Contents

LaTeX Seminar Spring Semester, 2010

Clifford E. Weil

Abstract

This document presents basics and some advanced topics about TeX and LaTeX. It ends with items aimed specifically at those in mathematics, the natural sciences and engineering.

1 Introduction.

A brief history will give the reader some insight into why LaTeX is designed the way it is. In 1977 Donald Knuth, of Stanford University, was motivated to use the rising power of computers to produce professional typesetting including the very technical kind needed for mathematics, engineering, science and the like. He called the result TeX for Tau Epsilon Chi. The TeX user creates a source file that contains text and commands using only the keystrokes that appear on a standard typewriter keyboard. The commands tell the program to perform certain actions such as start a new page or to produce a certain symbol such as a Σ or far more complicated structures such as a table. The file is then compiled and a .dvi file or now, more commonly, a .pdf file is produced that can be printed. Using commands rather than keys strokes permits an essentially limitless number of symbols and special structures. And indeed new commands are still being added today as the need arises. After more than a decade of adjusting and expanding the program, in the early 1990s Knuth announced that, in the interest of stability, he would make, nor permit, further development of the TeX program.

In 1985 Leslie Lamport developed a companion to TeX that he called LaTeX, *A Document Preparation System*, which expanded and simplified the use of TeX. It consisted of a large number of macros that made producing a professional-looking document much easier. His motivation was the philosophy that an author should be concerned only about the content of an article and not about formatting. The first version was called version 2.09 (perhaps because it's the start of the decimal expansion for e) but a serious problem developed. As changes and additions were made, users were forced to continually update their LaTeX implementations. Failure to do so might mean that they wouldn't be able to compile a file created using the latest version. One such addition was the use of an one of several European languages in addition to English. To solve this problem the LaTeX2ϵ project was launched in 1993 and released in 1996. It is this version still in use today, which is a testimony to its stability. The idea was

to create a version of LaTeX (called the kernel) that wouldn't be altered. Any extensions were to be acomplished by the use of "packages". Just how these packages are used will be explained as we work through the structure of a LaTeX document.

2 Some TEX Basics.

First, if your computer doesn't have one, install a *Tex Implementation*. For Mac users running OS X, go to http://www.uoregon.edu/~koch/texshop/obtaining.html. In the second paragraph click MacTeX.mpkg.zip if you have the required room. If not, get the current version of TeXShop from the first paragraph. If you're a Windows users you'll need a TEX editor and a TEX compiler. For an editor go to http://www.texniccenter.org and download *TexnicCenter*. For a TEX compiler go to http://www.miktex.org and download *MikTex*. Some Windows users prefer *WinEdit*, a commercial product that can be obtained from http://www.winedt.com Unix and Linux users can get information at http://www.tug.org/texlive/.

To produce a document using TEX or LaTeX , the user (you) produces a "source file" using your text editor from your *Tex Implementation*. The source file contains text and commands using only standard (ASCII) typewriter keystrokes. Compiling the source file with the compiler provided by your *Tex Implementation* produces the "output file" as well as a log file. Any "errors" in the source file are listed in the log file with help finding the errors and hints to correct them. Examples will follow after covering some TEX basics.

The existence of *commands* in a TEX source file is what distinguishes TEX and LaTeX from generic word processing programs such as Microsoft Word. Some commands determine the overall formating of the document such as text height, text width and margins, while others standard characters with accents or underlining, while still others are used to enter special content elements such as Greek letters and mathematical symbols.

2.1 Special Keystrokes.

Certain keystrokes are not interpreted as text by TEX, but rather have special meaning. They are: `\, $, _, ^, &, ~, %, {, }`, and `#`. The keystroke, `\` tells the TEX compiler that what follows is a command; not text. The use of the `$` is to toggle between in line text mode and in line math mode. In math mode the shape of the characters is different. For example, to produce $f'(x) = x$, type `$ f'(x)=x $`. Note that *italics* is used in math mode. Another feature of math mode is that all space bar keystrokes are ignored. Consequently typing `$ f ' ( x ) = x $` produces the same output as `$ f'(x)=x $`. Two `$$` toggles between in line text and displayed style math mode.

The `&` is the alignment character. It is used to separate columns in a list such as a table or a matrix, or to set the point of alignment in a multi-line mathematical calculation. You may already be familiar with the use of `_` to

denote subscript and ^ for superscript. Both can be used only in math mode. The ~ is called a "sticky space". It produces a space, but TEX isn't allowed to break a line at a sticky space. Typing % tells TEX to ignore all of the following text until the next carriage return. The two braces {} are used to enclose mandatory arguments for some types of commands TEX will issue an error message if they don't come in matching pairs. The # key is sometimes used when creating user designed commands.

Each of these characters can be inserted as text in a document as indicated in the following table.

To produce	Type
\	`$\backslash$`
$	`\$`
_	`\_`
^	`\^{}`
&	`\&`
~	`\~{}`
%	`\%`
{	`\{`
}	`\}`
#	`\#`

2.2 Example

A simple example will illustrate the use of some of these special keystrokes. The following text could easily appear in any elementary geometry textbook.

The Pythagorean Theorem states that the length of the hypotenuse, h, of a right triangle is related to the lengths, a and b of the other two sides by the formula, $h^2 = a^2 + b^2$.

It is produced by typing

```
\documentclass{article}
\begin{document}
The Pythagorean Theorem states that the length of the hypotenuse,
$h$, of a right triangle is related to the lengths, $a$ and $b$
of the other two sides by the formula, $h^2=a^2+b^2$.
\end{document}
```

The three lines

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

are examples of commands. The important aspect of this example is how the formula $h^2 = a^2 + b^2$ is produced. It also illustrates one use of { and }. For now we will begin the study of commands.

2.3 Simple Commands.

The simple commands come in two varieties. First are those starting with a \ followed by one non alphabetic character. There are several examples of such commands in the above table. Other examples of commands of this type are illustrated in the subsection, "Spacing" on page 6. In addition many accent marks are produced using a command of this type. See page 62 for a list of some accents. For example à is produced by typing `\‘a`. (For clarity, some authors type `\‘{a}`.) A simple command applies only to the character immediately following it; it can't apply to a space, which explains why it's necessary to type `\^{}` to produce ^.

A noteworthy example of a command of this simple type is a backslash followed by a space; that is, \␣ (A space is denoted by ␣.). In text mode, TEX interprets several consecutive spaces as just one space. In math mode, TEX ignore spaces completely. But the command \␣ will produce a space in the output file no matter what comes before it.

The second variety of simple command starts with a \ followed by one or more alphabetic characters; no non-alphebetic characters are permitted. In particular a space isn't permitted. In fact a space or any non alphabetic character indicates the end of the command. Moreover these commands are case-sensitive. All of the Greek letters are produced with such commands. (For a list of them again see page 62.) For example to produce σ , type `$\sigma$` while Σ is produced by typing `$\Sigma$`. (The summation symbol, $\sum$ is produced by typing `$\sum$`.) Note that all Greek letters must be produced in math mode. Epsilon, theta and phi have two forms. Neither form of epsilon is used to denote membership in a set. For example $x \in A$ is produced by typing `x\in A`. Special letters such as æ, and all mathematical symbols are produced with such commands. (See pages 62–65.) Another such command worthy of special attention is `\dots`, which is used in such expressions as $1, 2, \ldots, n$. **Do not** produce these dots with a string of 3 periods. Two additional commands of this type that are of special interest are `\i` and `\j` . They produce the corresponding letter, but without the dot allowing an accent mark is placed over either letter. For example, typing `\u\i` produces ĭ, which looks better than ĭ. In mathematical expressions rather than using `$l$` for the letter, it's better to use `$\ell$`. The first produces l, while the second produces ℓ.

Another example of a command that is quite useful and, in fact, has been used several times in this document, is the command `\verb` , by which text can be made appear in the output document exactly as it is typed from the keyboard. First select a keyboard character that won't be included in the special text. Then type `\verb` followed by this character, the text to appear as typed, and concluded by the special character. For example, type `\verb~\ell~` to produce `\ell`.

Warning. Using a space to end a command of this type can cause a problem if a space is desired after the command. Rather than typing just a space, instead type the command \␣ introduced above. It not only signals the end of the command, but also inserts a space. On the other hand if the command is to

end the sentence, just type a period. It signals the end of the command and produces the period. The same is true of the other sentence-ending characters; ! and ?.

2.4 Advanced Commands.

These commands also begin with a \ followed by a string of letters, but in addition they require one or more mandatory arguments. Also optional arguments may be permitted. Mandatory commands are placed inside of a pair of braces, { }, while the optional ones are placed between a pair of brackets, [ ]. The order in which the arguments are listed can be important. Additional accents are produced with elementary commands of this type. For example to produce o̧, type `\c{o}` while typing `\r{o}` produces o̊. Again see page 62 for a complete list of accents. Another example is the command, `\underline` used to underline text. For example <u>text to be underlined</u> is produced by typing `\underline{text to be underlined}`.

Often more than one mandatory argument is needed. For example to create used defined commands are created with the command `\newcommand`, which has two mandatory arguments. The first contains the name of the command being created (starting with at \) and the second being the definition of the command. For example the command `\etec` is created by typing `\newcommand{\etec}{enterotoxigenic E. coli}`. Then typing `\etec` enters, "enterotoxigenic E. coli" without being concerned about mistyping it.

For the two special commands, `_` and `^` arguments are sometimes necessary but not always. For example to produce x_1 it suffices to type `$x_1$`, but to produce x_{n+1} it's necessary to type `$x_{n+1}$`. We will encounter many, many more commands of this type as we go along.

2.5 The Hyphen Key.

The hyphen key is used to produce four horizontal lines of different lengths. First is the usual hyphen obtained by striking the hyphen key once while in text mode. In math mode striking the hyphen key once produces a minus sign. Compare the hyphen, -, to the minus sign,−. In regular text mode two consecutive strikes of the hyphen key is use to indicate a range of values. For example pages 4–9 is produced by typing `pages 4--9`. And finally three consecutive strikes of the hyphen key produces a dash. For example, "The Hyphen Key—A Versatile Character".

Even though TEX hyphenates automatically, there are cases where a hyphen can be used, but TEX doesn't insert one. For example, if a prefix begins a word that TEX knows how to hyphenate, TEX will not know how to hyphenate the result. For example, TEX knows how to hyphenate, "private", but not, "semiprivate". In such cases the command `\-` can be used to inform TEX where hyphens may be inserted.

2.6 Quotes.

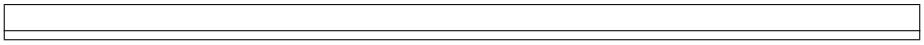
Quotation marks are often wrongly typed because on a typewriter they are produced from just one keystroke. But, as you know, in printing there's a difference between opening and closing quotation marks. To get the opening quotation marks, hit the ` key (the lower case of the tilde key) twice, producing “. Right quotation marks are obtained by striking the ' key (the lower case of the quotation marks key) twice, which produces ”.

2.7 Spacing between Words and Sentences.

In TEX the space between words and the space between sentences are different and both are “rubber” spaces; that is, they can be changed slightly in order to “justify” the text. It's reasonable to assume that a single strike of the space bar determines between word space. But two strikes of the space bar doesn't determine between sentence space because TEX interprets several consecutive strikes of the space bar as just one strike. So TEX interprets the space following a period as between sentences space if the letter before the period is a lower case letter. If the period is preceded by an upper case letter, then TEX thinks it's just the end of an abbreviation and inserts between words space. So if a sentence ends with an upper case letter followed by a period, such as, “I belong to AARP.” TEX is told that the period is indeed the end of a sentence by typing `AARP\@.` instead of just `AARP.` On the other hand, if an abbreviation ends in a lower case letter, such as etc., but isn't at the end of the sentence type `etc.\␣` to tell TEX that it should insert just between words space rather than between sentence space. Another possibility is to use the sticky space instead of the space bar; that is, typing `etc.~` but don't strike the space bar after the `~`. Keep in mind that TEX isn't allowed to break a line at a stick space. So it's always a good idea to follow abbreviations such as Mr., Mrs., Ms., Dr., etc. with a sticky space.

2.8 How TEX Views a Line

TEX views almost everything in terms of boxes; every character is placed in a box, every word is a box consisting of all of the characters in that word, etc. In particular every line in the output document is a box, which looks like the following box.



The line through the middle of the box is called the `baseline` and the distance between the baseline of two consecutive baselines is `\baselineskip`. It's in the form of a command because it can be used as a (vertical) length. See Section 2.10 below. The distance from the baseline and the top of the box is `\height` and the distance from the bottom of the box to the baseline is `\depth`. Finally the `\totalheight` is the sum of `\depth` and `\height`.

2.9 Horizontal Space.

Because TEX adjusts horizontal space automatically, it's usually not necessary for the user to make any adjustments. But when it is necessary, there are six standard positive horizontal space commands for that purpose: `\,`, `\:`, `\;`, `\quad`, `\qquad` and `\␣`.

The command	Produces this much space
`\,`	| |
`\:`	| |
`\;`	| |
`\quad`	| |
`\qquad`	| |

In addition `\!` produces negative space equal to that of `\,`. The command `\,` is used most often in integration formulas to put a small amount of space between the function and the differential. For example, to produce $\int f(x)\,dx$ type `$\int f(x)\,dx$`.

A specific amount of horizontal space is created with the command `\hspace{`*length*`}` where the length must be given units of some kind; for example, in, pt, mm, cm and em (the width of the letter, M). For a complete list of acceptable measures of length see page 62. Horizontal space created this way is ignored at the beginning of a line. If some horizontal space is desired even at the beginning of a line, use `\hspace*{`*length*`}` instead.

It's also possible to enter horizontal space equal to the space occupied by some specific text with the command `\phantom{`*block of text*`}`. For example typing `\phantom{text}` enters horizontal space equal to that used by the word, "text". Enter space equal to "text". This command can be very useful as will be seen.

Another very useful horizontal space command is `\hfill`. It can be used to evenly space two, three or more entries along a line such as

MTH 132 Section 23 First Hour Exam January 26, 2007

which is produced by typing

```
MTH 132 Section 23\hfill First Hour Exam\hfill January 26, 2007.
```

The command `\hfill` has two cousins: `\dotfill` and `\hrulefill`. The first fills the space with dots on the baseline; the second, with a solid line. For example

Student Name_____ Student PIN_____ Section number_____

is produced by typing

```
Student Name\hrulefill\hrulefill\hrulefill
Student PIN\hrulefill\hrulefill Section number\hrulefill
```

Note how the multiple appearance of the command `\hrulefill` affects the length of each line. The same is true of `\hfill` and `\dotfill`. These space-

filing commands can be used together as in

MTH 132 First Hour Exam Section Number_____

2.10 Vertical Space.

First, it should be noted that one carriage return is the same as one strike of the space bar in the output file. This feature makes it easy to find specific parts of a document, such as a short mathematical expression. Another feature is in locating errors in the source file. When the compiler runs on the source file, if it finds a TeX error (which it often does), it identifies the line in the source file where the error occurs. So creating new lines in the source file with carriage returns makes it easier to find errors. This process works because a single carriage return starts a new line in the source file, but simply produces a space in the output file. To begin a new paragraph in the output file enter two consecutive carriage returns in the source file. More than two consecutive carriage returns is the same as two carriage returns. Vertical space in the output file can't be created with additional strikes of t he carriage return.

There are three vertical space commands that create a small amount of vertical space: `\smallskip`, `\medskip`, and `\bigskip`. These three distances are rubber and therefore can be adjusted by TeX to maximize the look of the document. Specific vertical space is inserted with the commands `\vspace{`*length*`}` and `\vspace*{`*length*`}`. They are completely analogous to the corresponding horizontal space commands. Finally the command `\vfill` does for vertical distance what `\hfill` does for horizontal distance.

3 The Structure of a LaTeX Document.

A LaTeX document consists of a preamble and the body of the document. Usually, but not always, the document ends with a bibliography. We begin with the preamble. The preamble of every LaTeX document begins with the command `\documentclass{`*class_name*`}`. The most common choices for *class_name* are: `article`, book, report, beamer, and . Many of the documents you will produce will be done in `article` class. (The class `beamer` is used for presentations. It and the `letter` class will be discussed later in this document.) Once the document class has been selected a large number of commands become available and several parameters are set. These will be discussed later in this section.

The `documentclass` command has an optional argument placed between `documentclass` and the mandatory argument *class_name*; specifically `\documentclass[`*option1, option2, ...*`]{`*class_name*`}`.

3.1 Popular Class Options.

Except for `beamer`, the default font size (size of the characters) is 10pt. There are two other possibilities; namely, 11pt and 12pt, that can be specified in the

optional argument. (Producing larger characters will be discussed later.) The paper size may also be changed in the optional argument. The default setting is 8.5in × 11in, called `letterpaper`. Some other choices are `legalpaper` and the most common Eurpoean paper size, `a4paper`. In addition, it's possible to type a document in two column form by placing, `twocolumn` in the optional argument. For example,
`\documentclass[12pt,a4paper,twocolumn]{article}` tells LaTeX that the font size to used is 12 point, the paper size to use is a4 paper and that the document is to be typeset in two columns.

Long documents that are to be bound, such as a book, usually have text printed on both sides of the paper and have a slightly larger margin on the inner side of each page (left for odd pages and right for even pages) to facilitate binding. For that reason in the `book` class the margin settings for odd and even pages are different. It's possible to impose this same margin structure in the `article` and `report` classes by placing `twoside` in the optional argument. (The default is `oneside`.) See the discussion of page styles on page 11 to better understand what these different selections change.

When a displayed mathematical expression is numbered, the number is placed after the display flushed to the right margin. To move the number to the left of the display flushed with the left margin, use the option `leqno`. Displayed mathematical expressions are automatically centered. To force them to be flushed to the left margin, use the option `fleqn`.

3.2 Other Preamble Entries.

After selecting the document class and any desired options, the next choices to make are the packages to be used. This is done with the command `\usepackage` that has one mandatory argument which may be preceded by an optional argument. For any document containing a substantial amount of mathematical material, the packages to use are: `amsmath`, `amssymb`, and `amsthm`. There are others, too numerous to list here, that are needed for specific purposes. Several examples such as `url` and `graphics` will be discussed in the sequel.

The title of the document is created with the command `\title{`*title*`}`. LaTeX will automatically center the title and start new lines in the case of title too long to fit on one line. If the title is lengthy, a shorter version can be indicated for use in headings or other such structures. The syntax is `\title[`*Short Title*`]{`*Title*`}`.

The command `\author` is used to enter all author information. For example to enter the name and address of just one author, type
`\author{William R. Author, 8426 North Main Street, Fairview, VA}`.
The author information will be centered on the line immediately following the title. However, the user must put in line breaks with `\\` if the text runs into the margin. A carriage return will not create a new line. Alternatively, typing
`\author{William R. Author\\8426 North Main Street\\Fairview, VA}` will list the information as

William R. Author

8426 North Main Street
Fairview, VA

For multiple authors, use `\author{`*author1_info*`\and` *author2_info*`}` etc. If only names are used, LaTeX will display them centered on one line. If addresses are included, they are displayed according to how they are typed. For example typing

```
\author{William R. Author, 8426 North Main Street, Fairview, VA}
\and
\author{Irma A. Writer, 1842 West Jefferson Street, Washington, NV}
```

produces

William R. Author, 8426 North Main Street, Fairview, VA
Irma A. Writer, 1842 West Jefferson Street, Washington, NV

while typing

```
\author{William R. Author\\8426 North Main Street\\Fairview, VA}
\and
\author{Irma A. Writer\\1842 West Jefferson Street\\Fairview, VA}
```

produces

William R. Author	Irma A. Writer
8426 North Main Street	1842 West Jefferson Street
Fairview, VA	Fairview, VA

Like the `\title` command, the `\author` command also has an optional argument, which is used to enter a short version of the authors. For example, if there are five authors for a document, the short version could include only first initials and last names. For example `\author[W. Author and I. Writer]` ....

LaTeX will automatically enter the current date centered under the author(s), unless a different date is selected by using the command `\date{`*desired_date*`}`. To omit the date entirely, type `\date{}`.

The best way to get double spacing or one-and-one-half spacing is to use the package, `setspace` by typing `\usepackage{setspace}` in the preamble. To use double spacing or one-and-one-half spacing throughout the document, type the command `\doublespacing` or `\onehalfspacing` in the preamble. To produce a portion of the document in double spacing, precede that portion of the text with `\begin{doublespace}` and end it with `\end{doublespace}`. For one-and-one-half spacing, replace `doublespace` with `onehalfspace`. If `\begin{doublespace}` occurs after the beginning of a new paragraph, the entire paragraph is double spaced.

Other items to include in the preamble will be discussed later.

3.3 The Start of the Body.

To begin the document, type `\begin{document}` followed by `\maketitle`. In the `article` class, the title appears on the first page of the document. But it's possible to force the production of a separate title page with the option `titlepage` in the `\documentclass` command. The body of the document ends with the command `\end{document}`.

If an abstract is desired, begin by typing `\begin{abstract}`. Then type the text of the abstract and finish it with `\end{abstract}`. The abstract will appear as it does in this article, immediately following the author information (and the date if one is included). Note that the margins are increased and the font size is smaller. These adjustments are made automatically by LaTeX. Producing an abstract is an example of a *environment*, a subject to be discussed in great depth later in Subsection 5 on page 18.

3.4 Page Style.

A page may have a header and/or a footer. The information that appears in them is determined by the `pagestyle`. The four possible page styles are in the following list. The default page style for the `article` and `report` classes is `plain` and for the `book` class, is `headings`.

`plain` The header is empty and the page number appears in the center of the footer at the bottom of the page.

`empty` The header and footer are both empty. No page numbers are printed anywhere on the page.

`headings` This is the page style that should be used if a heading at the top of each page is desired. The header contains both the page number and a page title. If the `twoside` option has been selected, then the page number appears in the right side of the header on odd pages and in the left side of the header on even pages. In both cases the heading is the section title (in the `article` class) or the current chapter title (in the `book` and `report` classes). The page style used for this document is `headings`.

`myheading` This page style is identical to the `headings` page style except the user selects the headings with the command `\markright{`*heading*`}` for the same heading on both odd and even pages or for different headings, `\markboth{`*odd_page_heading*`}{`*even_page_heading*`}`. The heading command used should appear in the preamble, but the heading can be changed at any point in the document by repeating the command at the point where the change is to be made with the new heading(s) in the argument(s) of the command.

To change page style from the default, type `\pagestyle{`*style*`}` in the preamble. Typing the same command at an appropriate place in the body changes the page style for the current page and all subsequent pages. The command `\thispagestyle{`*style*`}` changes the page style for only the current page.

Warning. Part of the command `\maketitle` is `\thispagestyle{plain}`. Consequently if the page on which the title is to appear should have no page

number at the bottom, the command `\thispagestyle{empty}` **must** be inserted **after** the `\maketitle` command.

If the `myheadings` page style doesn't supply enough flexibility, the package, `fancyhdr` provides an addition page style called `fancy`, which allows the user to design the headings. Details can be found on page 224, section 4.2.2 of [6].

3.5 Page Numbering.

The page style determines where the page number will appear on the page. Wherever it appears the page number may be displayed in any of the styles `arabic, roman, Roman` (upper case roman), `alpha`, and `Alpha` (upper case alphabet), with arabic being the default. The style is changed using the command `\pagenumbering{`*numbering style*`}`. For example `\pagenumbering{Roman}` numbers the pages using upper case Roman numerals. The actual page number (no matter in what style it is displayed) is set using the *counter* called `page`. (LaTeX has a large number of counters. We will meet more of them later.) The value of that counter is incremented each time a new page is created even if the page number isn't printed anywhere on the page. Also the value can be changed using the command, `\setcounter{page}{`*page number*`}` or `\addtocounter{page}{`*adjustment*`}`. One example where the ability to use different number styles and to adjust the corresponding counter is an article that begins with a preface and/or a table of contents or any such elements that should be numbered in lowercase roman numerals while the remaining text should be numbered in arabic. For example to begin with a table of contents, after the `\begin{document}` command, type `\pagenumbering{roman}` and then type `\tableofcontents` followed by `\newpage` to begin a new page and then type the command `\pagenumbering{arabic}` followed by `\setcounter{page}{1}`. The result will be that the next page to be printed will be numbered 1. For example, the beginning of the body for the source file of this document is

```
\begin{document}        \newpage
\maketitle              \pagenumbering{arabic}
\pagenumbering{roman}   \setcounter{page}{1}
\tableofcontents        \begin{abstract}
```

3.6 Page Layout.

Figure 1 below shows what a page in LaTeX looks like.

The circle is at 1 inch from the top and left of the page. All of the dimensions shown in the figure are self-explanatory except for `\oddsidemargin`. For documents prepared in the `oneside` style, this command controls all indicated margins. But when the `twoside` style is used (either by default as in the `book` class) this command sets the margin for odd pages while the margin for even pages is set be the command `\evensidemargin`. Two dimensions that are not displayed in the figure are `\paperheight` and `\paperwidth`. The first is the sum of all of the vertical dimensions and the second, the sum of all of the horizontal dimensions. All of these commands represent lengths that are set automatically

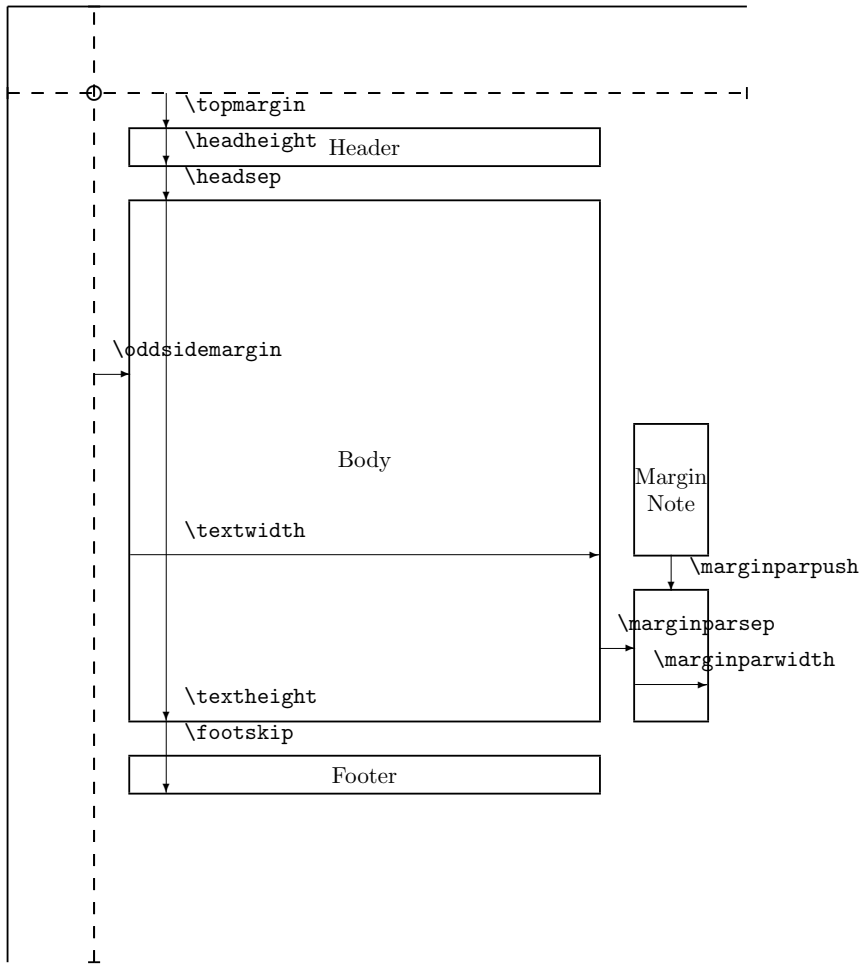


Figure 1: Page Layout Parameters

by the selected *document class* and any selected options. Consequently they can be used in space commands such as `\hspace` and `\vspace`. For example `\hspace{.5\textwidth}` will insert horizontal space equal to half of the text width.

These length can be changed, however because of the connections between these dimensions, they can't be changed independently. But there are situation in which changing some of them would be beneficial. For example, for a one page document, such as an exam, it may be desirable to expand the height and/or width of the text on the page. To increase the text height, first reduce the `topmargin` by typing

`\addtolength{\topmargin}{-2in}`. (As with counters, LaTeX has several *lengths* all of which can be altered in a similar fashion.) Then type `\addtolength{\textheight}{2in}`. Subtracting and adding the same amount is essential to maintain the sum of all of the vertical lengths. Provided you haven't selected the option `twoside`, to change the width, first type the command `\addtolength{\oddsidemargin}{-2in}` followed by `\addtolength{\textwidth}{2in}`. (With `twoside` you must also subtract 2 inches from `\evensidemargin` as well.) With these two changes, the text will be 2 inches wider and 2 inches higher, but will be centered on the paper. The other lengths should remain as they are set by LaTeX.

Because all of these dimensions are lengths, they may be used in the mandatory arguments of spacing commands; for example `\hspace{.5\textwidth}` will insert horizontal space equal to half of the text width.

4 The Body of a LaTeX Document Continued.

4.1 Fonts — Families, Series and Shapes.

In the standard TeX implementation there are three font families, two font series and four font shapes. The three font families are: Roman, typewriter and sans serif. (Sans is the French word meaning without and serifs are the short lines that appear in some letters such as T which, in sans serif, is T.) The default family for the `article, book, report` and `letter` document classes is Roman while sans serif is the default for the `beamer` document class. Examples of the typewriter font have appear in the previous sentence and is many other places in this document. Such text is produced by typing `\texttt{`*text to be in the typewriter font family*`}`. To change to sans serif, type `\textsf{`*text to be in sans serif*`}`, which will produce text to be in sans serif. It's possible to change back to Roman briefly using the command `\textrm`. For example typing `\textsf{some text to be in sans serif \textrm{and} one word in Roman}` produces some text to be in sans serif and one word in Roman. (A popular alternative to the commands introduced above is the *declaration* form. For example to enter text in, say, the sans serif family, type `{\sf` *text to be produced in the sans serif family*`}`, but this form has some serious limitations as explained below.)

The two font *series* are medium (default) and boldface. All three font families are available in medium, but only Roman and sans serif text can be produced in boldface. To produce boldface use the command `\textbf`. For example to produce **text in boldface** type `\textbf{`*text in boldface*`}`. The command `\textmd` can be used to insert non-bold text within a string of bold text. **Some bold face text** with **one word of text in medium series.** To get **Some text in sans serif boldface**, type `\textbf{\textsf{`*Some text in sans serif boldface*`}}`. (This "nesting" of commands doesn't work if the *declaration* form is used.)

The four font shapes are upright (default), italics, small caps and slant. All three font families are available in the upright, italics and slant, but sans

serif isn't available in small caps. To change from one shape to another use `\textup, \textit, \textsc,` or `\textsl`. For example to produce TEXT IN SMALL CAPS type `\textsc{`*Text in Small Caps*`}`. To produce *Text in sans serif italics* type `\textsf{\textit{`*Text in sans serif italics*`}}`.

Warning. When changing from the italics (or slant) to either of the other two shapes, a very small amount of extra space is needed to account for the right-leaning of the last italic character. The *declaration* forms don't insert this needed space. Compare the following two identical pieces of text, the first produced with the declaration form and the second, with
`\textit{fast talk} is to be ignored`.
fast talk is to be ignored.
fast talk is to be ignored.
Note the small amount of extra vertical space after the italics in the second version.

Not all combinations of font families, series and shapes are available. The following table indicates which combinations are available and when available, which are available in bold face as well.

	Roman	Typewriter	Sans Serif
upright		no bold	
italics		no bold	NA
small caps		no bold	NA
slant		no bold	no bold

Table 1: Font Families vs Shapes

4.2 Character Sizes.

All three of the font families are available in 10 different sizes listed in the following table. The corresponding command is used in a declaration form to produce characters of that size. The default size is `\normalsize`.

`\tiny`	smallest	`\Large`	larger
`\scriptsize`	very small	`\LARGE`	larger yet
`\footnotesize`	smaller	`\huge`	still larger
`\small`	small	`\Huge`	largest
`\normalsize`	normal		
`\large`	large		

For example to change a few words to a larger size for emphasis, type `{\LARGE Pay very close attention to this}` which will come out as Pay very close attention to this. Note that these character sizes are imposed using the *declaration* style. These sizes are computed in proportion to the font size selected in the optional argument to `\documentclass` in the preamble. Any one of these commands can be put in the preamble resulting in the entire body of the document being produced in that size.

4.3 Line and Page Breaking.

The command `\newline` will fill the remainder of the current line with space and start a new line. But it's better to use `\\` which, by itself is the same as `\newline`, but has an optional argument. For example `\\[2mm]` will fill the remainder of the current line with space, and then skip down 2mm before starting the next line. This feature is useful and can be employed anywhere the `\\` command is used. It also has a cousin, `\\*` which prevents a page break from occurring before starting the new line. The same optional argument `[2mm]` is available for this form of the newline command.

A related command is `\linebreak` which has one optional argument. The major difference between it and `\newline` is that the remaining space in the current line is distributed between the words already in the line resulting in a line that ends at the right margin. If the remaining space is substantial, the result can be ugly. The optional argument is `[`*imp-digit*`]`, where *imp-digit* is an integer between 0 and 4 establishing the importance of breaking the line at that point. A value of 0 means it's not too important to break there, whereas `\linebreak[4]` means the break is mandatory; that is, it's equivalent to `\linebreak`. The command `\nolinebreak[`*imp-digit*`]` works in exactly the opposite way to prevent a line break. There are other ways to prevent a line break. For example, using a "sticky space", `~` as in `Dr.~Smith`. Sometimes it's desirable to keep an entire string of words on one line. The easiest way to accomplish this is as follows. `\mbox{`*the string of words to be kept on one line*`}`. LaTeX treats what's in the brackets as one object rather that several objects. The `\mbox` command is described in detail on page 34.

The commands related to page breaks are `\newpage, \pagebreak[`*imp-digit*`]`, and `\nopagebreak[`*imp-digit*`]`. These commands are analogous to the corresponding ones for line breaking.

4.4 Document Divisions.

The three LaTeX document classes, *book, report, and article* provide the user with the ability to organize a document into divisions, subdivisions, etc. The primary division name for the *article* document class is *section*, followed by subdividions *subsection, subsubsection, paragraph* and *subparagraph.* To begin a section, the user types `\section{`*Section Title*`}`. When the file is compiled, LaTeX automatically does the formating of the section title putting some vertical space before the title, producing the title in bold face and in a larger font size. In addition the sections are automatically numbered consecutively. This feature means the author doesn't need to remember what the next section number should be and permits the insertion of a new section in a revision of the document with the assurance that the renumbering of the sections will be done right. Dividing a section into subsections is accomplished with the command `\subsection{`*Subsection Title*`}`. As with section titles, LaTeX does the formatting and the numbering. A subsection is numbered in the form $m.n$ where m is the number of the section in which the subsection appears and n means that

the subsection is the n^{th} subsection in the section.

The document class *report* adds as the primary division name, *chapter*. A chapter is begun with the command `\chapter{`*Chapter Title*`}`. A new chapter is automatically begun on a new page and all chapters are numbered consecutively with *sections* numbered in the form *m.n* and then subsection numbered in the form *m.n.k* etc. The document class *book* adds at its primary division name, *part*. The command `\part{`*Part Title*`}` begins a new page and always on an odd numbered page.

All of these division commands have an optional argument that allows a shortened version of the division name to be entered. For example `\section[`*The Body Continued.*`]{`*The Body of a LaTeX Document Continued.*`}`. When available, the short version is used in heading as in the top of this page.

4.5 Labeling.

Labeling is the facility by which the author may automatically refer to any part of the document from any other place in the document. First, to refer later in the document to the page on which some specific text appears, at the point in the source file where that text is entered, simply type `\label{`*marker*`}`. Then the command `\pageref{`*marker*`}` will insert the number of the page on which the label *marker* appears. For example in the source file typing
`this is important text \label{imp} that will be referenced later` and later typing, `on page \pageref{imp} you will find some important text` will insert the number of the page where that `important text` appears.

To refer to a specific section of the document by section number and page number, begin the section with `\section{`*Section title*`}\label{impsec}`. Then later in the document to refer to that section, type
`by Section \ref{impsec} on page \pageref{imp}` and the section number will be inserted as well as the page on which it appears. Subsections and all of the other *divisions* discussed above can be referenced in a similar manner. Other items that are numbered automatically by LaTeX, such as tables, figures, equations, and bibliographic entries, can be labeled and referenced similarly. Specifics will be discussed when each topics is discussed.

4.6 Footnotes.

To enter a footnote about some word or phrase, immediately after the text to be footnoted, type `\footnote{`*the text for the footnote*`}`. LaTeX will draw a short line at the bottom of the page followed by the footnote number and footnote text. The footnote number is obtained from the counter, `footnote`. To use an number other than that provided by the counter, type
`\footnote[number]{`*text*`}`.

There is a variant of footnoting specifically designed for authors wishing to thank a funding agency or a person for support. In the preamble, immediately after the *author name* type `\thanks{`*the gushing note of thanks*`}`. The thanks will appear as a footnote, but instead of a number, with a symbol, such as an *.

For example `\author{William R. Author\thanks{Supported in part by the NIH.}, 8426 North Main Street, Fairview, VA}`

The footnote marker can be changed using the command `\renewcommand{\thefootnote}{`*marker_style*`{footnote}}` where the other choices for *marker_style* are `\roman, \Roman, \alph, \Alph` and `\fnsymbol`. The last choice uses *, † and 7 other symbols.

4.7 The Auxiliary File.

All of the data needed to do the automatically numbering and referencing that have been discussed are stored in a separate file *file_name*.`aux` which is created by LaTeX the first time the file *file_name* is compiled and is rebuilt on each subsequent run. For that reason, sometimes the source file needs to be compiled twice for some of the automatic features to appear. This aspect will be even more apparent when using the programs BibTeX and `MakeIndex` discussed later.

5 Introduction to Environments.

Having discussed what is necessary to know to produce a document consisting mainly of simple text, we now learn how to produce special structures such as quotations, lists, tables, figures, bibliographies, and displayed mathematical expressions. Such structures are constructed in *environments*. An environment begins with the command `\begin{`*environment_name*`}`, some of which have mandatory and optional arguments.An environment ends with the command `\end{`*environment_name*`}`. In fact the entire document is typed in the *document* environment so when LaTeX encounters the command `\end{document}` it stops compiling text. (Consequently, extra text can be safely stored after the `\end{document}`.) Another environment already discussed is the *abstract* environment. Several environments will be introduced in what follows. We start with another easy example which is similar in structure to the *abstract* environment.

5.1 Center.

To center text or any structure, type
`\begin{center}`
The text or other structure to be centered
`\end{center}`.

The amount of text to be centered may be longer than one line as in the following example.

This text is being centered on the page using the *center* environment which inserts line breaks automatically, but, obviously, doesn't justify. But you may start a new line anywhere you wish with the command `\\`.

The command `\begin{center}` automatically ends the previous paragraph. The same can't be said for other environments as will be seen.

5.2 Quotes.

There are two environments for displaying quotes; *quote* and *quotation*. Both indent the text to be quoted by equal amounts on both sides and present the quoted text is smaller font size. The difference is that *quote* starts a new paragraph by inserting an empty line, but no indent, while *quotation* starts a new paragraph by indenting first line, but doesn't skip a line. For example typing

```
\begin{quote}
It hits on a dull, overcast Monday morning. I awake realizing
there is no party in sight for the weekend. I'm out of bread,
and I've got a dry skin problem. So I say it aloud to myself,
``What's a nice girl like me doing in a dump like this?''

The draperies are dirty (and will disintegrate if laundered),
the arms of the sofa are coming through. There is Christmas
tinsel growing out of the carpet. and some clown has written
in dust on the coffee table, YANKEE GO HOME.
\end{quote}
```

will produce

> It hits on a dull, overcast Monday morning. I awake realizing there is no party in sight for the weekend. I'm out of bread, and I've got a dry skin problem. So I say it aloud to myself, "What's a nice girl like me doing in a dump like this?"
>
> The draperies are dirty (and will disintegrate if laundered), the arms of the sofa are coming through. There is Christmas tinsel growing out of the carpet. and some clown has written in dust on the coffee table, YANKEE GO HOME.[1]

Replacing `quote` by `quotation` results in

> Last December world leaders met in Copenhagen to add more hot air to the climate debate. That is because although the impacts humanity would like to avoid—fire, flood and drought, for starters—are pretty clear, the right strategy to halt global warming is not. Despite decades of effort, scientists do not know what "number"—in terms of temperature or concentrations of greenhouse gases in the atmosphere—constitute a danger
>
> When it comes to defining the climate's sensitivity to forcings such as rising atmospheric carbon dioxide levels, "we don't know

[1] At Wit's End, Erma Bombeck, Fawcett Publishing, 1965.

> much more than we did in 1975," say climatologist Stephen Schneider of Stanford University, who first defined the term "climate sensitivity" in the 1970s. "What we know is if you add watts per square meter to the system, it's going to warm up."[2]

5.3 Verbatim Text.

The text above that looks like it came from a typewriter was produced using the *verbatim* environment. All text typed between `\begin{verbatim}` and `\end{verbatim}` is typeset exactly as it is typed. In particular the user must insert the carriage returns. A short piece of text that appears in line is done with the command `\verb` followed by any character that isn't part of the verbatim text to follow. The text to be so printed is ended with the same character it began. For example to produce `verbatim text` type `\verb=verbatim text=`. Any keyboard character other than `v e r b a t i m t e x t` could be used in place of `=`. The `verbatim` environment is commonly use to insert a computer program in a document.

5.4 Lists

Next we introduce the list environments: *itemize, description and enumerate*. Each generates a list of items, but they differ in how the items are presented. Itemize and enumerate indent and mark each item, while description indents just a very small amount, but doesn't mark. Itemize marks each item with a • while enumerate marks items with consecutive numbers (arabic, roman, Roman, alph, or Alph). All three list structures have the same basic format; namely,
`\begin{`*list_name*`}`
`\item` The first item in the list.
`\item` The second item in the list.
`\item` The third item in the list.
`\end{`*list_name*`}`

To demonstrate the difference the same list will be set in all three list environments beginning with itemize.

- The first item in the list.
- The second item in the list.
- The third item in the list.

Next the description environment is demonstrated.

The first item in the list.

The second item in the list.

[2] *Scientific American*, David Biello, January 2010, page 14.

The third item in the list.

Finally the enumerate environment is demonstrated.

1. The first item in the list.
2. The second item in the list.
3. The third item in the list.

A list may included another list as an item. The two lists may have the same or different list names. In the case of the same list, the second level markers are different from the first level. For example, in itemize the second level items are marked with the short dash created with two consecutive hyphens –, the third level, * and the fourth level, ·. For the enumerate environment the second level is delineated by lower case letters enclosed in parentheses, the third by lower case Roman numerals and the fourth by upper case letters.

The command `\item` has an optional argument that can be used to change the default marker. For example typing `\item[(a)] This item` produces, "(a) This item" in any of the three list environments. In the description environment, the marker is flush to the left margin and followed by the small indent characteristic of this list environment.

The `enumerate` permits changing all of the labels in an enumerate list with one command. For example putting `\usepackage{enumerate}` and typing `\begin{enumerate}[a)]` will result in all items having label of the form a) but using the letters of the alphabetical in order. It also eliminates the indent.

a) The first item in the list.

b) The second item in the list.

c) The third item in the list.

To insert text between two items in an enumeration list, end the list after the item where the text is to be inserted. Then begin an enumerate list, but with the appropriate number. To do so the *counter* must be changed. The counter's name is `enumi` (for the first level). It is automatically set to 0 with the `\begin{enumerate}` command. So follow that command with the command
`\setcounter{enumi}{`*the previous number*`}`. The resulting list will begin with the next number.

The above process can be automated as follows. First in the preamble created a new counter called, say *last* by typing `\newcounter{last}`. Then after the last item in the first list and before the `\end{enumerate}` command, type,
`\setcounter{last}{\value{enumi}}`. After beginning the new list environment and before the first item, type
`\setcounter{enumi}{\value{last}}`.

Because some authors find the distance between items in these list environments to be excessive, the `paralist` package was developed to provide compact versions of all three of these list environment. It it contains and extends the `enumerate` package. It makes available the list environments in the right hand column of the following table.

List Name	Compact Version
itemize	compactitem
description	compactdesc
enumerate	compactenum

For example typing

```
\begin{compactenum}[a)]
\item The first item in the list.
\item The second item in the list.
\item The third item in the list.
\end{compactenum}
```

produces
a) The first item in the list.
b) The second item in the list.
c) The third item in the list.

5.5 Tabbing and Tables.

5.5.1 The `tabbing` Environment.

The first method for producing tables is designed to imitate the process of setting tab stops on a typewriter and is called the *tabbing environment*. Once the `\begin{tabbing}` command has been issued, the tabs are set with the command `\=` in either of two ways. You may type the text to appear before the first tab in the first row followed by `\=`, which sets the tab, and proceed similarly until all tabs have been set and then issue a new page command, `\\`. Then type the text to go in the first column of the second row followed by `\>`, (the command that moves to the first tab stop). Continue until all entries in the second row have been entered. For example typing

```
\begin{tabbing}
Department\=Course 1 \=Course 2 \=Course 3\\
Math\>1825\>103\>114
\end{tabbing}
```

produces

Department Course 1 Course 2 Course 3
Math 1825 103 114

If more space is desired before the tab is set, it can be added with any of the horizontal space commands discussed earlier except for `\hfill` and its cousins. For example typing

```
\begin{tabbing}
Department \=Course 1\quad \=Course 2\; \=Course 3\\
Math\>1825\>103\>114
\end{tabbing}
```

produces

Department	Course 1	Course 2	Course 3
Math	1825	103	114

The second method avoids adding extra space. You begin by typing the longest amount of text to appear in the first column of any row followed by the `\=` command and then type the longest text that will appear in the second column followed by `\=`. Proceed to set the remaining tabs in a similar fashion, but at the end of the first line, type `\kill` instead of the usual end of line command. In either case, type the text to go before the first tab followed by `\>` to move to the tab stop and then continue until all of the text for that line is complete. Then enter the `\\` command to start the next line. The alternate method of setting the tab stops is illustrated by typing

```
\begin{tabbing}
Mathematics \=MTH 1825 \=CHEM 117 \=CHEM 203 \kill
Department\>First\>Second\>Third\\
Mathematics\>MTH 1825\>MTH 103\>MTH 114\\
Physics\>PHY 201\>PHY 202\>PHY 300\\
Chemistry\>CHEM 116\>CHEM 117\>CHEM 203
\end{tabbing}
```

which produces

Department	First	Second	Third
Mathematics	MTH 1825	MTH 103	MTH 114
Physics	PHY 201	PHY 202	PHY 300
Chemistry	CHEM 116	CHEM 117	CHEM 203

Note that there's no new line command at the end of the last line. The `\end{tabbing}` ends the line. Also note that it's not necessary to insert a blank line with a carriage return before typing `\begin{tabbing}` because the `tabbing` environment automatically begins a new paragraph. Consequently any strikes of the return key are ignored.

It's possible to insert another tab stop in any line following the first line by just typing `\=` after the last tab stop. However typing `\=` before the last tab stop simply repositions the next tab stop.

If the table produced by the `tabbing` environment is too long to appear on one page, LaTeX will automatically insert a page break and continue the table on the next page.

Warning. The commands `\‘, \’,` and `\=` are redefined in the *tabbing environment*. To produce any of the accents that are normally produced with these three commands, type `\a‘, \a’` and `\a=` instead. For example type `\a’e` to produce é in the `tabbing` environment. In addition the command `\-` is redefined. But new hyphenation in the `tabbing` environment isn't necessary.

5.5.2 The `tabular` Environment.

The second method for producing tables is done in the *tabular* environment, which differs from the `tabbing` environment in two important aspects. First, a table produced by the `tabular` environment must appear on the same page. Consequently if such a table is too long for the current page, LaTeX issues a new page command and puts the table on the next page. Second, a table produced with the `tabbing` environment must be in a separate paragraph, while a table produced with the `tabular` environment can occur in the current paragraph or can be in a separate paragraph. Consequently, to start the table in a new paragraph, the carriage return that was superfluous when starting a `tabbing` environment is required for a tubular environment.

A `tabular` environment begins with `\begin{tabular}[`*pos*`]{`*cols*`}`. The mandatory (second) argument sets the number of columns the table is to have and how the text in each is to be aligned. For example `{lcr}` indicates that there are to be three columns; the text in the first column is to be left aligned, in the second the text is to be centered and in the third the text is to be right aligned. For example the table

C.1	C.2
a	1
b	2

is produced with

```
\begin{center}
\begin{tabular}{cc}
C.1&C.2\\
a&1\\
b&2
\end{tabular}
\end{center}
```

It has two columns; each centered. Note that the text for each column in each row is separated by inserting an &. Also a row is ended with the newline command \\. Vertical lines can be inserted before and/or between columns with the `|` keystroke. For example enter `{|l|c|r|}` puts a vertical line before the first column, between each column and after the last column. For example replacing the first line for the table above with `\begin{tabular}{|c|c|}` results in

C.1	C.2
a	1
b	2

Horizontal lines are inserted with the command `\hline` which is inserted before the first row and after each \\ including the last row in order to produce

C.1	C.2
a	1
b	2

from

```
{\begin{center}
\begin{tabular}{|c|c|}\hline
C.1&C.2\\ \hline
a&1\\ \hline
b&2\\ \hline
\end{tabular}\end{center}}
```

The optional argument `[`*pos*`]` is used only when the table is to appear in the current line rather than in a separate paragraph. The value of the optional

argument determines how the table is to be placed relative to the baseline of the current line. Use `t` to put the top of the table at the baseline of the current line and `b` to put the bottom of the table at the baseline of the current line. If no value is inserted or if `c` is inserted, the table is centered on the baseline of the current page. For example

option `t`

C.1	C.2
a	1
b	2

option `c`

C.1	C.2
a	1
b	2

option `b`

C.1	C.2
a	1
b	2

As was demonstrated in the above examples, it's best to type the first row of the table starting on a new line in the source file. (This isn't necessary, but good for later editing.) Begin with the text for the first column followed by an &, which indicates the end of the text for the first column. Next type the text for the second column of the first row followed by a &. Continue in this fashion until the first row has been completed and type \\ to indicate the start of the second row. Again it's beneficial begin on a new line in the source file. Type the text for the second row in the same fashion as the first row and end with \\. Continue until all rows of the table have been typed. To complete the table, type `\end{tabular}`.

Note that unlike the `tabbing` environment, the width of each column is automatically adjusted to accommodate the longest entry in that column. For example to produce the table

Food	$135	11.25%
Rent	$250	20.83%
Transportation	$320	26.67%
Utilies	$80	6.67%
Medical	$275	22.92%
Misc	$140	11.67%

type

```
\begin{center}
\begin{tabular}{|c|c|c|}
\hline
Food&\$135&11.25\%\\ \hline
Rent&\$250&20.83\%\\ \hline
Transportation&\$320&26.67\%\\ \hline
Utilies&\$80&6.67\%\\ \hline
Medical&\$275&22.92\%\\ \hline
Misc &\$140&11.67\%\\ \hline
\end{tabular}
\end{center}
```

The output would look better if the dollar signs, $, and percent signs, %, were

lined up, which can be done using the structure `@{`*content*`}` in the mandatory argument of the `\begin{tabular}` command. It can be used to insert the same text, symbol or character at the beginning of or at the end of any column. For example to put a dollar sign before each amount in the second column, replace the second occurrence of `c|` in `|c|c|c|` by `@{\ \$}l|`. The purpose of the compulsory space `\ ` is to produce some space before the dollar sign. Without it the dollar sign would rub against the vertical line separating the two columns. Similarly to put a percent sign after each figure in the last column, replace the final `c|` by `r@{\%\ }|`. The result is

Food	$135	11.25%
Rent	$250	20.83%
Transportation	$320	26.67%
Utilies	$80	6.67%
Medical	$275	22.92%
Misc	$140	11.67%

obtained by typing

```
\begin{center}
\begin{tabular}{|c|@{\ \$}l|r@{\%\ }|}
\hline
Food&135&11.25\\ \hline
Rent&250&20.83\\ \hline
Transportation&320&26.67\\ \hline
Utilies&80&6.67\\ \hline
Medical&275&22.92\\ \hline
Misc &140&11.67\\ \hline
\end{tabular}
\end{center}
```

Note that in the new version, the dollar signs and the percent signed are all aligned.

The look of the table could be further improved by making the line separating the first and second column a bit thicker. To do so, the command `\vline` is employed. It must be inserted in a `@`-expression. In this case at the beginning of the first such expression. Thus augmented, becomes the following version of the table.

Food	$135	11.25%
Rent	$250	20.83%
Transportation	$320	26.67%
Utilies	$80	6.67%
Medical	$275	22.92%
Misc	$140	11.67%

Note the location of the `\vline` command in the first line of the table, which is `\begin{tabular}{|c|@{\vline\ \$}l|r@{\%\ }|}`.

It's possible to entitle the table as, "Monthly Budget" by combining all three columns into one columns as the first row with the title displayed in it. The command `\multicolumn{`*num*`}{`*pos*`}{`*content*`}` is inserted at the beginning of a row or immediately before one of the alignment letters,`l, c,+ or\verb`=. It combines *num* of the following columns into one column whose width is the combined width of the combined columns, whose content is *content* aligned according to *pos*. For example

Monthly Budget		
Food	$135	11.25%
Rent	$250	20.83%
Transportation	$320	26.67%
Utilies	$80	6.67%
Medical	$275	22.92%
Misc	$140	11.67%

has `\multicolumn{3}{|c|}{Monthly Budget}\\ \hline \hline` for its first row. It's important to know that for the `\multicolumn` command a column begins with a positioning character, `l, c,+ or\verb`= and includes everything up to but excluding the next positioning character.

One final improvement can be made by replacing the second `\hline` with `\cline{1-3}`. In general the command `\cline{`*m*–*n*`}` must come immediately after `\\`. It put a line from the left side of the m^{th} column to the right side of the n^{th}. There may be more than on `\cline` command after a `\\`. The first row of the refined table is
`\multicolumn{3}{|c|}{Monthly Budget}\\ \hline \cline{1-3}` and the resulting table is

Monthly Budget		
Food	$135	11.25%
Rent	$250	20.83%
Transportation	$320	26.67%
Utilies	$80	6.67%
Medical	$275	22.92%
Misc	$140	11.67%

As mentioned above if a table is too long to be included on the current page, a `\newpage` command is issued and the table moved to the next page. It would be better to fill the remaining space on the current page with text that follows the table. In addition, a caption to the table is often desired. The next environment accommodates both of these goals.

5.6 The *table* Environment

The *table* environment is a *float* structure; that is, one that is designed to move automatically to a place where it can fit. The user has some control over its possible locations. The environment also allows the user to assign a

caption to the table and to include the table in a list of tables if such a list is desired. A *table* environment begins with the command `\begin{table}[`*pos*`}`. The optional argument *pos* can be used to **try** to place the table. Below are the choices and what each means.

`h` *here* If possible the float should appear at the point of the text where it is entered. If there isn't enough room at the bottom of the page to include the float, it will be put at the top of the next page (if permitted).

`t` *top* If there is room for the text already on the current page to appear under the float, it should be placed at the top of the current page. If not, then it goes at the top of the next page (if permitted).

`b` *bottom* The float is placed at the bottom of the current page after the preceding text on the current page if there's sufficient room and any room under the current text should be filled by text following the float. If not, then the float is placed at the bottom of the next page (if permitted).

`p` *page* This option is used to put this float on a special page reserved for it (and perhaps other floats).

`!` Suspends all constraints on floats. These constraints are described below.

More than one choice may be listed. The default selection is `tbp`. The constraints that apply to floats are:

1. no more than two floats at the top of a page and no more than 70% of the page can be occupied by floats at the top of the page.

2. no more than one float at the bottom of a page and no more than 30% of the page can be occupied by a floats at the bottom of the page.

3. many more that don't need to be mentioned here.

It's these constraints that can be most frustrating to a user trying to place a float. Suspending them with the option `!` should result in the desired placement.

A caption for the figure is entered with the command `\caption{`*Caption text*`}`. Including a caption is important for two reasons. It's the caption that's listed in the list of tables and if the table is to be referenced later using a `\ref` command the `\label` command be attached to the caption; not to the table environment.

A typical table environment might look like this.

```
\begin{table}[htb!]
\begin{center}
\begin{tabular}
Insert the table here
\end{center}
\caption{caption text}\label{key}
```

`\end{table}` Note that the `\label` command is associated with the caption and not with the table. When the file is compiled, LaTeX determines which *number* the table is and then precedes the *caption text* with the text, so that the output might look like, "Table 3. Test Scores".

There is a second float environment, the *figure* environment that is designed to contain figures, usually, but not exclusively, those that are imported. The environment works exactly as does the *table* environment. So all that needs to be explained is how to import graphics files.

5.7 The *graphicx* Package

The latest package used for importing graphics into a LaTeX document is the package, *graphicx*, which replaced the package, *graphics*. The package *graphicx* recognizes that pdf is quickly replacing post script as the output format of choice. It can import graphics in several different form, pdf, jpg, gif, tiff, but it can't deal with eps (encapsulated postscript) files. If you use Mathematica, or Maple to create your graphics files, you will need to export them as a pdf file rather that an eps file. (Double-clicking an eps file on a Mac will automatically convert the file into a pdf file.)

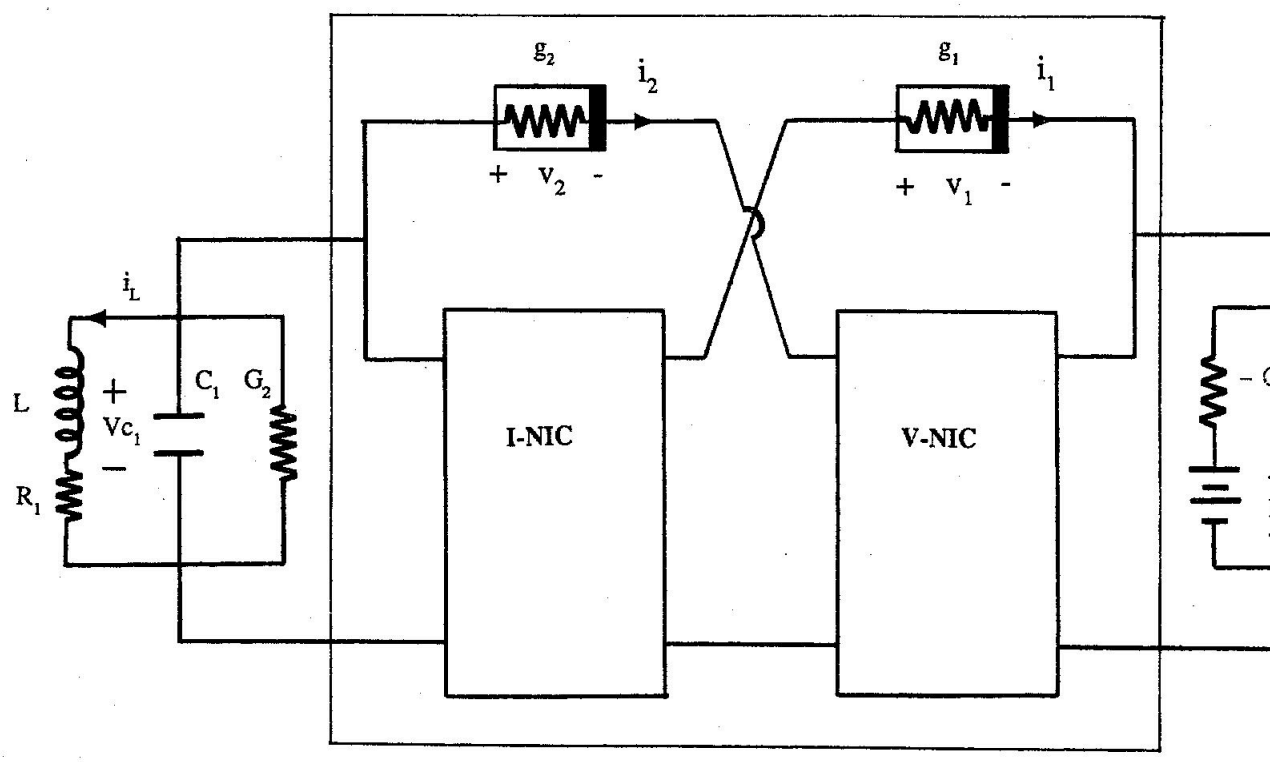
The command used to import the graphics file is `\includegraphics` which has an optional argument and a mandatory one. The optional one comes first and the mandatory one, which comes second, contains the name of the graphics file; for example, `graphics_file.pdf` although the suffix may be omitted. In particular, the command `\includegraphics{`*graphics file*`}` will put the graphics immediately after the word preceding the command, even if the command is on the next line. To put it below the current line of text, put two carriage returns after the line. To line the graphics up with the left margin, precede the command with `\noindent`. For example typing

```
Here's a graphics file, in pdf format,  presented as is.

\noindent\includegraphics{circuit.pdf}
```

will produce

Here's a graphics file, in pdf format, presented as is.



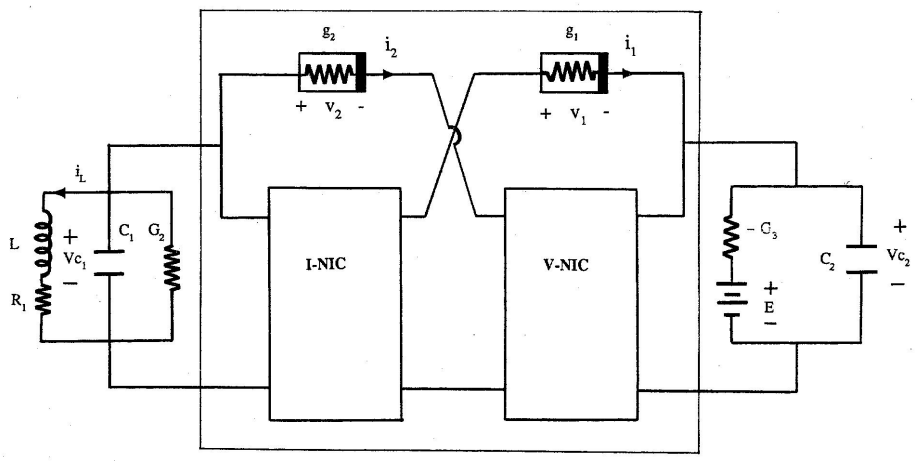
But as you can see, the file is too large to fit on the page. That's where the optional argument is used. Its syntax is [*key=value*]. There are two *keys*: `width` and `height`. For example the optional argument `[width=5in]` will expand or shrink the graphics to a width of 5 inches. The height is also adjusted so that the ratio of width/height remains unchanged. For example typing

```
Here's the same file shrunk to fit exactly in the page.

\noindent\includegraphics[width=\textwidth]{circuit.pdf}
```

will produce

Here's the same file shrunk to fit exactly in the page.



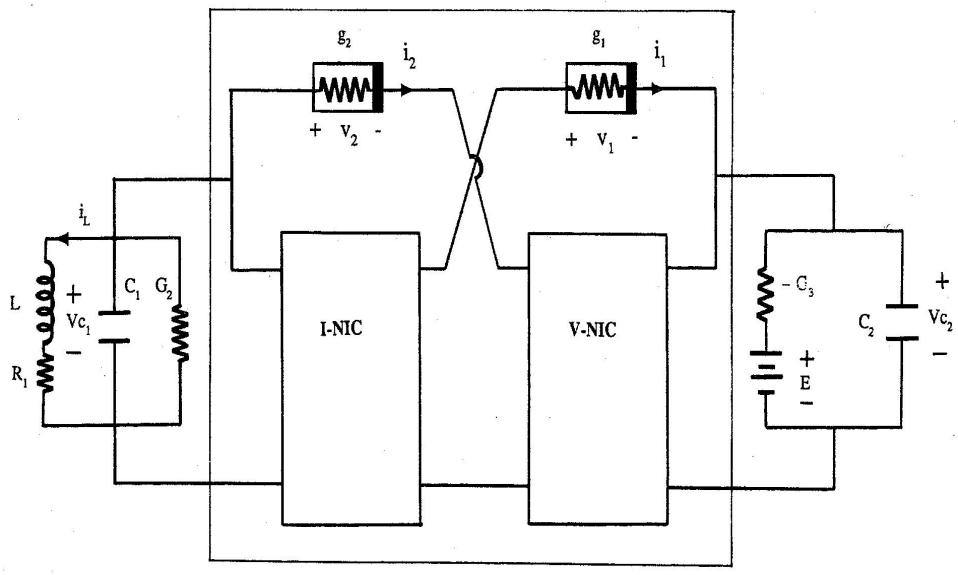
It's also possible to change both the width and length with `[width=5in,height=3in]`. For example typing

```
Here's the same file enlarged vertically.

\noindent\includegraphics[width=5in,height=3in]{circuit.pdf}
```

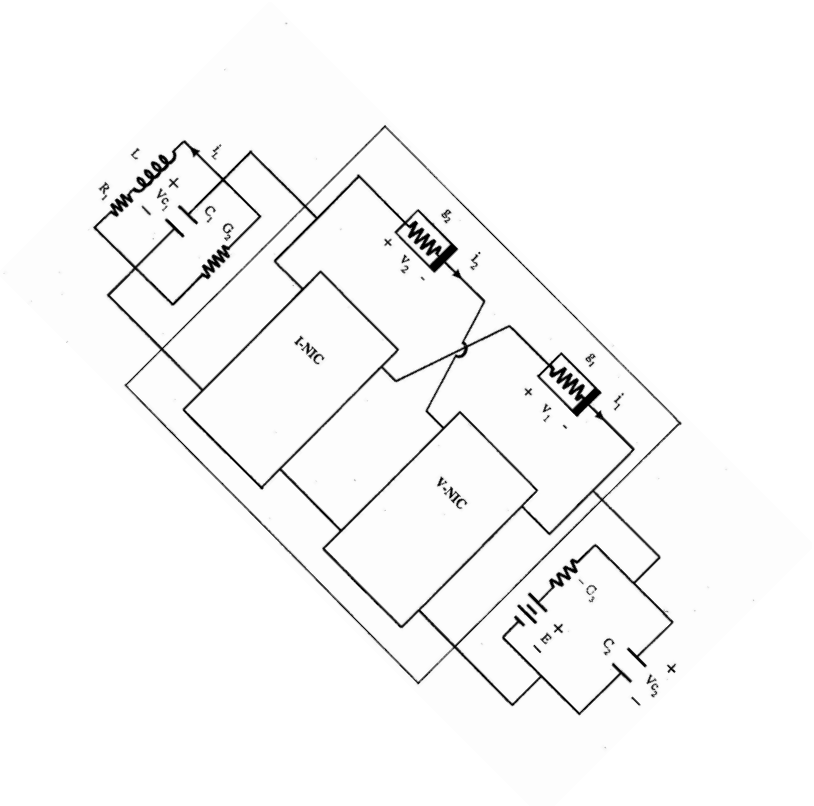
will produce

Here's the same file enlarged vertically.

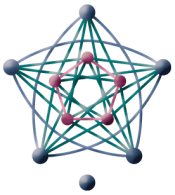


It's also possible to use the optional argument to rotate the graphics using the key `angle` measured in degrees. For example `[angle=-45]` will rotate the graphic around the bottom, left corner by -45 degrees. The point of rotation can be altered with the option, `origin=` and choices `c` for center, `t` for top, `r` for

right, and `B` for baseline. Any sensible combination of these options is allowed. The default is `bl`.



Note that is all of the cases above, there is an extra line between the text and the `\includegraphics`. Without it, if there's room, a small graphics would



be placed as follows.

But wait! There's more. Suppose the graphics file you wish to display is so large that there isn't room for it on the current page. In that case LaTeX inserts a `\newpage` command and puts the graphics at the top of the next page leaving the current page with a large blank space at the bottom which could contain text from after the graphic. Or suppore you wish to include a caption and label the graphics as, say, Fig. 1. The `figure` environment deals with both of these concerns.

5.8 The *thebibliography* Environment

A bibliography is produced with the environment *thebibliography* which is another list environment, but with one mandatory argument and one optional argument for each item. Specifically, the bibliographic entries begin with the command `\bibitem[`*user key*`]{`*entry id*`}` followed by the data that identifies the reference being entered. If no *user key* is entered, a number is automatically assigned as the *key* just as is done for an *enumerate* list. If one entry in a bibliography has an optional *user key*, then all entries must have *user keys*.

A bibliography starts with the command `\begin{thebibliography}{`*widest key*`}`. If numbers are used as *keys* and there are fewer than 9 bibliographic entries, the mandatory argument should be `{9}`. With 10 or more but fewer than 100, use `{99}`. On the other hand if all bibliographic entries are to be labeled with a *user key* of, say, a sequence of letters and/or numbers, the longest of which is 4 characters, then use `{XXXX}` for the *widest key*.

For example the bibliography at the end of these notes is produced by typing

```
\begin{thebibliography}{9}

\bibitem{HD04} Helmut Kopka and Patrick W. Daly, \textit{Guide to
\LaTeX}, fourth edition, Addison-Wesley, 2004.

\bibitem{LL94} Leslie Lamport, \textit{\LaTeX, A document preparation
system}, second revised edition, Addison-Wesley, 1994.

\bibitem{MG05} Frank Mittelbach and Michael Goossens, \textit{The
\LaTeX{} Companion}, second edition, Addison-Wesley, 2005.

\bibitem{AMS} \textit{User's Guide for the} \texttt{amsmath} \textit{
Package}, version 2.0, Amer. Math. Soc., revised, 2002,
\url{http://www . ams . org / tex / amslatex . html}.

\end{thebibliography}
```

Another option is

```
\begin{thebibliography}{XXXX}

\bibitem[HD04]{HD04} Helmut Kopka and Patrick W. Daly, \textit{Guide to
\LaTeX}, fourth edition, Addison-Wesley, 2004.

\bibitem[LL94]{LL94} Leslie Lamport, \textit{\LaTeX, A document preparation
system}, second revised edition, Addison-Wesley, 1994.

\bibitem[MG05]{MG05} Frank Mittelbach and Michael Goossens, \textit{The
\LaTeX{} Companion}, second edition, Addison-Wesley, 2005.
```

```
\bibitem[AMS]{AMS} \textit{User's Guide for the} \texttt{amsmath} \textit{
Package}, version 2.0, Amer. Math. Soc., revised, 2002,
\url{http://www . ams . org / tex / amslatex . html}.

\end{thebibliography}
```

which would produce

References

[HD04] Helmut Kopka and Patrick W. Daly, *Guide to LaTeX*, fourth edition, Addison-Wesley, 2004.

[LL94] Leslie Lamport, *LaTeX, A document preparation system*, second revised edition, Addison-Wesley, 1994.

[MG05] Frank Mittelbach and Michael Goossens, *The LaTeX Companion*, second edition, Addison-Wesley, 2005. 3.4, 5.8

[AMS] *User's Guide for the* `amsmath` *Package*, version 2.0, Amer. Math. Soc., revised, 2002, http://www.ams.org/tex/amslatex.html.

The mandatory argument *entry id* in each `\bibitem` is used to cite the reference in the body of the document by typing `\cite[`*extra*`]{`*entry id*`}`. The *extra* information in the optional argument further identifies the actual material being referenced. For example, " `see \cite[pages 5--8]{MG05}`" would produce, "see [6, pages 5–8]".

5.9 Boxes and the `minipage` Environment

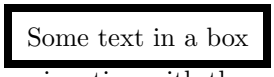
5.9.1 Boxes.

There are three types of boxes: **LR**, **Rule** and **Par**. We'll begin with the first, **LR** Boxes. These boxes come in two verities: plain and framed. The difference is that the framed verity has a visible frame around it. The contents of a **LR** box are typeset from left to right in a single line with no line breaks permitted. Such a box can be constructed easily with the command `\mbox`, which has one mandatory argument. For example *some text that must appear on one line* is produced by typing `\mbox{\textit{text that must appear on one line}}`. The more versatile box command is, `\makebox[`*width*`][`*pos*`]{`*text*`}`. The *width* sets how long the box will be. The default is the length of the *text*. If a *width* has been entered, the *pos* determines how the *text* will be aligned in the box: `l, c, r` with `c` be ing the ddfault. Typing `\makebox[2\width][l]{`*a small bit of text*`}` produces *a small bit of text* . The command `\width` that appears in this example is the length of *a small bit of text*. A box of width `0pt` permits over-writting. For example to produce $ type `\makebox[0pt][l]{/}$`.

To actually see the box that is produced by the previous two command. The simpler of them, corresponding to `\mbox` is `\fbox`. For example `\fbox{`*some text*`}` produces *some text*. Corresponding to `\makebox` is `\framebox`, which

has the same syntax as does `\makebox`. So that `\framebox[1.5\width][r]{`*a small bit of text*`}` produces . With any frame box it's possible to increase the width of the frame with the command `\fboxrule` and the space at either end with the command `\fboxsep`. For example typing

```
\setlength{\fboxrule}{1mm}\setlength{\fboxsep}{2mm}
\fbox{Some text in a box}
```

yields 

In conjunction with these boxes come the very useful command `\raisebox` which is used to create and raise or lower a **LR** box. The syntax for the command is `\raisebox{`*lift*`}[`*height*`][`*depth*`]{`*contents*`}`. The first mandatory argument is the amount that the box is to be raised, which can be negative. The default setting for *height* and *depth* are `\heigth` and `\width` respectively. (See Section 2.8 for the meaning of these commands.) Either of these can used to determine the *lift*. For example a very useful one for those authors wishing the use the Greek letter χ for the characteristic function is `\raisebox{\depth}{$\chi$}`, but rather than typing it repeatedly, define a newcommand,

```
\newcommand{\charf}[1]{\raisebox{\depth}{$\chi$}{\ensuremath{_{#1}}}}.
```

Then get χ_A instead of χ_A by simply typing `\charf{A}`.

As a final note, it is important to know that all text placed in one of these boxes is typeset in text mode even if the box is created in math mode.

5.9.2 Rule Boxes

The command `\rule` has one optional arguments, *lift*, and two mandatory arguments, *width* and *height*. It produces a black line whose height is determined by *height* and whose width is *width*. For example `\rule{3in}{2mm}` produces . The optional argument, *lift* may be either positive or negative and will move the rule vertically the designated amount.

5.9.3 Paragraph Boxes.

Paragraph boxes are to pages what LR boxes are to lines. They are created with the `\parbox` command, which has first three optional arguments followed by two mandatory arguments. The first optional argument, *pos*, has three choices: `t, c, b` with `c` being the default. The baseline of the current line is aligned with the base line of the top, center, or bottom line of the text in the box respectively. The second optional argument, *height*, can be a specific length, such as 2in or a multiple of a fixed length, such as `\height`, the height of the text in the box, which is the default setting. Only if the height is other that its default does the third optional argument, *inner-pos*, get selected. It has three choices, `t, c, b` with `c` being the default. The first mandatory argument is *width*, which, like *height*, can be assigned a specific number or a multiple of a standard length such

as \textwidth or \linewidth. By the way, it's possible to put a frame around a paragraph box by putting the \parbox command inside of the argument of a \fbox command.

5.9.4 The minipage Environment.

Environments, footnotes and other structures can't be placed in a paragraph box, but they can go in a mini page. The syntax is
\begin{minipage}[pos][height][inner-pos]{width}.

pos This argument is applicable only if the mini page is to be used in line. The options are b,t,c(default).

height This argument is used only to set the height for a value other than height of text on mini page.

inner-pos If [height] is designated, this argument determines how the text will be positioned on page.

width This mandatory argument sets the width of mini page.

6 Some Useful Packages

6.1 The url Package.

Above the command \url appears, which is made possible by the url package. Entering internet and email addresses can be tricky because a hyphen is often part of such an address, so LaTeX must be prevented from breaking a line with a hyphen in an address. The package url is designed to do just that and to typeset these items in the typewriter family. In the preamble type \usepackage{url} and then when you wish to enter, say an internet address, type \url{http://www. ams. org/ tex/ amslatex. html} and the result will be the internet address with a linebreak http://www.ams.org/tex/amslatex.html. The spaces in the argument tell LaTeX where it is allowed to break the line.

6.2 The milticol Package

The class option twocolumn typesets a document in two columns. If both formats are desired in the same document, it's possible, with the commands \twocolumn and \onecolumn, to toggle between the two formats, but each time a change is made, a new page is begun. If the transition is from two columns to one, the left column is completed on the page before the right one is begun which can produce a large blank space. The better way is to use the multicol package. To do so, put \usepackage{multicol} in the preamble, which makes the multicols environment available. To begin two or more columns, type \begin{multicols}{*num cols*}, where *num cols* is an integer between 1 and

10 indicating the number of columns desired. LaTeX automatically inserts some vertical space, and starts producing text in the number of columns requested. If the space left on the page is small, a `\newpage` command is issued and the multicolumn text begins on the next page. When the `\end{multicols}` command is met, LaTeX stops producing the text in the multicolumn format and balances the columns. It's possible to start a second `multicols` environment inside an existing one with the expected outcome.

If some introductory text is desired before beginning the multicolumn text, it's advisable to put that text in the first optional argument of the `\begin{multicols}[`*prefix text*`]`. (Note the location of the optional argument following the mandatory one.) Doing so makes that text part of the multicolumn environment preventing it from appearing at the bottom of a page while the multicolumn text begins at the top of the next page. A second optional argument `[`*skip*`]` allows the user to change the amount of vertical space skipped before resuming the text format in force before the multicolumn text was begun. For example `\begin{multicols}{2}[\centerline{The next paragraph is typeset in two columns.}]`, which is exactly the command issued at the beginning of the next paragraph.

The next paragraph is typeset in two columns.

In addition to the `multicols` environment, the `multicol` package provides two additional parameters that can be altered by the user. The first of these is the length `\columnsep` which, as it's name suggest, is the distance between the columns. Its value can be changes by either of the commands, `\setlength` or `\addtolength`. It's recommended that this distance not be diminished because the resulting text might be difficult to read. However increasing it might make sense stylistically. One such case would be if it's desired to have a line separating the columns. To do so, set (or increase) the length, `\columnseprule` to the desired thickness. Its default value is 0pt. If it's increased to, say 2pt, then its recommended that the `\columnsep` be increased by 2pt as well. Finally, the length of a line in each column is the length `\linewidth`, which in single column text is the same as `\textwidth`.

6.3 The `picinpar` Package

The `picinpar` package permits the insertion of special material, such as a photograph or a special logo, to be inserted in a paragraph of text through the use of the `window` environment. It has one mandatory argument which, contrary to the usual practice, is enclosed in brackets rather than in braces. The

argument consists of four dismas. The first of these parts the paragraph where the winond determines where the obgraph, at the left margin, in gin with the choice of `l, c,` the actual object to be in the for any explanatory text about this paragraph is graphics was

A Snowflake

tinct parts separated by comis the number of the line in dow is to begin. The secject is located in the parathe center or at the right maror `r`. The third position if for window and the last place is the object. For example in included with the command `\begin{window}[4,c,\includegraphics[width=.3\textwidth]{images3}, \centerline{A Snowflake}]`. The paragraph must end with the command `\end{window}`.

7 The Bibliography Using BibTEX

For those who will be authoring several manuscripts about related subjects and in that process will be drawing references from a large collection of article, book or other sources, building bibliographies with BibTEX would be more efficient than constructing them individually from scratch. To do so requires one (or more) database file(s), which must be in the same folder as the source file. Each *entry* (or *record*) in such a database begins with an *entry type* (`book, article` (The full list is given below in subsection 7.3 on page 39.), followed by an *entry id* that uniquely determines the entry, and then the data needed to produce a reference in the bibliography. The command `\cite{`*entry id*`}` in the body of the source file will cause the associated reference to be included in the bibliography. (Find more information about the `\cite` command on page 34.) To include a reference that is never cited in the document, type `\nocite{`*entry id*`}` anywhere in the body of the source file or to include all references in the database(s), type `\nocite{*}` again anywhere in the body of the source file.

7.1 How the BibTEX System works

To create a bibliography from the elements discussed above, put the command `\bibliographystyle{`*style*`}` in the preamble to select a style for the bibliography. There are many possibilities. (See the next subsection for partial list of styles.) The command `\bibliography{database_1.bib, database_2.bib, ...}` is placed in the source file where the bibliography is to appear. To produce the bibliography, compile the file and then run the program `BibTeX` which will use information in the *file_name*`.aux` file to produce two new files; *file_name*`.blg` and *file_name*`.bbl`. (The BibTEX program comes with each TEX implementation.) The first of these two files is just a log file. The latter file contains the actual bibliography that will be imported into the source file at the point where the command `\bibliography{database_1.bib, database_2.bib, ...}` is situated. In addition, the file can be edited, but don't do so until the final run of `BibTeX` because it's rebuilt with each run.

7.2 Bibliography Styles

The bibliography *style* determines the formatting of each reference and how the bibliography is ordered. Every TEX implementation comes with the following four bibliography style files.

plain Entries are ordered alphabetically by last name of the first author and identified with a numeric *key*.

unsrt Entries are ordered by first appearance in a `\cite` or `\nocite` command and identified with a numeric *key*.

alpha Entries are ordered as in `plain` but with a *key* consisting of the first three letters in the first author's last name followed by the last two digits of the year of publication.

abbrv Same as `plain` but with a *key* consisting of abbreviations of names and other fields.

Many periodicals have their own bibliography style files. These can usually be downloaded from the publication's web page. Any style file obtained in such a way should be placed in the `/texmf/tex/bibtex/` folder on your computer.

Those *styles* that order *entries* by author's last name have a problem if the data for an *entry* doesn't have an author. In that case BIBTEXuses other information, such as editor or organization, for sorting. When the style doesn't list the *entries* by number, it constructs a different *key* for ordering. This *key* (not to be confused with the *entry id*) is what appears in brackets in the output file where a `\cite` command occurs in the source file. If the *style* file produces an unacceptable *key*, the use can change it as will be explained later.

7.3 BibTEX Databases

A database is a plain text (ASCII) file consisting of a collection of *entries* (or *records*). The name of the file must end with the suffix, .bib. Each *entry* must begin with @*entry type*{*entry id*, and end with a closing }. The *entry id* is the unique identifier for the entry and consequently no two *entries* can have the same *entry id* in a database file and preferably, in no other database file. The possible entry types are

1. `article`
2. `book`
3. `booklet`
4. `inbook`
5. `incollection`
6. `inproceedings`
7. `manual`
8. `mastersthesis`
9. `misc`
10. `phdthesis`

11. `proceedings`
12. `techreport`
13. `unpublished`

7.3.1 The Structure of a BibTEX Database

The remainder of an *entry* consists of *fields* The *entry type* determines what *fields* must or may occur in the *entry*. Each *field* begins with its name followed by an equal sign (=) and then the text for that field enclosed in a pair of quotation mark (" ") and ends with a comma (,). The possible *fields* are

1. `address`
2. `annote`
3. `author`
4. `booktitle`
5. `chapter`
6. `edition`
7. `howpublished`
8. `institution`
9. `journal`
10. `key` (for a specific *cite key*)
11. `month`
12. `note`
13. `month`
14. `number`
15. `organization`
16. `pages`
17. `publisher`
18. `school`
19. `series`
20. `title`
21. `type`
22. `volume`
23. `year`

article Required Fields: author, journal, title, year. Optional Fields: volume, number, pages, month, note.

book Required Fields: author or editor, title, publisher, year. Optional Fields: volume or number, series, address, edition, month, note.

inproceedings Required Fields: author, title, booktitle, year. Optional Fields: editor, volume or number, series, pages, address, month, organization, publisher, note.

7.3.2 A Typical `article` *entry*

```
@article{AHP92,
```

```
author = "Anderson, J. M. and Housworth, E. A. and Pitt,  L. D.",
title = "The Spectral Theory of Multiplication Operators",
journal = "Mathematika",
volume = "39",
year = "1992",
pages = "136--151"
}
```

7.3.3 A Typical `book` *entry*

```
@book{BBS97,
author ="Bruckner, A. M. and Bruckner, J. B. and Thomson, B. S.",
title ="Real Analysis",
edition ="",
publisher  ="Prentice-Hall",
year ="1997",
address ="Englewood Cliffs, NJ"
}
```

7.3.4 Abbreviations and Preamble

If new commands would be useful in a BibTeX database, they are entered at the beginning of the file using the following syntax. `@PREAMBLE{"\providecommand{\url}[1]{\texttt{#1}}"#`
`"\providecommand{\singleletter}[1]{#1}"#`
⋮`}`
Abbreviations are entered next in a similar fashion. `@STRING{pams="Proc. Amer. Math. Soc."}`

```
@STRING{raex = "Real Analysis Exch."}
@STRING{jcs = "J. Chem. Soc."}
@STRING{jfm = "J. Fluid Mech."}
```

7.3.5 The `author` or `editor` Field

Of all of the fields that appear in *entries* the `author` field requires the most explanation. If there is more than one author, their names are separated by the word "and". Each name consists of four parts; the *Last* name, a possible *prefix* to it, a possible *suffix* to it and a *First* name. Only the *Last* name is required. A *prefix* might be something like, von or van der or de la. A *suffix* is something like Jr. or III. To make it easy for BibTeX to find the *Last* name it's best to enter a name in the form *Last* name, *First* name. To help BibTeX get the parts of a name right, parts may be enclosed in a pair of braces ({ }).

7.3.6 Several Examples of Names

Arnold, Edward M. or Edward M. Arnold is also acceptable
van der Werder, Arthur
di Bari, Cristina Maria

Dutta, Tapan Kumar
{Lee Peng Yee}
de Lucia, Paolo
{Fernandez Long de Foglio}, Susana
D'Aniello, Emma
Enrique de Amo, Artero
Persson, Lars Erik
da Cunha, Marisa Ortegoza
\MakeUppercase{l}a Torre, Davide
de Vries, Martijn
\MakeUppercase{d}i Fratta, Giovanni
\MakeLowercase{D}eSouza, Geraldo
{Morgan, II}, John C.

7.3.7 The `title` Field

Many bibliography styles capitalize only the first word in the title of an article. So to assure that proper nouns in titles are capitalized type then as `{Smith}`. It may seem that `{S}mith` would work as well, but for very technical reason, it's not recommended.

7.3.8 Cross Referencing

In case several *entries* appear in the same source, such as a conference report, the fields from one entry can be used in another entry. For example

```
@INPROCEEDINGS{PWCPJ,
author = "Janot, J.",
title = "Closing the light sbottom mass window from a compilation of e+e- -> hadron data",
page = 81,
crossref = "PWC"
}
@PROCEEDINGS{PWC,
title= "Proceedings of the Workshop on $e^+e^-$ Collisions",
booktitle= "Proceedings of the Workshop on $e^+e^-$ Collisions",
editor="Zerwas,  P.M.",
year= 2002
}
```

7.3.9 Multiple Bibliographies

It's possible to produce a separate bibliography for each chapter, or at the end of specific divisions or several bibliographies at the end of the document. To learn how this is done, see *The LaTeX Companion*, [2, Section 12.6, pages 747–756].

8 Long Documents.

When producing a long document several new aspect arise. First, the size of the source file for a long document can be clumsy. Second in a long document it's desirable to have a table of contents and also, perhaps, an index. In this section we deal with all of these matters.

To more efficiently handle a long document, it's convenient to create the source file in segments. For this purpose LaTeX provides the notion of a root file with sub files that are imported into the document using the `\include` command. For example suppose a thesis is being written that has an abstract, an introduction, three chapters and a bibliography. At the outset the root file would look like this.

```
\documentclass{book}
\usepackage{packages}
Other preamble entries
\begin{document}
\pagenumbering{roman}
\tableofcontents
\newpage
\pagenumbering{arabic}
\setcounter{page}{1}
\maketitle
\begin{abstract}
Text of the abstract
\end{abstract}
\include{intro}
\include{chapt1}
\include{chapt2}
\include{chapt3}
\include{biblio}
\end{document}
```

Then five separate files are created for each of the five parts. Because it's never certain at the beginning what will go into the bibliography, the `biblio` file should be updated as the others are in production. These files contain only text; no `\begin{document}` etc. commands. All of these files must be in the same directory. While working on the `intro` file, and building the `biblio` file, the root file is slightly altered as follows.

```
\documentclass{book}
\usepackage{packages}
Other preamble entries
\includeonly{intro,biblio}
\begin{document}
\maketitle
\pagenumbering{roman}
```

```
\tableofcontents
\newpage
\pagenumbering{arabic}
\setcounter{page}{1}
\begin{abstract}
Text of the abstract
\end{abstract}
\include{intro}
\include{chapt1}
\include{chapt2}
\include{chapt3}
\include{biblio}
\end{document}
```

Once the introduction is finish (at least in its first form) and chapter 1 is started and the command `\includeonly{intro,biblo}` is changed to `\includeonly{chapt1,biblo}`. (Some prefer to comment out the first `\includeonly` command and add the second.) Each time LaTeX is run, the auxiliary file is updated; not replaced. Thus all labels and citations created during the construction of the Introduction are preserved and consequently, if any are referenced in the new file, they will be recognized when the root file is compiled. Continue in this fashion to construct all of the files. Then remove all of the `\includeonly` commands and compile the file. Each time LaTeX sees an `\include` command, it enters and `\clearpage` command thereby starting the next file on a new page. For that reason the files must represent separate parts of the document.

8.1 Table of Contents.

To create a table of contents for any document, simply type `\tableofcontents` at the place in the document where you wish the table of contents to appear, usually immediately after the title page (if there is one) and the abstract. (See the example on page 43.) Adding `\newpage` will start the rest of the document on a new page. It takes two runs to create the table of contents. Putting the command `\pagenumbering{arabic}` numbers the table of contents in lower case Roman numerals. Begin a new page, change the page numbering to `arabic` and reset the `page` counter to 1 to begin the body of the document. A List of Figures (Tables) can be added after the Table of Contents with the command `\listoffigures` (`\listoftables`).

8.1.1 Depth.

By default the divisions listed below are automatically included in the table of contents.

Book	Report	Article
Part	Chapter	Section
Chapter	Section	Subsection
Section	Subsection	Subsubsection

The user can change the depth with the command `\setcounter{tocdepth}{`*desired_depth*`}` or `\addtocounter{tocdepth}{`*desired_change*`}`. For example, if the depth is set to 4, then paragraph names will be included.

8.1.2 Adding Items.

The command `\tableofcontents` produces a file named *main_file_name*`.toc` to which entries can be added. For example sections entered with the command `\section*` are not automatically put into the table of contents nor is the bibliography. To add the section given by `\section*{Extra Section}`, type `\addcontentsline{toc}{section}{Extra Section}`. In this form, the Table of Contents might look something like

Contents

1. Introduction

Extra Section

To align the names of the two sections, augment the command as `\addcontentsline{toc}{section}{\protect\numberline{}{Extra Section}` resulting in

Contents

1. Introduction

Extra Section

The References section is added to the table of contents of this document using this technique.

8.1.3 The `hyperref` Package.

The `hyperref` package turns all items listed in the Table of Contents, all bibliographical citations, all pages listed in the Index and all urls into hypertext links. To avoid possible conflicts with other packages, the command `\usepackage{hyperref}` should be the last package listed in the preamble. By default, all links created by the package are enclosed in boxes, which is preferable for black-and-white output, but for color output the option `colorlinks` the links appear in red, except for the urls for which the color is magenta. It can be changed with the option `urlcolor=`*desired color*. (In this document the color is changed to blue.) Another useful option is `backref` . After each biographical entry it lists the division numbers where that reference is cited in the document. For additional information see [2, pages 35–67].

8.2 Index.

Creating an index is far more complicated than creating a table of contents, because the author must specify each item to be included in the index. This is accomplished with a command beginning with `\index`, which has a very complicated structure due to the different ways in which entries in the index may appear. To create the index the file must be compiled and then a program must be run on a file created by LaTeX. Finally the location of the index must be selected by typing the command `\printindex` at the point in the source file where the index is to appear, usually just before the `\end{document}` command. Details are provided below in Section 8.2.2

8.2.1 The Command `\index`.

The process of producing the index will begin by discussing how and where to designated index entries in the text. Entires in an index often have subentries under them and in some cases those subentries also have subentries, but others stand alone with no subentries. To create a stand alone entry, at the point in the source file where this item appears, type `\index{`*entry*`}`. For example typing `\index{Roman}` will cause the word, "Roman" to appear in the index followed by the page number in the output file where the command appears. Typing the same command later in the source file will produce a second page number in the index separated from the first by a comma. It must be emphazied that the command must be typed exactly as before. For example `\index{ Roman}` will create a different entry in the index as will `\index{Roman }`. To avoid making any such errors, it's best to never use the space bar when creating index entries. All stand alone entries in the index will appear in alphabetic order.

To create a entry that will have subentries, type `\index{`*entry*`!`*sub_entry 1*`}`. For example typing `\index{environments!abstract}` will cause the word, "environments", to appear in the index, with no page number listed, but with , "abstract" listed under it and slightly indented and a page number. As with stand alone entries, typing the same command later in the text will cause a second page number to be listed after, "abstract". In addition a second subentry can be created by typing `\index{`*entry*`!`*sub_entry 2*`}`. For example typing `\index{environments!center}` will cause, "center", to appear under, "environments" in the index. All subentries to an entry appear under that entry in alphabetic order.

Subentries may have subsubentries listed under them by typing `\index{`*entry*`!`*sub_entry*`1`*subsub_entry 1*`}`. For example typing `\index{environments!lists!enumerate}` will cause, "enumerate" to appear under, "lists", which will appear under, "environments", in the index. Additional similar entries but will different subsubentries will cause additional entries under, "lists", to appear in alphabetic order.

The appearance of any page number in the index can be altered if, for example, one page reference is more important than the others. For a stand alone entry, typing `\index{`*entry*`|\textbf}` will cause the page number for that par-

ticular reference to the *entry* to appear in boldface. Similar commands such as `\textit` or `\textsc` may also be used. For entries with subentries, altering the appearance of the page number only makes sense for the last subentry. For example `\index{environments|\textit!lists!enumerate}` would be senseless because no page number is listed for the *main_entry*. Only `\index{environments!lists!enumerate|\textit}` would be meaningful.

To indicate that an entry is discussed on several consecutive pages, type `\index{`*entry*`|(}` at the beginning of the discussion and `\index{`*entry*`|)` at the end. The page number will be listed as a range of number; for example 45–51. To refer an entry to another entry, type `\index{`*entry*`|see{` *alt_entry*`}`. For example `\index{plain}|see{page style}` refers, "plain", to, "page style". Similarly `see also{`*alt_entry*`}` can be used as well.

The way an entry or subentry appears in the index may be altered slightly or significantly without changing where in the index it appears. For example typing `\index{amsmath@\texttt{amsmath}}` will displays, "amsmath", in the index as `amsmath`. Likewise typing `\index{newpage@\verb=\newpage=}` will cause the command, "`\newpage`", to appear in the index alphabetically ordered under, "newpage" as it happens in the index of this document.

Finally, it's best not to put an `\index` inside of an environment. If the same command occurs elsewhere in the document, the reference will appear twice, each with a different page number instead of one appearance with two or more page numbers listed.

The `\index` command gives new meaning to the characters, `!, @, |` and `"`. The use of the first three of these characters has been explained above. The quote character, `"` allows one of these characters to be typeset. For example typing `\index{at@"@}` will cause the `@` symbol to be printed alphabetically where the word, "at" would appear. Similarly `\index{bar!vertical@"|}` will cause a vertical bar `|`, to be printed under the heading, "bar" where the word, "vertical" would be.

8.2.2 Creating the Index File.

To actually create the index, first include the package `makeidx` with a `\usepackage`, put the command `\makeindex` in the preamble, and compile the file. In that process, LaTeX will create a file named *file_name*`.idx`. Then run the program MakeIndex. (How that is done depends on particular TeX implementation you're using.) That program creates files *file_name*`.ilg` and *file_name*`.ind` (which is actually an environment) that LaTeX inputs when it runs next. That is when the index appears at the point where the command `\printindex` appears.

If the package `hyperref` is loaded, then the page numbers in an index will be hyper links.

9 Typesetting Mathematics.

LaTeX employs a different font for characters used as mathematical symbols. So be certain that every symbol you enter that is to be mathematics is set in "math mode". If a mathematical expression is to be included as part of the text in a line, it is enclosed between a pair of $ signs. For example to typeset the sentence, "The factoring of squares formula is $(a^2 - b^2) = (a - b)(a + b)$." you type `The factoring of squares formula is $(a^2 - b^2)=(a - b)(a + b)$.` (Spaces are ignored in math mode, so type as many as you need to make reading the source document as easy as possible.) Longer and more complicated expressions are displayed. Displayed expressions are enclosed between a pair of $$ signs, or inside of the special mathematics environments introduced below. Use the first technique when the expression to be displayed is less than a line in length and you don't wish to number it. For example type

```
The formula for the factoring of the difference of $n^\text{th}$
powers is
$$ a^n - b^n = (a - b)(a^{n-1} +a^{n-2}b + a^{n-3}b^2+ \dots +
a^2b^{n-3} +ab^{n-2} + b^{n-1})$$
```

to get,
The formula for the factoring of the difference of n^{th} powers is

$$a^n - b^n = (a - b)(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \cdots + a^2b^{n-3} + ab^{n-2} + b^{n-1})$$

(Note that exponents longer than one character must be contained in a pair of braces { }.) Some authors will type `\cdots` to get the dots in the center of the line, but LaTeX (with the `amsmath` package) knows from the contents where to put them according to the convention adopted by the AMS.

9.1 Fractions/Binomials, Modulo, and Roots.

The commands for producing fractions and binomial expressions are similar. For fractions use the command `\frac{`*numerator*`}{`*denominator*`}`. For example $\frac{a^2}{b^3}$. A slightly smaller character size is used for fractions. For fractions in displayed formulas, the normal character size is used. If a fraction within text has expressions with exponents, the smaller font may be hard to read. In that case replace `\frac` with `\dfrac` (d for display) to get $\dfrac{a^2}{b^3}$. Sometimes in displayed formulas a smaller fraction is desired. In that case type `\tfrac` (t for text). For binomial expressions,

An analogous command structure is available for binomial coefficients. For example `$\binom{m}{n}$` will produce $\binom{m}{n}$ while `$\dbinom{m}{n}$` will produce $\dbinom{m}{n}$. Likewise in display mode `$$\binom{m}{n}$$` will produce $$\binom{m}{n}$$ and `$$\tbinom{m}{n}$$` will produce $$\tbinom{m}{n}$$.

The basic command for producing modulo expressions such as $x \equiv y \mod n$ is `x\equiv y\mod{n}`. The variation `\bmod{n}` diminishes the space before

mod, $x \equiv y \bmod n$. Parentheses are placed around mod n with `\pmod` and `\pod` eliminates the word mod, $x \equiv y \ (\mathrm{mod}\ n)$ and $x \equiv y \ (n)$.

Roots are created with the command `\sqrt[`*n*`]{`*expression*`}`. For example `\sqrt[3]{x^2+1}` produces $\sqrt[3]{x^2+1}$ in text and $\sqrt[3]{x^2+1}$ in displayed formulas. There is no `\dsqrt` form as might be expected, but any mathematical expression in text can be given its displayed version with the command `\displaystyle` which is a declaration type command. As would be expected there's a corresponding command `\textstyle` to produce text style in display mode.

9.2 The `amssymb` and `eucal` Packages.

LaTeX provides an collection of upper case in math mode called the Computer modern calligraphic letters. They are $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{H}, \mathcal{I}, \mathcal{J}, \mathcal{K}, \mathcal{L}, \mathcal{M}, \mathcal{N}, \mathcal{O}, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}, \mathcal{V}, \mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}$ which are produced with the command `$\mathcal{`*WANTED LETTERS*`}$`. Loading the `amssymb` package makes the Blackboard Bold and Fraktur fonts available in math mode as well along with many special characters. The Blackboard Bold fonts are available only in upper case and are produced with the command `$\mathbb{`*BLACKBOARD LETTERS*`}$`. The complete set of letters is $\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}, \mathbb{E}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathbb{I}, \mathbb{J}, \mathbb{K}, \mathbb{L}, \mathbb{M}, \mathbb{N}, \mathbb{O}, \mathbb{P}, \mathbb{Q}, \mathbb{R}, \mathbb{S}, \mathbb{T}, \mathbb{U}, \mathbb{V}, \mathbb{W}, \mathbb{X}, \mathbb{Y}, \mathbb{Z}$. The Fraktur font family is available in math mode in both upper and lower cases. They are produced with the command `$\mathfrak{`*Letters In Fraktur*`}$`. The entire list of characters is $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}, \mathfrak{E}, \mathfrak{F}, \mathfrak{G}, \mathfrak{H}, \mathfrak{I}, \mathfrak{J}, \mathfrak{K}, \mathfrak{L}, \mathfrak{M}, \mathfrak{N}, \mathfrak{O}, \mathfrak{P}, \mathfrak{Q}, \mathfrak{R}, \mathfrak{S}, \mathfrak{T}, \mathfrak{U}, \mathfrak{V}, \mathfrak{W}, \mathfrak{X}, \mathfrak{Y}, \mathfrak{Z}, \mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \mathfrak{d}, \mathfrak{e}, \mathfrak{f}, \mathfrak{g}, \mathfrak{h}, \mathfrak{i}, \mathfrak{j}, \mathfrak{k}, \mathfrak{l}, \mathfrak{m}, \mathfrak{n}, \mathfrak{o}, \mathfrak{p}, \mathfrak{q}, \mathfrak{r}, \mathfrak{s}, \mathfrak{t}, \mathfrak{u}, \mathfrak{v}, \mathfrak{w}, \mathfrak{x}, \mathfrak{y}, \mathfrak{z}$. In particular, `${\mathfrak c}$` is commonly used to denote the cardinality of the real line, $\mathbb{R}$.

The package and option `[mathscr]eucal` provides an additional complete set of upper case characters in math mode. They are produced with the command `$\mathscr{`*LETTERS*`}$` and look like $\mathscr{A}, \mathscr{B}, \mathscr{C}, \mathscr{D}, \mathscr{E}, \mathscr{F}, \mathscr{G}, \mathscr{H}, \mathscr{I}, \mathscr{J}, \mathscr{K}, \mathscr{L}, \mathscr{M}, \mathscr{N}, \mathscr{O}, \mathscr{P}, \mathscr{Q}, \mathscr{R}, \mathscr{S}, \mathscr{T}, \mathscr{U}, \mathscr{V}, \mathscr{W}, \mathscr{X}, \mathscr{Y}, \mathscr{Z}$.

9.3 The `amsthm` Package.

There are many different styles for presenting assertions such as definitions, lemmas, theorems, corollaries, etc. The `amsthm` permits the use of all of the popular ones. Each different assertion is stated with an environment that is created by the user with the command `\newtheorem` , which must be in the preamble. This command has two mandatory arguments and two optional ones. The syntax using the only two mandatory arguments is `\newtheorem{`*name*`}{`*heading*`}` where *name* is the name of the environment and *heading* is the heading that will announce the assertion. For example the command `\newtheorem{theorem}{Theorem}` creates an environment, `theorem`. In the body of the document when `\begin{theorem}\label{th1}` is typed, LaTeX begins an environment on a new line after a small amount of vertical space and, flush with the left margin, produces, **Theorem** followed by a number identifying the theorem that can be recalled with the command `\ref{th1}`. The user types the statement of the theorem which LaTeX sets in *italics*. When `\end{theorem}`

is typed, LaTeX inserts a small amount of vertical space and reverts to the previous font family. The default method for numbering the assertions is 1, 2, 3, etc. If the user wishes to number by section number; i.e., 4.1, 4.2, etc for the first and section theorems in the fourth section, then the command employs one of the optional arguments; specifically, `\newtheorem{theorem}{Theorem}[section]`. The numbers 1, 2, 3, etc. assigned to the assertion come from a counter, `theorem` which is automatically created. When a second type of assertion is created, the user has an option to use the same counter as is used by `theorem` or to introduce a new one. To introduce a new one, type say `\newtheorem{lemma}{Lemma}` in the preamble. To use the counter already created, type `\newtheorem{lemma}[theorem]{Lemma}` in which case only a new environment is created, `lemma` but not a new counter. The numbering would be **Lemma 1.**, **Theorem 2.** (or **Lemma 4.1.**, **Theorem 4.1.** if [section] was used in the definition of `theorem`).

The style in which assertions are presented as described above represents the default style, *plain*. The `amsthm` package supplies two other styles, `definition` and `remark`. The command `\theoremstyle{`*style*`}` changes the style. Examples of all three styles how to change from one to the other follow. First is the default style, *plain*

Theorem 1. *This assertion is set in the theorem style with the default numbering scheme.*

Definition 2. This assertion is set in the *definition* theorem style. Note that the font shape is upright, which is the only change from plain.

Remark 1. This assertion is set in the remark theorem style. Note that the heading is in italics rather than bold and that the statement is again in the upright shape.

Note the small amount of vertical space between each assertion. LaTeX knows when to insert such space and when not to insert any. The amount of space between Theorem 1 and Definition 2. is the same as that before Theorem 1 and after Remark 1. The user shouldn't attempt to alter this spacing.

Below are the entries in the preamble that are needed to produce the above assertions.

```
\newtheorem{theorem}{Theorem}
\theoremstyle{definition}
\newtheorem{definition}[theorem]{Definition}
\theoremstyle{remark}
\newtheorem{remark}{Remark}
```

Note that `remark` has its counter, but that `definition` uses the `theorem` counter.

```
\begin{theorem}\label{th1}
This assertion is set in the theorem style with the default
numbering scheme.
```

```
\end{theorem}
\begin{definition}\label{def1}
This assertion is set in the \textit{definition} theorem style.
Note that the font shape is upright, which is the only change
from \text{plain}.
\end{definition}
\begin{remark}\label{rmk1}
This assertion is set in the remark theorem style. Note that the
heading is in italics rather than bold and that the statement is
again in the upright shape.
\end{remark}
```

Note that all of the assertions are labeled for possible reference elsewhere in the document.

Frequently instead of identifying an assertion by a number, it's preferable to identify it by a name. The preamble command `\newtheorem*` with the same two mandatory arguments is used in such a case. For example typing `\newtheorem*{WO}{Well Ordering Principle}` in the preamble and in the document typing

```
\begin{WO}
Every non-empty set can be well-ordered.
\end{WO}
```

will produce

Well Ordering Principle. *Every non-empty set can be well-ordered.*

All of the environments created with either of the preamble commands `\newtheorem` or `\newtheorem*` have an optional argument that can be used for inserting additional information about the assertion before its statement. For example, typing

```
\begin{lemma}[see \cite{T}]
If $f$ is differentiable at $x$, then $f$ is continuous at $x$.
\end{lemma}
```

will produce
Lemma 2. (see [7]) *If f is differentiable at x, then f is continuous at x.*

assuming that the famous text book by George Thomas is the seventh item in the bibliography.

Also provided in the `amsthm` package is the `proof` environment, which, as its name suggests, is to contain the proof of an assertion. Typing `\begin{proof}` automatically begins the proof with, *Proof.* unless the optional argument is used. For example to start with something like, Proof of Lemma 3.1. type `\begin{proof}[\textsc{Proof of Lemma 3.1}]`. Ending the environment with `\end{proof}` produces the default end-of-proof symbol □ at the end of the line. For example

and that completes the proof. □
A problem arises if the proof ends with a displayed expression because the □ comes then at the end of the line after the end of the displayed expression; for example,

$$(a^n - b^n) = (a-b)(a^{n-1} + a^{n-2}b + a^{n-3}b^2 + \cdots + a^2b^{n-3} + ab^{n-2} + b^{n-1}.$$

□

The command `\qedhere` is provided to put the □ at the end of the last line of a displayed expression, but it won't work with if the display is created with a pair of dollar signs. In that case replace the pair of dollar signs with `\begin{equation*}` and `\qedhere\end{equation*}`. To understand why, see the discussion of this, and other environments, coming up next .

9.4 The amsmath Package.

9.4.1 Multi-Lined Expressions and Numbering.

Use this package when typing any mathematics that will require numbered or multi-line expressions. It provides you with several environments for these purposes: `equation, align, gather`, and `multline`. Each of these environments numbers each line in the environment. To suppress the numbers, follow the name with a $*$; for example `equation*`. In addition the environment `split` is provided to be used inside of a number-producing environment. LaTeX produces a small amount of (rubber) vertical space before beginning the environment and more after the environment ends. So adding vertical space should be avoided. For ease of editing the source document, always put the `\begin{...}` command on a new line and start the environment text on the next line.

The `equation` environment produces a single line of mathematics which is numbered. For example

$$\sum_{i=1}^{\infty} p(f(t_i))\mu(\sigma_i) \leq \sum_{i=1}^{\infty} g(t_i)\mu(\sigma_i) \leq \overline{\int_{\Omega} p(f(t))\,dt + \epsilon} \tag{1}$$

is created by typing

```
\begin{equation}\label{eq1}
\sum_{i=1}^\infty p(f(t_i))\mu(\sigma_i)
\leq \sum_{i=1}^\infty g(t_i)\mu(\sigma_i)
\leq \overline{\int_\Omega}p(f(t))\,dt +\epsilon
\end{equation}
```

To produce the same expression without the number, replace *equation* with *equation**. Its use was mentioned at the end of the previous segment on page 52.

To refer to a numbered equation later in the document, type `\begin{equation}\label{eq1}` and then typing `\eqref{eq1}` will produce (1).

For a calculation that requires several lines, use the `align` environment. (It replaces the `eqnarray` environment supplied by LaTeX.) A simple example of its use is

$$|f(x) - f(y)| = |h(x) - h(p_k^i)| \le L|x - p_k^i| \le L(|x - v_i| + |v_i - p_k^i|) \tag{2}$$
$$\le L(|x - v_i| + |v_i - x_{k+1}^i|) \le L(|x - y| + |x - y|) = 2 \cdot L|x - y| \tag{3}$$

which is produce by typing

```
\begin{align}
|f(x)-f(y)|&=|h(x)-h(p^i_k)|\label{eq2}
 \le L|x-p^i_k|
 \le L(|x-v_i|+|v_i-p^i_k |)\\
& \le L(|x-v_i|+|v_i-x^i_{k+1}|)\label{eq3}
 \le L(|x-y|+|x-y|)= 2\cdot L|x-y|
\end{align}
```

Note that the equation number (3) is on the following line whereas it ought to be immediately after the second line. By inserting `\!` in several places, the number can be coaxed into moving up. If this technique doesn't create enough room at the end of the line for the number, Create a third line by breaking the second line before the second $\le$ sign.

New lines are created with the `\\` command. Note that there isn't a new line command at the end of the last line nor should there be. If one is included, too much vertical space results. To suppress the number at the end of a line, insert the command `\notag` before the new line command `\\`. The point of alignment in each line is at the ampersand `&`. It should precede a relation character ($=, \le, \ge, <, >$ etc.) in order to have the appropriate amount of space on each side of the relationship symbol. If, as in the following example, the new line begins with a + sign or any character indicating the continuation of the previous line, it's best to indent the line just a bit. For example typing

```
\begin{align*}
Y_n &= \sum_{\nu =
1}^{n}{a_{n\nu}\lambda_{\nu}}\mu a_{\nu}b_\nu
= \sum_{\nu =1}^{n} a_{n\nu}\lambda_{\nu}
[\sum_{r= 1}^{\nu}ra_r - \sum_{r = 1}^{\nu - 1}ra_r]\\
&= \sum_{\nu =1}^{n-1}\Delta_{\nu}(a_{n\nu}\lambda_{\nu})
\sum_{r= 1}^{\nu}ra_r + {a}_{n\nu}\lambda_n\sum_{r = 1}^n ra_{r}\\
& \phantom{=} + \sum_{\nu = 1}^{n-1}a_{n, \nu + 1}\lambda_{\nu +
1}\frac{1}{\nu}t_{\nu} +
\frac{(n + 1)a_{n\nu}\lambda_nt_n}{n}.
\end{align*}
```

produces

$$\begin{aligned} Y_n &= \sum_{\nu=1}^{n} a_{n\nu}\lambda_\nu \mu a_\nu b_\nu = \sum_{\nu=1}^{n} a_{n\nu}\lambda_\nu [\sum_{r=1}^{\nu} r a_r - \sum_{r=1}^{\nu-1} r a_r] \\ &= \sum_{\nu=1}^{n-1} \Delta_\nu (a_{n\nu}\lambda_\nu) \sum_{r=1}^{\nu} r a_r + a_{n\nu}\lambda_n \sum_{r=1}^{n} r a_r \\ &\quad + \sum_{\nu=1}^{n-1} a_{n,\nu+1}\lambda_{\nu+1} \frac{1}{\nu} t_\nu + \frac{(n+1)a_{n\nu}\lambda_n t_n}{n}. \end{aligned}$$

The command `\phantom{=}` is used to move the + sign to the right an amount equal to the size of an = sign. The lack any equation numbers is due to the use of the $*$ version of the `align` environment.

The *split* environment is used to associate just one number with a multi-line calculation or expression. To accomplish this goal, type

```
\begin{equation}\label{eq2a}
\begin{split}
| f(x) -f(y)|& = |h(x) -h(p^i_k)|
 \le  L|x- p^i_k|
 \le  L(| x-v_i|+|v_i -p^i_k |)\\
& \le  L(| x-v_i|+|v_i-x^i_{k+1}|)\\
 &\le  L(| x-y|+|x-y|)= 2\cdot L|x-y|
\end{split}
\end{equation}
```

which produces

$$\begin{aligned} |f(x) - f(y)| = |h(x) - h(p_k^i)| &\le L|x - p_k^i| \le L(|x - v_i| + |v_i - p_k^i|) \\ &\le L(|x - v_i| + |v_i - x_{k+1}^i|) \\ &\le L(|x - y| + |x - y|) = 2 \cdot L|x - y| \end{aligned} \tag{4}$$

The option `centertags` causes the tag is placed vertically in the center of the display. It is the default setting. The other choice is `tbtags`. With this option, the number is placed at the end of the last line if the *leqno* (the default) has been selected. If the other choice, `reqno` has been selected, then the number will be at the beginning of the first line.

If the last line is too long, the number can be forced to the end of the line following the expression. For example

$$\begin{aligned} |f(x) - f(y)| = |h(x) - h(p_k^i)| &\le L|x - p_k^i| \le L(|x - v_i| + |v_i - p_k^i|) \\ &\le L(|x - v_i| + |v_i - x_{k+1}^i|) \\ &\le L(|x - y| + |x - y|) = 2 \cdot L|x - y| < M \cdot |x - y| < M \cdot \delta < \epsilon \end{aligned} \tag{5}$$

The command, `\raisetag` is available to solve this problem. By experimenting, determine by how much the tag needs to be raised. In this case `\raisetag{2\baselineskip}` placed after `\end{split}` is the correct amount.

The `split` environment can be used similarly inside of all of the others except for `multline`.

The `align` environment can be used to produce two (or more) columns of aligned equations. The points of alignment and the points where the columns end must be designated with ampersands. For example

$$8 \equiv 0 \bmod 8 \qquad 9 \equiv 1 \bmod 8 \qquad 10 \equiv 2 \bmod 8 \tag{6}$$
$$11 \equiv 3 \bmod 8 \qquad 12 \equiv 4 \bmod 8 \qquad 13 \equiv 5 \bmod 8 \tag{7}$$
$$14 \equiv 6 \bmod 8 \qquad 15 \equiv 7 \bmod 8 \qquad 16 \equiv 0 \bmod 8 \tag{8}$$

is the result of typing

```
\begin{align}
8&\equiv 0\bmod{8}&9&\equiv 1\bmod{8}&10&\equiv 2\bmod{8}\\
11&\equiv 3\bmod{8}&12&\equiv 4\bmod{8}&13&\equiv 5\bmod{8}\\
14&\equiv 6\bmod{8}&15&\equiv 7\bmod{8}&16&\equiv 0\bmod{8}
\end{align}
```

You can designate a displayed expression by what ever symbol you wish, say (*), by typing `\tag{*}` after the expression but before the line is ended by a `\\` or before the end of the environment. Using `\tab*{*}` omits the parentheses.

It is sometimes useful to number the lines of a calculation with the same number by adding a suffix, such as (3a), (3b), (3c) etc. This variation can be done using the `subequations` environment. For example in the above example

```
\begin{subequations}
\begin{align}
8&\equiv 0\bmod{8}&9&\equiv 1\bmod{8}&10&\equiv 2\bmod{8}\\
11&\equiv 3\bmod{8}&12&\equiv 4\bmod{8}&13&\equiv 5\bmod{8}\\
14&\equiv 6\bmod{8}&15&\equiv 7\bmod{8}&16&\equiv 0\bmod{8}
\end{align}
\end{subequations}
```

produces

$$8 \equiv 0 \bmod 8 \qquad 9 \equiv 1 \bmod 8 \qquad 10 \equiv 2 \bmod 8 \tag{9a}$$
$$11 \equiv 3 \bmod 8 \qquad 12 \equiv 4 \bmod 8 \qquad 13 \equiv 5 \bmod 8 \tag{9b}$$
$$14 \equiv 6 \bmod 8 \qquad 15 \equiv 7 \bmod 8 \qquad 16 \equiv 0 \bmod 8 \tag{9c}$$

Some displayed mathematics contains too many lines to be completed before the end of the page on which it starts. In this case LaTeX will move the entire calculation to the next page and spread out the text on the previous page to fill the page by using large sections of white space. The result can be ugly. To correct the situation, you can tell LaTeX that the calculation can be broken part way through by typing the declaration, `\allowdisplaybreaks` inside the environment producing the calculation. Or, if you prefer, you can put the declaration in the preamble and it will apply to all

calculation environments. If you know where you want the display to be broken, type `\displaybreak` immediately before the end of line command, `\\`. However, the `split` environment can not be broken.

Neither `gather` nor `multline` employs ampersands because no alignment is involved. The `gather` environment simply centers each of its lines giving a number to each line unless the * version is selected. To have one number associated to the entire collections of lines, put `gather*` inside of an `equation` environment as was done with `split` above. The `multline` environment moves the first line to the left hand margin (unless the option `leqno` has been selected), centers the following lines except the last one, which is moved to the right leaving room only for the equation number.

If you want, say expressions created with the `equation` environment to be numbered according to the section in which it occurs, put the command,
`\numberwithin{equation}{section}` in the preamble.

To enter non-mathematical text in the midst of a mathematical expression (displayed or in-line), use the command `\text{`*text*`}`. The `\intertext{`*text*`}` command allows you to put text between two lines of a displayed calculation without ending the environment to do so (nor disturbing the alignment in such environments). As with `\text` the text entered is governed by the same rules for entering regular text.

To put a } adjusted to encompass all lines of a multi-line expression, use one of the two variations, `gathered` and `aligned`. For example to produce

$$\left.\begin{aligned} y' &= f(x,y) \\ y(x_0) &= y_0 \end{aligned}\right\} \quad \text{Initial Value Problem}$$

type

```
$$\left.\begin{aligned}
y'&=f(x,y)\\
y(x_0)&=y_0
\end{aligned}
\right \}
\quad \text{Initial Value Problem}$$
```

Before the `amsmath` package was available, a structure similar to the one above was needed to produce something like

$$f(x) = \begin{cases} 0 & \text{if } x \text{ is a rational number} \\ 1 & \text{if } x \text{ is an irrational number} \end{cases}$$

which is somewhat clumsy and easy to get wrong. But with `amsmath` the above is done by typing

```
$$
f(x)=
\begin{cases}
0&\text{if $x$ is a rational number}\\
1&\text{if $x$ is an irrational number}
\end{cases}
$$
```

Note that there is no space between `&` and `\text{if` .... The reason is that some space is automatically included after the `&`. To number the expression, put it between

`\begin{equation}\label{}` and `\end{equation}` in place of the pair of double dollar signs. The environment may be used in line as well as in display mode.

9.4.2 Matrices.

The `amsmath` package provides 6 environment for producing matrices. They are: `matrix, pmatrix, bmatrix, Bmatrix, vmatrix, Vmatrix`, and `smallmatrix`. Each is an offshoot of the `tabular` environment in that columns are separated by & signs. To produce the matrix, $\begin{matrix} 1 & 2 & 3 & 4 \\ a & b & c & d \\ -3 & -2 & -1 & 0 \end{matrix}$, type

```
$\begin{matrix}
1&2&3&4\\
a&b&c&d&\\
-3&-2&-1&0
\end{matrix}$
```

The next four matrix environments enclose the matrix in different symbols; `pmatrix` encloses the matrix in parentheses, `bmatrix`, in brackets [ ], `Bmatrix`, in braces { }, `vmatrix` in a pair of vertical lines | |, and `Vmatrix` in a pair of double vertical lines ‖ ‖. The height of the symbols involved is expanded to match the height of the matrix. For example,

```
$$\begin{Vmatrix}
1&2&3&4\\
a&b&c&d&\\
-3&-2&-1&0
\end{Vmatrix}$$
```

will create

$$\begin{Vmatrix} 1 & 2 & 3 & 4 \\ a & b & c & d \\ -3 & -2 & -1 & 0 \end{Vmatrix}.$$

Of course these matrix structures can be assigned a number by replacing the two pairs of dollar signs with `\begin{equation}` and `\end{equation}`.

The above examples are used mostly in display mode. Use `smallmatrix` for in line math mode. However, you must supply the inclosing symbol and adjust the height of the symbols. See the next paragraph. Often in matrices dots are desired. For example

$$\text{Let } A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & \dots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & & & a_{2,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m,1} & a_{m,2} & \dots & \dots & \dots & \dots & a_{m,n} \end{pmatrix}$$

The matrix has 7 (actual) columns. The second row is produced by typing `a_{2,1}&a_{2,2}&a_{2,3}&\hdotsfor{3}&a_{2,n}`, the third by `\hdotsfor{7}` and the last row, by `a_{m,1}&a_{m,2}&\hdotsfor{4}&a_{m,n}`. The argument of the command, `\hdotsfor` is the number of columns that are to be taken up by dots.

9.4.3 Additional Useful Structures.

The height of bracketing symbols (referred to as *delimiters*[3]) like those used to enclose matrices can by controlled by typing, for example, `\left` and `\right`. However, this technique often produces a symbol that is too tall. For example $\left| \int_a^b f^p \, dt \right|^{\frac{1}{p}}$. They have two other disadvantages as well. First, if two delimiters are nested, such as $(a + (b + c))$, the outer pair isn't increased in size. Second, each `\left` must be paired with a corresponding `\right`. If the expression inside of a pair of delimiters stretches over two lines in a displayed formula, the first `\left` must be balanced with a `\right.` before the end-of-line command, `\\` and `\left.` must appear at the beginning of the next line (or the line in which the right delimiter appears). LaTeX provides the additional choices of `\big`, `\Big`, `\bigg`, and `\Bigg`. But with these the spacing is not quite right. So the `amsmath` package provides commands that produce bracketing symbols of varying heights with the appropriate spacing. They are `\bigl \bigr; \Bigl \Bigr; \biggl \biggr; \Biggl \Biggr`. One must try each one and pick the one that looks best. For example $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, is produced by typing `$\bigl(\begin{smallmatrix}a&b\\c&d\end{smallmatrix}\bigr)$`, but for $\left(\begin{smallmatrix} 1 & 0 & 2 \\ 0 & 3 & 4 \\ 2 & 5 & 0 \end{smallmatrix}\right)$ the `\Big\` and `\Bigr` combination works best. These are not required to appear in pairs which is useful it a situation like the one alluded to above. Also two different sizes can be paired; for example, $\left(\int_a^b f(x) \, dx \right)$.

LaTeX abhors using the same symbol for different purposes. The character, |, produced in math mode with the shifted forward slash, is used for a is a factor of b, $a|b$, for absolute values, $|x|$, for norms, $||X||$ or to denote the restriction of a function to a subset of its domain, $f|_A$. But LaTeX has different symbols for the last three. For absolute value use the set of delimiters, `\lvert` and `\rvert`. Compare `$|x|$` to the result from `\lvert x\rvert`; $|x|$ versus $|x|$. Their size can be adjusted using the commands introduced in the previous paragraph. For example, $\left| \int_a^b |f(t)|^p \, dt \right|^{\frac{1}{p}}$. The pair of commands `\lVert` and `\rVert` work for norms in the same fashion. Compare $||X||$ to $\|X\|$. To indicate restriction, the command `\restriction` is available. It and produces $f \restriction_A$. If the horizontal space seems to much, reduce it with a `\!`first; for example $f \restriction_A$ is produced with `$f\!\restriction _A$`. (a new command is useful for this purpose.) To denote inner produce, don't use the $<$ and $>$ symbols but rather the pair, $\langle$ and $\rangle$. Compare $< x, y >$ to $\langle x, y \rangle$. As might be expected these delimiters

[3]Delimiter sets include (), {}, | |, $\langle\rangle$, $\| \|$, $\lceil \rceil$, and $\lfloor \rfloor$.

can be increased in size as needed. Rather than use `\int` twice for a double integral, three times for triple integrals, etc. use `\iint` for double integrals, `\iiint` for triple integrals, `\iiiint` for quadruple integrals and `\idotsint` for more although this command produces the same outcome as `\int\cdots\int`. Also for $\oint$ use `\oint`.

Before proceeding, this would be a good place to discuss the `\newcommand` command in further detail. All such commands must go in the preamble. The command has two mandatory arguments and one optional one that, when it is used, is placed between the two mandatory ones and is a number indicating the number of arguments for the command being defined. For example, to easily place absolute signs around a character, in the preamble type `\newcommand{\abs}[1]{\left\lvert#1\right\rvert}` and then to put absolute value signs around an expression, type `\abs{`*expression*`}`. For example $|xy|$ is produced by typing `$\abs{xy}$`. Also larger expressions are automatically accommodated. For example, $\left|\frac{a}{b}\right|$. A similar command can be defined to produce $\|A\|$. Another possibility is `\newcommand{\rest}{\,\restriction}`.

`\newcommand` has one very annoying feature when used to insert text rather that a mathematical expression. For example the command `\st` was suggested earlier, but if it is typed followed by a strike of the space bar, no space appears. For example, typing `For all $x$ \st $x\in A$.` what appears is, "For all x such that$x \in A$.". The reason is that LaTeX interprets the strike of the space bar as singling the end of the command. Even is an additional strike of the space bar is inserted, it is ignored by LaTeX. One possible fix is to augment the definition of the command to contain a space; that is, `\newcommand{\st}{such that }`. The problem with this solution is that if `\st` comes at the end of a sentence, the space is inserted before the period—unacceptable. The better solution is to follow the command with a mandatory space; that is, a backslash followed by a space bar, or a pair of braces, `{}`.

Besides the command `\vec{x}` to produce the simple expression $\vec{x}$ commands are available to put arrows pointing in each direction or both directions over or under expressions. They are `\overleftarrow, \overrightarrow, \overleftrightarrow, \underleftarrow, \underrightarrow,` and `\underleftrightarrow`. They work in the same way as `\vec` does, with one mandatory argument, except that they will expand automatically to include all characters in the mandatory argument. For example, typing `$\overleftrightarrow{abcd}$` produces $\overleftrightarrow{abcd}$.

For left or right arrows between symbols with expressions over the arrow or under the arrow the commands `\xleftarrow` and `\xrightarrow` are available. Each has one optional argument and one mandatory argument. The mandatory argument contains the expression to go over the arrow and the optional one, the expression to go under the arrow. For example, `$X\xrightarrow[\text{onto}]{\text{1-1}}Y$` produces $X \xrightarrow[\text{onto}]{\text{1-1}} Y$. The mandatory argument may be empty, `{ }` to omit an expression above the arrow.

It's possible to have two lines in a subscript such as $\lim\limits_{\substack{n\to\infty \\ n \text{ even}}} a_n$. Such a construction is accomplished by typing

`$\lim_{\substack{n\to\infty\\ n\text{ even}}} a_n$`. `\substack` may also be used in superscripts.

LaTeX provides several commands for putting a modifier over a character. For example $\hat{a}$ is produced by `$\hat{a}$`. It's possible to put any modifier over another character in math mode with the command `\overset`. The command `\underset`reverses the positions of the modifier and the character. Each has two mandatory arguments. The first contains a modifier and the second a character to be modified. For example, `$\underset{\bullet}{A}$` produces $\underset{\bullet}{A}$. One useful application is to construct $\sum_{n=1}^{\infty}$ which is accomplished by typing `$\underset{n=1}{\overset{\infty}{\sum}}$`. If you wish to use this form several times, it would be a good idea to create a new command. In the preamble type `\newcommand{\mysum}[2]{\underset{#1}{\overset{#2}{\sum}}`. To produce the same output, simple type `$\mysum{n=1}{\infty}$`. The optional command [2] indicates that the new command is to have 2 mandatory arguments. So the command could be used to produce $\sum_{i=1}^{n}$ by typing `\mysum{i=1}{n}`.

To put symbols lower left, upper left, lower right, and/or upper right of a sum or sum-like character, the command `\sideset` is provided. For example in math mode, typing `\sideset{_{l,l}^{l,u}}{_{r,l}^{r,u}}\sum` produces ${}_{l,l}^{l,u}\sum_{r,l}^{r,u}$. It's not necessary to have symbols in all "four corners". For example ${}^{l,u}\sum$ is produced by typing `$\sideset{^{l,u}}{}\sum$`.

The commands such as `\sin,\tan \log` are fine as far as they go, but there are others that you might need. These commands are used in math mode to produce the name of the function in text font and insert a small space after the function name before the next mathematical symbol. For example `\sin x` produces $\sin x$. The command `\DeclareMathOperator` allows you to produce function names that aren't provided. For example typing `\DeclareMathOperator{\dist}{dist}` in the preamble (All of these commands must be in the preamble.) produces $\operatorname{dist}(A,B)$ when `$\dist(A,B)$` is typed. If a limit is required for the new math operator, use the * form. For example $\operatorname{ess\,sup}_{x\in A} f(x)$ is produced by first typing `\DeclareMathOperator*{\esssup}{ess\,sup}` in the preamble and then in the document typing `$\esssup_{x\in A}f(x)$` in the document.

The sticky space, `~`, can be used to prevent an unwanted line break at a space, but there are times when a line break should be prevented at a hyphen. For example in semi-group. The command to prevent line breaks in such situations is `\nobreakdash`. To use it, insert it just before the hyphen where a break should not occur. For example `semi\nobreakdash-group`.

9.5 Additional Font Families for Math Mode.

The font family used in math mode is essentially the *italic* font family with some minor changes in spacing. The *Roman* font family can be use in math mode with the command `\text{`*text in Roman font family*`}`. For *sans serif* and *typewriter* use `\textsf` and `\texttt` respectively. Similarly the commands for bold face

and the different shapes work in math mode and produce the same output as they do in text mode. Some authors prefer to use the usual Roman version of, d, in the differential of an integral, $\int f(x)\,\mathrm{d}x$, which is produced by typing `$\int f(x)\,\mathrm{d}x$`. (Rather than typing `\,\mathrm{d}\` repeatedly create a new command.) To embolden a character in math mode, for example $\boldsymbol{a}$ as opposed to $\mathbf{a}$, use the command `\boldsymbol`, which puts a character as it appears in math mode, in boldface. For example $\boldsymbol{x \cdot y}$ is produced by typing `$\boldsymbol{x\cdot y}$`.

There are additional font families available in math mode as well, but not in all shapes. These are blackboard bold (`\mathbb`), calligraphy (`\mathcal`), Fraktur (`\mathfrak`), and Euler (`\mathscr` with the package `eucal` and the option *mathscr*). The blackboard bold, calligraphy and Euler are available in upper case only. In particular the symbol, $\mathfrak{c}$ used to denote the cardinality of the real line is produced by typing `$\mathfrak{c}$`. For the real line use $\mathbb{R}$ produced with `\mathbb{R}` . Both of these are candidates for new commands; `\c` and `\R`.

10 Commutative Diagrams.

Simple commutative diagrams can be constructed using the package, `amscd`, which must be entered after the package, `amsmath`. Only diagrams with vertical and horizontal arrows can be made. These arrow may point in either direction (left or right for horizontal arrows and up or down for vertical ones). In addition symbols may be entered on either or both sides of these arrows. The commands that produce theses four types of arrows are: `@>>>` for arrows pointing to the right, `@<<<` for arrows pointing left, `@VVV` for arrows pointing down and `@AAA` for arrows pointing up. The symbols to accompany the arrows are inserted between the characters. For example `@>f>1-1>` produces a right arrow with an f over it and 1-1 beneath it. All of these commands must be given inside of the environment, `CD`. For example

```
$$\begin{CD}
X@>f>\text{1-1}> Y\\
@VhV\text{onto}V @AgA\text{iso}A\\
X/G@=Y/H
\end{CD}$$
```

produces

$$\begin{array}{ccc} X & \xrightarrow[\text{1-1}]{f} & Y \\ {\scriptstyle h}\big\downarrow{\scriptstyle \text{onto}} & & {\scriptstyle g}\big\uparrow{\scriptstyle \text{iso}} \\ X/G & = & Y/H \end{array}$$

11 Commands

Dimensions.

mm (millimeter), cm (centimeter), in (inch), pt (point 1in = 72.27 pt), pc (pica = 12pt), bp (big point 1in = 72 bp), em = width of M, ex = height of x.

Greek Letters (All in Math Mode.)

Lower Case.

α=\alpha	β=\beta	γ= \gamma	δ =\delta
ϵ=\epsilon	ε=\varepsilon	ζ=\zeta	η =\eta
θ=\theta	ϑ=\vartheta	ι=\iota	κ=\kappa
λ=\lambda	μ=\mu	ν=\nu	$\varkappa$=\varkappa
ξ=\xi	o=o	π=\pi	ϖ=\varpi
ρ=\rho	ϱ=\varrho	σ=\sigma	ς =\varsigma
τ=\tau	υ=\upsilon	ϕ=\phi	φ=\varphi
χ=\chi	ψ=\psi	ω=\omega	

Upper Case.

Γ=\Gamma	Δ=\Delta	Θ=\Theta	Λ=\Lambda
Ξ=\Xi	Π=\Xi	Σ=\Sigma	Υ=\Upsilon
Φ=\Phi	Ψ=\Psi	Ω=\Omega	Π=\Pi

Hebrew Letters (All in Math Mode).

$\aleph$=\aleph $\beth$=\beth $\daleth$=\daleth $\gimel$=\gimel

Some Letters from Other Languages.

œ=\oe	Œ=\OE	æ=\ae	Æ=\AE	å=\aa	Å =\AA	¡=!‘
ø=\o	Ø=\O	ł=\l	Ł=\L	ß=\ss	SS=\SS	¿=?‘

Miscellaneous Symbols

†=\dag ‡=\ddag §=\S ©=\copyright ¶=\P £=\pounds

Accents.

à =\‘a	á=\’a	â=\^a	ä =\"a	ã =\~a
ā =\=a	ȧ=\.a	ă=\u{a}	{a}&\verb{a}=a̋	\H{a}=
o͡o=\t{oo}	ą=\c{a}	a̲=\b{a}	å =\r{a}	

Accents in Math Mode.

$\acute{a}$=\acute{a}	$\bar{a}$ =\bar{a}	$\breve{a}$=\breve{a}	$\check{a}$ =\check{a}
$\dot{a}$=\dot{a}	$\ddot{a}$ =\ddot{a}	$\dddot{a}$=\dddot{a}	$\ddddot{a}$ =\ddddot{a}
$\grave{a}$=\grave{a}	$\mathring{a}$ =\mathring{a}	$\hat{a}$=\hat{a}	$\widehat{abc}$=\widehat{abc}
$\tilde{a}$=\tilde{a}	$\widetilde{abc}$=\widetilde{abc}	$\vec{a}$=\vec{a}	

Function Names.

sin =\sin	cos =\cos	tan =\tan	cot =\cot
sec =\sec	csc =\csc	arcsin=\arcsin	arccos=\arccos
arctan=\arctan	sinh =\sinh	cosh =\cosh	tanh =\tanh
coth =\coth	exp =\exp	ln =\ln	log =\log
deg =\deg	det =\det	dim =\dim	Pr =\Pr
arg =\arg	gcd =\gcd	max =\max	min =\min
lim =\lim	lg =\lg	inf =\inf	sup =\sup
lim inf=\liminf	lim sup=\limsup	hom =\hom	arg =\arg

Binary Operation Symbols.

$\pm$=\pm	$\cap$=\cap	$\circ$=\circ	$\bigcirc$=\bigcirc
$\mp$=\mp	$\cup$=\cup	$\bullet$=\bullet	$\Box$=\Box
$\times$=\times	$\uplus$=\uplus	$\diamond$=\diamond	$\Diamond$=\Diamond
$\div$=\div	$\sqcap$=\sqcap	$\lhd$=\lhd	$\bigtriangleup$=\bigtriangleup
$\intercal$=\intercal	$\sqcup$=\sqcup	$\rhd$=\rhd	$\bigtriangledown$=\bigtriangledown
$*$=\ast	$\vee$=\vee \lor	$\unlhd$=\unlhd	$\triangleleft$=\triangleleft
$\star$=\star	$\wedge$=\wedge \land	$\unrhd$=\unrhd	$\triangleright$=\triangleright
$\dagger$=\dagger	$\oplus$=\oplus	$\oslash$=\oslash	$\setminus$=\setminus
$\ddagger$=\ddagger	$\ominus$=\ominus	$\odot$=\odot	$\wr$=\wr
$\amalg$=\amalg	$\otimes$=\otimes	$\circledcirc$=\circledcirc	$\circleddash$=\circleddash
$\boxdot$=\boxdot	$\boxminus$=\boxminus	$\boxplus$=\boxplus	$\boxtimes$=\boxtimes
$\cdot$=\cdot	$\centerdot$=\centerdot	$\barwedge$=\barwedge	$\veebar$=\veebar
$\Cup$=\Cup	$\Cap$=\Cap	$\curlyvee$=\curlyvee	$\curlywedge$=\curlywedge

Relational Symbols.

$\blacktriangleleft$ \blacktriangleleft	$\leq$ \le \leq	$\geq$ \ge \geq	$\sim$ \sim
$\blacktriangleright$ \blacktriangleright	$\gg$ \gg	$\doteqdot$ \doteq	$\simeq$ \simeq
$\subset$ \subset	$\supset$ \supset	$\approx$ \approx	$\asymp$ \asymp
$\subseteq$ \subseteq	$\supseteq$ \subseteq	$\cong$ \cong	$\smile$ \smile
$\sqsubset$ \sqsubset	$\sqsupset$ \sqsupset	$\equiv$ \equiv	$\frown$ \frown
$\sqsubseteq$ \sqsubseteq	$\sqsupseteq$ \sqsupseteq	$\propto$ \propto	$\bowtie$ \bowtie
$\in$ \in	$\ni$ \ni	$\prec$ \prec	$\prec$ \prec
$\vdash$ \vdash	$\dashv$ \dashv	$\preceq$ \preceq	$\succeq$ \succeq
$\models$ \models	$\perp$ \perp	$\parallel$ \parallel \\|	$\mid$ \mid |
$\lessdot$ \lessdot	$\gtrdot$ \gtrdot	$\rtimes$ \rtimes	$\ltimes$ \ltimes
$\approxeq$ \approxeq	$\backsim$ \backsim	$\backsimeq$ \backsimeq	$\bumpeq$ \bumpeq
$\Bumpeq$ \Bumpeq	$\circeq$ \circeq	$\curlyeqprec$ \curlyeqprec	$\Doteq$ \Doteq
$\curlyeqsucc$ \curlyeqsucc	$\doteqdot$ \doteqdot	$\eqcirc$ \eqcirc	$\eqsim$ \eqsim
$\eqslantless$ \eqslantless	$\geqq$ \geqq	$\eqslantgtr$ \eqslantgtr	$\ggg$ \ggg
$\geqslant$ \geqslant	$\gtrapprox$ \gtrapprox	$\gtreqless$ \gtreqless	$\gtreqqless$ \gtreqqless
$\fallingdotseq$ \fallingdotseq	$\gtrless$ \gtrless	$\gtrsim$ \gtrsim	$\leqq$ \leqq
$\leqslant$ \leqslant	$\lessapprox$ \lessapprox	$\lesseqgtr$ \lesseqgtr	$\lesssim$ \lesssim
$\precapprox$ \precapprox	$\preccurlyeq$ \preccurlyeq	$\precsim$ \precsim	$\triangleq$ \triangleq
$\succsim$ \succsim	$\succsim$ \succsim	$\succcurlyeq$ \succcurlyeq	$\thicksim$ \thicksim
$\risingdotseq$ \risingdotseq	$\thickapprox$ \thickapprox	$\Subset$ \Subset	$\subseteq$ \subseteq
$\ll$ \ll	$\Supset$ \Supset	$\supseteqq$ \supseteqq	$\vartriangle$ \vartriangle

Most of the symbols above can be negated by placing a `\not` before the command. For example $\notin$ is produced by typing `\not\in`. The following table gives alternate negations

Negated Relational Symbols.

$\nless$ `\nless`	$\nleq$ `\nleq`	$\nleqslant$ `\nleqslant`	$\nleqq$ `\nleqq`
$\lneq$ `\lneq`	$\lneqq$ `\lneqq`	$\lvertneqq$ `\lvertneqq`	$\lnsim$ `\lnsim`
$\nprec$ `\nprec`	$\npreceq$ `\npreceq`	$\precnsim$ `\precnsim`	$\nsim$ `\nsim`
$\nmid$ `\nmid`	$\precnapprox$ `\precnapprox`	$\nshortmid$ `\nshortmid`	$\nvdash$ `\nvdash`
$\nvDash$ `\nvDash`	$\ntriangleleft$ `\ntriangleleft`	$\nsubseteq$ `\nsubseteq`	$\subsetneqq$ `\subsetneqq`
$\gneqq$ `\gneqq`	$\ntrianglelefteq$ `\ntrianglelefteq`	$\gvertneqq$ `\gvertneqq`	$\lnapprox$ `\lnapprox`
$\neq$ `\ne \neq`	$\varsubsetneq$ `\varsubsetneq`	$\gnapprox$ `\gnapprox`	$\gneq$ `\gneq`
$\ngtr$ `\ngtr`	$\varsubsetneqq$ `\varsubsetneqq`	$\subsetneqq$ `\subsetneqq`	$\ngeqslant$ `\ngeqslant`
$\ngeq$ `\ngeq`	$\ngeqslant$ `\ngeqslant`	$\ngeqq$ `\ngeqq`	$\gnapprox$ `\gnapprox`
$\gnsim$ `\gnsim`	$\succnapprox$ `\succnapprox`	$\nsucc$ `\nsucc`	$\succnsim$ `\succnsim`
$\nsucceq$ `\nsucceq`	$\nshortparallel$ `\nshortparallel`	$\ncong$ `\ncong`	$\ncong$ `\ncong`
$\nvDash$ `\nvDash`	$\ntriangleright$ `\ntriangleright`	$\nVdash$ `\nVdash`	$\nsupseteq$ `\nsupseteq`
$\nsupseteqq$ `\nsupseteqq`	$\ntrianglerighteq$ `\ntrianglerighteq`	$\supsetneq$ `\supsetneq`	$\supsetneqq$ `\supsetneqq`
$\notin$ `\notin`	$\varsupsetneqq$ `\varsupsetneqq`		

Delimiters.

$\lfloor$ `\lfloor`	$\rfloor$ `\rfloor`	$\lceil$ `\lceil`	$\rceil$ `\rceil`	$\langle$ `\langle`	$\rangle$ `\rangle`
$\ulcorner$ `\ulcorner`	$\urcorner$ `\urcorner`	$\llcorner$ `\llcorner`	$\lrcorner$ `\lrcorner`		

Arrows.

$\circlearrowleft$ `\circlearrowleft`	$\circlearrowright$ `\circlearrowright`	$\curvearrowleft$ `\curvearrowleft`
$\curvearrowright$ `\curvearrowright`	$\dashrightarrow$ `\dasharrow`	$\dashleftarrow$ `\dashleftarrow`
$\dashrightarrow$ `\dashrightarrow`	$\Downarrow$ `\Downarrow`	$\downarrow$ `\downarrow`
$\downdownarrows$ `\downdownarrows`	$\downharpoonright$ `\downharpoonright`	$\gets$ `\gets`
$\hookleftarrow$ `\hookleftarrow`	$\hookrightarrow$ `\hookrightarrow`	$\Leftarrow$ `\Leftarrow`
$\leftarrow$ `\leftarrow`	$\leftarrowtail$ `\leftarrowtail`	$\leftharpoondown$ `\leftharpoondown`
$\leftharpoonup$ `\leftharpoonup`	$\leftleftarrows$ `\leftleftarrows`	$\Leftrightarrow$ `\Leftrightarrow`
$\leftrightarrow$ `\leftrightarrow`	$\leftrightarrows$ `\leftrightarrows`	$\leftrightharpoons$ `\leftrightharpoons`
$\leftrightsquigarrow$ `\leftrightsquigarrow`	$\Lleftarrow$ `\Lleftarrow`	$\Longleftarrow$ `\Longleftarrow`
$\longleftarrow$ `\longleftarrow`	$\Longleftrightarrow$ `\Longleftrightarrow`	$\longleftrightarrow$ `\longleftrightarrow`
$\longmapsto$ `\longmapsto`	$\Longrightarrow$ `\Longrightarrow`	$\longrightarrow$ `\longrightarrow`
$\looparrowleft$ `\looparrowleft`	$\looparrowright$ `\looparrowright`	$\Lsh$ `\Lsh`
$\mapsto$ `\mapsto`	$\multimap$ `\multimap`	$\nearrow$ `\nearrow`
$\nwarrow$ `\nwarrow`	$\restriction$ `\restriction`	$\Rightarrow$ `\Rightarrow`
$\rightarrow$ `\rightarrow`	$\rightarrowtail$ `\rightarrowtail`	$\rightharpoondown$ `\rightharpoondown`
$\rightharpoonup$ `\rightharpoonup`	$\rightleftarrows$ `\rightleftarrows`	$\rightleftharpoons$ `\rightleftharpoons`
$\rightrightarrows$ `\rightrightarrows`	$\rightsquigarrow$ `\rightsquigarrow`	$\Rrightarrow$ `\Rrightarrow`
$\Rsh$ `\Rsh`	$\searrow$ `\searrow`	$\swarrow$ `\swarrow`
$\to$ `\to`	$\twoheadleftarrow$ `\twoheadleftarrow`	$\twoheadrightarrow$ `\twoheadrightarrow`
$\Uparrow$ `\Uparrow`	$\uparrow$ `\uparrow`	$\Updownarrow$ `\Updownarrow`
$\updownarrow$ `\updownarrow`	$\upharpoonleft$ `\upharpoonleft`	$\upharpoonright$ `\upharpoonright`
$\upuparrows$ `\upuparrows`		

Negated Arrows.

$\nLeftarrow$ `\nLeftarrow`	$\nleftarrow$ `\nleftarrow`	$\nLeftrightarrow$ `\nLeftrightarrow`
$\nleftrightarrow$ `\nleftrightarrow`	$\nRightarrow$ `\nRightarrow`	$\nrightarrow$ `\nrightarrow`

Mathematical Symbols in Two Sizes.

$\sum$ $\displaystyle\sum$	`\sum`	$\cap$ $\displaystyle\bigcap$	`\bigcap`	$\odot$ $\displaystyle\bigodot$	`\bigodot`
$\int$ $\displaystyle\int$	`\int`	$\cup$ $\displaystyle\bigcup$	`\bigcup`	$\otimes$ $\displaystyle\bigotimes$	`\bigotimes`
$\oint$ $\displaystyle\oint$	`\oint`	$\sqcup$ $\displaystyle\bigsqcup$	`\bigsqcup`	$\oplus$ $\displaystyle\bigoplus$	`\bigoplus`
$\prod$ $\displaystyle\prod$	`\prod`	$\vee$ $\displaystyle\bigvee$	`\bigvee`	$\uplus$ $\displaystyle\biguplus$	`\\biguplus`
$\coprod$ $\displaystyle\coprod$	`\coprod`	$\wedge$ $\displaystyle\bigwedge$	`\bigwedge`		

References

[1] American Mathematical Society. *User's Guide for the* `amsmath` *Package*, 2.0 edition, 2002. `http://www.ams.org.tex.amsmath.html`.

[2] Michael Goossens and Sebastian Rahtz. *The LaTeX Web Companion: Integrating TeX HTML and XML*. AW, 1999. 8.1.3

[3] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voss. *The LaTeX Graphics Companion*. Addison-Wesley, second edition, 2007.

[4] Helmut Kopka and Patrick W. Daly. *Guide to LaTeX*. Addison-Wesley, fourth edition, 2004.

[5] Leslie Lamport. *A document preparation system*. Addison-Wesley, second revised edition, 1994.

[6] Frank Mittelbach and Michael Goossens. *The LaTeX Companion*. Addison-Wesley, second edition, 2004. 3.4, 5.8

Index