screen exist. Among the more common are the following:

- (for displaying a `.dvi` file) `xdvi` for UNIX and LINUX platforms and `yap` (*y*et *a*nother *p*reviewer) for windows platforms. These programs are described more fully in Section A.14.
- (for displaying a PostScript file) `ghostview`, which is available for almost all platforms.
- (for displaying a PDF file) Adobe `acroread`, which is available for almost all platforms.

These programs are invoked by double-clicking ML on an icon for the program and then opening the desired file, by double-clicking ML on an icon for the file to be displayed, by executing an appropriate statement as a command in a *Shell* window, or by double-clicking ML on the name of the file in a directory displayed on the screen.[17]

A.1.5 Printing the Document

Programs (`xdvi`, `yap`, ghostview, `acroread`) for displaying a file on the screen often have an item in the File menu to request printing of the file on an available printer, and many times the print utility thereby invoked offers options for double-sided printing, scaling of the output, ... Often, a simple command like

UNIX/LINUX/MAC	Windows
`lp` *filename*`.ps`	`print` *filename*`.ps`

submitted from a *Shell* or *Command* window will request a printed copy of the file, though the destination printer must be known to the operating system and able to translate PostScript.[18] In the case of the typed command, the extension must this time be explicitly present, since the programs `lp` and `print` make no assumptions about file type.[19] In any case, the resulting output for our sample file is shown in Table A.4.

[17] See the *Local Guide* for specifics at your site.
[18] Printers without this capability are rare.
[19] Again, check the *Local Guide* for details on how to print a file.

A.2 Specification of Global Style: The Preamble

The standard document classes have default settings for the page setup (area allocated to text, margins, paragraph indent, paragraph separation, line spacing, etc.). Only occasionally are these defaults actually appropriate for the document being prepared. Therefore, it is frequently necessary to modify the global style of the document. Commands that accomplish this end are normally placed in the *preamble*—the section *between* the command `\documentclass` and the command `\begin{document}`—and affect the entire document. The most commonly used commands in the preamble invoke the general command

`\setlength{`*LengthParameter*`}{`*Value*`}`

to set the length parameter identified by *LengthParameter* to the value specified by *Value*. A length can be a positive or negative value given in points[20] (abbreviated `pt`), centimeters (abbreviated `cm`), or inches (abbreviated `in` or, when we wish to make the dimension immune to global changes of scale, `truein`). Further, to give LaTeX some flexibility in placing text on the page, *rubber* values can be specified. For example, a value of `9pt plus 1pt minus 2pt` gives LaTeX authority to "cheat" on the value over the range from 7 points to 10 points. The most useful length parameters are identified in Table A.5. Thus, the commands

```
\setlength{\textheight}{9truein}
\setlength{\textwidth}{6truein}
\setlength{\oddsidemargin}{0.25truein}
\setlength{\topmargin}{-0.5truein}
\setlength{\parindent}{20pt}
\setlength{\parskip}{6pt plus 2pt minus 1pt}
```

placed in the preamble will change the defaults to specify a 6"×9" area of text centered on an 8.5"×11" page with a 20 point paragraph indent and an *extra* 5–8 points between paragraphs. Indeed, these are (almost[21]) the settings used for this book.

A few issues of global style are specified by an optional argument in the command `\documentclass`. The most common of these arguments specifies the type size. By default, the document will be set in 10 point type, which is standard for the main text in many journals. To change the size of the characters *throughout* a document, modify the `\documentclass` command to[22]

`\documentclass[`*TypeSize*`]{`*Class*`}`

where, in the `article`, `book`, and `report` classes, *TypeSize* can be one of `10pt` (the default), `11pt`, or `12pt`. The `slides` class *admits* these specifications of size, but *ignores* them. The main text in this book is set in LaTeX's 10 point type, which is also common in many technical journals.

Several other global specifications can be included in the preamble or as optional arguments in the command `\documentclass`, including specifications regarding positioning and style of page headers, definitions of new commands, more subtle changes in the global style, selection of two-column format, specifications to anticipate ultimate printing on both sides of the page, etc. For details, the reader is referred to *The LaTeX Manual*.

Specific instructions for document styles can be saved in files for easy incorporation in documents as they are created. Several possibly useful *templates* are enumerated in Table A.6. Consult the

[20] The *point*, which is a standard printer's measure of length, is 1/72" (72 points per inch).

[21] The difference lies in the specification of the left margins. For this book (which uses double-sided printing), LaTeX's ability to specify one left margin (with `\oddsidemargin`) for odd-numbered pages and a different left margin (with `\evensidemargin`) for even-numbered pages has been exploited to keep the text more fully out of the binding than would otherwise be the case.

[22] By convention in LaTeX, *optional* arguments to a command are enclosed in *square brackets* and *mandatory* arguments are enclosed in *curly braces*.

Table A.5: Selected parameters that affect the size and positioning of the text area on a page. The given default values apply to the `article` style. Further information can be found in Section 6.4.1, at the very end of Section C.5.3, and in Fig. C.3 in *The LaTeX Manual*.

`\textheight`	(default 7.375") Specifies the vertical dimension of the region of the page occupied by text, excluding the head and the foot of the page. Its value is commonly 9" for an 8.5"×11" page.
`\textwidth`	(default 4.75") Specifies the horizontal dimension of the region of the page occupied by text. Its value is commonly 6" for an 8.5"×11" page.
`\oddsidemargin`	(default 0.75") For all pages (single-sided output) or odd-numbered pages (double-sided output), specifies the left margin—distance from the left edge of the page to the left edge of the region occupied by text—to be 1" plus the specified value. The value 0.25" will center 6" wide text on an 8.5"×11" page.
`\evensidemargin`	(default 0.75") Only for even-numbered pages with double-sided output, specifies the left margin—distance from the left edge of the page to the left edge of the region occupied by text—to be 1" plus the specified value. The value 0.25" will center 6" wide text on an 8.5"×11" page.
`\topmargin`	(default 0.25") Specifies the top margin, i.e., the distance from the top edge of the page to the top edge of the header above the main region occupied by text, to be 1.0" plus the specified value. By default, the height of the header (specified by `\headheight`) and the separation of the header from the main text (specified by `\headsep`) together add to 0.5", so—if the defaults for `\headheight` and `\headsep` are accepted—we can *pretend* that `\topmargin` specifies the distance from the top edge of the page to the top edge of "real" text to be 1.5" plus the specified value. Thus, for example, the value −0.5" will center 9" high text on an 8.5"×11" page and place the header line—if any—0.5" above the first line of text.
`\parindent`	(default 15.0 points) Specifies the paragraph indent.
`\parskip`	(default 0.0 points) Specifies the *extra* space *between* paragraphs.

Table A.6: Templates in the directory `$HEAD/tex`.

`lu_article.template`	Specifies a general article style, including a 6"×9" text area centered on an 8.5"×11" page.
`lu_cpl.template`	Specifies the style of CPL publications.
`lu_sloan.template`	Specifies the two-column style used for the proceedings of the Sloan/Lawrence conference, including capacity for a two-column wide title and abstract on the first page.
`lu_letter.template`	Supplies a starting format for business letters.
`lu_memo.template`	Supplies a starting format for memos.

Table A.7: Commands for changing type style. *Note*: Some of these commands may exhibit unexpected behaviors when used in math mode and one of them (`\sc`) is invalid in math mode.

Language	Command	Function	Sample
TEX	`{\rm` *text*`}`	Sets *text* in medium, Roman, upright type.	Sample
TEX	`{\sl` *text*`}`	Sets *text* in medium, Roman, slanted type.	*Sample*
TEX	`{\it` *text*`}`	Sets *text* in medium, Roman, italic type.	*Sample*
TEX	`{\tt` *text*`}`	Sets *text* in medium, typewriter, upright type.	`Sample`
TEX	`{\bf` *text*`}`	Sets *text* in bold, Roman, upright type.	**Sample**
TEX	`{\sc` *text*`}`	Sets *text* in medium, Roman, small caps type.	SAMPLE
TEX	`{\sf` *text*`}`	Sets *text* in medium, sans serif, upright type.	Sample
LATEX	`{\em` *text*`}`	Sets *text* in emphasized type.	(See text.)
LATEX	`\emph{`*text*`}`	Sets *text* in emphasized type.	(See text.)

Local Guide for information about possible additional templates available at your site.

A.3 In-Text Specification of Local Style

In addition to the global issues discussed in Section A.2, documents are affected locally by many less extensive changes of style. We have already mentioned the use of a blank line to trigger a new paragraph. Commands for accomplishing other common local changes are described briefly in this section.

A.3.1 Type Style

In the terminology of type setters, the phrase *type style* refers collectively to combinations of three independent characteristics of the type. In the most sophisticated description, type style is specified by selecting the *series* (medium, bold), *family* (Roman, sans serif, typewriter), and *shape* (upright, italic, slanted, small cap) of the desired style.[23] Each characteristic is specified independently of the other two. Repeated selection of all three characteristics, however, is cumbersome, so—as enumerated in Table A.7—LATEX provides simpler ways to select the more common combinations. Some of these mechanisms make use of *declarations*,[24] which change the type style until it is explicitly changed again. If, for example, we included the declaration `\bf` at some point in our code, everything in our document from that point until we change the style with another declaration would be set in bold, Roman, upright type. To limit the scope of this declaration to some portion of the text, the text to be "emboldened", *including the declaration*, would be enclosed in curly braces (as shown in the table). Once LATEX's processing has passed beyond the point of the closing curly brace, the declaration is no longer in effect and the type style reverts to what it was before the opening curly brace.

Emphasis of a word or a phrase now and then would, in normal use, be achieved with *italic* type. In LATEX, the declaration `\it` might be used. The declaration `\em`, however, is preferred because it is sensitive to the current type style. When the current type style is Roman, `\em` will shift to italic type; when the current type style is italic, `\em` will shift to Roman type. Thus, `\em` can be used *inside* a phrase that is already being emphasized. Emphasized text inside of emphasized text that is itself embedded in Roman text will be set in Roman text to contrast with the italic style of the text that immediately surrounds it.

[23] The default is medium series, Roman family, upright shape—the type style used for this book.

[24] Because it is sometimes difficult to remember which "commands" are commands and which are actually declarations, we index both as commands.

Note also in Table A.7 that emphasized text can be achieved not only with the *declaration* `\em` but also with the *command* `\emph`. The former form is perhaps preferable for longer phrases; the latter would be used for a word or two. Each achieves the same effect.

To be especially fastidious, we should recognize that return from italic or emphasized type to other type styles may result in too little space between the last *italic* or *emphasized* letter and the first Roman (say) letter. The command `\/` inserted immediately after the last italicized or emphasized letter instructs LaTeX to insert the last letter's *italic correction*; use of this correction will usually improve the legibility of the final copy. Thus, the careful way to specify an emphasized word is, for example, `{\em and\/}`. (The italic correction is *not* necessary in *all* situations, e.g., if the following character is a period or a comma, which explains why it cannot be automatically and always inserted.)

In using one or another of these simple specifications, we are accepting particular combinations of series, family, and shape. LaTeX provides two ways to take advantage of the full flexibility with *separate* specification of these three characteristics. For example, we might use declarations to specify bold, sans serif, upright type with the construction

`{\bfseries\sffamily\upshape` *text*`}`

Alternatively, we might use the commands

`\textbf{ \textsf{ \textup{` *text* `} } }`

to achieve the same end with a shorter phrase. The full set of available declarations and commands includes

`\mdseries`, `\bfseries`, `\textmd{`...`}`, `\textbf{`...`}`

to specify medium and bold series, respectively,

`\rmfamily`, `\sffamily`, `\ttfamily`, `\textrm{`...`}`, `\textsf{`...`}`, `\texttt{`...`}`

to specify Roman, sans serif, and typewriter families, respectively, and

`\upshape`, `\itshape`, `\slshape`, `\scshape` ,

`\textup{`...`}`, `\textit{`...`}`, `\textsl{`...`}`, `\textsc{`...`}`

to specify upright, italic, slanted, and small-cap shapes. A quick return to the default specified in the originally invoked document class is accomplished with the declaration `\normalfont` or the command `\textnormal`.

A.3.2 Type Size

Type size is independent of type style. The global type size for a particular document is specified in the command `\documentclass` as described in Section A.2. A local change in type size is accomplished by one or another of the declarations in Table A.8. The selected type size is determined relative to the global type size specified in the command `\documentclass`. Again curly braces are used to limit the scope of the change in size.

To combine a change in type *style* with a change in type *size*, place the specification of type size *first*, e.g., `{\Large\bf` *text*`}` to select large bold-face type.

Table A.8: Commands for changing type size. Note that the actual size produced by each command is relative to the global type size (10 pt, 11 pt, or 12 pt) in use. Note also that, in some classes, adjacent members in this sequence may be assigned to the *same* type size.

Command	Function	Sample
`{\tiny` *text*`}`	Sets *text* in tiny type.	Sample
`{\scriptsize` *text*`}`	Sets *text* in scriptsize type.	Sample
`{\footnotesize` *text*`}`	Sets *text* in footnotesize type.	Sample
`{\small` *text*`}`	Sets *text* in small type.	Sample
`{\normalsize` *text*`}`	Sets *text* in normalsize type.	Sample
`{\large` *text*`}`	Sets *text* in large type.	Sample
`{\Large` *text*`}`	Sets *text* in Large type.	Sample
`{\LARGE` *text*`}`	Sets *text* in LARGE type.	Sample
`{\huge` *text*`}`	Sets *text* in huge type.	Sample
`{\Huge` *text*`}`	Sets *text* in Huge type.	Sample

A.3.3 White Space

Some aspects of the way the components (text, tables, graphs, ...) of a document are placed in the available space are controlled by the length parameters set in the preamble and discussed in Section A.2. Additional features of LaTeX that give control over this feature of a document include

- The command `\noindent` to suppress paragraph indentation for a single paragraph.
- The commands `\newpage` and `\clearpage` to force a new page. The first of these commands terminates the current page and starts a new page. The second *also* flushes out any accumulated tables and figures that have not yet been output.
- The length parameter `\baselineskip` to specify the linespacing. This parameter is set with the command `\setlength` described in Section A.2. The default value is 12 points, which yields single spacing. The value 18 points specifies space and a half, and the value 24 points specifies double spacing. This specification must be placed *after* the command `\begin{document}`, and it can be changed whenever appropriate to the text. See, however, Section A.3.4 for a comment about limitations of this approach to setting the linespacing.
- The commands `\quad` and `\qquad`, which insert an en space and an em space, respectively. (An en space is about the width of the letter x, and an em space is about the width of the letter M, both in the current font.)
- The command `\vspace{`*Value*`}` to generate vertical space. Here, *Value* can be any defined length parameter, e.g., `\vspace{\parskip}`, or a specific length, e.g., `\vspace{2.0truein}`. If the space happens to occur at the top of a new page, it will be omitted, but the command `\vspace*{`*Value*`}` will force the space to be included even if it is at the top of a page.
- The command `\hspace{`*Value*`}` to generate horizontal space. Here, *Value* can be any defined length parameter, e.g., `\hspace{\parindent}`, or a specific length, e.g., `\hspace{36.0pt}`. If the space happens to occur at the end of a line, it will be omitted, but the command `\hspace*{`*Value*`}` will force the space to be included even if it is at the end of a line.
- The commands `\settowidth`, `\settoheight`, and `\settodepth`, which return lengths for the corresponding dimensions of the box that will enclose whatever is the argument of the

command. No printed output is produced by these commands. We might, for example, define a new length parameter `\dblast` as the width of the box accommodating two asterisks with the two commands

```
\newlength{\dblast}
\settowidth{\dblast}{**}
```

and then insert the space for two asterisks (without actually displaying them) with the command

```
\hspace{\dblast}
```

These commands are described in detail at the end of Sections 6.4.1 and C.13.1 in *The LaTeX Manual*.

A.3.4 Environments

Sometimes, the formatting to be accomplished requires a more sophisticated definition of a change than is possible within the framework of a simple declaration or command with a few arguments. LaTeX's *environments* are more versatile and flexible than declarations and commands. For example, to present text in a narrower paragraph, as is conventionally done with extended quotations, we might invoke one of the constructions

```
\begin{quotation}
Text of quotation.
\end{quotation}
```

or

```
\begin{quote}
Text of quotation.
\end{quote}
```

Normally, the command `\begin{...}` will be placed on a line by itself, the text may spread over several lines, and the command `\end{...}` will be placed on a line by itself. Left and right margins are indented equally. Unless the text includes its own formatting specifications, however, the first construction will indent each new paragraph and will place *no* extra space between paragraphs while the second will place extra space between paragraphs and will *not* indent each new paragraph.

For a second example, if we wished to center a line or lines, we might invoke the similar construction

`\begin{center}` *First line* `\\` *Second line* `\\` *Third line* `\\` ... `\end{center}`

Normally, the command `\begin{...}` will be placed on a line by itself, the text may spread over several lines, each line in the final output will in the code be separated from its predecessor with the command `\\`, and the command `\end{...}` will be placed on a line by itself.

These two examples introduce the construction called an *environment*, which in general has the format

`\begin{`*EnvironmentName*`}` ... `\end{`*EnvironmentName*`}`

We met the `document` environment early on. Beyond the `quotation`, `quote`, and `center` environments, the most commonly used environments include `verbatim`, `flushleft`, `flushright`, `itemize`, `enumerate`, `list`, `displaymath`, `equation`, `eqnarray`, `figure`, `table`, `tabular`, `minipage`, `picture`, and `thebibliography`. Environments can be thought of as mini-documents within a document. They have default settings, but many of them possess parameters that the user can modify to customize the details of the environment. Further, as described in Sections 3.4 and C.8 in *The*

LaTeX Manual, individual users can supplement the standard set by defining additional environments suited to the circumstances of the document. Individual exercises in this book, for example, are formatted by a user-defined environment.

The length parameter `\baselineskip` interacts in interesting ways with the formatting of text contained within environments. Changing `\baselineskip` will affect only the spacing in the main text of the document, including text in `itemize`, `enumerate`, and `verbatim` environments. Footnotes, material in `tabular` environments, table and figure captions, and perhaps other components will remain single spaced unless—as described in Section C.3.2 in *The LaTeX Manual*—the parameter `\baselinestretch` is *also* changed with the command `\renewcommand`. This additional change, however, has interesting side effects. (For example, the spacing of lines within footnotes will be changed but the spacing between separate footnotes on the same page remains single!) Tampering with global style can at times have unintended (and unwanted) consequences. *Beware*!

A.3.5 LaTeX Packages

As a publishing system, LaTeX is infinitely extendable. Rather than incorporate all manner of special tools within the basic program, its designers provided a means by which additional features could be added through the use of *packages*, the features of which can be made available in any specific code by placing the command

`\usepackage{` *PackageName* `}`

in the preamble *The LaTeX Companion* describes numerous packages, including `amstex`, `babel`, `color`, `graphics`, `graphicx`, `graphpap`, `ifthen`, `latexsym`, `imakeidx`, `verbatim`, `pict2e`, and `showidx`. A brief description of many of these packages is included in Section C.5.2 of *The LaTeX Manual*. The statement `texdoc` *PackageName* at a *Shell* prompt also will bring up documentation on several of these packages.

In addition, some programs, especially those with notebook capabilities, have the ability to write the contents of the notebooks into LaTeX source files. Usually, those programs make use of their own special LaTeX commands and supply program-specific LaTeX packages which must be accessible, either at the proper point in the LaTeX directory structure or in the directory from which you run LaTeX when processing a LaTeX file produced by the programs. Details on those additional packages will be found in the manuals provided by the vendors of the programs.

A.3.6 Miscellaneous Other Capabilities

Several additional capabilities merit particular comment:

- To force LaTeX to keep words together on a single line and/or to prevent LaTeX from inserting additional space as it justifies a line, use a tilde `~` instead of a space between the words. The tilde should also be used after a period that does *not* end a sentence. For example, we should type '`Mr.~Cook`' rather than[25] '`Mr.␣Cook`' in our file so as to produce 'Mr. Cook' rather than 'Mr. Cook' in our output. In the second form, there is more space between the abbreviation and the name because LaTeX automatically inserts extra space after each period (which is assumed to mark the end of a sentence). LaTeX may add even more space in the second case as it justifies the line between the margins.
- To generate a dash, use `-`, `--`, or `---`, depending on the length of the required dash. One hyphen produces a hyphen, as in two-column; two hyphens in a row produce a slightly longer

[25] The symbol '␣' underscores the presence of a space at the indicated point; it is *not* a character explicitly present in the code.

dash (an en dash in printers' terminology), conventionally used to indicate ranges, e.g., 10–12; three hyphens in a row produce a still longer line (—, an em dash in printers' terminology), sometimes called a *punctuation* dash and conventionally used in pairs as an alternative to parentheses. Note that *none* of these constructions is a minus sign −, which has yet a different length and is produced only in math mode. (See Section A.4.)

- To generate opening and closing (double) quotation marks, use `` and '', i.e., two opening or closing (single) tics in a row, respectively. Do *not* use the symbol " for *either* of these punctuation marks.
- To specify a footnote, use the command `\footnote{`*Text of footnote.*`}`. The specified text will be placed at the bottom of the page and keyed to the text with an automatically generated number that starts with 1 at the beginning of the document.
- To start a new section, use the command `\section{`*Title of Section*`}`. In the `article` class, section numbers start with 1 at the beginning of the document and are automatically generated and incremented; the section number is included in the printed section title, which is displayed in a type size and font specified in the document class, and indentation will be suppressed in the *first* paragraph of each new section. [The commands `\subsection` and `\subsubsection` function in a similar way for subsections and subsubsections in the document. In the `book` class, the commands `\chapter` and `\part` (an aggregation of chapters) are also available.]
- To control hyphenation, insert the command `\-` at the point in a word where LaTeX is permitted to insert a hyphen as it fills and justifies lines. LaTeX has its own rules for hyphenation, but they are not infallible. Sometimes LaTeX needs help. Note that words containing even one indication of an optional hyphen will not be hyphenated at any other point(s) in the word.
- To suppress hyphenation of a single word altogether, place the word as the argument of an `\mbox` command, e.g., `\mbox{customize}`.

A.4 Including Equations

Most scientific documents will have equations, all of which must be specified in LaTeX's math mode. Shifting from the default text mode to math mode can be accomplished in several ways. For short, unnumbered equations or mathematical symbols that appear *in a line of text*, the equation or symbol must be enclosed in one of the two constructions `\(...\)` or `$...$`, which are equivalent. If we want one or more *displayed* equations, with or without automatically generated equation numbers, we need to use the form `\begin...\end`. Three standard environments use this construction. The `displaymath` environment centers a single equation on a line by itself, does not number the equation, and can be specified by the shortened form `\[...\]`. The `equation` environment centers a single equation on a line by itself and numbers the equation automatically. The `eqnarray` environment—see *The LaTeX Manual*—is used for a string of equations and for long equations that will not fit on one line; its syntax allows vertical alignment of two or more displayed and numbered equations. To facilitate references to numbered equations produced in the `equation` and `eqnarray` environments, LaTeX provides the command `\label{`*ReferenceName*`}` for defining a symbolic label *within* the environment and the command `\ref{`*ReferenceName*`}` to permit referencing the equation by its number in the text. Further, the command `\pageref{`*ReferenceName*`}` permits referencing the *page* on which the equation occurs in the text.[26] (*Warning*: The `displaymath`, `equation`, and `eqnarray` environments will generate error messages if they contain blank lines.)

[26] When these references are used, the information written into the `.aux` file becomes important and the code must be processed by `latex` or `pdflatex` *twice*, once to write the correct information into that file and a second time to read and make use of that information. Other environments, such as `table` and `figure` also admit a `\label` command and can make use of this capability for symbolic internal references to those components. The label command can also be used inside the argument of `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, and `\footnote` and in the text following an `\item` command in an `enumerate` environment to facilitate reference to these components of the document.

LaTeX also has many symbols and structures, common in mathematical formulas, that can be used only in math mode. Subscripts and superscripts, roots, Greek letters, integral signs, summation signs, the `array` environment, and many more are available.[27] To reiterate, these symbols and structures can only be used if LaTeX is in math mode. So, to insert the symbol Θ in the final output, we place the command `$\Theta$` in the code.[28] (Here, the first dollar sign puts LaTeX in undisplayed math mode and the second returns LaTeX to text mode.) Since LaTeX follows its own rules about spacing when in math mode, spaces in math mode are usually ignored. (A space, however, is necessary in such contexts as `\omega t` to separate the command `\omega` from the character `t`.) The commands `\,`, `\!`, `\:`, and `\;` can be used in math mode to fine tune the spaces between elements in the equation by adding a thin space, a negative thin space, a medium space, and a thick space, respectively. It is, for example, customary to separate a differential element from what precedes and follows with a thin space, e.g., $(x + y)\, dx\, dy$ rather than $(x + y)dxdy$.[29]

Following the conventions for the type setting of mathematical text, all symbols in a math environment are automatically italicized. The most common departure from this convention occurs with the names of the standard mathematical functions, which are conventionally set in Roman type. Special math mode commands of the form `\cos` and `\sin` should be used for these functions.[30] Except for `\sc`, the declarations shown in Table A.7 can also be used in math mode to specify text in something other than italic type but, in some contexts it may be preferable to use one of the commands `\mathrm`, `\mathit`, `\mathbf`, `\mathtt`, etc. described in Section 3.3.8 of *The LaTeX Manual*. Remember, however, that spaces are ignored in math mode, so a shift to Roman type with the command `\mathrm`, for example, to include a *two*-word phrase, will result in the words running together unless the *space* command `\␣` is used to insert an *explicit* space between the words. Sometimes, the better means to insert Roman text in equations is to use the command `\mbox` to escape temporarily to text mode.

Including bold characters in *equations* is especially tricky. The command `\mathbf` "emboldens" only *some* characters in the processed formula and, in particular, leaves lower case Greek symbols in light face. The `\boldmath` declaration causes *everything* in math mode to be bold but cannot itself be *invoked* in math mode. To include a bold face, lower case θ, for example, we must use the construction

```
$ ... \mbox{ \boldmath $\theta$ } ... $
```

where the command `\mbox`, which *is* valid in math mode, temporarily shifts to text mode so the `\boldmath` declaration can be invoked and the math mode expression in the argument of the command `\mbox` will then be made bold; the bold presentation is confined by the command `\mbox` to the desired part of the total expression.

A.5 Including Lists

Among the most commonly used environments are those that facilitate creation of lists, including the `itemize` environment for bulleted lists (such as the one in Section A.3.6) and the `enumerate` environment for lists in which the items are lettered or numbered in sequence (such as the one in Section A.17). Each environment is introduced and terminated with the standard construction `\begin` ... `\end`. Within the environment, each new item is introduced with the command `\item`. Thus, the structure might look like

[27] For a complete list of all symbols and the commands that create them, see Section 3.3 on Mathematical Formulas in *The LaTeX Manual*.

[28] In math mode, Greek letters can be produced by commands that simply name the letter, e.g., `\Theta` for Θ and `\omega` for ω. Only the initial letter of the name is capitalized to produce an upper-case letter. Note, however, that commands for letters that are identical to Arabic letters, e.g., `\Kappa`, do not exist.

[29] Technically and officially, the differential element dx should be written with a *Roman* d and an italic x, i.e., $\mathrm{d}x$, but this convention is rarely followed.

[30] See Table 3.9 in *The LaTeX Manual* for a complete listing.

```
\begin{...}
\item Text of first item.
\item Text of second item.
\item Text of third item.
\end{...}
```

The bullets and the numbers are, of course, generated automatically and, within the `enumerate` environment, the numbers are automatically incremented. Details will be found in Sections 2.2.4, 6.6, C.6.2, and C.6.3 in *The LaTeX Manual*. Note in particular

- The optional argument for the command `\item`, which provides a means to override the automatic label for that item and replace it with an explicit stipulated label. For example, the command `\item[DMC]` will cause the item to be labeled 'DMC' and the command `\item[]` will suppress the label altogether. Note that, if the text of the item itself begins with something enclosed in square brackets, the command introducing that item must be written `\item{}` to prevent the text in square brackets from being interpreted as the desired label.
- The length parameters `\itemsep`, `\parskip`, and `\parsep`, which provide a means to override default spacings within the environment.
- The way to change the "bullets" in the `itemize` environment. The symbols used to label the "bulleted" items in the various levels of nested `itemize` environments are created by the commands `\labelitemi`, `\labelitemii`, `\labelitemiii`, and `\labelitemiv`. The default sequence for marking items at each level is •, – *, and ·. Each of these symbols can, however, be changed by redefining the corresponding command. For example, the command

  ```
  \renewcommand{\labelitemii}{$\circ$}
  ```

  executed in the preamble will change the symbol – to ∘ for the second level in nested `itemize` environments.
- The way to change the form of the labels for each item in the `enumerate` environment. The labels at the various levels of nested `enumerate` environments are determined by invoking one of the commands `\theenumi`, `\theenumii`, `\theenumiii`, and `\theenumiv`, whose action is in turn determined from the value of the corresponding one of the counters `enumi`, `enumii`, `enumiii`, and `enumiv`. The default sequence for marking items at each level is '1.', '(a)', 'i.', 'A.', i.e., arabic number with period, lower-case letter in parentheses, lower-case Roman number with period, and upper-case letter with period. These defaults, however, can be changed with a command like

  ```
  \renewcommand{\theenumi}{(\Alph{enumi})}
  ```

  which will change the labels used at the first level in `enumerate` environments to upper-case arabic letters in parentheses. The command `\Alph` could be replaced with `\alph`, `\roman`, `\Roman`, or `\arabic` to translate the underlying counter into lower-case arabic letters, lower-case Roman numbers, upper-case Roman numbers, or arabic numbers, respectively. As it turns out, whatever is specified in the argument of the commands translating the counters, a period will be appended by LaTeX at the first, third, and fourth levels, and enclosing parentheses will be supplied at the second level.

A.6 Including Tables

Creation of columnar arrangements is facilitated by LaTeX's `tabular` environment, the details of which are involved and are fully described in *The LaTeX Manual*. The `tabular` environment can

Table A.9: Format for inserting a table.

```
\begin{table}
  \caption{ ... }
  \label{ ... }
  \begin{center}
    \begin{tabular}{ clr }
       Row 1, Col 1 & Row 1, Col 2 & Row 1, Col 3 \\
       Row 2, Col 1 & Row 2, Col 2 & Row 2, Col 3 \\
       Row 3, Col 1 & Row 3, Col 2 & Row 3, Col 3
    \end{tabular}
  \end{center}
\end{table}
```

be invoked anywhere in the code and the resulting table will be placed at the point at which the environment appears, perhaps even in the middle of a line of text. Conventionally, however, tables are placed at the top or bottom of the page,[31] and LaTeX provides the `table` environment to facilitate proper placement of a table,[32] though the bare `table` environment contains nothing other than its name that suggests its use for tables. Any text at all can be incorporated in the `table` environment and will be placed on the page as would a table. Most commonly, however, the `table` environment will embrace a `tabular` environment defining the table itself and will use the command `\caption` to specify the caption of the table and the command `\label` to specify a symbolic label for use in referring to the table within the document.[33] Frequently, the `tabular` environment will itself be embraced in a `center` environment to control the horizontal placement of the table on the page. Thus, the common structure for the code defining a table would be of the form shown in Table A.9.[34,35]

Here, individual elements in a row are separated from one another by ampersands, and the end of each row—except the last—is marked with the command `\\`. In this example, the caption, including an automatically generated phrase and number of the form 'Table 3:', will appear *above* the table. Positioning the commands `\caption` and `\label` *after* the `center` environment would place the caption *below* the table. Whichever position is adopted, the command `\label` must appear *after* the command `\caption`. Wherever the command `\label` is positioned, it will place an entry in the `.aux` file so that the commands `\ref` and `\pageref` will function here as described for equations at the beginning of Section A.4.

The illustrative argument `clr` of the `tabular` environment requires a bit more explanation.

[31] Actually, the command `\begin{table}` has an optional argument (`\begin{table}[`*OptArg*`]`), which can assume any of the values `h`, `b`, or `t` for placement of the table at the place where the `table` environment appears (`h`, for here) or at the bottom (`b`) or top (`t`) of the page. In the absence of an explicit specification, LaTeX makes its own decision about placement.

[32] If there is space on the current page when the `table` environment is encountered, the table will be placed on that page. Otherwise, the table will be placed on a subsequent page.

[33] The `\caption` and `\label` commands can be placed anywhere within the `table` environment. Captions will commonly be placed either above or below the captioned component. In some contexts (see the warning at the end of Section A.10), placing the caption above the item is preferable.

[34] LaTeX automatically uses the specified caption as an entry for a list of tables. There is, however, a limit to the length of such entries. Especially long captions may generate error messages. To avoid this problem, we can—and sometimes *must*—exploit an optional argument to the command `\caption{`...`}` which allows the user to dictate the entry made to the list of tables. Unless a list of tables is to be generated, some authors argue that we should routinely specify a null value for the optional argument by writing the command `\caption[]{`...`}`, though we shall here not follow that recommendation. See *The LaTeX Manual* for details.

[35] See Sections 3.6 and C.10 in *The LaTeX Manual* for *much* more detail on ways to line things up in columns. Note particularly (1) the command `\multicolumn`, which provides for entries that span more than one column, and (2) ways to create horizontal and vertical rulings in the table.

In general, this argument will be a string of `c`'s, `l`'s and `r`'s, one for each column in the table. Each specifies the position (centered, left justified, right justified), respectively, of the entry in the corresponding column. In the example, there are three columns, with entries in the first column centered, entries in the second column left justified, and entries in the third column right justified. The argument is mandatory but will, of course, be a string appropriate to the table being constructed rather than the specific string `clr`.

By default, tables will be produced with no vertical or horizontal rulings. See Section C.10.2 in *The LaTeX Manual* for a description of the means to add these rulings.

A.7 Including Illustrations

Graphics displays frequently appear in technical documents. Within LaTeX, means to incorporate graphics include[36]

- A cut and paste method (Section A.7.1).
- A method for incorporating a properly constructed PostScript or PDF file that invokes the command `\includegraphics` from the package `graphicx` (Section A.7.2).
- A method that exploits the `tikz` package of macros that facilitate describing the graphical display and its formatting in the same way that LaTeX itself facilitates describing the text and its formatting (Section A.7.3).
- The built-in `picture` environment (Section A.7.4), which has limited capability but is sometimes just the ticket for simple displays.

Most often, commands incorporating figures will be bracketed in a `figure` environment, which facilitates locating the illustration at the top or bottom of the current (or a following) page.[37] In most cases, we need to know the final vertical extent of the figure in order to specify the size of an appropriate space. Further, the commands `\caption` and `\label` are available to caption the figure and to place appropriate entries in the `.aux` file so that the commands `\ref` and `\pageref` will function here as described in the first paragraph of Section A.4.

A.7.1 Using Cut and Paste

The cut and paste method requires only that LaTeX be instructed to set aside appropriately sized and positioned space into which the illustration will be placed after the document has been printed. The segment of the code that creates and labels space for a figure will have the general form

```
\begin{figure}
  \caption{ ... }
  \label{ ... }
  \vspace{ ??.?truein }
\end{figure}
```

As illustrated here, the caption, including an automatically generated phrase and number of the form 'Figure 3:', will appear *above* the space left for the figure. Positioning the commands `\caption` and `\label` *after* the command `\vspace` would place the caption *below* the figure. Note also that,

[36] We elect here to discuss only a very few of the numerous available packages, choosing those of the broadest applicability. Packages for drawing Feynman diagrams, organic molecules, musical scores, and many other displays are described in *The LaTeX Graphics Companion*.

[37] The `figure` environment also admits the optional argument described in footnote 31.

whichever position is adopted, the command `\label` must appear *after* the command `\caption`. For this method, you need only have printed copies of your figures.[38] The format of the files used for the figures is irrelevant and the document can be produced with any of the methods described in Sections A.1.2 and A.1.3.

A.7.2 Using the `graphicx` Package

Incorporation of graphic images in a LaTeX document is viewed by LaTeX as the responsibility of the device driver. To enable that process, LaTeX allows the passing of commands to the device driver. Both the format of such commands and the variety of capabilities thus available depends entirely on the device driver. In essence, this process uses the TeX (not LaTeX) command `\special`, but the user is unaware of that underlying command because it is invoked behind the scenes.

The simplest means for importing graphical displays into a LaTeX document—i.e., of commanding LaTeX to pass the proper information on to the device driver—exploits the `graphicx` package,[39] whose features are made available by including that package with the statement

```
\usepackage{graphicx}
```

in the preamble of our LaTeX code.[40]

After placing the command `\usepackage{graphicx}` in our preamble, we incorporate the graphical display itself by placing a statement like

```
\includegraphics[height=dimension]{filename}
```

at the point where the PostScript or PDF figure defined in by the file *filename* is to be inserted.[41] In general, the image would be centered horizontally on the page by inserting this command in a `center` environment, e.g.,

```
\begin{center}
\includegraphics[height=dimension]{filename}
\end{center}
```

For example, the code[42]

[38] Appropriately translated, the substance of footnote 34 applies here to the entry placed automatically in the list of figures.

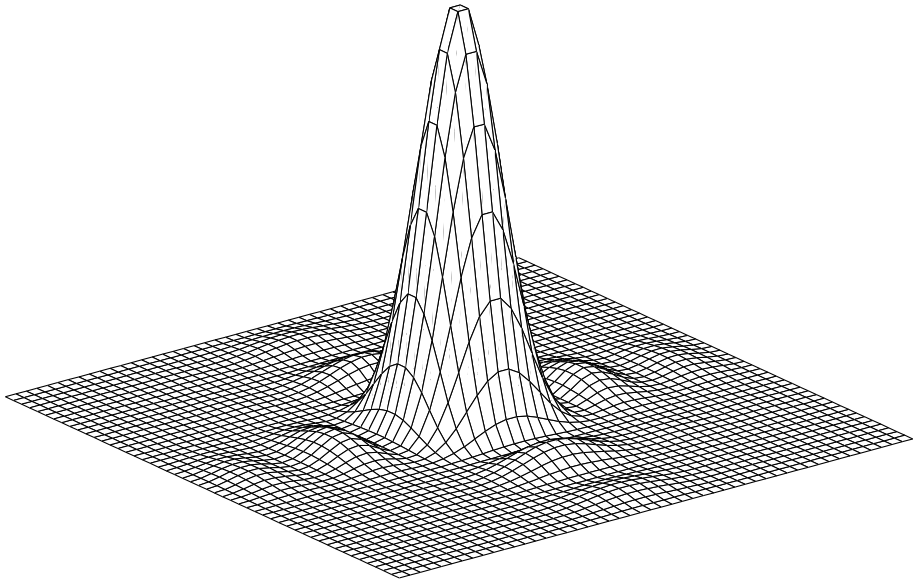
[39] The `graphics` package might also be used, but the syntax of the commands it defines is marginally less convenient than the syntax of the parallel commands in the `graphicx` package, i.e. the *extended* `graphics` package.

[40] More generally, this statement admits an optional argument to the desired graphics driver explicitly. Frequently, that driver will be `dvips` and the statement would be `\usepackage[dvips]{graphicx}`, but numerous other drivers exist, and many may be installed at your site. See the *Local Guide* for more information. For ultimate flexibility in the formatting of graphics files, we elect to leave that choice to the default.

[41] If you use `latex`, `dvips`, and `ps2pdf` as described in Section A.1.2 and item 2 in Section A.1.3 to produce PostScript and PDF documents, your figures must be available as `.ps` or `.eps` files, and—even if the file type is omitted in the `\includegraphics` command—the processing will correctly identify and incorporate the figures. If you seek to produce a PDF document directly with `pdflatex` as described in item 1 in Section A.1.3, then—even if the file type `.pdf` is omitted in the`\includegraphics` command—the processing will correctly identify and incorporate the figures. If your figures are PostScript, you must use the first route to the final document; if your figures are PDF, you must use the second route. Software to convert PostScript figures to PDF figures and *vice versa* is described in Section A.11. If both PostScript and PDF files exist for all figures *and* you omit the file type in the `\includegraphics` command, then you can use either `latex` or `pdflatex` for the processing.

[42] Appropriately translated, the substance of footnote 34 applies here to the entry in the list of figures.

Figure A.1: Mesh surface representation of the irradiance produced by a square aperture.



```
\begin{figure}
  \caption{Mesh surface representation of the irradiance
  produced by a square aperture.}
  \label{LATEX:irrad}
  \begin{center}
  \includegraphics[height=3.0truein]{diffract}
  \end{center}
\end{figure}
```

will produce Fig. A.1 if the file `diffract` is stored in the current default directory.[43,44] Here, the specification of the parameter `height` in the optional argument of the command `\includegraphics` causes the figure to be scaled to the specified dimension vertically and scaled by the same fraction horizontally to preserve its aspect ratio. As in the example in Section A.7.1, the figure here has been inserted *before* the command `\caption`, so that the caption will appear *above* the figure. Further, the command `\label`, which must *follow* the command `\caption`, defines a symbolic label that can be used within the document to refer to the figure.

Warning for OCTAVE users: All graphics toolkits in OCTAVE will produce proper on-screen graphs and will output PostScript files of these graphs that can be displayed with ghostview. If, however, the graph is to be incorporated in a LaTeX document by the procedures described above, graphs produced by all toolkits will be properly incorporated in the `.dvi` file but those produced by the `qt` toolkit *may* not then translate with `dvips` to the subsequent PostScript file. When the latter is the objective, use `gnuplot` (or maybe `fltk`). A bit of testing may be necessary.

The command `\includegraphics` admits several optional arguments. The argument `width` can also be included. If `width` is specified *instead of* `height`, the height will be scaled to preserve

[43] Omission of the file type in the statement including the graph prepares the way to use either `latex-dvips` to produce a PostScript file using `latex` and `dvips` (Section A.1.2), in which case the file `diffract.ps` or `diffract.eps` will be used, or `pdflatex` to produce a PDF file (Section A.1.3), in which case the file `diffract.pdf` will be used. The appropriate file must, of course, be accessible in either case.

[44] The label following the word 'Figure' in the output will, of course, reflect the document class in use and the unit of the document in which the figure appears.

the aspect ratio. If *both* `height` and `width` are specified, each will be respected and the aspect ratio of the figure may be distorted; specification of *neither* will cause the display to be produced in its original size. Arguments that allow overriding of the bounding box specified in the PostScript file and arguments that permit clipping a portion of a fuller figure are described in *The LaTeX Graphics Companion*.

A.7.3 Using the `tikz` Package

The `tikz` package, which is routinely included in present-day LaTeX distributions, provides an assortment of LaTeX macros to facilitate the drawing of simple—and even complicated—figures. Basically, the statement `\usepackage{tikz}` in the preamble followed in the document itself by statements embedded in a `tikzpicture` environment will specify the desired graph and incorporate the graph in a PDF or Postscript file created as described in Sections A.1.2 and A.1.3.[45] Be aware that the code inserted in the `tikzpicture` environment describes the picture in the same way that LaTeX provides the text and its formatting. Just as one imagines the final formatted document as the LaTeX coding is constructed, so one has to imagine the resulting picture as one constructs the descriptive code to create it in the final document. `Tikz` is *not* a WYSIWIG drawing program like `PAINT` or `TGIF`. One of its advantages over the route described in Section A.7.2, which imports figures produced in other programs, is that the route of this section guarantees that the fonts used in the figures will be identical to those used in the rest of the document.

The graphics system provided by `tikz` is impressively elaborate and versatile. As such, learning its capabilities, especially its more sophisticated capabilities, will require substantial effort. Typing the command[46]

```
texdoc pgf
```

at a *Shell* window to your operating system will probably bring up links to a number of manuals, including the main—and voluminous (1100-plus pages!)—manual in the file `pgfmanual.pdf`. Further, googling `tikz` will bring up links to numerous documents, including in particular a link to `pgfmanual.pdf` and a link to the much more compact tutorial-style introduction in the file `minimaltikz.pdf`. Once you have selected any of these items, you may in some operating systems have to look in your Downloads folder to access the item.

This section quickly orients you to the general features of `tikz` without in any way pretending to be complete. To that end, having included the statement[47]

```
\usepackage{tikz}
```

in the preamble of a document, we produce Fig. A.2 by placing the coding in Table A.10 at the appropriate point in the document itself. This coding briefly illustrates some of the more elementary capabilities of `tigz`. Note specifically the following:

- Optional arguments to any `tikz` command are included in (square) brackets.
- By default, specified coordinates are expressed in centimeters, though any recognized LaTeX unit of length can be specifically stipulated. The optional argument `[x=1.0in,y=1.0in]` following `\begin{tikzpicture}` changes the default to inches. Once a figure has been defined, its size can be easily altered simply by changing this optional argument.

[45] Note that, when displayed on the screen with `xdvi` or `yap`—see Section A.1.4—the intermediate `.dvi` file produced in Section A.1.2 will *not* contain the correct figures.

[46] The designation `pgf`—Portable Graphics Format—reflects the name of the engine underlying the entire system.

[47] Some installations may also require inclusion of the packages `pgf` and `xcolor`.

- *Every* statement *must* end with a semicolon. Any statement can be spread over several lines with only the last line so terminated.

- The statement `\draw [fill=black] (0,0) circle [radius=0.1];` draws a black-filled circle centered at the origin and having radius 0.1 inches, thus marking the origin of the coordinate system.[48] Available colors include red, green, blue, cyan, magenta, yellow, and many others. You can also define your own color using a command like[49]

  `\definecolor{`*ColorName*`}{rgb}{`r, g, b`}`

  where *ColorName* is the name you assign to the color and r, g, b specify the r, g, and b components that make up the color. Available shapes include `rectangle`, `ellipse`, and `arc`.[50]

- The statement `\draw [<->, line width=2] (0,1.2)--(0,0)--(2.3,0);` draws a line connecting the specified points in the order given, i.e., draws the axes intersecting at the origin. Any number of points can be included in the path. The optional argument `<->` places an arrow at both ends of the line,[51] and the argument `line width` species the weight of the line, by default, in points.[52]

- Text is placed where specified with the statements `\node at (0,1.3) {\Large\bf y};` and `\node at (2.4,0.0) {\Large\bf x};`. Note that LaTeX stipulations of font size and style are recognized by `tikz`. Note also that the text is, by default, *centered* at the specified point. See the TIKZ manuals for ways to override that default positioning.

- The statement `\draw [ultra thick, domain=0:2.0] plot (\x, {sin(pi*\x r)} );` illustrates how to draw a smooth curve defined by a function, many of which are available within `tikz`. Note also the availability of the irrational number π with the simple coding `pi`.[53]

- The construction

```
\foreach \x in {0.0,1.0,2.0} {
\draw (\x,0.1 )--(\x,-0.1);
\node at (\x, -0.4){\x};};
```

  creates a loop executing the statements enclosed in `{...}` for each value in the list following the keyword `in`. The two loops in this code place tick marks on the axes and label those marks. Note the semicolons in this construction.

Clearly the command `\draw` is a versatile command that admits numerous embellishments through the use of keywords and/or optional arguments. Conveniently, `tikz` determines the size of the space to be used in the document to reflect the size of the picture as constructed with the measures provided in the `tikz` coding.

When features of the package `tikz` are invoked in the LaTeX code, producing the formatted document must be done carefully. If there are no complications (external figures, table of contents, internal references) to be incorporated, the simple statement

[48] Note that the default unit—here inches—for radius can be overridden with an explicit stipulation, e.g., `[radius=0.1cm]`.

[49] The command `\definecolor` here used is defined within the `tikzpicture` environment and does not require use of the LaTeX package `color`.

[50] See the manuals for the details of how to specify the dimensions of these shapes.

[51] The separate arguments `->` and `<-` will place an arrow at the end or the beginning of the line, respectively.

[52] The argument `line width=2` could be replaced, among others, by `ultra thin`, `very thin`, `thin`, `thick`, `very thick`, and `ultra thick`, and the default unit could be overridden with an argument like `line width=0.1cm`.

[53] Another way to draw smooth curves is to use a calculational aid in another program to generate a probably long list of coordinates for many points on the curve and then to insert that list as the argument of a `\draw` command.

Figure A.2: An illustrative `tikz` figure. Note that specified colors will be converted to a gray scale unless the display device—screen or printer—can display colors.

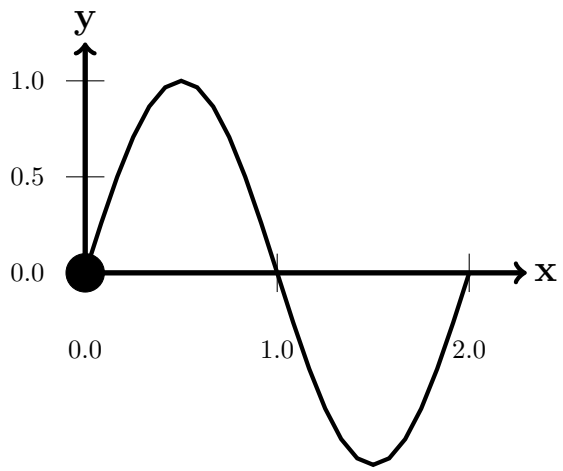


Table A.10: Coding to produce Fig. A.2.

```
\begin{figure}
\caption{An illustrative {\tt tikz} figure.}
\label{LATEX:tigzfig}
\begin{center}
\begin{tikzpicture}[x=1.0in,y=1.0in]
\draw [fill=black] (0,0) circle [radius=0.1];
\draw [<->, line width=2] (0,1.2)--(0,0)--(2.3,0);
\node at (0,1.3) {\Large\bf y};
\node at (2.4,0.0) {\Large\bf x};
\draw [ultra thick, domain=0:2.0] plot (\x, {sin(pi*\x r)} );
\foreach \x in {0.0,1.0,2.0} {
  \draw (\x,0.1 )--(\x,-0.1);
  \node at (\x, -0.4){\x};};
\foreach \y in {0.0,0.5,1.0} {
  \draw (-0.1,\y)--(0.1,\y);
  \node at (-0.3, \y) {\y};};
\end{tikzpicture}
\end{center}
\end{figure}
```

`pdflatex` *filename* (Default file type `.tex`)

will produce a PDF file and the statements

`latex` *filename* (Default file type `.tex`)
`dvips -o` *filename*`.ps -t letter` *filename* (Default file type `.dvi`)
`ps2pdf` *filename*`.ps`

will produce a PostScript and then a PDF file, though figures defined by `tikz` will not be properly

rendered in the intermediate `.dvi` file. If a table of contents or internal references or both are involved, two—and maybe three–passes through `latex` or `pdflatex` will be necessary. If there are PostScript figures and *no* PDF figures to be incorporated, only `latex` will work; if there are PDF figures (and *no* PostScript figures) or hyperlinks to be incorporated, only `pdflatex` will work. Finally, if there are both PostScript and PDF files defining figures, files of one type will have to be converted to the other type before either of these routes to a finished document will work.[54]

If you wish to short circuit learning detailed `tikz` code, you might wish to explore a WYSIWYG editor `TikzEdt` that provides a graphical interface for creating figures and translates that figure into the corresponding `tikz` code. This program, which is free, can be downloaded from the web site `www.tikzedt.org`. Documentation is also available at that site.

This section provides a woefully incomplete introduction intended more to wet your appetite than to make you an expert. The effort invested to study the manuals identified in the second paragraph of this section will be richly rewarded.

A.7.4 Using the `picture` Environment and `pict2e` Package

Finally, we merely mention the `picture` environment, described in Sections 7.1 and C.14.1 of *The LaTeX Manual*, and the supplementary `pict2e` package, described in *The LaTeX Companion*. These components add several commands that facilitate the detailed construction of at least simple graphical displays.

A.8 Including a Table of Contents, a List of Figures, and a List of Tables

A table of contents, a list of figures, and a list of tables can be included in the formatted document by the commands `\tableofcontents`, `\listoffigures`, and `\listoftables`. Each

- results in the output of an auxiliary ASCII file of information with file type `.toc`, `.lof`, and `.lot`, respectively,
- requires a second pass through `latex` or `pdflatex` to incorporate the list in the finished document, and
- places the list in the output at the point at which the command appears.

When these commands are invoked, each of the components of the text defined by the commands `\part`, `\chapter`, `\section`, `\subsection`, and `\subsubsection` will automatically generate a line in the `.toc` file;[55] each use of the `\caption` command in a `figure` or `table` environment will automatically generate a line in the `.lof` or `.lot` file. Then, in the second pass—which will always be necessary—through `latex` or `pdflatex`, these files are read and each line results in an entry in the corresponding list.

Two commands allow explicit entry of information into one or another of these files at the point where the command is inserted in the LaTeX source file, specifically

- the command `\addcontentsline{`*file*`}{`*unit*`}{`*entry*`}`
- the command `\addtocontents{`*file*`}{`*text*`}`

[54] See Section A.11 for details about that conversion.

[55] Note that not all of these options are available in all document classes. For example, `\part` and `\chapter` are available only in the `book` class. Note also the counters `secnumdepth` and `tocdepth` that control the depth to which sections are numbered and the depth to which sections are catalogued in the table of contents. The default values of these counters can be overridden using `\setcounter` in the preamble.

Here, *file* is one of `toc`, `lof`, and `lot`, *unit* is one of `part`, `chapter`, `section`, `subsection`, or `subsubsection` if *file* is `toc`, `figure` if *file* is `lof`, and `table` if *file* is `lot`.

These two commands have quite different effects. The first adds a *bona fide* entry to the corresponding file. For example, the line

```
\addcontentsline{toc}{subsection}{Specially marked point}
```

will add to the table of contents an entry that is left-justified at the subsection level, contains the text "Specially marked point", and includes a row of dots and the appropriate page number.[56] The second merely adds text—no dots or page number—to the corresponding file. For example, the line

```
\addtocontents{toc}{\vspace{24pt}This is a note.\vspace{24pt}}
```

inserts a note preceded and followed by a bit of extra vertical space. Any textual note is left-justified as dictated by the section level at which it appears in the document.

Additional details about the issues discussed in this section are laid out in Sections 4.1 and C.4 of *The LaTeX Manual*. Note, in particular, the optional argument in the command `\caption` and the several sectioning commands (`\chapter`, `\section`, ...), which provide control over the entry each command places in the table of contents, list of figures, and list of tables.

A.9 Including an Index

Among many capabilities, LaTeX is able to generate an index, though not automatically. In essence, we must place the commands[57]

```
\usepackage{imakeidx}
\makeindex
```

in the preamble and the command

```
\printindex
```

at the point in the code at which the index is to be printed—usually the very end. Then, we indicate what items are to be indexed by placing commands like

`\index{`*Item to be indexed*`}`

at appropriate points throughout the document. (The detailed structure of the argument to the command `\index` is described in Section 4.5 and Appendix A of *The LaTeX Manual*.) When the file *filename*`.tex` is processed by `latex` or `pdflatex`, the presence of these commands results in the addition of three auxiliary ASCII files named

- *filename*`.idx` containing one line for each index entry,
- *filename*`.ilg` containing a log of messages created when, behind the scenes, the idx file is automatically processed through `makeindex`, and

[56] If you are creating a linked document, this entry in the table of contents will also appear in the navigation panel of that document.

[57] Earlier versions of LaTeX used the package `makeidx`, which is still available. We here recommend using `imakeidx` because it makes available a few more features than are available in the previous package and it eliminates the need to run an auxiliary program to format the index.

- *filename*`.ind` produced by `makeindex` and containing the formatted index that is then read by `\printindex` to create the actual index in the document.

With `imakeidx`, the index is automatically created and incorporated in the document because an execution of `makeindex` is inserted automatically at the appropriate point when `latex` or `pdflatex` is run. Errors and warnings in that step are compiled in the `.ilg` file (which should be examined because on-screen display of those glitches may well scroll by to quickly to be comprehended).[58]

The above-described process yields an index in the default format. Numerous options can be exploited by adding optional arguments in the `\makeindex` command. For example, the command

```
\makeindex [options=-s ind_style, intoc]
```

stipulates that the style defined by the file `ind_style.ist` should be invoked and the index should be provided—argument `intoc`—with an entry in the table of contents. The construction of the style file is fully described in Section 12.4 in *The LaTeX Companion* as identified in Section A.18.

To assist in constructing the index, an auxiliary package called `showidx`, which works both with `latex` and with `pdflatex`, can be invoked to print the index entries on each page as marginal notes. This package is invoked simply by including the command

```
\usepackage{showidx}
```

in the preamble, though it appears as if this command must be inserted *after* the `\makeindex` command described above. Unfortunately, when printing in a 6"×9" area on 8.5"×11" paper (`letterpaper`), the marginal insertions of index entries will extend outside the edges of the page and, in `.ps` and `.pdf` files, the portions that are off the page will not be displayed in `ghostview` or `acroread`. Conveniently, if the `.dvi` file is displayed on the screen with `xdvi` (UNIX) or `yap` (Windows), those marginal notes will be visible in their entirety, even if they extend beyond the boundaries of the page, though that route is available only if figures are provided in `.ps` or `.eps` format. For the purposes of checking the index entries, one workaround is to shrink the area of the page in which printing of the text is allowed. For example, for this document, replacing the commands

```
\setlength{\textwidth}{6.0truein}
\setlength{\oddsidemargin}{0.5truein}
\setlength{\evensidemargin}{0.0truein}
```

in the preamble with the commands

```
\setlength{\textwidth}{4.0truein}
\setlength{\oddsidemargin}{1.0truein}
\setlength{\evensidemargin}{1.0truein}
```

will shrink the text width and reset the margins so there will be space for the marginal notes. To be sure, the pagination of the text will also change, but at least the marginal notes will be visible, even in the `.ps` and `.pdf` files. In different situations, tampering with the above illustrated length parameters and, perhaps also, tampering with the length parameters `\marginparwidth` (which sets the width of the marginal boxes) and `\marginparsep` (which sets the space between the text and the boxes) as well as seeking ways to shrink the entire output including marginal notes on each page as a unit may yield fruit.[59]

[58] With `makeidx`, a separate explicit processing of the `.idx` file with `makeindex` to produce the `.ind` file and a further pass through `latex` or `pdflatex` was necessary to include the index in the document.

[59] For example, the `-x` option to `dvips` can scale the size of each page output to the `.ps` file.

A.10 Including Hyperlinks

Properly constructed, a LaTeX source file can be used to create a PDF document that includes a navigation panel and hyperlinked references for easy viewing with a compatible PDF viewer. Only a few changes need be made to the source files so far described to turn entries in the table of contents, internal references in the body of a document, and page references in an index into hyperlinks to the appropriate points in the document, though these features will appear only in PDF files created via the route described in item 1 in Section A.1.3 above. Specifically,

- the `\documentclass` statement at the beginning of the file must not specify any particular driver, i.e. should be of the form `\documentclass{`*Class*`}`.
- if used at all, the `\usepackage{graphicx}` command in the preamble must not include any options specifying a graphics driver
- figures must be provided as PDF files or defined using the package `tikz`. PostScript descriptions of figures will not be properly rendered.
- the command `\usepackage{hyperref}`, perhaps with some optional arguments, must be added in the preamble and, to make sure that other included packages do not overwrite redefinitions made by `hyperref`, should be the *last* package included.

This edited source file is then processed by `pdflatex` (Section A.1.3) at least twice to include hyperlinks, and tables of contents, figures, and tables. With these simple additions, a navigation panel in the PDF file containing chapter, section, subsection, and ... subdivisions of the document will be created. Further, all entries in the tables of contents, list of figures, and list of tables, all internal references created with `\ref` commands, and all index entries will be converted into hot links within the document.

The command `\usepackage{hyperref}` in the preamble is the minimum necessary to create a hyperlinked PDF file. By default, hot links in the document will be displayed in a red box. To override that default, one can exploit options in the `\usepackage` command. For example, the statements,

```
\usepackage{color}
\definecolor{MyPurple}{rgb}{0.577,0.000,1.000}
\usepackage[colorlinks=true,%
linkcolor={MyPurple},bookmarksnumbered=true,linktocpage=true]{hyperref}
```

in the preamble of the LaTeX source file will set the stage for automatic creation of hyperlinks for all references, setting the color of those hyperlinks to `MyPurple` as specified by the RGB values in the `\definecolor` statement and removing the enclosing box, stipulating that the bookmarks in the navigation panels should include chapter and section numbers, and stipulating that page numbers rather than section titles should be the linked entries in the Table of Contents.[60] The final PDF file is then produced by processing the source file with `pdflatex` (Section A.1.3)—at least twice and perhaps three times. To avoid error messages, all auxiliary files hanging over from a previous processing of the source file *without* hyperlinks through `latex` or `pdflatex` should be deleted before processing the file with hyperlinks.[61]

The files created in the user's directory when a properly constructed `.tex` file with hyperlinks, index, and table of contents is processed through `pdflatex` to produce a linked PDF file are[62]

[60] The default value of the `bookmarksnumbered` and `linktocpage` options is `false`.
[61] See item 25 in Section A.17 for ways to achieve this deletion.
[62] We here assume you are making an index and using `imakeidx`. The sequence and auxiliary files will be different if you are not making an index and/or if you are using `makeidx`.

- (after first pass through `pdflatex`) `.aux`, `.idx`, `.ilg`, `.ind`, `.log`, `.out`, `.pdf`, and `.toc`. The `.idx`, `.ilg`, and `.ind` files will be present only if you are creating an index with `imakeidx`. The `.out` file contains information about hyperlinks; the rest are as described in item 1 of Section A.1.3, in Section A.8, and in Section A.9. At this point, the PDF file does not have the navigation panel or the table of contents but it does have the index. All of these files are ASCII text files, though the PDF file may contain some non-printing characters. As such they can all be displayed in an available text editor and understood.
- (after the second pass and, if necessary, the third pass through `pdflatex`) the same as in the previous bullet, though many of those files have been updated. At this point, the navigation panel, the internal hyperlinks, and the index have all been created.

A WARNING: Links to figures and tables will point to the line containing the caption for the figure or table. Captions for these items are perhaps better placed *above* the item rather than below the item.

A CONVENIENCE: If the commands `\documentclass` and `\usepackage{graphicx}` are phrased *without* options *and* the files in all `\includegraphics` commands are presented *without* file type, then the file processed with `latex` will look for `.ps` and `.eps` graphics files and the file processed with `pdflatex` will look for `.pdf` files. All files to be sought must, of course, exist.[63] This feature makes it easy to process the *same* source file both with `latex` and with `pdflatex`.

A.11 Converting `.eps` and `.ps` Files to `.pdf`

When files describing figures are created, it is often easier to output those files from the creating program as `.eps` or `.ps` files rather than as `.pdf` files. Unfortunately, if you desire to produce the final output with `pdflatex`, files describing pictures need to be `.pdf`. PostScript files must therefore be converted into PDF before running `pdflatex`. Short of asking someone who already knows how to achieve that transfer, rummaging on the web for guidance may be a frustrating experience. Once you find the right tools, however, the process is quite simple. Basically, to convert the files `figure.eps` and `figure.ps` to a useful PDF file involves the two steps

1. `ps2pdf figure.eps` (which will yield `figure.eps.pdf`)
   `ps2pdf figure.ps` (which will yield `figure.pdf`)

2. `pdfcrop figure.eps.pdf` (which will yield `figure.eps-crop.pdf`) or
   `pdfcrop figure.pdf` (which will yield `figure-crop.pdf`)

Here, step 1 creates the `.pdf` file and step 2 removes extraneous white space around the perimeter of the first-created `.pdf` file. Second arguments to both `ps2pdf` and `pdfcrop`, as in

```
ps2pdf figure.eps figure.pdf
pdfcrop figure.pdf figurecrop.pdf
```

can be used to relieve the awkwardness of the file names. In addition, `pdfcrop` has a number of options. The statement `pdfcrop --help` will provide a list of those options on the screen. Sometimes the tight cropping provided by the illustrated commands will clip the edges a bit too closely. The option `--margins ...`, as in the statement

```
pdfcrop --margins 10 figure.pdf figurecrop.pdf
```

[63] See Section A.11 for a means to convert `.ps` and `.eps.` files to `.pdf` and *vice versa*.

will provide a 10-pixel margin of white space around the image. See the help message for further details.

To simplify the process, the MS DOS batch files listed in Section A.A.1.1, respectively, and the UNIX *Shell* scripts listed in Section A.A.2.1, respectively, exploit the second argument for both `ps2pdf` and `pdfcrop` to control the file names and, when executed with statements like

MS DOS	UNIX	
`ceps2pdf figure`	`./ceps2pdf figure`	(for `.eps` file)
`cps2pdf figure`	`./cps2pdf figure`	(for `.ps` file)

will (1) leave in the involved directory a cropped PDF file whose name `figure.pdf` differs from that of the original PostScript in only the file type and (2) remove any temporary files created along the way.

The above procedure is a bit tedious if you have more than a few files to convert. If you work in a *Command* window (Windows) or a *Shell* window (UNIX), the steps

- Create a temporary directory and create the several files listed in Section A.A.1 (Windows) or A.A.2 (UNIX) and the file listed in Section A.A.3 (Windows *and* UNIX) into that directory. These files can be created by direct typing or they can be copied from the directory `$HEAD/tex`.
- Copy all `.ps` *or* all `.eps` files to be converted into that temporary directory. *This step guards against disaster should the process to come be run in the initial directory and fail.*
- Create a file containing a list of the names of the `.eps` or `.ps` files in the directory. The statements

MS DOS	UNIX
`dir /b *.eps > dir.txt`	`ls -1 *.eps > dir.txt`
`dir /b *.ps > dir.txt`	`ls -1 *.ps > dir.txt`

  utilizing the option `/b` in Windows and the option `-1` (one, not el) in UNIX produce the desired file (filename, including file type).
- Strip the file type by running the python program `ExtractFileName.py` (Section A.A.3.1). Here the same code works in both Windows and UNIX. This action will create the file `nameonly.txt` containing only the names of the files to be converted.
- Effect the conversions by running the script `rdfile` with statements like

MS DOS	UNIX
`rdfileeps` (for eps files)	`./rdfileeps` (for eps files)
`rdfileps` (for ps files)	`./rdfileps` (for ps files)

  This action will convert each file in turn from `.eps` or `.ps` to `.pdf` and leave no intermediate files in the directory.
- Delete the `*.eps` or `*.ps` files in the temporary directory.
- Move the `*.pdf` files to the directory from which the `.eps` and `.ps` files were moved temporarily, leaving both the original `.ps` and `.eps` files intact but adding the `.pdf` files, so either `latex` or `pdflatex` can then be used to create the final document.

will accomplish that conversion

The reverse process of converting a `.pdf` file to `.ps` or `.eps` format is easier. Specifically, the statements

`pdftops` *FileName*`.pdf` *FileName*`.ps`
`pdftops -eps` *FileName*`.pdf` *FileName*`.eps`

will effect this conversion. The task of creating scripts to automate this process when many files are to be converted is left to the reader.

A.12 Using Conditional Expressions in LaTeX

The LaTeX package `ifthen` provides a capacity to include text and LaTeX commands conditionally, i.e., depending on the state of a Boolean flag. If the package is to be used, the preamble of the LaTeX source file must contain the command

```
\usepackage{ifthen}
```

Once that command has been processed, we must—also in the preamble—then define one or more Boolean flags and set their values with commands like

`\newboolean{`*FlagName*`}`
`\setboolean{`*FlagName*`}{`*FlagValue*`}`

where *FlagName* can be any name that does not conflict with names already used and *FlagValue* will be either `true` or `false`. Finally, at the point in the source file where some text is to be included conditionally, we would place the command

`\ifthenelse{\boolean{`*FlagName*`}}`
`{`Text to be inserted if *FlagValue* is `true`.`}`
`{`Text to be inserted if *FlagValue* is `false`.`}`

Either text can be null, in which case the opening and closing braces should still appear side by side (`{}`), and either text can include LaTeX commands—which will be executed—and can spread over several lines. In particular, the text can include commands to input additional files. Indeed, the customization in *CPSUP* is achieved by including or excluding files depending on the state of several flags. For example, in the preamble of the LaTeX source file for *CPSUP*, the command

```
\usepackage{ifthen}
```

appears and several flags are defined and set with commands like, for example,

```
\newboolean{LATEX} \setboolean{LATEX}{true}
```

Finally, the command

```
\ifthenelse{\boolean{LATEX}}
{ \input{FileName} }
{  }
```

is placed at the point where the file containing the LaTeX portions of this book would be inserted. If the file identified is in the working directory, only its name need be included; otherwise, a full path—absolute or relative—must be specified. At this point, the LaTeX Appendix will be included if the flag is `true` and omitted if the flag is `false`.

More complicated logical operations can be constructed with the operators `\and`, `\or`, and `\not`. For example, if the Boolean flags `NUMREC` and `FORTRAN` are defined and set, then the composite statement

```
\ifthenelse{\boolean{NUMREC} \and \boolean{FORTRAN} }
{ \input{FileName} }
{  }
```

would include *FileName* only if *both* `NUMREC` and `FORTRAN` are true.

In the effort to render files processable with either `latex` or `pdflatex` and also to create easily either a printable document or a linked document, the package `hyperref` must be invoked only when a linked document is desired. In order to avoid manual editing of the source file, one can insert in the preamble the statements[64]

```
\usepackage{ifthen}
\newboolean{PRINT}

\typeout{} \typein[\trueorfalse]{true (for print), false (for linked):}
\ifthenelse{ \equal{\trueorfalse}{true} \or \equal{\trueorfalse}{false}}
{ \setboolean{PRINT}{\trueorfalse} }
{\typeout{} \typeout{Must be either "true" or "false". Try again.} \end{document}}
```

which define the boolean variable `PRINT`, ask for entry of either `true` or `false` at execution time, and terminate execution if an invalid value is entered. Then, the lines

```
\usepackage{color}
\ifthenelse{\boolean{PRINT}}{}
{\definecolor{MyPurple}{rgb}{0.577,0.000,1.000}
\usepackage[colorlinks=true, linkcolor={MyPurple}]{hyperref}}
```

also in the preamble, will include the package `hyperref`, but only if `PRINT` is false.

One further conditional command included in `latex` and `pdflatex` without adding a package facilitates testing whether a file exists. For example, the statement

```
\IfFileExists{test.tex}{\input{test.tex}{}
```

inputs `test.tex` if it exists and simply moves on to the next statement if it doesn't exist. This statement is useful if the file to be input is created in the first pass through `latex` or `pdflatex` and then read in a subsequent pass. The "false" clause could alternatively be used to print an error message and terminate execution, e.g.,

```
\IfFileExists{test.tex}{\input{test.tex}{\typeout{File not found} \end{document}}
```

A.13 Error Messages Generated by LaTeX

However carefully the code is created, sooner or later LaTeX will generate an error message, which will have the general form

```
! error identification
l. number text read
                    text not read
?
```

[64] Inclusion of the package `ifthen` can be omitted if it had been entered previously for other reasons.

The exclamation point is followed by a message that explains the nature of the error. The lower case `l` (el) is followed (1) by the number of the line in the code at which the problem is detected and (2) the text surrounding the problem.[65] Finally, the question mark prompts for user input. A simple ⟨RETURN⟩ at the question mark will instruct LaTeX to continue to read the file unless a fatal error has occurred (though LaTeX may be forced to make assumptions in order to continue); entry of the character `x` (followed by ⟨RETURN⟩ will exit immediately from the program. Many errors are caused by a missing delimiter or by failure to specify a switch to math mode *before* a mathematical expression or back to text mode *after* a mathematical expression. All of the reported errors must be identified and fixed (by editing the code) before LaTeX will produce the desired finished product. A full description of the messages that may be produced as well as an enumeration of other user inputs at the `?` will be found in Chapter 8 in *The LaTeX Manual*.

A.14 The Page Previewer for `.dvi` Files

Since LaTeX is not wysiwyg, we must use a page previewer to see exactly what our document looks like without printing it out many, many times. The *Local Guide* describes the page previewer available at your site. UNIX systems usually provide a version of `xdvi` and Windows systems usually provide a version of `yap`,[66] but many such previewers exist and, on personal computers especially, the previewer may be accessed by a mouse click in a GUI that also gives access to other components of the LaTeX package. To use this previewer, we first create the `.dvi` file by running our code through LaTeX as described in Section A.1. Then we enter a command like [67,68]

`xdvi` *filename* or `yap` *filename* (The default extension for the input file is `.dvi`.)

or select an appropriate item from a menu (as described in *The Local Guide*). Presently, the first page of the document will appear on the screen. Some versions of `xdvi` display an array of buttons along the right edge of the *Xdvi* window; other, probably newer, versions replace those buttons with drop-down menus and icons in a toolbar. Clicking ML on any of the various buttons or selecting items from one of the menus effects the associated action. For `xdvi` and its clones, possible actions include the options shown in Table A.11. In addition, (1) moving the cursor to any point in the window and pressing (and holding) any of the three mouse buttons will generate a magnified version of a region whose size is determined by which button is pushed and (2) some versions of `xdvi` have a window along the left edge in which you can click ML on a page number to request immediate display of the selected page—though note that these numbers simply count pages from the beginning of the displayed document and may not correspond to the numbers actually printed on the pages in the document.

Beyond the controls made available through buttons and/or menus around the periphery of the *Xdvi* window, `xdvi` responds to a number of keystrokes issued when the the cursor is moved into the area displaying text. Some of these are identified in Table A.12; all are described in the `xdvi` documentation, e.g., the UNIX `man` page.

In some environments, a GUI may provide an easy way to use the mouse to work on LaTeX code, process it repeatedly with LaTeX, and examine the changes with an on-screen previewer each time. In the absence of a GUI giving access to all of these features (text editor, LaTeX, screen previewer, printer, ...), one convenient strategy for using a text editor *and* a page previewer simultaneously involves invoking the previewer with a command like

[65] The point at which LaTeX encounters difficulty is conveyed by the downward displacement of the text after that point, though that point is not always where the offense actually occurs.

[66] *Y*et *A*nother *P*reviewer

[67] Additional information about `xdvi` on UNIX systems can be found in the on-line help, accessed in many systems by typing the command `man xdvi`.

[68] If explicit specification of the input file is omitted, some implementations of `xdvi` will bring up a browser in which the desired `.dvi` file can be selected. Other versions may open the most recently displayed file, in which case the browser can be invoked by selecting 'Open' from the FILE menu.

Table A.11: Some of the features available through selection from menus (first column) in some versions of `xdvi`, through clicking ML on buttons in other versions of `xdvi`. The features actually available by these means may vary from version to version of `xdvi`.

Menu	Button	Action
FILE→Reload	Reread	Rereads current `.dvi` file.
FILE→Quit	Quit	Quits previewer.
ZOOM→Shrink by 1	100%	Displays text at full size (quite large).
ZOOM→Shrink by 3	33%	Displays text at 33% of full size.
ZOOM→Shrink by 4	25%	Displays text at 25% of full size.
ZOOM→Shrink by 6	17%	Displays text at 17% of full size.
NAVIGATE→First Page	First	Moves to first page.
NAVIGATE→Page-10	Page-10	Moves to the tenth page before the one displayed.
NAVIGATE→Page-5	Page-5	Moves to the fifth page before the one displayed.
NAVIGATE→Prev	Prev	Moves to previous page
NAVIGATE→Next	Next	Moves to next page.
NAVIGATE→Page+5	Page+5	Moves to the fifth page after the one displayed.
NAVIGATE→Page+10	Page+10	Moves to the tenth page after the one displayed.
NAVIGATE→Last Page	Last	Moves to last page.
OPTIONS→PostScript	View Page	Allows selection among two or more options, including displaying figures (the initial state) and replacing them with properly sized rectangles.
FILE→Open	File	Brings up a browser for selection of new `.dvi` file.

Table A.12: Keystrokes for controlling `xdvi`.

s	Size text to fit window.	n	Move to next page.
1s	(one-s) Largest text.	p	Move to previous page.
*n*s	Apply shrink factor *n*.	*n*n	Advance *n* pages.
g	Move to last page.	*n*p	Back up *n* pages.
1g	(one-g) Move to first page.	q	Exit program.
*n*g	Move to (absolute) page *n*.		

`xdvi` *filename* `&` (UNIX) or `yap` *filename* `&` (Windows)

which will start `xdvi` or `yap` but detach it from the launching *Shell* window, leaving the *Shell* window free for other uses. Then, with the `.tex` file in a text editor detached from the launching *Shell* window, we can edit the text, save the edited version, use the *Shell* window to invoke LaTeX on the edited file, and then simply click ML at an appropriate point in the displayed text to instruct `xdvi` or `yap` to reread the `.dvi` file.[69] Thus, the effect of each new set of edits can be examined quickly without repeatedly starting and exiting from the previewer.

A.15 The Spell Checker in UNIX[70]

The programs `ispell` and `aspell`, either or both of which may be part of your LaTeX distribution, are common spell checkers. If either is installed at your site (see the *Local Guide*), it can be invoked

[69] In some environments, rereading an updated file will happen automatically without an explicit user request.
[70] To the author's knowledge, no comparable stand-alone spell checker exists for Windows.

with a command like[71]

`ispell` *filename* or `aspell check` *filename*

or perhaps with a mouse click in a GUI. For files with extension `.tex`, `ispell` enters its TeX mode and will not identify every TeX or LaTeX command as a misspelled word. Note, however, that these commands are simply *ignored* by `ispell` and `aspell`; their correctness or legitimacy as commands to LaTeX is *not* assessed.

A.16 A Sample Document

The following sample document demonstrates some of the commands explained above and also introduces some useful commands that have not yet been mentioned. (The explanatory *comments* following the `%` sign can be included in the code but have no effect on the output produced.)

```
\documentclass{article}                          % Mandatory command; select
                                                 %   default 10 point type

\setlength{\textheight}{9truein}                 % Set height of text area
\setlength{\topmargin}{-0.5truein}               % Center text on 11'' height
\setlength{\textwidth}{6truein}                  % Set width of text area
\setlength{\oddsidemargin}{0.25truein}           % Center text on 8.5'' width
\setlength{\parskip}{6pt plus 1pt minus 1pt}     % Specify extra space
                                                 %   between paragraphs,
                                                 %   allowing LaTeX to adjust
                                                 %   space 1 point up or down
\setlength{\parindent}{40pt}                     % Set paragraph indent

\begin{document}                                 % Mandatory command

\begin{center}                                   % Large, bold, centered title
{\Large\bf A Sample Document for Your Perusal}
\end{center}

\begin{flushleft}                                % Two lines; flush left
{\em Author\/}: J.~Q.~Student \\                 % \\ forces a new line
                                                 % Note italic correction
                                                 % Tilde prevents extra space
{\em Date\/}: \today
\end{flushleft}

In this document we have used the \verb+\setlength+ commands to modify
\LaTeX's default page setup to make fuller use of an 8.5''$\times$11''
page.\footnote{Note the special command $\backslash${\tt LaTeX}
provided for the display of the \LaTeX{} ``logo''.}
The title could have been generated using the three commands
\verb*+\title{+\ldots\verb*+}+, \verb*+\author{+\ldots\verb*+}+, and
\verb*+\date{+\ldots\verb*+}+ before the \verb*+\begin{document}+
```

[71] Additional information about `ispell` or `aspell` can be found by typing the command `ispell` with no arguments or the command `aspell help`. Even more detailed information may be available in the on-line help, accessed in many systems by typing the command `man ispell` or the command `man aspell`. Googling `ispell` or `aspell` will surely also provide links to detailed descriptions.

```
command---i.e., in the preamble---followed by the command
\verb*+\maketitle+ after the beginning of the document.  The command
\verb*?\verb*+ ... +? will cause \LaTeX{} to print whatever
is between the plus signs
(which could be virtually any specified character) exactly as it is.
The asterisk is optional, and it makes \LaTeX{} highlight the spaces that
occur inside the \verb+verbatim+ environment.

\noindent Notice that a blank line creates a new paragraph.  The
\verb+\noindent+ command {\em suppresses\/} the standard paragraph
indentation. In this special space we will also demonstrate some math
environment operations.  Consider the equation
\begin{equation}
\int_{0}^{\infty}\: e^{-x}\, dx = 1,
\label{LATEX:demo}
\end{equation}
where a medium sized space, specified by the command \verb+\:+, is
put between the integral sign $\int_{0}^{\infty}$, and
the integrand and a small space, specified by the command \verb+\,+,
is inserted between the integrand and the $dx$ in Eq.~(\ref{LATEX:demo}).
The equation
\begin{equation}
\frac{ \partial e^{x_0 y^2} } {\partial x_0 } = y^2 e^{x_0y^2}
\end{equation}
demonstrates partial derivatives and fractions.  Note the automatic
generation and placement of equation numbers in the output. Also note
that pages are automatically numbered, though the actual printing of
page numbers can be suppressed with the commands \verb+\pagestyle+ and
\verb+\thispagestyle+.

\end{document}                                  % Mandatory command
```

The output produced when this short sample file is processed through LaTeX and `dvips` and then printed is shown in Table A.13, though the page header on that page is not included in the output. Note that, to obtain the proper internal reference to the first equation, this document must be processed *twice* by LaTeX. The file itself is named `texsample2.tex` and can be copied from the directory `$HEAD/tex`.

A.17 Miscellaneous Other Features

LaTeX has an enormous number of additional features beyond those enumerated above. In particular, be aware of

1. All of the environments listed in Section A.3.4.
2. All of the packages listed in Section A.3.5.
3. The command `\today`, which returns today's date in the form "month date, year".
4. The command `\newcommand`, which permits us to define commands supplementing the standard commands. For example, the commands

   ```
   \newcommand{\beq}{\begin{equation}}
   \newcommand{\eeq}{\end{equation}}
   ```

Table A.13: Output produced by the code in Section A.16, except that the equations here are numbered (A.1) and (A.2) rather than (1) and (2) and the footnote is labeled [a] rather than [1]. Further, the page number that would appear has been deleted in this display.

A Sample Document for Your Perusal

Author: J. Q. Student
Date: 13 February 2023

In this document we have used the `\setlength` commands to modify LaTeX's default page setup to make fuller use of an 8.5"×11" page.[a] The title could have been generated using the three commands `\title{...}`, `\author{...}`, and `\date{...}` before the `\begin{document}` command—i.e., in the preamble—followed by the command `\maketitle` after the beginning of the document. The command `\verb*+␣...␣+` will cause LaTeX to print whatever is between the plus signs (which could be virtually any specified character) exactly as it is. The asterisk is optional, and it makes LaTeX highlight the spaces that occur inside the `verbatim` environment.

Notice that a blank line creates a new paragraph. The `\noindent` command *suppresses* the standard paragraph indentation. In this special space we will also demonstrate some math environment operations. Consider the equation

$$\int_0^\infty e^{-x}\,dx = 1, \tag{A.1}$$

where a medium sized space, specified by the command `\:`, is put between the integral sign $\int_0^\infty$, and the integrand and a small space, specified by the command `\,`, is inserted between the integrand and the dx in Eq. (A.1). The equation

$$\frac{\partial e^{x_0 y^2}}{\partial x_0} = y^2 e^{x_0 y^2} \tag{A.2}$$

demonstrates partial derivatives and fractions. Note the automatic generation and placement of equation numbers in the output. Also note that pages are automatically numbered, though the actual printing of page numbers can be suppressed with the commands `\pagestyle` and `\thispagestyle`.

[a] Note the special command `\LaTeX` provided for the display of the LaTeX "logo".

---

in the preamble will permit us to type the commands `\beq` and `\eeq` rather than the longer forms, measurably simplifying the typing of a document containing many displayed equations. Details will be found in Sections 3.4.1 and C.8.1 in *The LaTeX Manual*.

5. The command `\renewcommand` for changing commands that already exist. When a command already exists (as, for example, when a command is defined by the selected document class), `\newcommand` will fail. In those circumstances (as we have seen already in Section A.5), we need the command `\renewcommand`. As one particular example, the built-in command `\today` which, by default, formats the date in American style (e.g., April 3, 1938) can be changed to format the date in European style (3 April 1938) with the command

```
\renewcommand{\today}{\number\day\space \ifcase\month\or January%
\or February\or March\or April\or May\or June\or July\or August%
\or September\or October\or November\or December\fi \space\number\year}
```

The percent signs at the end of the first two lines are *not* superfluous; they guarantee that LaTeX will see this command as a single line and hence that extraneous spaces will not appear in

the output for occasional dates. Note also that this modification invokes TEX's *case* structure, which is introduced by the `\ifcase` command and terminated by the `\fi` command.

6. The command `\pagestyle{`*Style*`}`. Placed in the preamble, this command selects a particular (global) page style *for the entire document*. Recognized styles include `plain`, `empty`, `headings`, and `myheadings`. Details will be found in Sections 6.1.2 and C.5.3 in *The LATEX Manual*.

7. The command `\thispagestyle{`*Style*`}`. Placed at any point, this command selects a particular page style *for the current page*, overriding the global specification for that page alone. Recognized styles are the same as for the command `\pagestyle`. This command in the form `\thispagestyle{empty}` is commonly used to suppress page numbering on the first page of a several-page document. Details will be found in Sections 6.1.2 and C.5.3 in *The LATEX Manual*.

8. ReVTEX, which contains files defining the `aps` document class. These files have been created by the American Institute of Physics (AIP), the American Physical Society (APS), and the Optical Society of America (OAS) and are intended for use in preparing manuscripts for ultimate publication in the journals published by these organizations. Full documentation is contained in *The REVTEX Input Guide* prepared by the AIP, the APS, and the OAS.[72]

9. The command `\input`, which simply inputs the file specified in its argument. Details are described in Section 4.4 of *The LATEX Manual*.

10. The calligraphic type style for producing upper-case calligraphic letters in math mode. This style can be invoked either with the declaration `\cal` or the command `\mathcal`. It is described in Sections 3.3.2, 3.3.8, and C.7.8 of *The LATEX Manual*.

11. Methods for placing accents over characters. For example, the command `\"{o}` will produce ö, the command `\~{n}` will produce ñ, and the command `\c{c}` will produce ç. A full listing of the possibilities will be found in Section 3.2.1 and Table 3.1 of *The LATEX Manual*.

12. The `alltt` and `verbatim` packages, which allow incorporation of computer code by reference to the actual file containing the computer program itself. On the surface, it would appear as if a file containing computer code could be incorporated into a LATEX document with the simple command sequence

    `\begin{verbatim}` `\input{`*FileName*`}` `\end{verbatim}`

    The problem comes because the backslash that introduces the command `\input` will, in the `verbatim` environment, be treated as an ordinary character and will *not* be recognized as introducing a command. If, instead, we invoke the alternative `alltt` environment with the LATEX commands

    `\begin{alltt}` `\input{`*FileName*`}` `\end{alltt}`

    the problem is solved, since `\`, `{`, `}`, and one or two other characters are treated specially within the `alltt` environment and the embedded command `\input` will now be properly recognized as a command to read the specified file into the LATEX source stream at this point. This new environment, however, will be available only if the `alltt` package is explicitly added with the command

    `\usepackage{alltt}`

    placed in the preamble to the LATEX file.

[72] Version 4.1, which was released in final form on 11 August 2010, is compatible with LATEX2$_\varepsilon$. The APS website `publish.aps.org/revtex4` (*no* `www.`) contains up-to-date information about ReVTEX4, links to an assortment of manuals (including one titled *RevTEX4.1 Author's Guide*, and a link from which that version can be downloaded. ReVTEX is automatically included in many standard LATEX distributions.

Actually, there is at least one situation in which the `alltt` environment isn't quite up to the task. If we want to read in a C program that specifies newline characters with `\n`, LaTeX embellished with `alltt` will complain that `\n` is an undefined command. For this case (and, of course, for the others as well), we need instead exploit the `verbatim` *package* (not environment), which is made available by placing the command

```
\usepackage{verbatim}
```

in the preamble. In particular, this inclusion defines a *command* `\verbatiminput`, invoked with a statement like

```
\verbatiminput{FileName}
```

When the `verbatim` package is invoked, we also have available a new environment—the `comment` environment, which can be used to "bracket" extended text that one wants to exclude from processing by LaTeX.[73]

13. The ten characters (\$, &, {, }, %, #, _, ~, ^, \), which have special meanings to LaTeX and will not normally be printed as characters. The first seven of these characters can be printed *as characters* by preceding the character with a backslash, e.g., to print \$, type `\$` in the code. The last three, however, are trickier because `\~` and `\^` are themselves commands for accents over the following letter and `\\` is the command for a new line. These last three characters can, however, be printed by invoking the constructions `\verb+~+`, `\verb+^+`, and `\verb+\+`. Here, the `\verb` command enters verbatim mode, the *immediately following* `+` sign—any character can be used—flags the beginning of the text to be presented in verbatim mode, and the final `+` sign—better, repeat of the first character—flags the end of the text to be presented in verbatim mode. The backslash can also be produced in math mode with the command `$\backslash$`.

    In some environments and in footnotes, the command `\verb` is forbidden. Another way to specify the printing of some of these special characters is to use the TeX (not LaTeX) command `\char`. In its raw form, this command simply takes a number, e.g., `\char98`, which specifies the insertion of character 98 from the current font. We can, however, also specify the character by invoking the 'tic' operator, which instructs TeX to *calculate* the numeric code from the character itself. In particular, some of the special characters that *cannot* be invoked directly can be invoked with the command `\char`. Thus, for example, the command `\char97` will produce the character 'a' while the commands `\char‘b`, `\char‘\^`, `\char‘\~`, and `{\tt \char‘\\}` will produce the characters 'b', '^', '~', and '\', respectively. Similarly, the command `{\tt \char‘>}` will produce '>', which differs in sometimes desirable ways from the character '>' produced by `$>$`.

14. The commands that allow us to create boxes as described in Sections 6.4.3 and C.13.3 in *The LaTeX Manual*. These commands include `\mbox`, `\makebox`, `\fbox`, `\framebox`, `\parbox`, and `\rule` and the `minipage` environment. All of these commands "block" the contents of the box into a structure that LaTeX sees as a single unit. The commands `\fbox` and `\framebox` also place a printed rectangular box around the contents of the box created.

15. The notion of counters. In the off-the-shelf version of LaTeX, each new invocation of the `enumerate` environment starts a counter off again at the beginning. Sometimes, we might wish to have the numbers in a new enumeration pick up from where the numbers in the previous one ended. We should, of course, achieve that objective in a way that does not depend on knowing explicitly what the starting number in the new environment should be. Basically, we have to invent a way to tell LaTeX to remember the value at which it ended and retrieve that

[73] The `verbatim` *package* also redefines the `verbatim` *environment* in ways, however, of little consequence unless the text in the environment is *very* extensive. If the `verbatim` *package* is invoked, one particularly significant change in the `verbatim` *environment* results in the complete *ignoring* of any characters following the statement `\end{verbatim}` in the same line. It is best always to place the statement `\end{verbatim}` on a line by itself.

value when a new environment is entered. The remembered value must be stored in a LaTeX variable called a *counter*, which must be defined with the command

```
\newcounter{hold}
```

where `hold`, which can*not* contain numeric characters, is the name chosen for the counter. Then, the structure

```
\begin{enumerate}
\item Text of first item.
\item Text of second item.
\setcounter{hold}{\value{enumi}}
\end{enumerate}
... Text not in enumerate environment.
\begin{enumerate}
\setcounter{enumi}{\value{hold}}
\item Text of third item.
\item Text of fourth item.
\setcounter{hold}{\value{enumi}}
\end{enumerate}
```

will achieve the desired end. In the first `enumerate` environment, the counter `enumi`, which is used behind the scenes to keep track of the number of items, is initialized to zero. It is then incremented by 1 with each new item. At the end of the environment, `enumi` stores the number of the last item. The command `\setcounter` in the first environment saves the current value of `enumi` in the counter `hold`, which survives the exit from the environment. Thus, immediately on entry to the second environment, we can use a "reflection" of the first use of `\setcounter` to restore the value that `enumi` had reached at the end of the first environment.

16. Parameters influencing placement of "floats" (figures and tables). Sometimes LaTeX seems to have a mind of its own with regard to placement of floating objects on the current or later pages. Many of these "problems" can be cleared up by tampering with the default values of the parameters that limit the number of floats that can be put at the top or bottom of the page or, even more, by changing the parameters that stipulate the maximum fraction of a page that can be devoted to floats or the minimum amount of text that must appear on a page. These parameters are enumerated at the end of Section C9.1 in *The LaTeX Manual*. For single column presentations, the most important of these parameters are `\topfraction`, `\bottomfraction`, `\textfraction`, and `\floatpagefraction`. The command `\renewcommand` must be used to change the values of these parameters.

17. The way to change section, page, figure, table, and footnote "numbering". By default in the `article` class, sections, subsections, subsubsections, pages, figures, and footnotes are labeled with Arabic numbers, e.g., Section 3.1.2. The format of that label as determined from the underlying counters `section`, `subsection`, `subsubsection`, `page`, `figure`, `table`, and `footnote` can be changed. If, for example, we wished the sections to be labeled with upper-case letters, the subsections with lower-case letters, and the subsubsections with arabic numbers (the default), we would simply execute the instructions

    ```
    \renewcommand{\thesection}{\Alph{section}.}
    \renewcommand{\thesubsection}{\thesection\alph{subsection}.}
    ```

    in the preamble.[74] Similarly, the commands

    ```
    \renewcommand{\thetable}{\Roman{table}}
    \renewcommand{\thepage}{\roman{page}}
    ```

[74] Note the explicit periods and the inclusion of the section "number" in the definition of the subsection "number".

Table A.14: Structure to produce full-width text at top of double-column text.

```
\documentclass[twocolumn]{article}
   ... Whatever preamble we want
\begin{document}

\twocolumn
[  \begin{center}
     {\large\bf Title} \\[12pt]       % Bold title; extra space
     {\large\bf Author} \\[12pt]      % Bold author; extra space
   \end{center}
   \begin{quotation}
   \noindent ... Text of abstract.
   \end{quotation}
]

   ... Text of paper

\end{document}
```

---

will change table "numbering" to upper-case Roman numbers and page "numbering" to lower-case Roman numbers. Essentially, the "numbers" on these components of a document are generated when LaTeX executes the command `\the...`, where `...` stands for the name of the appropriate counter. Redefining this command changes the translation LaTeX applies to the counter when generating the "number".

18. The way to create a full-width heading and abstract above (and on the *same* page as) a two-column presentation of text, which exploits an optional argument to the *command* `\twocolumn` (not the optional argument `twocolumn` to the command `\documentclass`). One format that achieves this end uses the code is shown in Table A.14.[75] If this pattern is to work, there can be *no* commands that produce output between the command `\begin{document}` and the command `\twocolumn`.

19. The means to change the label in the caption of a figure. By default, LaTeX introduces the caption of the first figure with the label 'Figure 1:', incrementing the number automatically with each subsequent figure. The word 'Figure' in this caption is the default value of the command `\figurename`, but that portion of the label can be changed with a simple command like

    ```
    \renewcommand{\figurename}{Fig.}
    ```

    in the preamble. Changing the colon after the number is harder, since the specification of that punctuation is embedded in the class file in use and is not brought to the outside in a simple command. To change the colon to a period, for example, we need to find the file `article.cls`, search through the file for colons, replace the critical ones with periods, save the file in the default directory with a new name, and then specify *it* rather than the standard file in the `\documentclass` command at the beginning of the LaTeX source file.[76]

[75] In more recent versions of LaTeX, the optional argument `[12pt]` to put a bit of extra space after the title and author appears to generate error messages. The "fix" is to replace `[12pt]` with a full command, specifically `\vspace{12pt}`.

[76] The file `article.cls` in the version of LaTeX installed at Lawrence University has only two colons that are not in comments; both should be changed.

20. The means to change the label in the caption of a table. By default, LaTeX introduces the caption of the first table with the label 'Table 1:', incrementing the number automatically with each subsequent table. The word 'Table' is supplied by the command `\tablename` and can be changed in the way described for `\figurename` at item 19 above. The colon can also be changed as described in the previous item, though changing the colon for figures will change it in the same way for tables.

21. The means to change the label in the title of a chapter. By default, LaTeX introduces the title of the first chapter with the label 'Chapter 1' on a line by itself, incrementing the number automatically with each subsequent chapter. The word 'Chapter' is supplied by the command `\chaptername` and can be changed in the way described for `\figurename` at item 19 above.

22. The means to change the title above the table of contents and above the index. By default, LaTeX labels the table of contents 'Contents' and the index 'Index'. The word 'Contents' is supplied by the command `\contentsname` and the word 'Index' is supplied by the command `\indexname`. Each can be changed in the way described for `\figurename` at item 19 above.

23. Using the `array` environment to create matrices in math mode. As with the `tabular` environment (see Section A.6), an optional argument specifies the positioning of entries in the columns, ampersands separate entries in each row, and the command `\\` marks the end of each row. Thus, for example, the LaTeX code

```
\[
\left(\begin{array}{cccc}
1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12
\end{array}\right)
\]
```

    will produce the display

$$\left(\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{array}\right)$$

    The entries in each cell can, of course, be much more elaborate than the simple choices made here.

24. The LaTeX package `needspace`, which facilitates avoiding awkward page breaks conditionally. For example, when a section title occurs close to the bottom of a page, LaTeX may place the title and only a single line of the text before turning the page. Sensible styles would dictate that the section be started on a fresh page. The command `\newpage` will, of course, achieve that objective. As a document experiences further edits, however, that page break may not any longer be appropriate. Placing the command

```
\usepackage{needspace}
```

    in the preamble and then, at the point where a conditional page break might be wise, inserting one of the commands

```
\needspace{5\baselineskip}      or      \Needspace{5\baselineskip}
```

    will result in turning the page, but only if five or fewer lines remain at the bottom of the page. The amount of space left by `\needspace` will depend some on what penalties are in effect but will usually be close enough to be acceptable; further, `\needspace` will leave a ragged bottom even if `\flushbottom` is in effect. The command `\Needspace` will leave the requested space, will take longer to execute, and will leave a ragged bottom; the command `\Needspace*` will produce a flush bottom if `\flushbottom` is in effect. The unit of measure is the size of the length parameter `\baselineskip`. The 5 in this example can, of course, be whatever number seems appropriate. Note that there is *no* multiplication sign after the number.

25. The Windows batch command `cleanTEX` defined by the batch file

```
REM delete all TEX intermediate files
del *.aux *.log *.dvi *.toc *.ilg *.idx *.bak *.sav *.out *.ind
```

and the Unix shell script defined by the file

```
#!/bin/bash
# delete all TEX intermediate files
rm -f *.aux *.log *.dvi *.toc *.ilg *.idx *.bak
rm -f *.sav *.out *.ind
```

which facilitate removing all TeX/LaTeX auxiliary files[77] in the directory in which the command or script is executed. These files can be created by direct typing or can be copied from the directory `$HEAD/tex`. The Windows file is executed by the simple statement `cleanTEX` at a command window prompt, though a path to the file may be included if the file is not located in the directory to be purged of TeX/LaTeX auxiliary files; the UNIX file is executed by the simple statement `./cleanTEX`, though the characters `./` may be replaced by the path to the file if the file is not located in the directory to be purged of TeX/LaTeX auxiliary files.

26. The Windows batch command `cleanbak` define by the the batch file

```
REM Delete .bak files
del /s *.bak
del /s *.sav
```

and the UNIX shell script `cleanbak` defined by the the file

```
#!/bin/bash
# Delete *.bak, *.sav, and *~ files
find . -name '*.bak' -exec rm {} \;
find . -name '*.sav' -exec rm {} \;
find . -name '*~' -exec rm {} \;
```

which facilitate removing all backup files[78] in the directory in which the command or script is executed **and** in all subdirectories below that directory. These files can be created by direct typing or can be copied from the directory `$HEAD/tex`. The Windows file is executed by the simple statement `cleanbak` at a command window prompt, though a path to the file may be included if the file is not located in the directory at the top of the tree to be purged of backup files; the UNIX file is executed by the simple statement `./cleanbak`, though the characters `./` may be replaced by the path to the file if the file is not located in the directory at the top of the tree to be purged of backup files.

A.18 References

As access to the web has expanded, more and more of the information that once was printed is available more readily on the web. Searching for help on specific issues, either on the web as a whole or (perhaps more effectively) more narrowly on the TeX Users Group site `www.tug.org` will yield valuable results. Further, when LaTeX is installed in UNIX and UNIX-based operating systems (which includes Mac computers), the Shell command `man`, e.g. `man hyperref`, may bring up a

[77] Files with file types `.aux`, `.log`, `.dvi`, `.toc`, `.ilg`, `.idx`, `.bak`, `.sav`, `.out`, and `ind`.
[78] Files with file types `.bak` and `.sav` and files for which the last character in the name is `~`.

page describing the indicated entity. In all full LaTeX installations, the Shell command `texdoc`, e.g. `texdoc hyperref`, will bring up a description (though some of those descriptions are in languages other than English).

Numerous books have also been written for a variety of audiences and with objectives ranging from general discussions to very specific focus on particular tasks. Among the more common books are the following:

A Document Preparation System: LaTeX User's Guide and Reference Manual (Updated for LaTeX2$_\varepsilon$), Leslie Lamport (Addison-Wesley Publishing Co., Reading, MA, 1994). [ISBN 0-201-52983-1 or 978-0201529838] Throughout this aAppendix, this book is referred to as *The LaTeX Manual*.

The LaTeX Companion, Michel Goossens, Frank Mittelbach, et al. (Addison-Wesley Publishing Co., Reading, MA). [ISBN 0-201-54199-8 (first edition, 1994); 0201362996 or 978-0201362992 (second edition, 2004]

LaTeX for Engineers and Scientists, David J. Buerger (McGraw-Hill Book Co., New York, 1990). [ISBN 0-07-008845-4]

The TeXbook, Donald E. Knuth (Addison-Wesley Publishing Co., Reading, MA, 1984, 1986). [ISBN 0-201-13448-9 or 978-0201134483]

TeX for the Impatient, Paul W. Abrahams with Karl Berry and Kathryn A. Hargreaves (Addison-Wesley Publishing Co., Reading, MA, 1990). [ISBN 0-201-51375-7]

The LaTeX Graphics Companion, Michael Goossens, Sebastian Rahtz, and Frank Mittelbach (Addison–Wesley Publishing Co., Reading, MA). [ISBN 0-201-85469-4, first edition, 1997; 0321508920 or 978-0321508928, second edition, 2007]

A Guide to LaTeX, Helmut Kopka and Patrick W. Daly (Addison–Wesley Publishing Co., Reading, MA). [ISBN 0-201-39825-7, third edition, 1999; 0321173856 or 978-0321173850, fourth edition, 2003]

Math Into LaTeX (Third Edition), George Grätzer (Birkäuser, Boston) [ISBN 0-8176-4131-9 or 978-0201433111, 2000]

The LaTeX Web Companion Integrating TeX, HTML, and XML, Michael Goossens, Sebastian Rahtz, Eitan M. Gurari, and Ross Moore (Pearson Education, Inc.) [ISBN 0201433117 or 978-0201433111, 1999]

Additional books will likely surface in a search for `LaTeX` on the Amazon or Barnes and Noble websites, though that search may also generate a number of hits for documents about rubber.

A.19 Exercises

A.1. Study carefully the LaTeX code presented in Section A.16 and the resulting output in Table A.13, making sure you understand both the syntax of each command and the effect it produces in the output.

A.2. Use LaTeX to write a letter to a friend. Format your letter so that it has

- a centered block at the top containing your name and address (`center` environment);
- a *right*-justified date (using `\today` and either `\hfill` or the `flushright` environment);
- a *left*-justified block containing an inside address (`flushleft` environment);
- a *left*-justified salutation;
- the body of the letter, containing more than one paragraph;
- a closing (e.g., Sincerely, With love, ...) positioned 3.5" from the left margin; and

- your name, spaced far enough below the closing to allow for your signature and aligned with the closing.

Suppress page numbers altogether on this letter. Include both your LaTeX code and the processed output in what you submit as a solution to this exercise.

A.3. Examine one (or more) of the templates listed in Table A.6 and, in a document produced with LaTeX, explain the function of each command it contains. Include both your LaTeX code and the processed output in what you submit as a solution to this exercise.

A.4. The folder `$HEAD/tex` contains the three files `radio.ps`, `radio.eps`, and `radio.pdf`, each of which contains a description of a graph showing the number of nuclei of each of three species A, B, and C as a function of time as the 1000 nuclei of A initially present decay radioactively first to B and then to C. The equations describing the system are

$$\frac{dA}{dt} = -k_A A \quad ; \quad \frac{dB}{dt} = k_A A - k_B B \quad ; \quad \frac{dC}{dt} = k_B B$$

and the graph shows the solution of these equations when the initial conditions are $A(0) = 1000$, $B(0) = C(0) = 0$ for the case $k_A = k_B = 0.1$. Prepare a document containing this figure, the differential equations, the initial conditions, and a brief description of the graph in English. Be sure that your text includes explicit references to the figure and the equations. You might also contrive to include a footnote somewhere. Include both your LaTeX code and the processed output in what you submit as a solution to this exercise. *A Crutch*: The file `$HEAD/tex/sampledoc.tex` provides a start on the creation of a suitable source file for this exercise.

A.5. Generate LaTeX code to produce the memo

M E M O R A N D U M

Current date

TO: *Insert name of recipient here.*

FROM: *Insert name of sender here.*

SUBJECT: *Insert topic here.*

MESSAGE:

Insert message here.

Make sure that the date placed in the memo is the date the memo was processed (i.e., use the command `\today`). Include both your LaTeX code and the processed output in what you submit as a solution to this exercise. *Suggestion*: You may want to save this template in a file, say `memo_template.tex`, so you have it available as a starting point for the multitude of memos you will subsequently write.

A.6. In connection with a grant you received from the XYZ Foundation, you need periodically to submit a progress report. The format and content of that report are dictated by the Foundation, and the report is created by filling in the information indicated in italics in the template on page 45. Create a LaTeX source file that you can save and use repeatedly to facilitate generating each required report, i.e., create a LaTeX source file that, when processed, will produce the output shown below. Pay meticulous attention to the spacing in the heading and to the spacing and the rulings in the table.

PROGRESS REPORT to XYZ FOUNDATION

Due date of report

Grant Number: *Grant number* **Date of Award:** *Date of award*

Title of Award: *Title of award*

Principal Investigator (PI): *Name of PI*

Investigator's Institution: *Name and address of institution*

---

1. Please describe progress made since last report:
   Insert response.
2. Please list talks given since last report, including title and venue:
   Insert response.
3. Please give full citations for each publication since last report:
   Insert response.
4. Please describe any unanticipated difficulties encountered:
   Insert response.
5. Please identify any individuals beyond the PI who have contributed more than ten hours per week to the project during the past time period:
   Insert response.
6. Please summarize expenses since the last report:

Category	Amount	Amount
Original grant		*$xxx,xxx*
Total expenses reported last time		*$xxx,xxx*
Available funds at start of current period		*$xxx,xxx*
Expenses in current period:		
Salaries	*$xxx,xxx*	
Fringe benefits	*$xxx,xxx*	
Equipment	*$xxx,xxx*	
Supplies	*$xxx,xxx*	
Travel	*$xxx,xxx*	
Page charges	*$xxx,xxx*	
Other: *Identify*	*$xxx,xxx*	
Total for current period		*$xxx,xxx*
AVAILABLE FUNDS FOR REMAINDER OF PROJECT		*$xxx,xxx*

Signed: _____ **Date:** _____

Typed Name of PI

Template to be reproduced in Exercise A.6.

A.A Listings

This section provides listings of the several batch files/shell scripts and the one PYTHON program involved in converting PostScript files to PDF files. Their use is described in Section A.8, and the files themselves reside in the directory `$HEAD/tex`.

A.A.1 ... for Windows

A.A.1.1 Batch File `ceps2pdf.bat`

```
REM ceps2pdf.bat    (Convert EPS to PDF in Windows)
echo off
ps2pdf %1.eps tmp.pdf
pdfcrop tmp.pdf %1.pdf
del tmp.pdf
```

A.A.1.2 Batch File `cps2pdf.bat`

```
REM cps2pdf.bat    (Convert PS to PDF in Windows)
echo off
ps2pdf %1.ps tmp.pdf
pdfcrop tmp.pdf %1.pdf
del tmp.pdf
```

A.A.1.3 Batch File `rdfileeps.bat`

```
REM rdfile.bat   (Convert each file in nameonly.txt to PDF in Windows)
del dir.txt
for /f %%a in (nameonly.txt) do (
   ps2pdf %%a.eps tmp.pdf
   pdfcrop tmp.pdf %%a.pdf
)
del tmp.pdf
del nameonly.txt
```

A.A.1.4 Batch File `rdfileps.bat`

```
REM rdfileps.bat   (Convert each file in nameonly.txt to PDF in Windows)
del dir.txt
for /f %%a in (nameonly.txt) do (
   ps2pdf %%a.ps tmp.pdf
   pdfcrop tmp.pdf %%a.pdf
)
del tmp.pdf
del nameonly.txt
```

A.A.2 ... for UNIX

A.A.2.1 Shell Script ceps2pdf

```
#!/bin/bash
# ceps2pdf    (Convert EPS to PDF in UNIX)
filename=$1
ps2pdf $filename.eps tmp.pdf
pdfcrop tmp.pdf $filename.pdf
rm -f tmp.pdf
```

A.A.2.2 Shell Script cps2pdf

```
#!/bin/bash
# cps2pdf    (Convert PS to PDF in UNIX)
ps2pdf $1.ps tmp.pdf
pdfcrop tmp.pdf $1.pdf
rm -f tmp.pdf
```

A.A.2.3 Shell Script rdfileeps

```
#!/bin/bash
# rdfileeps    (Convert each file in nameonly.txt to PDF in UNIX)
cat nameonly.txt | while read filename
  do
     ps2pdf $filename.eps tmp.pdf
     pdfcrop tmp.pdf $filename.pdf
     rm -f tmp.pdf
  done
rm -f dir.txt
rm -f nameonly.txt
```

A.A.2.4 Shell Script rdfileps

```
#!/bin/bash
# rdfileeps    (Convert each file in nameonly.txt to PDF in UNIX)
cat nameonly.txt | while read filename
  do
     ps2pdf $filename.ps tmp.pdf
     pdfcrop tmp.pdf $filename.pdf
     rm -f tmp.pdf
  done
rm -f dir.txt
rm -f nameonly.txt
```

A.A.3 ... for Windows and UNIX

A.A.3.1 Python Script `ExtractFileName.py`

```
# David Cook
# ExtractFileName.py
# Strip file names in dir.txt and store results in nameonly.txt

# Open and read in existing file from execution of dir /b or ls -1
stream = open("dir.txt","r")
all_lines = stream.readlines()
stream.close()

current_line = "\n"
current_heading = ""

# Open a file for writing out
outfile = open("nameonly.txt","w")

# Loop over each line of the ‘all_lines‘ variable,
# stripping off the file type, leaving only the
# name to be written to the output file.
for line in all_lines:
  lw = (line.split(".")[0]).strip()
  outfile.write(lw +"\n")
outfile.close()
```

Index

M

P

R

T

U

W

X

Y