LaTeX for Math and Science

2nd edition

Cordelia Csar Elizabeth Dyer Fernand Garin
Alan Leong Thomson Nguyen Dan Volmar

February 1, 2011

LIST OF TABLES

1.1 What It Does

LaTeX dominates academic publishing in science and mathematics. If you want an academic career in these fields (particularly mathematics), you must learn to love LaTeX. Those that merely want to type up their homework can get by with a casual fondness.

While not as friendly as the cartoon paperclip, LaTeX is not at all difficult to use. In fact, the great news is that LaTeX possesses a very flat learning curve. This means that once you get into it, getting at all the most advanced features is not that difficult. A little practice is all that is needed to produce textbook-quality technical documents. Unfortunately, for all of its virtues, LaTeX is not all that friendly to the beginner. In fact, that small bit of practice necessary for getting textbook-quality is not that far from what is needed to do some basic typesetting in LaTeX.

It will run on any modern computer system (and most unmodern computer systems) and is well supported by a community of developers, enthusiasts, and fanatical zealots. It can be extended to satisfy a wide variety of publishing needs, and a vast repository of such user-made packages are available on the Internet. Most significantly for academia: LaTeX is free. Departments are too cheap to buy licenses for big commercial software products, and students are too lazy to learn how to use a pirated copy of Mathematica. Before long, you may decide to junk Microsoft Word and the stupid paperclip and do all of your word processing with LaTeX.

1.2 Where It Came From

The TeX language was written by Stanford computer scientist Donald Knuth in the late 1970s. Knuth is best known for *The Art of Computer Programming* books which are of biblical proportions in academic computer science. Upset by the poor quality of the typesetting in the published editions of his books, Knuth decided that he could do better. The first version of TeX was completed in 1978 and ran on a huge mainframe computer at Stanford. Although TeX began as a research project, the mathematics community became interested because it promised to be a cheap solution for academic publishing. The

American Mathematical Society even sponsored their own TEX implementation, predictably titled $\mathcal{AMS}$-TEX.

In addition to TEX, Knuth also wrote METAFONT, a language for producing document fonts from mathematical curves. This system, and its successor METAPOST, are still used for generating fonts and figures for technical documents.

The TEX language itself is rather primitive. You could typeset an entire document in pure TEX if you wanted to, but you could also drive on the wrong side of the road. Donald Knuth wrote what is called Plain TEX, a set of higher-level commands written in simpler TEX commands, to make things easier for mere mortals. However, even in Plain TEX, the author has to do more work setting the type than actually composing the document. Various languages derived from TEX have been developed to be more functional than Plain TEX. Of these, LATEX is the most prevalent. It was written in 1984 by research computer scientist Leslie Lamport. Thus, LATEX is not a software package like Microsoft Office or even a stand-alone computer language. Rather, it is what is called a macro language—a set of commands written in TEX. In practice, it functions like a typical markup language such as HTML. Development on LATEX still continues: the current version is called LATEX 2_ε and small updates are made to the system about every six months. The community is working towards the next major release, aptly, LATEX3.

1.3 The Strange Name

The funny typesetting of the word TEX is supposed to be a rendition of the Greek letters $\tau\epsilon\chi$, which are transliterated as *tex*. It should be pronounced "tekh" where "kh" indicates a voiceless velar fricative (as a German would say "J.S. Bach"). However, most lazy Americans simply say "tek." The name is in honor of Caltech, where Donald Knuth did his graduate work. A popular folk etymology is that the name came from Knuth's verbal reaction to the lousy typesetting of the second volume of *The Art of Computer Programming*: "blech."

LATEX is for Lamport TEX, after the initial author Leslie Lamport. It is usually pronounced "lay-tek(h)," although some say "lah-tek(h)." To say "le-tek(h)" is incorrect and should be strictly shunned. Apparently, the crazy typesetting Knuth used for the word TEX was contagiouis and was embraced by Lamport as the word is printed as LATEX.

1.4 The LaTeX System

As was painfully explained earlier, LaTeX itself is a document language. More specifically, it is a macro language for the TeX language. Just like all computer languages, it must either be compiled or interpreted in order to be of any use. The actual document preparation takes place in a text editor, such as Emacs or vi. Vi is included in any POSIX-compliant operating system, such as UNIX and most Unix-like operating systems. Emacs is commonly found on UNIX and Unix-like operating systems, and more recently, these editors can be found on Mac OS X as well (which is really just bad UNIX). You can use DOS EDIT or Windows Notepad if you like wasting time. A usable document usually carries a `.tex` extension is an ordinary ASCII text file.

In general terms, a compiler is a software program that translates a human-readable source file into machine code that can be executed by a computer. Similarly, the LaTeX compiler translates a `.tex` file (the source) into a machine-readable format. The output of the standard LaTeX compiler is what is called DVI for math for "device independent." A `.dvi` file will appear identical on any kind of printer or computer display. However, viewing a `.dvi` file on anything other than a UNIX or Unix-like operating system proves to be a chore as DVI viewers are less common on Mac OS and Microsoft Windows. Compilers have also been written to output a `.tex` document directly into PostScript, PDF, or even HTML. It is also possible to convert from DVI to these formats. PostScript is a language common to most larger laserjet printers, and allows you to print the document. Additionally, PostScript was a common way to exchange documents over the Internet prior to the advent of PDF files. Adobe's Portable Document Format (PDF) is in some ways the successor to PostScript files for exchange over the Internet. They are readable on nearly any piece of computer hardware that has a sufficiently-sized screen, and enough memory to store the file. It is important to note that it is typically difficult to impossible to recover a `.tex` document from a compiled document. Thus, it is important to keep the source files around if further editing is anticipated.

LaTeX itself is fairly comprehensive and can accommodate a wide variety of publishing needs. It can, however, be extended with the addition of user-defined macro packages. A package is an independent source file that adds features to the language by defining them in terms of existing LaTeX commands (these are called macros). This is the same principle by which LaTeX is derived from Plain TeX; LaTeX is actually one big TeX macro package. An author can include a package in a document and make use of these new features. Packages can greatly simplify document creation by providing additional commands

that would take time and/or extensive programming knowledge to implement. The American Mathematical Society provides macro packages that extend on the already rich mathematical typesetting ability of LaTeX. This is collectively known as the $\mathcal{AMS}$-LaTeX system, and it can be considered to be the mathematician's god package. Also available are packages that add support for new classes of documents, graphics, illustrations, publishing in different languages, and even printing chess boards and crossword puzzles.

BibTeX is another important extension to TeX and LaTeX. BibTeX automates the creation of bibliographies and textual citations. A separate source file is used to store the publication data for books, journals, and other articles that can be referred to by a TeX document. An author can create a list of commonly used citations and quickly refer to them in any document. When compiled, BibTeX automates the formatting of citations and generates the bibliography, all according to a customizable style package. A large number of BibTeX packages are available to handle different academic, legal, and professional citation styles.

Fonts for use with LaTeX are usually written in METAFONT or METAPOST and then compiled. Since fonts are defined by geometric curves (the curious can Google "Bézier curve"), they can be extremely versatile. The same font can be used on any computer system for which a METAFONT or METAPOST compiler is available. Often, technical illustrations are written in METAPOST. Upon execution, the LaTeX compiler will reference the fonts that it needs and render them appropriately.

The advantage of this approach is that LaTeX is not married to any specific operating platform. A `.tex` document can be written on any computer system with a text editor. It is only necessary to provide the compiler for a given operating system. With a bit of work, LaTeX will run on anything that you can connect to a monitor, keyboard, and printer to, or even those systems that you cannot connect a printer to. Furthermore, regardless of platform, LaTeX will produce identical output.

However, compiling and configuring a complete LaTeX system is a demanding and sometimes painful undertaking. To remedy this, some people have been kind enough to pre-package all of the necessary software and automate the process. These complete, ready-to-use LaTeX systems are called distributions. MiKTeX is the most popular for Windows. proTeXt is derived from MiKTeX and can ostensibly be run directly from a CD or flash drive. teTeX is a common distribution on UNIX and Mac OS X, and TeX Live is a cross-platform collection of distributions.

While there are die hards who would rather face certain death than stop

using Emacs, and another group of die hards who await the day when Emacs users will see the light and use vi, some people feel that a text editor is not the most productive environment for producing LaTeX documents. Furthermore, many people are not familiar with Emacs or vi, and have no need to learn the intricacies of either editor. To this end, software developers have produced a number of alternatives. LyX and TeXmacs attempt to reproduce the friendliness of Word or WordPerfect. These programs are pseudo-WYSIWYG ("what you see is what you get") interfaces to an underlying LaTeX system. However, this approach also obscures the high degree of control the cryptic-looking source code provides. The LaTeX front-end is another approach; a frontend is simply a text editor with some creature comforts like syntax highlighting to help you navigate your source code, toolbars for common commands, help with matching parentheses, and project management. These usually integrate with an existing LaTeX distribution and link to the LaTeX compiler; this allows you to compile your document without ever leaving the front-end. WinEdt and TeXnicCenter are popular front-ends for Windows, TeXShop is popular for Mac, and there are many implementations for UNIX systems. For those that still can't get enough of their UNIX terminal, AUCTeX adds additional TeX functionality to Emacs and LaTeX-suite will extend the vim implementation of vi with LaTeX-friendly features.

1.5 Documentation and the TeX Community

Because there are so many different components that make up a LaTeX system, documentation is unfortunately very scattered. The information that you need may not be part of LaTeX itself, but a specific package or perhaps your distribution. The various distributions are usually very good at providing the documentation to the packages that they support. However, there are so many packages available that a problem might only be solved by an appeal to the larger TeX community. The Comprehensive TeX Archive Network (CTAN) and the TeX Users Group are the two largest Internet communities that provide repositories of TeX packages and documentation. The benefit of the collaborative nature of LaTeX is its extreme versatility, but there is a price to be paid. There is no stupid cartoon paperclip to answer your questions. Fortunately, the answers are available to those willing to make a little effort.

LaTeX differs from common word processors in that it requires the author to indicate the logical structure of the document. This involves identifying sections, subsections, titles, and paragraphs among others. Only then will LaTeX derive a text document that looks remotely like what you intended it to be. In this chapter, we will cover the basic structure of every LaTeX document in addition to simple formatting commands available in LaTeX. We begin with the syntactic structure of basic LaTeX documents.

2.1 Structure

Like any programming language, LaTeX enjoys a very strict and unforgiving structure.

Command Structure

Commands in LaTeX have a very intuitive and flexible structure, like most programming languages. Most of the commands that you will encounter in LaTeX will look like this:

```
\command{argument}
```

where the name of the command replaces `command`, and any argument or modifiers you have replace `argument`. Some examples include

```
\texttt{Text} \textit{Text} \textbf{Text}
```

The first example will make text a cool typewriter font, while the second and third examples will make your text *italic* and **bold**, respectively. Note that all commands start with a backslash.

Environment Structure

Environments are special commands that modify large blocks of text. Environments almost always start with `\begin{environment}`, and end with `\end{environment}`, where the name of the environment replaces *environment*. The portion of the document between the `\begin{}` and `\end{}` is in the selected environment. An environment affects the behavior of LaTeX. For

example, here is an environment called `texttt` which alters the font LaTeX uses:

```
\begin{texttt}
  This is typewriter text
\end{texttt}
```

Documents themselves are also environments:

```
\begin{document}
Document goes here.
\end{document}
```

The second example is something that will be used quite often: it's the beginning and ending arguments for a document!

Document Structure

Here's a sample document that outlines the basic structure of LaTeX:

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

\title{Some Really Important Results}
\author{My Name}

\begin{document}
\maketitle

  \chapter{Chapter Name}
  A Document Body would go here.

  \section{The First Section}
  Here's the first section of my document.  It's
  wicked cool.

  \section{The Second Section}
  Here's the second section of my document.  This
  is twice as cool as the previous one.
```

```
    \subsection{A Subsection}
    Sometimes things aren't important enough to get
    their own section, so here's a subsection.

    \subsubsection{Subsubsections Exist?}
    I guess they do.

    \paragraph{Paragraph Title}
    Why paragraph have titles, I have no idea.

    \section*{The Third Section}
    This is the third section of my document, but
    this will show without a number because of the
    asterisk.
  \end{document}
```

When compiled, it'll look something like this:

Chapter 1

Chapter Name

A Document Body would go here.

1.1 The First Section

Here's the first section of my document. It's wicked cool.

1.2 The Second Section

Here's the second section of my document. This is twice as cool as the

previous one.

1.2.1 A Subsection

Sometimes things aren't important enough to get their own section, so here's a subsection.

Subsubsections Exist?

I guess they do.

Paragraph Title Why paragraph have titles, I have no idea.

The Third Section

This is the third section of my document, but this will show without a number because of the asterisk.

Preamble

One can see that before the actual document body, there's some weird text at the beginning of our sample document. This is called the preamble of the document, and tells LaTeX exactly how you want your document to be structured. Our sample document uses the `article` document class, at 12pt font. The title of your document goes in between the braces in the command `\title{}`, while the author name(s) go in `\author{}`. You'll see that one package has been declared with `\usepackage{}`, `amsmath`; packages and `amsmath` will be covered in later chapters, so we'll just ignore it for now. The preamble is ended with the command `\begin{document}`, which begins our document. One warning: In order for our title to display in LaTeX, the titling command `\maketitle` must be used *immediately after* `\begin{document}` has been declared.

Body

In most LaTeX documents, you'll want to split your text up into discrete sections or parts in order to make reading your document accessible and easy to your readers. LaTeX makes this easy with special commands that are placed at strategic places in your document. To reiterate, Table 2.1 shows the available sectioning commands.

Table 2.1: Sectioning commands

Command	Description
`\chapter{...}`	Starts a numbered chapter (Chapter 1)
`\section{...}`	Starts a numbered section (Section 1.1)
`\subsection{...}`	Starts a numbered subsection (Subsection 1.1.1)
`\paragraph{...}`	Starts a paragraph (w/o numbering)
`\subparagraph{...}`	Starts a subparagraph (w/o numbering)
`\part{...}`	Starts a part (with Roman numeral numbering)

The mechanics of these commands will be explained later, but for now, it'll suffice to say that each successive command creates a smaller title, similar to the chapters and sections you see in your usual textbook. It should be noted that the `\chapter` command is available only in the `report` or `book` class.

Note: If you want any of the numbered commands without the numbers (i.e., unmarked chapters), simply add a * at the end of the command. For example, `\section*{Section}` will yield:

Section

instead of

1.1 Section

2.2 Formatting

Just like any other word processor, LaTeX enjoys a good deal of strict and unforgiving formatting commands.

Emphasizing Words

Just like any word processor, you can underline and *italicize* words whenever you see fit. Underlining is done with the command `\underline{}`, where the

underlined text goes between the braces. Italicization is done with `\textit{}`, with the desired italicized text between the braces. Likewise, `\textbf{}` gives bold face.

```
I like to \underline{underline} text here and
\underline{everwhere}. I like it \textit{so}
much, I can't \textit{\underline{stop!}}
```

I like to underline text here and everywhere. I like it *so* much, I can't *stop!*

Left/Right/Center Justification

Justification in LaTeX is easy to do. If you want to left-align text, you must put the text you want left-justified in the `flushleft` environment. That is between the commands `\begin{flushleft}` and `\end{flushleft}`.

```
\begin{flushleft}
  Left-aligned text is fun.  I like left-aligned
  text. I think you should like left-aligned text
  too!
\end{flushleft}
```

Left-aligned text is fun. I like left-aligned text. I think you should like left-aligned text too!

If you have not figured it out already, right-aligned text is done with the `flushright` environment.

```
\begin{flushright}
  Whoa, this text is right-aligned.  What the heck?
  Who uses right-aligned text anyway?
\end{flushright}
```

Whoa, this text is right-aligned. What the heck? Who uses right-aligned text anyway?

Centering text is also possible in LaTeX, except instead of using the `flushcenter` environment (which doesn't even exist), we use the `center` environment by using the `\begin{center}` and `\end{center}` commands.

```
\begin{center}
  This text is centered.  Centered text makes me
  feel at peace. These examples make no sense,
  don't they?
\end{center}
```

This text is centered. Centered text makes me feel at peace. These examples make no sense, don't they?

Quotes and Verses

When using quotation marks, it's probably a good idea *not* to use the " key for opening and closing quotes. Rather, we use two grave accent (``) characters for opening quotes, and two single quotes ('') for closing quotes.

Bad I think it was Kant who said "Let my people go." Or perhaps that was Moses.

Good I think it was Kant who said ``Let my people go.'' Or perhaps that was Moses.

If your quotation is long (say, more than four lines long), it might be a good idea to use the `\begin{quote}` and `\end{quote}` commands, as they will automatically be indented in text. However, separate paragraphs within the `quote` environment won't be indented like regular paragraphs. To do that, use the `quotation` environment instead. Furthermore, to typeset poetic verses, use the `verse` environment.

Quote:

```
One of my favorite quotes from \textit{Principles
of Mathematical Analysis} goes something like this:
\begin{quote}
  If f is a continuous complex function on [a,b],
  there exists a sequence of polynomials $P_n$ such
  that $\lim_{n\rightarrow \infty} P_n(x)=f(x)$
  uniformly on [a,b]. If f is real, the $P_n$ may
  be taken real.
\end{quote}

Math is very fun!
```

One of my favorite quotes from *Principles of Mathematical Analysis* goes something like this:

> If f is a continuous complex function on [a,b], there exists a sequence of polynomials P_n such that $\lim_{n\rightarrow\infty} P_n(x) = f(x)$ uniformly on [a,b]. If f is real, the P_n may be taken real.

Math is very fun!

Quotation:

```
This text will contain a quotation with indentation
very soon.  Maybe about now!
\begin{quotation}
  This quotation has indentations at the beginning
  of every paragraph.

  The quick brown fox jumps over the lazy dog.
  Unfortunately, he jumped into hazardous terrain
  and died.

  Yoo hoo!
\end{quotation}
```

This text will contain a quotation with indentation very soon. Maybe about now!

> This quotation has indentations at the beginning of every paragraph.
>
> The quick brown fox jumps over the lazy dog. Unfortunately, he jumped into hazardous terrain and died.
>
> Yoo hoo!

Verse:

```
I know only one English poem by
heart. It is about Humpty Dumpty.
\begin{flushleft}
  \begin{verse}
    Humpty Dumpty sat on a wall:\\
    Humpty Dumpty had a great fall.\\
    All the King's horses and all
    the King's men\\
    Couldn't put Humpty together again.
  \end{verse}
\end{flushleft}
```

I know only one English poem by heart. It is about Humpty Dumpty.

> Humpty Dumpty sat on a wall:
> Humpty Dumpty had a great fall.
> All the King's horses and all the King's men
> Couldn't put Humpty together again.

Line and Page Breaks

LaTeX automatically breaks up lines, adds spaces between words, and auto-hyphenates words when necessary. If you have noticed from this reader, every line is optimized so the lengths are the same. Normally, the first line of every paragraph is automatically indented, with no additional space between

paragraphs. When it becomes necessary to add a line break, two commands are available: `\\` and `\newline`. Both will break an additional line. LaTeX will automaticallyl spill over to another page when there is too much content for one page; hence, premature page breaks are done with the command `\newpage`.

If you would like to doublespace a document, this is not easily accomplished with LaTeX; however it is fairly straightforward with the use of a package. See Section 8.3 in the packages chapter.

The Space Between Words[1]

To get a straight right margin in the output, LaTeX inserts varying amounts of space between the words. It inserts slightly more space at the end of a sentence, as this makes the text more readable. LaTeX assumes that sentences end with periods, question marks, or exclamation marks. If a period follows an uppercase letter, this is not taken as a sentence ending, since periods after uppercase letters normally occur in abbreviations.

Any exception from these assumptions has to be specified by the author. A backslash in front of a space generates a space that will not be enlarged. A tilde '~' character generates a space that cannot be enlarged and additionally prohibits a line break. The command `\@` in front of a period specifies that this period terminates a sentence even when it follows an uppercase letter.

The additional space after periods can be disabled with the command `\frenchspacing`, which tells LaTeX *not* to insert more space after a period than after ordinary characters. This is very common in non-English languages, except bibliographies. If you use `\frenchspacing`, the command `\@` is not necessary.

2.3 The Title Block

A title block is an easy way to make your document look professional; it is very easy to do in LaTeX. A title block in LaTeX contains the title of the document, the author(s), and optionally the date. You can place a title block either at the top of your document or within a cover page, which is on a separate page and is not numbered; however, the cover page is only natively available in the `report` and `book` classes (you have to specify the `titlepage` option as an optional argument to the `article` document class to get it).

To make a title page (or a title block), you need to declare its contents by including

[1] Oetiker, Tobias, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LaTeX 2ε*. May 2006. p. 34

```
\title{Why Word Sucks}
\author{The Berkeley \LaTeX{} DeCal Staff
\thanks{Supported by Professor Ribet} \\ Berkeley, CA
\and The \LaTeX{} Community \\ Around the World}
\date{2005}
```

in the preamble, or any place before the command to make the cover page given by `\maketitle`. The `\maketitle` command should then be placed right after the `\begin{document}` command.

As you can see in the sample declaration of the contents of the title block, you can use the `\and` command to introduce multiple authors. The `\thanks` command creates footnotes (automatically numbered, of course; would you expect anything less from LaTeX?) at the bottom of the cover page. This can be accomplished by using the `\\` command. If you do not use the `\date` command, today's date will automatically be inputted.

2.4 References

The way that LaTeX deals with cross-references to other sections, tables, figures, or theorems is one of the best attributes of LaTeX. Usually, when referencing different sections, tables, figures, pages, etc., you have to manually look up what the number is and place it in your document. This is fine if you never make any changes to your document, but we all know that everything seems to be revised constantly. This can make it a huge pain to go back through all of your references to change all of them by one number. LaTeX avoids this annoyance by having a way to define and recall references.

In order to reference something in LaTeX, you need to define a label for the reference. This is done by the command `\label{marker}` where `marker` is the reference that you will need to remember to reference the part later. The `\label` command saves the last number that was generated so it is usually best to put it in the same line as the `\begin` command for that part to make things standardized.

If you are cross-referencing a table or a figure, you will need to put the label in the `\caption` command. Here is an example below:

```
\begin{figure}[!h]
  \includegraphics{filename}
  \caption{I hate graphics in \LaTeX{}}
  \label{fig:hategraphics}
```

```
\end{figure}
```

Now if you want to reference the number of the thing you labeled, use the `\ref{marker}` command using the `marker` that you chose before. If I wanted to reference the figure example above, I would type

```
\ref{fig:hategraphics}
```

You can also use `~\ref{marker}` without a space between the word that precedes it. To use this with the figure example above, I would type

```
Figure~\ref{fig:hategraphics}
```

You can also reference the page number of a label by using the `pageref` command that is used in the same way as the `ref` command.

2.5 Footnotes

In order to create footnotes in a LaTeX document, you use the `\footnote` command. So that you know you are using footnotes correctly, footnotes should be placed after the word or sentence (after the comma or period) they are referring to. In the event a footnote within a section command is required, precede the `\footnote` command with `\protect`, like so:

```
\section{Foo\protect\footnote{This is a footnote!}}
```

Here is an example of a footnote in action:

```
I like bananas. I know that mangoes are sweet. I like
papayas. But I know that nothing can beat\ldots{}
\LaTeX{}! Hell yes!\footnote{Based on a chant from %
World Youth Day.}
```

I like bananas. I know that mangoes are sweet. I like papayas. But I know that nothing can beat…LaTeX! Hell yes![2]

[2] Based on a chant from World Youth Day.

One common usage of LaTeX is the typesetting of mathematical discourse and documents with mathematical content. LaTeX comes with a good deal of functionality in this area and has become a fairly standard tool in the math community. In addition, the math typesetting abilities of LaTeX can be further expanded by the use of the $\mathcal{AMS}$-LaTeX packages.

Packages in general will be discussed later, but for now, it suffices to add

```
\usepackage{amsmath,amsfonts,amssymb,amsthm}
```

to the preamble of any document you are typesetting with math content. This places additional environments and symbols at your disposal that make life far easier.

3.1 Math Mode

The key tool for typesetting mathematical content is math mode. Math mode is accessed through a variety of math environments. In math mode, the basic behavior of LaTeX is altered in a few important ways. First of all, the letters on the keyboard become constants and variables. For example, an `a` in math mode will be displayed as a while in normal text, it is an a. This gives them an italicized appearance to make them typographically different from their textual counterparts. Note that this is not the same as typesetting text in italics, and math mode should never be used to italicize text.

While in math mode, the fast majority of characters on your keyboard will continue to function as normal. In addition to the letters available on the keyboard, there are a wide variety of other characters and symbols available through various LaTeX commands. LaTeX also offers an extensive array of hats, bars, and other useful accents.

The math mode chapters do not have many examples. However, there is a selection of examples following the chapters.

3.2 Math Environments

LaTeX is extremely powerful, not least in its ability to typeset math. There are three math environments: `math`, `displaymath`, and `equation`. The inline environment allows the inclusion of math in a normal line of text, while the

displayed environment enables one to set off lines of math. The equation environment is essentially the same as the displayed environment, but each instance is numbered sequentially. $x+y=z$ is an example of the inline environment, while

$$x+y=z$$

is the same thing in the displayed environment.

Inline Math

Like other LaTeX environments, inline math, or more accurately, the `math` environment, is delimited by `\begin` and `\end` commands. In TeX, inline math was set off by dollar signs. This convention was carried over to LaTeX, which allows us a shortcut, as all those `\begin` and `\end`s would get very tedious very quickly. Using the `$` convention, the above example would appear in one's source as

```
$x+y=z$ is an example of the inline environment.
```

Displayed Math

The displayed math environment is formally delimited by

```
\begin{displaymath}
\end{displaymath}
```

However, one can also use `\[` at the beginning and `\]` at the end of the statement. Thus, using square brackets as the delimiters,

$$x+y=z$$

would appear in one's source as

```
\[x+y=z\]
```

Equations

The equation environment returns to the conventional delimiters for environments. It is accessed by `\begin{equation}` and `\end{equation}`. Our example equation in the equation environment would appear as

$$x+y=z \tag{3.1}$$

Stars (Well, Asterisks Really)

Various environments and structures in LaTeX will automatically number themselves, like the `equation` environment above. However, unlike `equation`, not everything has a separate unnumbered counterpart. To avoid the numbering, an asterisk (*) is used to disable numbering. This fact is of little concern now, but it is a fact to file away, as it will come up again later.

3.3 Commands

If you take a look at the keyboard, you'll see a number of symbols that look fairly useful for typing math, including `+`, `-`, `*`, `/`, `=`, (, and), essentially the same things you find on a small calculator. Which is all well and good, up to a point, but what about typing homework? Or textbooks for that matter, as your math textbooks were most likely done with LaTeX. It is possible to include symbols in math environments that are not available on one's keyboard. This is achieved through various commands. With two important exceptions, math commands, like many other LaTeX commands, begin with a backslash, \.

Subscripts and Superscripts

The two exceptions to the commands beginning with a backslash are ^ and _, the commands for superscript and subscript, respectively. Thus, x^2 is entered as `x^2` and x_2 as `x_2`. Note that ^ and _ only include the character immediately following, unless one encloses the subscript or superscript in brackets. Therefore, x^{12} is entered as `x^{12}`, not as `x^12`, which is x^12.

Math Symbols

If you think back to the last homework assignment you did or the last textbook you read, you'll probably remember that math uses all sorts of symbols such as ∞, $\sum$, $\int$, and a whole host of non-Roman letters. Appendix B contains a table of various math symbols. The commands work in essentially the same way as other LaTeX commands. Just remember that they only work in math mode.

Frequently, it seems like a lot of bother to go sifting through appendices looking for a command. Fortunately, many of the commands have fairly intuitive names, so if you're willing to risk errors, you can be lazy and guess. At this point, you might be thinking, "You said think about my last homework. I

didn't use half that stuff. I had Greek letters, hats, and transposes coming out of my ears!" We will get there in a few short paragraphs, so don't worry.

Delimiters

In a very basic sense, delimiters are things like parentheses and brackets. Parentheses and square brackets can be typeset with the matching key on the keyboard. Other delimiters have specific commands. Beyond the commands for different delimiters, it is important to know how to control the size of delimiters. Let's say you wanted to put parentheses around a summation for some reason.

$$(\sum_{k=0}^{10} k)$$

looks pretty silly because the parentheses are so small. We can use `\left` and `right` to correct this problem. When immediately followed by the approximate delimiter (i.e., one uses (with `\left`), LaTeX will size the delimiter to what it thinks is the right size. By writing

```
\left(\sum_{k=0}^{10}k\right)
```

we get

$$\left(\sum_{k=0}^{10} k\right)$$

which looks a lot better. It is important to note that the left and right hand delimiters must be paired. For example, attempting to compile `$\left($` will result in an error. It is possible to have a single left or right delimiter by pairing it with a `\left.` or `\right.` as appropriate. If you only want a delimiter one one side, you can use `\left` or `\right` without a delimiter to complete the pair. You also have the option of choosing the size of a delimiter with `\big`, `\Big`, `\bigg`, and `\Bigg`, which are used like `\left` and `\right` and give progressively larger delimiters. The existence of these commands may seem a little puzzling, given that LaTeX will size delimiters on its own, if you ask it to. However, there are times when you will want to do it yourself.

3.4 More Math Commands

Some of the major commands that take arguments are `\frac` and `\sqrt`, which typeset fractions and roots, respectively. One writes a root by `\sqrt{expr}`,

Table 3.1: Delimiters (those with commands)

Delimiter	Command
{	\{
}	\}
\	\backslash
⟨	\langle
⟩	\rangle
‖	\Vert or |
⌊	\lfloor
⌋	\rfloor
⌈	\lceil
⌉	\rceil

where `expr` is replaced by what goes under the root. If one wants the nth root, the full command is actually `\sqrt[n]{expr}`, so $\sqrt[3]{x}$ is written as `\sqrt[3]{x}`. Likewise, fractions take the form of `\frac{`*num*`}{`*denom*`}`. Combinations, such as $\binom{a}{b}$ are written as {*combinations*}`\choose{`*objects*`}` or `\binom{combo}{obj}`.

Log-like Operators

Occasionally, you will want to type things like `sin` and `det`, things that are both words and symbols. LaTeX provides commands for such operators to preclude any need to exit math mode and to allow for appealing spacing (shown in Table 3.2).

Accents

Another group of math commands are what one might describe as modifiers for letters. This would include bars and hats. It might not come as a surprise that the command for $\hat{x}$ is `\hat{x}`. There is also `\widehat`, which looks a bit better on capital letters. Compare $\hat{T}$ to $\widehat{T}$. This happens because `\widehat` extends over everything enclosed in the brackets, whereas `\hat` simply centers a hat over what's enclosed in the brackets. As a result, one can have $\widehat{xyz}$ (`\widehat{xyz}`) rather than $x\hat{y}z$ (`\hat{xyz}`). `\tilde` works the same way as `\hat`. There is also `\widetilde`. $\bar{x}$ is obtained by `\bar{x}`. It is worth noting that `\widebar` does not exist. Use `\overline` to accomplish a similar effect.

Table 3.2: Log-like operators

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

Table 3.3: Uppercase Greek letters

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

3.5 Greek and Other Fancy Letters

Greek Letters

For the Greek alphabet, the commands are simply a backslash followed by the name of the letter. For example, β is `\beta`. As always, commands are case sensitive, so `\Gamma` is Γ rather than γ.

Table 3.4: Lowercase Greek letters

Letter	Command	Letter	Command
α	`\alpha`	ν	`\nu`
β	`\beta`	ξ	`\xi`
γ	`\gamma`	π	`\pi`
δ	`\delta`	ρ	`\rho`
ϵ	`\epsilon`	σ	`\sigma`
ζ	`\zeta`	τ	`\tau`
η	`\eta`	υ	`\upsilon`
θ	`\theta`	ϕ	`\phi`
ι	`\iota`	χ	`\chi`
κ	`\kappa`	ψ	`\psi`
λ	`\lambda`	ω	`\omega`
μ	`\mu`		

Table 3.5: Math fonts

Font	Command
$\mathbb{NQRZ}$	`\mathbb{}`
text	`\textrm{}`
text	`\text{}`
math	`\mathrm{}`
bold	`\mathbf{}`
ABCXYZ	`\mathsf{}`
italics	`\mathit{}`
$\mathcal{ABCXYZ}$	`\mathcal{}`
$\mathfrak{ABCXYZ}$	`\mathfrak{}`

Fancy Fonts

The $\mathcal{AMS}$ packages provide several fonts that allow for some rather useful letters and symnbols that are not available through previously discussed commands. These fonts are selected by `\[font command]{text}`. This is most easily illustrated by an example such as `$\mathbb{R}$`. This command typesets a capital R in the math board bold font. Compiled, it gives $\mathbb{R}$, the familiar R for the reals.

Other fonts available from LaTeX and the $\mathcal{AMS}$ packages are shown in Table 3.5.

Table 3.6: Dots

Symbol	Command
…	`\ldots`
⋯	`\cdots`
⋮	`\vdots`
⋱	`\ddots`

The board bold font is the most commonly used one. Also of note are `\textrm{}` and `\mathrm{}`. Both of these fonts override the default italicization in math mode. `\textrm{}` typesets text in a style identical to normal text mode and `\mathrm` typesets text in normal upright letters, but slightly different for math. A more detailed discussion of these differences can be found in Oetiker's *The Not So Short Introduction to LaTeX*. Refer to Appendix B for examples of some of these fonts. If you would like to check on any of the other fonts, just experiment with a LaTeX compiler. Be aware that some of the fonts only work for capital letters.

Dots

An ellipsis (three dots) is often handy in math mode. LaTeX provides a wide variety of ellipses. They are shown in Table 3.6.

4.1 Matrices

LaTeX provides several different environments for making matrices and table-like structures. Common to all of these environments is the use of the ampersand, &, to delineate columns and the line break \\ to separate rows. Each row must be on its own line, though columns need not line up. Although it is not necessary to maintain any manner of alignment, maintaining any manner of alignment of the columsn between rows can be helpful, especially in larger tables.

Let us start by making a simple matrix using the `array` environment. Note that to use the `array` environment, one must be in math mode first. Suppose we want to type the following matrix:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Then we want an array with a 1 and a 0 in the first row, and a 0 and a 1 in the second row. Thus, the code is

```
\begin{array}{rr}
  1 & 0 \\
  0 & 1
\end{array}
```

However, this will not enclose the numbers in parentheses. As previously noted, the large left and right parentheses can be typeset using the commands `\left` and `\right`. Also, note that `{rr}` has been appended to `\begin{array}`. This indicates the vertical alignment of the columns. The first `r` indicates the first column should be right-aligned, while the second `r` indicates the same for the second column. Text in the columns can be aligned left, right, or center using `l`, `r`, and `c`, respectively. Thus

```
\left(
\begin{array}{rr}
  1 & 0 \\
  0 & 1
\end{array}
\right)
```

gives the completed matrix.

The $\mathcal{AMS}$ packages provide various environments for matrices beyond `array`. `matrix` is essentially the same as `array`, as it creates a matrix without delimiters. However, `matrix` automatically centers the entries within the columns. `pmatrix` and `vmatrix` are two other matrix environments, and they come with delimiters. Consider our previous example matrix written using `pmatrix` and `vmatrix`, respectively:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

Additionally, `matrix` and similar environments can only handle matrices of up to ten columns. Should you need a matrix with more columns, precede the start of the matrix environment with

`\setcounter{MaxMatrixCols}{numcolumns}`.

4.2 Commutative Diagrams

While extensive graphics work in LaTeX becomes something of a computer geek black art, commutative diagrams can be drawn easily using the `\xymatrix{}` command. This command is part of the `xy` package by putting

```
\usepackage[all]{xy}
```

in the preamble. The `\xymatrix{}` command is used inside math mode, and behaves much like the `array` environment. For example,

```
\[
\xymatrix{G & H \\
J & K}
\]
```

will give

$$\begin{array}{cc} G & H \\ \\ J & K \end{array}$$

These can all be connected with arrows using the `\ar[argument]` command. The argument of the command indicates the direction of the arrow: `l`, `r`, `u`, and

d for left, right, up, and down, respectively. These directions can be combined, repeatedly if necessary. For example, `\ar[drrr]` points down three columns to the right. The source code for a basic commutative diagram would look like this:

```
\[
\xymatrix{G \ar[r] \ar[d] & H \ar[d] \\
J \ar[r] & K}
\]
```

This gives the diagram

$$\begin{array}{ccc} G & \longrightarrow & H \\ \downarrow & & \downarrow \\ J & \longrightarrow & K \end{array}$$

Furthermore, these arrows can be labeled using subscripts and superscripts. For vertical arrows, subscripts go to the left, and superscripts go to the right.

```
\[
\xymatrix{G \ar[r]_f \ar[d]^\tau & H \ar[d]_\sigma \\
J \ar[r]_g & K}
\]
```

$$\begin{array}{ccc} G & \xrightarrow{f} & H \\ \downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \sigma} \\ J & \xrightarrow{g} & K \end{array}$$

4.3 Piecewise Functions

The typesetting of piecewise functions is accomplished by using a built-in environment from the `amsmath` environment. Piecewise functions should be typeset almost exclusively in displayed math environments. It is possible to place a piecewise function in an inline math environment, but the large size of a piecewise function will make a mess of your page.

The `amsmath` package includes a `cases` environment for typesetting piecewise functions and other related functions. It works just like the `matrix` environment (also provided by `amsmath`), but only two columns are permitted. The left brace is also automatically printed before the array. Here is an example:

```
\[
\delta(x) = \begin{cases}
  0 & \mbox{if } x \neq 0 \\
  \infty & \mbox{if } x = 0
\end{cases}
\]
```

$$\delta(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ \infty & \text{if } x = 0 \end{cases}$$

4.4 Better Looking Math

There are several commands that are very useful for improving the appearance of your mathematical content.

Eqnarray

Warning: This environment is deprecated and has been supplanted by the `align` *environment provided by* `amsmath`*. It is only mentioned here for historical reasons.*

The `eqnarray` environment is similar to the `array` or `tabular` environment, but is at most three columns wide, and no alignment needs to be declared for the columns. Also, `eqnarray` is a math environment, meaning no dollar signs or other math mode environments are necessary. The environment comes in two flavors: `eqnarray` will number the rows, while `eqnarray*` will not. The environment is entered and exited like any other with `\begin{eqnarray*}` and `\end{eqnarray*}`. Here is an example:

```
\begin{eqnarray}
  \label{eq:2}
  3x+2&=&5 \\
  3x&=&3 \\
  x&=&1
\end{eqnarray}
```

which results in:

$$\begin{aligned} 3x+2 &= 5 && (4.1) \\ 3x &= 3 && (4.2) \\ x &= 1 && (4.3) \end{aligned}$$

Align

Note: This environment is the recommended way to typeset aligned equations.

The `align` environment, like `eqnarray`, is similar to the `array` or `array` environment, but unlike the `eqnarray` environment, this environment is provided by the `amsmath` package and offers more features than the provided `eqnarray` environment. Here is an example of the basic syntax of the `align` environment:

```
\begin{align}
  \label{eq:3}
  3x+2&=5 \\
  3x&=3 \\
  x&=1
\end{align}
```

which ends up as

$$3x + 2 = 5 \tag{4.4}$$
$$3x = 3 \tag{4.5}$$
$$x = 1 \tag{4.6}$$

Displaystyle

The `\displaystyle` command is the best friend of inline math. For the most part, `\displaystyle` will make things bigger. It will make inline fractions more readable, and for inline integrals, and general sums and products, it will place the indices above and below, rather than to the side of the operator. The use of the `\displaystyle` is best described with a few examples. In the following example, the compiled code will immediately follow the source.

```
Some inline math: $\frac{\sqrt{2}}{\sqrt{3}}$.
```

Some inline math: $\frac{\sqrt{2}}{\sqrt{3}}$.

```
Now with \texttt{displaystyle}:
$\displaystyle\frac{\sqrt{2}}{\sqrt{3}}$.
```

Now with `displaystyle`: $\displaystyle\frac{\sqrt{2}}{\sqrt{3}}$.

```
A bad looking sum: $\sum_{n=1}^\infty \frac{1}{n}$.
```

A bad looking sum: $\sum_{n=1}^{\infty} \frac{1}{n}$.

```
\texttt{displaystyle} to the rescue!
$\displaystyle \sum_{n=1}^\infty \frac{1}{n}$.
```

`displaystyle` to the rescue! $\displaystyle \sum_{n=1}^{\infty} \frac{1}{n}$.

Note that `\displaystyle` need only be used once in each math environment.

4.5 Theorems and Proofs

Theorems

Back in the dark ages of TEX[1], the command to begin a theorem was called `proclaim`. However, LATEX isn't as overjoyed about its theorems, so `proclaim` is gone and replaced by the `theorem` environment. If you open a math book, you'll notice that the theorems and theorem-like objects are all labeled differently. Some are called theorems, others lemmas and still others propositions, among other things. In the preamble of your document, you need to set up each of these different names separately using the `\newtheorem` command.

First, let's look at a theorem that just has a number, rather than a specific name:

Theorem 1. *Let V be a finite-dimensional vector space, and define $\psi : V \to V^{**}$ by $\psi(x) = \hat{x}$. Then ψ is an isomorphism.*

The first thing we had to do to typeset this fine theorem was define an environment for it via `newtheorem`. The command has two arguments, taking the form `\newtheorem{`*name*`}{`*Name*`}`. The first argument names the environment, and the second argument determines what things will be labeled. (The two arguments do not have to be the same, but it would be a bit weird for the `chicken` environment to be producing theorems.) In our example, the full command looked like `\newtheorem{theorem}{Theorem}`. The remainder of the code was

[1] One should not advertise that one actually used TEX, especially if LATEX was written before one was born, as one will get strange looks.

```
\begin{theorem}
  Let $V$ be a finite-dimensional vector space, and
  define $\psi:V\rightarrow V^{**}$ by $\psi(x)= %
  \hat{x}$. Then $\psi$ is an isomorphism.
\end{theorem}
```

But what happens if your theorem has a name?

Theorem 2 (Fermat's Last Theorem)**.** *If an integer* n *is greater than 2, then the equation* $a^n + b^n = c^n$ *has no solutions in non-zero integers* a, b, *and* c.

The only difference between this and the previous example is the addition of the name. We didn't have to add another `\newtheorem` because we had already defined a theorem environment, which is also why this is numbered as Theorem 2. So how did we add the name? When we opened the `theorem` environment, we did it with

```
\begin{theorem}[Fermat's Last Theorem]
```

This is all well and good, but what if you have a theorem that is so fantastically cool giving it a number would be cheapening the moment?

Morse Lemma. *Let* p_0 *be a non-degenerate critical point of a function* f *of two variables. Then we can choose appropriate local coordinates* (X, Y) *in such a way that the function* f *expressed with respect to* (X, Y) *takes one of the following three standard forms:*

1. $f = X^2 + Y^2 + c$
2. $f = X^2 - Y^2 + c$
3. $f = -X^2 - Y^2 + c$

If you recall, the way we got rid of numbering on other things was to add a *. The question is, do you add it to `\newtheorem` or the start of the environment? From experience, it seems as if adding it to `\newtheorem` would ensure that every instance of that environment will be unnumbered, and adding it to the start of the environment would just eliminate the number from that one instance. As it happens, the answer is `\newtheorem`, sort of. If you just tack on the asterisk, you'll get an error as `\newtheorem*` is not defined in LaTeX. However, it does exist with the inclusion of the `amsthm` package. To typeset the Morse Lemma, we declared the environment with `\newtheorem*{morse}{Morse Lemma}`.

Each different proclamation environment has its own counter by default, so you could have a Theorem 1, a Definition 1, a Lemma 1, and who knows what else numbered 1. But what if you wanted your theorems and lemmas numbered in succession so Theorem 1 was followed by Lemma 2 rather than Lemma 1? Covering the entire scope of counters would be unmanageable at this juncture, so we'll jsut cover the very basics and leave the rest for another time and place (quite possibly another book). As it turns out, achieving this end leads us back to the `\newtheorem` command. So how do you declare the lemma environment?

```
\newtheorem{lemma}[theorem]{Lemma}
```

There's one potential pitfall with this system. What happens if `theorem` has not been defined yet? Bad, bad things happen, so make sure you tie the counting to something that already exists.

There is another counting option that ties numbering to some subdivision of the document, creating Theorem 1.1, etc. This option is of the form

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

where *section* can be any subdivision of the document such as `section`, `chapter`, or `subsection`. Using this numbering scheme and tying the numbering to the subsection, we can create:

Theorem 4.5.0.1. *Every bounded, monotonic sequence is convergent.*

The `amsthm` package provides the `\newtheoremstyle{`*style*`}` command which lets you define what the theorem is all about by picking from three predefined styles: `definition` (fat title, roman body), `plain` (fat title, italic body), or `remark` (italic title, roman body).[2]

The Proof Environment

In addition to allowing for unnumbered theorems, the `amsthm` package provides the `proof` environment. Using the environment is fairly straightforward; just follow the `\begin{proof}` with what you want the proof labeled in square brackets. For example, if youw ant to include the proof of the Morse Lemma, your code would look something like:

[2] Oetiker, Tobias, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LaTeX 2ε*. May 2006. p. 60

```
\begin{proof}[Proof of Morse Lemma]
  the proof goes here
\end{proof}
```

The `proof` environment automatically places the QED symbol (□) at the end, leaving an appropriate amount of space in front. However, should it place the symbol in the wrong place, you can force the placement with `\qedhere`.

The following is a collection of LaTeX examples. First, the source code is displayed, immediately followed by the compiled code. Try not to pay attention to the mathematical content as it is culled from old homework sets and may or may not be correct.

5.1 Radicals and Fractions

This is an example of usages of radicals and fractions. The `amsmath` and `amssymb` packages are necessary.

```
\paragraph*{4.4}
Show that $1,\sqrt{2},\sqrt{3},\sqrt{6}$ are
linearly independent over $\mathbb{Q}$.
\subparagraph*{}
Suppose $1,\sqrt{2},\sqrt{3},\sqrt{6}$ are linearly
dependent over $\mathbb{Q}$. Then there exist
$p,q,r,s\in\mathbb{Q}$ such that
$p+q\sqrt{2}+r\sqrt{3}+s\sqrt{6}=0$ and
$p,q,r,s$ are not all zero. Then either $r$ or $s$
is nonzero, otherwise, $p=0$, or
$q\sqrt{2}\in\mathbb{Q}$, neither of which can be
the case if $p,q,r,s$ are not all zero. Then we have
\begin{displaymath}
  \sqrt{3}=\frac{-p-q\sqrt{2}}{r+s\sqrt{2}}
\end{displaymath}
and this is well defined because at least one of
$r$ and $s$ is not zero. Thus, there exist rational
numbers $e,f$ such that $\sqrt{3}=e+f\sqrt{2}$.
Squaring this, we have
\begin{align*}
  3 &= e^2+ef\sqrt{2}+2f^2 \\
  \frac{3-e^2-2f^2}{ef} &= \sqrt{2}.
\end{align*}
Which implies that $\sqrt{2}$ is rational.
Thus, we have a contradiction. Therefore, we can
```

```
conclude that $1,\sqrt{2},\sqrt{3},\sqrt{6}$ are
linearly independent over $\mathbb{Q}$.
```

4.4 Show that $1, \sqrt{2}, \sqrt{3}, \sqrt{6}$ are linearly independent over $\mathbb{Q}$.

Suppose $1, \sqrt{2}, \sqrt{3}, \sqrt{6}$ are linearly dependent over $\mathbb{Q}$. Then there exist $p, q, r, s \in \mathbb{Q}$ such that $p + q\sqrt{2} + r\sqrt{3} + s\sqrt{6} = 0$ and p, q, r, s are not all zero. Then either r or s is nonzero, otherwise, $p = 0$, or $q\sqrt{2} \in \mathbb{Q}$, neither of which can be the case if p, q, r, s are not all zero. Then we have

$$\sqrt{3} = \frac{-p - q\sqrt{2}}{r + s\sqrt{2}}$$

and this is well defined because at least one of r and s is not zero. Thus, there exist rational numbers e, f such that $\sqrt{3} = e + f\sqrt{2}$. Squaring this, we have

$$\begin{aligned} 3 &= e^2 + ef\sqrt{2} + 2f^2 \\ \frac{3 - e^2 - 2f^2}{ef} &= \sqrt{2}. \end{aligned}$$

which implies that $\sqrt{2}$ is rational. Thus, we have a contradiction. Therefore, we can conclude that $1, \sqrt{2}, \sqrt{3}, \sqrt{6}$ are linearly independent over $\mathbb{Q}$.

Note that inline fractions can be made larger by using `\displaystyle`.

5.2 Aligned Equations

The following is an example of the `\align*` environment. The `amsmath` package is necessary.

```
Conjugating the elements of $\mathbb{V}$ by the
transposition of $\mathbb{S}_4$, we have
\begin{align*}
  (12)(12)(34)(12) &= (12)(34) \\
  (12)(13)(24)(12) &= (14)(23) \\
  (12)(14)(23)(12) &= (13)(24) \\
  (13)(12)(34)(13) &= (14)(23) \\
  (13)(13)(24)(13) &= (13)(24) \\
  (13)(14)(23)(13) &= (12)(34) \\
  (14)(12)(34)(14) &= (13)(24) \\
```

```
  (14)(13)(24)(14) &= (12)(34) \\
  (14)(14)(23)(14) &= (14)(23) \\
  (23)(12)(34)(23) &= (13)(24) \\
  (23)(13)(24)(23) &= (12)(34) \\
  (23)(14)(23)(23) &= (14)(23) \\
  (24)(12)(34)(24) &= (14)(23) \\
  (24)(13)(24)(24) &= (13)(24) \\
  (24)(14)(23)(24) &= (12)(34) \\
  (34)(12)(34)(34) &= (12)(34) \\
  (34)(13)(24)(34) &= (14)(23) \\
  (34)(14)(23)(34) &= (13)(24)
\end{align*}
thus, $\mathbb{V}$ is closed under conjugation by
transpositions in $\mathbb{S}_4$ as $\mathbb{S}_4$
is generated by all transpositions on 4 letters,
we can conclude that $\mathbb{V} \lhd \mathbb{S}_4$.
```

Conjugating the elements of $\mathbb{V}$ by the transposition of $\mathbb{S}_4$, we have

$$
\begin{aligned}
(12)(12)(34)(12) &= (12)(34)\\
(12)(13)(24)(12) &= (14)(23)\\
(12)(14)(23)(12) &= (13)(24)\\
(13)(12)(34)(13) &= (14)(23)\\
(13)(13)(24)(13) &= (13)(24)\\
(13)(14)(23)(13) &= (12)(34)\\
(14)(12)(34)(14) &= (13)(24)\\
(14)(13)(24)(14) &= (12)(34)\\
(14)(14)(23)(14) &= (14)(23)\\
(23)(12)(34)(23) &= (13)(24)\\
(23)(13)(24)(23) &= (12)(34)\\
(23)(14)(23)(23) &= (14)(23)\\
(24)(12)(34)(24) &= (14)(23)\\
(24)(13)(24)(24) &= (13)(24)\\
(24)(14)(23)(24) &= (12)(34)\\
(34)(12)(34)(34) &= (12)(34)\\
(34)(13)(24)(34) &= (14)(23)\\
(34)(14)(23)(34) &= (13)(24)
\end{aligned}
$$

thus, $\mathbb{V}$ is closed under conjugation by transpositions in $\mathbb{S}_4$ as $\mathbb{S}_4$ is generated by all transpositions on 4 letters, we can conclude that $\mathbb{V} \lhd \mathbb{S}_4$.

5.3 Piecewise Functions

The following is an example of a piecewise function.

```
\paragraph*{3}
Let $m$ be a cardinal number such that $m+\aleph_0=C$.
Without the Axiom of Choice, show that $m=C$. Since
$m+\aleph_0=C$, we have $A\cup \omega \sim \lbrace
0,1 \rbrace^\omega$ where $A$ and $\omega$ are
disjoint. Let $f:\lbrace 0,1\rbrace^\omega \to A \cup
\omega$ be an equivalence. Consider the families of
```

```
functions $F_n$ and $G_n$ in $\lbrace 0,1\rbrace^
\omega$, for all $n\in \omega$ defined as follows:

\[
F_n(x) =
\begin{cases}
  0, & x \neq n; \\
  1, & x=n;
\end{cases}
\]
\[
G_n(x) =
\begin{cases}
  0, & x=n; \\
  1, & x \neq n;
\end{cases}
\]

Define a one-to-one and onto function
$g:\lbrace 0,1\rbrace^\omega \to \lbrace 0,1 %
\rbrace^\omega$ such that $f\circ g(F_n)\in
\omega$ for all $n\in \omega$ and $f\circ g(G_n)
\in A$ for all $n\in \omega$. Since $g$ is
one-to-one and onto, $f\circ g$ is an equivalence
from $\lbrace 0,1\rbrace^\omega$ to $A \cup \omega$.
Furthermore, since the set of all $F_n\in %
\lbrace 0,1\rbrace^\omega$ and the set of all
$G_n\in \lbrace 0,1\rbrace^\omega$ are both clearly
equivalent to $\omega$, we can conclude that there
is a subset $B$ of $A$ that is equivalent to $\omega$.
Thus, $A=(A-B) \cup B$. Hence, there is a cardinal
number $n$ such that $m=n+\aleph_0$. Therefore,
$m+\aleph_0=n+\aleph_0+\aleph_0=n+\aleph_0=m$. Thus,
$m=C$.
```

3 Let $\mathfrak{m}$ be a cardinal number such that $\mathfrak{m} + \aleph_0 = C$. Without the Axiom of Choice, show that $\mathfrak{m} = C$. Since $\mathfrak{m} + \aleph_0 = C$, we have $A \cup \omega \sim \{0,1\}^\omega$ where A and ω are disjoint. Let $f : \{0,1\}^\omega \to A \cup \omega$ be an equivalence. Consider the

families of functions F_n and G_n in $\{0,1\}^\omega$, for all $n \in \omega$ defined as follows:

$$F_n(x) = \begin{cases} 0, & x \neq n; \\ 1, & x = n; \end{cases}$$

$$G_n(x) = \begin{cases} 0, & x = n; \\ 1, & x \neq n; \end{cases}$$

Define a one-to-one and onto function $g : \{0,1\}^\omega \to \{0,1\}^\omega$ such that $f \circ g(F_n) \in \omega$ for all $n \in \omega$ and $f \circ g(G_n) \in A$ for all $n \in \omega$. Since g is one-to-one and onto, $f \circ g$ is an equivalence from $\{0,1\}^\omega$ to $A \cup \omega$. Furthermore, since the set of all $F_n \in \{0,1\}^\omega$ and the set of all $G_n \in \{0,1\}^\omega$ are both clearly equivalent to ω, we can conclude that there is a subset B of A that is equivalent to ω. Thus, $A = (A - B) \cup B$. Hence, there is a cardinal number n such that $m = n + \aleph_0$. Therefore, $m + \aleph_0 = n + \aleph_0 + \aleph_0 = n + \aleph_0 = m$. Thus, $m = C$.

5.4 XY

The following is an example of drawing commutative diagrams with the xy package. For the following code to function, the package xy must be included in the preamble of the document (`\usepackage[all]{xy}`).

```
Suppose $\phi:A\to B$ and $\psi:C\to D$ are
isomorphic field extensions. Then the following
diagram commutes.
\begin{displaymath}
\xymatrix{
A \ar[r]^\phi \ar[d]_\alpha & B \ar[d]_\beta \\
C \ar[r]^\psi & D}
\end{displaymath}
Thus, $\beta\circ\phi=\psi\circ\alpha$. Therefore
the diagram
\begin{displaymath}
\xymatrix{
C \ar[r]^\psi \ar[d]_\alpha^{-1} & D %
\ar[d]_\beta^{-1} \\
A \ar[r]^\phi & B}
\end{displaymath}
```

```
also commutes. Thus, $\psi$ is isomorphic to $\phi$.
Therefore, $\phi~\psi$ implies $\psi~\phi$.
\subparagraph*{}
Suppose $\phi:A\to B$ and $\psi:C\to D$ are
isomorphic field extensions. Furthermore, let
$\psi:C\to D$ and $\mu:E\to F$ be isomorphic field
extensions. The following diagrams commute.
\begin{displaymath}
\xymatrix{
A \ar[r]^\phi \ar[d]_\alpha & B \ar[d]_\beta \\
C \ar[r]^\psi & D}
\xymatrix{
C \ar[r]^\psi \ar[d]_\delta & D \ar[d]_\varepsilon \\
E \ar[r]^\mu & F}
\end{displaymath}
```

Suppose $\phi : A \to B$ and $\psi : C \to D$ are isomorphic field extensions. Then the following diagram commutes.

$$\begin{array}{ccc} A & \xrightarrow{\phi} & B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ C & \xrightarrow{\psi} & D \end{array}$$

Thus, $\beta \circ \phi = \psi \circ \alpha$. Therefore the diagram

$$\begin{array}{ccc} C & \xrightarrow{\psi} & D \\ {\scriptstyle\alpha}\downarrow{\scriptstyle -1} & & {\scriptstyle\beta}\downarrow{\scriptstyle -1} \\ A & \xrightarrow{\phi} & B \end{array}$$

also commutes. Thus, ψ is isomorphic to ϕ. Therefore, $\phi\ \psi$ implies $\psi\ \phi$.

Suppose $\phi : A \to B$ and $\psi : C \to D$ are isomorphic field extensions. Furthermore, let $\psi : C \to D$ and $\mu : E \to F$ be isomorphic field extensions. The following diagrams commute.

$$\begin{array}{ccc} A & \xrightarrow{\phi} & B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ C & \xrightarrow{\psi} & D \end{array} \qquad \begin{array}{ccc} C & \xrightarrow{\psi} & D \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\varepsilon} \\ E & \xrightarrow{\mu} & F \end{array}$$

Matrices are really wonderful if you want to make, well, matrices. But what if you wanted to put words in your matrix? Well, if you really wanted, you could probably use `mathrm` and make a giant ugly mess. But in reality, your scary mutant matrix would be a table. To avoid this untimely end to your sanity, the `tabular` environment exists. Much like matrices, ampersands are used to separate entries in a row and double backslashes are used to end lines. Additionally, one can control the justification of each of the individual columns and add vertical lines between the columns. In order to avoid continuing to describe tables in hazy terms, we are just going to cut to the chase and have an example.

```
\begin{tabular}{|c|c|}
  a & b \\
  c & d \\
\end{tabular}
```

makes the following table:

a	b
c	d

Now that was cool, but you've probably noticed that this example table has no horizontal lines, and as a result, is rather ugly. That's where `\hline` comes in. `\hline` draws a horizontal line. So if you want horizontal lines in the table above, you'd change things to:

```
\begin{tabular}{|c|c|}
  \hline
  a & b \\
  \hline
  c & d \\
  \hline
\end{tabular}
```

to get:

a	b
c	d

Note that you need a line break before the last `\hline` (if you didn't use `\hline`, you wouldn't need a line break after the last set of entries). So now that we have a pretty table, how did we get there?

6.1 Anatomy of the `tabular` Environment

To start, we see that things are laid out similarly to a matrix, with each cell in a row separated by an ampersand and rows being ended with line breaks. Notice that we put some arguments after the `\begin{tabular}`. The vertical bars, `|`, draw the lines down each column. The `c`'s center the contents of each cell. The other options are `l` and `r` for left and right justification, respectively. But what if you don't want vertical lines? You simply leave out the bars and continue as usual. By default, entries are centered vertically within the cell, but, not surprisingly, you have the option of changing this to either the top, `t`, or bottom, `b`. This option, should you desire it, goes in square brackets immediately following the `\begin{tabular}`. So, if we wanted a table with the entries at the bottom of the cells, we'd have:

```
\begin{tabular}[b]{|c|c|}
  \hline
  a & b \\
  \hline
  c & d \\
  \hline
\end{tabular}
```

which produces:

a	b
c	d

Fiddling with Columns

LaTeX automatically decides how wide to make each column, but, as with many things in LaTeX, you can override it, should you need to do so. Likewise, you can make a single cell span more than one column. To set the width of a column manually, you change the alignment. Instead of `l`, `r`, or `c`, you use `p{}` and put the width of the column inside the curly braces. `\multicolumn` and `\cline` are two other commands that override the layout of a table. `\multicolumn`

Table 6.1: Career statistics: Brant Brown

Year	Tm	G	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	BA	OBP	SLG
1996	CHC	29	69	11	21	1	0	5	9	3	3	2	17	.304	.329	.536
1997	CHC	46	137	15	32	7	1	5	15	2	1	7	28	.234	.286	.409
1998	CHC	124	347	56	101	17	7	14	48	4	5	30	95	.291	.348	.501
1999	PIT	130	341	49	79	20	3	16	58	3	4	22	114	.232	.238	.449
2000	FLA	41	73	4	14	6	0	2	6	1	0	3	33	.192	.224	.356
	CHC	54	89	7	14	1	0	3	10	2	1	10	29	.157	.248	.270
	TOT	95	162	11	28	7	0	5	16	3	1	13	62	.173	.237	.309
5 Seasons		424	1056	142	261	52	11	45	146	15	14	74	316	.247	.301	.445

lets you have a cell span that is more than one column. It takes three arguments: the number of columns spanned, the alignment within the cell, and the actual contents. `\cline{a-b}` is a variation on `\hline` which draws a horizontal line from column `a` to column `b`. There are other far more complicated things that can be done with table alignment, but you'll have to look in a bigger, far more complicated, book for that.

6.2 The `table` Environment and Captions

It would look awfully silly if tables were split in two at page breaks. As you might imagine, tables don't do that. If a table falls on a page break, LaTeX will move it entirely onto the next page. But what if you want to caption your table? You'd want the caption to ride along with the table over the page breaks, wouldn't you? Of course, you could always align the caption yourself, but why do extra work when you could use the `table` environment? `table` doesn't replace `tabular`; rather, it encloses `tabular`. Within the `table` environment, you can use `\caption` to caption your table (either above or below the `tabular` environment). Additionally, the `\begin{table}` command takes an optional argument that controls the placement of the table. The options are `h` to place the table where it appears in the source, `t` to place the table at the top of a page, `b` to place the table at the bottom of a page, and `p` to place the page on its own page of floats.

6.3 Example

Up to this point, we've had fairly silly, uninteresting examples. This section is just going to be one big example. To render the table exactly as typed, the `booktabs` and `graphicx` packages must be loaded.

```
\begin{table}
  \centering
  \caption{Career Statistics: Brant Brown}
  \label{tab:brant-brown}
  \resizebox{\textwidth}{!}{\begin{tabular}{@{}rrrr %
        rrrrrrrrrrrrr@{}}
    \toprule
    Year & Tm & G & AB & R & H & 2B & 3B & HR & RBI &
    SB & CS & BB & SO & BA & OBP & SLG \\
    \midrule
    1996 & CHC & 29 & 69 & 11 & 21 & 1 & 0 & 5 & 9 &
    3 & 3 & 2 & 17 & .304 & .329 & .536 \\
    1997 & CHC & 46 & 137 & 15 & 32 & 7 & 1 & 5 & 15
    & 2 & 1 & 7 & 28 & .234 & .286 & .409 \\
    1998 & CHC & 124 & 347 & 56 & 101 & 17 & 7 & 14 &
    48 & 4 & 5 & 30 & 95 & .291 & .348 & .501 \\
    1999 & PIT & 130 & 341 & 49 & 79 & 20 & 3 & 16 &
    58 & 3 & 4 & 22 & 114 & .232 & .238 & .449 \\
    2000 & FLA & 41 & 73 & 4 & 14 & 6 & 0 & 2 & 6 & 1
    & 0 & 3 & 33 & .192 & .224 & .356 \\
    \multicolumn{2}{r}{CHC} & 54 & 89 & 7 & 14 & 1 &
    0 & 3 & 10 & 2 & 1 & 10 & 29 & .157 & .248 & .270
    \\
    \multicolumn{2}{r}{TOT} & 95 & 162 & 11 & 28 & 7
    & 0 & 5 & 16 & 3 & 1 & 13 & 62 & .173 & .237 &
    .309 \\
    \addlinespace
    \multicolumn{2}{l}{5 Seasons} & 424 & 1056 & 142
    & 261 & 52 & 11 & 45 & 146 & 15 & 14 & 74 & 316 &
    .247 & .301 & .445 \\
    \bottomrule
  \end{tabular}}
\end{table}
```

To this point, we have been talking pretty much about how to get LaTeX to make text look the way you want, and we've always included the line `\documentclass{article}`. This is all well and good, and we get what we want, but you might be wondering: why only `article`? Does LaTeX do other things? And what's an article anyway?

LaTeX has several built-in document classes: `article`, `report`, `book`, `slides`, and `letter`, each of which are used pretty much for precisely what the name implies. `article` is also used, as you have learned, in more general situations as the generic document class. These are not on the only document classes in existence, as individuals and organizations can write their own, but they are the standard ones.

7.1 Standard LaTeX Document Classes

Article

`article` is used for, as the name implies, journal articles. It is the most basic LaTeX document class, as it has few special commands and quirks. (This may be less because `article` is somehow inherently superior and more because it is the document class everyone thinks of first. But maybe that makes it inherently superior.) The title and associated information is centered by the `\maketitle` command. If one wants to include an abstract, this can be done with the `abstract` environment.

Report

`report` is very similar to `article`. However, it is designed to be used for longer documents, and as such, has two additional sectioning commands: `\chapter` and `\part`. Additionally, as one might expect, the title information appears on a title page rather than simply at the top of the first page. The abstract gets its own page as well.

Slides

`slides`, not very surprisingly, makes slides. The slides are meant to be printed and projected, rather than projected from the computer as you might with a

PowerPoint. This isn't to say you can't do it, but people have written other document classes to achieve this end. Additionally, `slides` documents are a bit trickier in their construction than other types of documents. In general, it is best to stay away from `slides` unless you actually intend to print them and project them on an overhead. If you are interested in creating computer presentations for use on screen projectors, use Beamer, an add-on to LaTeX that provides just such a functionality. Continuing with `slides`, we're going to look at the general format first and then make some more general comments.

Format of a `slides` Document

```
\documentclass{slides}
preamble
\title{title}
\author{author}
\date{date}

\begin{document}
\maketitle

\begin{slide}
  slide content
\end{slide}

\end{document}
```

The things that one should take from this skeleton of an example is that the document class is called `slides`, not `slide` (this is a seemingly small detail, but when your documents won't compile because you forgot what the document class was called, that's going to become an important point); the title information does not go within a slide, it goes before the slides start; lastly, the contents of each slide are set off within a `slide` environment.

There's some information you probably didn't glean from the little skeleton (good job if you figured this out, because the information is not in the example to be observed). First, a fact that you may want to know or just care about: `slides` uses a different default font than the other LaTeX document classes. Additionally, it is your responsibility to make sure that what you want to go on one slide fits on one slide. If the stuff between the `\begin{slide}`

and `\end{slide}` is going to spill over onto another page, LaTeX lets it and simply doesn't number the overflow slide(s).

`slides` documents can have two other environments besides `slide`: `overlay` and `note`. `overlay` is used for making a slide designed to be placed over the slide environment preceding it. As a result, overlays aren't numbered in the same way as slides; the first overlay corresponding to the fourth slide would be numbered 4-a. In order to get the text on the overlay to line up correctly, it is best to include the text from the slide with the color set to white. This can be accomplished with `\textcolor{white}{text}`.

The `note` environment allows one to make notes corresponding to each slide. Each note is numbered according to its corresponding slide, so the first note corresponding to the fourth slide would be numbered 4-1.

Letters

The `letter` document class has several commands specific to it. It bears some similarity to the `slides` class because you can have multiple letters in one document. The return address is controlled by the `\address` command in the preamble. Each line of the return address (and the other command arguments in letters, if necessary) is separated by a `\\`. The start of the `letter` environment is structured as

```
\begin{letter}{addressee}
```

where *addressee* is both the name and address of the addressee. Next comes the `\opening{opening}` argument, which has as its argument the opening of the letter. The letter concludes with the `\closing{}` and `\signature{}` commands where `\closing` takes the closing of the letter and `\signature` the name and/or title appearing under the signature as their arguments. The `\cc` and `\encl` commands are optional and take the names of those receiving copies and the list of enclosures.

7.2 Nonstandard Document Classes

There are many nonstandard document classes out there, but two of the most popular ones are the KOMA-Script document class bundle and the `memoir` document class. They are meant to serve as replacements to the default LaTeX document classes, offering more features, and in both cases, typographically better documents. Besides those two, the $\mathcal{AMS}$ provides its own document classes for

articles, books, and monographs that are submitted to various mathematical journals.

KOMA-Script[1]

The KOMA-Script bundle (written by Markus Kohm) provides drop-in replacements for the `article/report/book` classes with emphasis on typography and versatility. There is also a letter class, different from all other letter classes. It also offers a package for calculated type areas in the way laid down by the typographer Jan Tschichold, a package for easily changing and defining of page styles, a package for getting not only the current date, but also the name of the day, and a package for getting the current time. All these packages may be used not only with KOMA-Script classes, but also with standard classes.

The equivalents to the `article`, `report`, `book`, and `letter` classes are `scrartcl`, `scrrprt`, `scrbook`, and `scrlttr2`, respectively. By default, the paper size for all four classes is set to A4 paper, so you have to manually change it to letter paper by using

```
\documentclass[letterpaper]{foo}
```

where `foo` is the name of the KOMA-Script document class you want to use. Furthermore, the default font is set to 11pt. Please see the KOMA-Script documentation for more details on the classes.

Memoir[2]

The `memoir` class is a flexible class for typesetting poetry, fiction, non-fiction, and mathematical works as books, reports, articles, or manuscripts.[3] It is meant to replace the standard LaTeX document classes, particularly the `report` and `book` document classes. Documents can use 9pt, 10pt, 11pt, 12pt, 14pt, or 17pt as the normal font size. Many methods are provided to let you create your particular design. The class incorporates over thirty of the more popular packages. It was written by Peter Wilson. See the documentation for more information on the wealth of customizations for the document class.

---

[1] http://www.ctan.org/tex-archive/help/Catalogue/entries/koma-script.html

[2] Taken from the README file in the CTAN directory (http://www.ctan.org/tex-archive/macros/latex/contrib/memoir/)

[3] This reader was typeset using the `memoir` class.

8.1 Package Overview

LaTeX itself is a large package of macros written in the TeX language. Accordingly, LaTeX has been designed to facilitate the creation and usage of user-defined packages to extend the functionality of the LaTeX system. The large LaTeX distributions, such as MiKTeX, ship with an enormous library of packages and feature some kind of package managemetn software. This makes it easy to download, install, and manage packages on a LaTeX installation. This chapter will summarize this and mention some of the most common LaTeX packages. Much more descriptive lists of packages are located in Appendix C.

8.2 Package Management

The base of a LaTeX installation (or any other flavor of TeX) is a directory called `\texmf` by default. Common locations for this directory include `/usr/share/texmf` on Mac OS X, Unix, and Unix-like operating systems, and `c:\texmf` on Windows operating systems. This directory will contain a collection of fonts, packages, scripts, source code, etc. The precise details will vary according to the distribution. TeX was designed for deployment on large UNIX systems, and the `texmf` tree is intended to be maintained by the system administrator and is generally off-limits to users. In the days before TeX distributions became common, an expert would have to manually compile and configure the installation for use in a network environment. This remains the case in many multi-user networks. For personalized needs, every user has a `localtexmf` directory in the home directory. The contents of the `localtexmf` tree override those of the `texmf` tree. This allows an individual user to load packages or fonts that are not in the main `texmf` tree without disturbing the delicate server-side installation and breaking TeX for everyone on the network.

However, the distinction between the main and local TeX trees is less clear for single-user operating systems or those with weak multi-user support (i.e., Windows and Mac OS 9 and earlier). In these cases, the users and administrator are the same person.

The details of managing packages is different depending on the distribution. It is best to refer to the documentation that comes with the LaTeX distribution to see exactly how it is done. The process is fairly straightforward in MiKTeX

as it features a graphical package management utility to automate the process. The package manager will connect to a CTAN mirror, download the package, install it, and rehash the TEX tree for you.

This can also be done manually. The documentation for your distribution or the package that you are installing should specify details. The process involves three generic steps. First, you must locate the package. Nearly all mainstream packages reside on the CTAN mirrors. Once you have the package in hand, you must place it in the desired location in your TEX tree, typically an intuitively-named directory in `localtexmf`. Lastly, you need to rehash the TEX tree. This rebuilds an index of all available packages that TEX maintains, allowing the computer to locate the package you want when you request it from within a document.

LATEX packages are typically installed in `\texmf\tex\latex`. For organizational purposes, you may want to create a subdirectory for the package. After copying the package, you must rehash the TEX tree. To save time, the LATEX compiler does not scan the TEX tree for packages when run. Rather, it refers to a configuration file to tell it which packages are installed, and where. Typing `texhash` from the command line should run the executable that will refresh this file. In Windows XP, the command line can be accessed by selecting Start → Run, and entering `cmd`. On Mac OS X, the command line can be accessed from the `terminal` application in the Utilities folder. On any Unix or Unix-like operating system, you should be knee-deep in ways to access a command line. It is important to note that to hash the TEX tree, you will need administrator privileges. This is common on Windows and Mac OS, but you will need to either be root or use a program such as `sudo` on any Unix or Unix-like operating system.

If your computer cannot find the appropriate programs to hash your TEX tree (this will be somewhat more common on Windows than the other operating systems), you may have to find where the binaries for your TEX installation are located to run `texhash`; refer to your documentation for specifics.

8.3 Packages of Interest

There are a ridiculous number of user-defined packages available on the Internet. A typical LATEX installation should include the most common packages, as well as some that are not. There are packages available to simplify typesetting tasks in law, medicine, linguistics, computer science, chemistry, music, and more, as well as mathematics. Even if you have a very specific need, someone

has probably written a package to make it easy. CTAN is the best and most organized source of packages, and when it doubt, Google is your best friend.

$\mathcal{AMS}$ Packages

The American Mathematical Society ($\mathcal{AMS}$) maintains a family of packages that are invaluable in typesetting mathematical discourse. The `amsmath` package provides an expanded set of math symbols, and should always be used when typesetting mathematics. The `amsfonts` package provides additional mathematical fonts and should also be included in nearly all mathematical documents. The packages `amssymb` and `amsextra` are less essential, but still good to include.

The `amsthm` package provides the $\mathcal{AMS}$ theorem environments. This allows for the insertion of $\mathcal{AMS}$-style theorem environments as described in Section 4.5.

Line Spacing

The `setspace` package adds the ability to doublespace documents in LaTeX. To doublespace a document, put

```
\usepackage{setspace}
```

and

```
\doublespace
```

in the preamble. Other line spacing options are available, and are described in the documentation of the `setspace` package.

Hyperref

The `hyperref` package provides commands for typesetting Internet URLs (using the `\href` command). This not only improves the formatting of the URLs, but also adds hyperlinks to the linked URL. This is especially useful if you plan to distribute your document as a PDF. Here's the syntax for the `\href` command:

```
\href{http://blah.org/blah.html}{name of link}
```

In addition, by passing the option `colorlinks=true` to `hyperref`, the rectangle box around links and internal references will be replaced by changing the color of the text itself. For internal references (especially in the table of contents and the use of `\ref`), the color is set to red; for hyperlinks set by `\href`, the default color is pink.

Babel

The `babel` package extends LaTeX's language support. Refer to `babel`'s documentation for help with any specific language.

LaTeX also gives you the ability to define your own commands. This ability is seldom necessary as a great variety of packages are distributed with the major LaTeX distributions and many others are available on the Internet from LaTeX-using organizations and individuals. Chances are that most things you want to do are converged either in LaTeX or an available package.

Should you find yourself in a situation where you need something that is not available in LaTeX or a package, or you have a bunch of commands that are used repeatedly, you can define a new command in the preamble of your document.

9.1 Defining New Commands

The `\newcommand` command allows you to define a command with a name that is not already in use by LaTeX or any of the packages you are using with your document. Usage is as follows:

```
\newcommand{name}{definition}
```

Additionally, you can also add a third argument as follows to specify the number of arguments for your new command:

```
\newcommand{name}[number of arguments]{definition}
```

The `name` argument is the name you want for your command. For example, `\card` might be the name you want. The definition argument is a set of commands that describes what the command should do. For example, if you want `\card` to double overline its argument, you would use `\overline{\overline{}}`. Thus, you would add

```
\newcommand{\card}{\overline{\overline}}
```

to the preamble of your document. Alternatively, you could use

```
\newcommand{\card}[1]{\overline{\overline{#1}}}.
```

This formation is essential if you want to define a command that has more than one argument.

For example, if you have the following document:

```
\documentclass{article}
\newcommand{\card}[1]{\overline{\overline{#1}}}

\begin{document}

\[
\card{A}=\aleph_0
\]

\end{document}
```

you would get an output that looks like

$$\overline{\overline{A}} = \aleph_0$$

9.2 Redefining Commands

If a command already exists, you cannot use it as a name for a new command. However, if you want to redefine how a command acts, you can do this with the `\renewcommand` command. Other than modifying existing commands, rather than defining new commands, `\renewcommand` behaves exactly like `\newcommand`.

10.1 Overview

LaTeX has some native features to automate the process of creating a bibliography and textual references. These are affected with the use of the `thebibliography` environment. This environment will allow you to define bibliographic references that can be cited in the main document. This method is adequate for a document that makes use of a few sources, but it has some serious shortcomings that make it cumbersome for use in larger projects. We will not discuss `thebibliography` in detail because it is outclassed by BibTeX.

Strictly speaking, BibTeX is a separate program from LaTeX, but it is included with nearly every LaTeX distribution.

To better understand the limitations of this system, consider the following example:

```
\begin{thebibliography}{99}
\bibitem{Widder} Widder, David V.
  \underline{Advanced Calculus}. 2nd ed.
  Mineola, NY: Dover, 1989.
\end{thebibliography}
```

When placed at the end of a document, this code defines a source with the key `Widder`. To cite this source in the body of the document, invoke the `cite` command.

```
For more hardcore multivariable calculus, consult
\cite{Widder}.
```

This will produce the following output:

> For more hardcore multivariable calculus, consult [1].
> ...*more stuff*...
> [1] Widder, David V. Advanced Calculus. 2nd ed. Mineola, NY: Dover, 1989.

where the reference will print at the end of the document.

LaTeX makes things somewhat easier by handling the indexing. However, there is not much flexibility in the citation style. You must type the citation

in the desired style yourself. In the above example, the source was type out in typical MLA style. This can be frustrating and time consuming if you are not a master in the style with which you are preparing your document, or if you have to change the style at some later time. This inflexibility can prevent you from using a style correctly. According to the MLA, *Advanced Calculus* should be cited parenthetically as (Widder), rather than with the index [1]. There is also no easy way to reuse the same sources in other documents, other than using the traditional cut and paste. BIBTEX was created to address all of these issues.

BIBTEX is not a package; it is an auxiliary program with its own compiler. It is not strictly for use with LATEX; it can be used with other flavors of TEX as well. Most LATEX distributions are preconfigured to use BIBTEX.

The design philosophy of BIBTEX is to divorce the bibliographic information and style from the main document. The bibliographic information is stored separately in a `.bib` (ASCII text) document and the style is defined by a style package. An author can create a large list of commonly-referenced works to be use din many different documents without repeatedly entering the data. Furthermore, style packages can be used to automate and customize the creation and appearance of textual references and the bibliography. A good BIBTEX installation will include a large number of academic, legal, and professional styles. Many journals and publishers provide their own style packages. Some individuals have even provided large `.bib` documents of works commonly cited in a specific academic field. The use of BIBTEX may unnecessarily complicate small documents, but it is an invaluable tool for composing large documents or many documents that reference the same sources.

10.2 The Bibliographic Database

As described above, BIBTEX expects the list of references to be stored in a separate ASCII text document. To do this, use your text editor of choice to create a document with a `.bib` file extension. This is where you will type your data. This example will use `sources.bib`. The format for an entry is relatively simple. Figure 10.1 provides a sample.

Notice that the entry begins with a declaration of the type of document being referenced (in this case, a book). This is significant because books, papers, theses, etc., may be formatted differently, according to the style package. Table 10.1 is a list of some of the more common entry types. For specifics, consult the documentation of the style package you are using.

```
@book{Szekeres:2004,
   author = "Peter Szekeres",
   title = {A course in modern mathematical %
     physics: groups, {H}ilbert space and %
     differential geometry},
   publisher = "Cambridge University Press",
   address = Cambridge,
   year = 2004,
}
```

Figure 10.1: A sample BIBTEX entry

Table 10.1: BIBTEX entry types

Type	Description
`@article`	An article published in a periodical.
`@book`	A book with a defined author and publisher.
`@conference`	An article printed in the proceedings of a conference.
`@inproceedings`	Same as `@conference`.
`@manual`	A technical document or reference manual.
`@mastersthesis`	A thesis for a master's degree.
`@misc`	For works that are hard to categorize (i.e., web sites).
`@phdthesis`	Like `@mastersthesis`, but for doctoral theses.

The entire entry is enclosed in braces. Following the declaration of the entry type, it is necessary to assign the entry a key name. This is the name by which the source will be referenced in the main document. In this case, the entry is named according to the author and the date of publication, `Szekeres:2004`.

Commas are used to separate the data fields. Neglecting a comma at the end of a field is the most common source of errors with BIBTEX. Each field is assigned a value with the `=` operator. You must enclose strings with multiple words in quotation marks or braces, otherwise the BIBTEX compiler will ignore the whitespace and become confused.

Table 10.2 is a partial list of the fields that BIBTEX understands. Not all fields may be displayed by a bibliographic entry; this depends on the style being used.

Table 10.2: BibTeX data fields

Field	Description
`address`	The address (usually just the city and state) of the publisher.
`author`	The author(s); special considerations to be discussed shortly.
`booktitle`	The title of the book, when citing an article printed in a book.
`edition`	The edition of the book; write it out in full.
`editor`	The editor(s); refer to the discussion of the author name.
`journal`	The name of the periodical in which an article is published.
`month`	The month of publication for a periodical; can be abbreviated.
`note`	For information that does not fit into any other category.
`publisher`	The name of the publisher. What else?
`title`	The title of the article or book.
`url`	The uniform resource locator of a document accessed online.
`volume`	The volume number of a journal or multiple volume book.
`year`	The year of publication.

Notice the use of capitalization in Figure 10.1. Some citation styles have different conventions for capitalization in the document title. For example, MLA style capitalizes every word except prepositions, whereas only the first word of a title and proper nouns are capitalized according to the Chicago style. It is best to let the style package handle the capitalization. Surrounding a word or letter in braces will force capitalization, as in `{H}ilbert`. This is best reserved for proper nouns. TeX commands for special characters can be used, such as `\"u` for ü, but should be enclosed in braces as well.

Author names can either be entered as *"Given name Surname"* or *"Surname, Given name"* (i.e., *"Grover Cleveland"* or *"Cleveland, Grover"*). BibTeX will format the author's name appropriately. If there is any ambiguity between the given name and the surname, use braces (for example, use *"Zack {de la Rocha}"* instead of *"Zack de la Rocha"*). Generally, the use of braces forces BibTeX to treat the enclosed quantity as a single entity.

To reference a work by multiple authors, simply insert `and` between each name. The style package will handle the rest. The list of authors may automatically be shortened and terminated with *et al*, *and others*, etc., if it becomes too long.

In general, an entry follows the following syntax:

```
@doctype{<key>,
    author = {...},
```

```
    title = {...},
    ...,
}
```

A number of entries constitutes a bibliography database (hencefore, "source file"). If you notice a commonly recurring stirng in your sources, such as multiple references to the same journal, you can define a macro to use as an abbreviation. For example, to abbreviate `The American Journal of Physics` as `AJP`, place the following declaration at the beginning of the source file:

```
@string{AJP = {The American Journal of Physics}}
```

Then to reference the journal in a bibliographic entry, define the journal field with `journal = ajp`. The abbreviation is not case-sensitive. The general format for a string declaration is

```
@string{abbr_name = {...} }
```

10.3 Generating the Bibliography

You must include a style package in order to produce a bibliography in your document. There are plenty of styles available online and you will most likely find one that fits your needs. The three most common styles in the humanities are those defined by the Modern Language Association (MLA style), the American Psychology Association (APA style), and the University of Chicago's Manual of Style (Chicago style). Only high school teachers (and some literature professors) care about MLA style, but implementations of the others hsould be in any respectable LaTeX distribution.

To generate a bibliography with BibTeX, you may need to include a macro package. A style may require macros that are defined in a specific package. The `natbib` package seems to work with most styles. Some styles come bundled with their own macro package: `natbib` includes some styles, and so does `jurabib`. Some common styles are listed in Table 10.3. In the document body, define the style you want by passing it to the `\bibliographystyle` command. At the point in the body where you want the bibliography to appear (usually at the end), pass the name of your source file to the `\bibliography` command.

As a side effect of BibTeX actually being a separate program, you must run both programs on your source file. First, use LaTeX to compile the document and ignore all of the compiler warnings. Then run the BibTeX compiler. In a properly configured LaTeX environment, this is done by inputting

Table 10.3: Common BibTeX style packages

Style Package	Description
`apa`	An implementation of APA style.
`apacite`	Another APA style package.
`apalike`	It's like APA, but not quite.
`chicago`	Do it Chicago style.
`chicagoa`	... another Chicago style.
`harvard`	Harvard style? Does anyone use this?
`jurabib`	A few useful styles, as well as footnote support.
`natbib`	Several styles common in the natural sciences.
`oxford`	Like Harvard style, but from Oxford.

```
bibtex documentname.tex
```

at the command line. Some editors have support for BibTeX in the interface. Assuming that BibTeX executes properly, you can recompile the document with LaTeX and the citations and bibliography should appear. If you notice errors, such as lots of question marks in your document, or if the LaTeX compiler keeps complaining about missing references, run LaTeX again. It can take several compiles for all the references to stabilize.

If you make changes to your source file, you will need to recompile with BibTeX and again with LaTeX to update all of the references.

As an example, let us say there was a paper that used a variant of the Oxford style that required the `jurabib` package (it actually comes with the package). The style itself is called `jox`. The name of the bibliography database was `sources.bib`. Included in the document preamble was

```
\usepackage{jurabib}
```

and just before the end of the main body was

```
\bibliographystyle{jox}
\bibliography{sources}
```

Of course, the `jurabib` package needed to be installed correctly for this to work (see Chapter 8). Also, the file `sources.bib` was located in the same directory as the LaTeX document. If it were somewhere else, its exact location must be specified. BibTeX expects the source file to have a `.bib` extension; the complete filename would have been needed if the extension was something else.

The babelbib package will let you specify a language for your bibliography. Also, with a package like bibunits or multibib, you can include more than one bibliography in the same document, say, one for each chapter. The moderately useful splitbib package will allow you to split up a bibliography and organize it by category. Check out their documentation to find out how to do this.

10.4 Textual References

BIBTEX will only include those references in the source file that are cited in your document. If you had created the source file sampled above and invoked BIBTEX in some random document, nothing would have happened (or, nothing *should* have happened).

The citation method will depend somewhat on your choice of style. It is generally accomplished by passing the key of the document you are referencing to the \cite macro. To cite our previous example:

```
The theory of general relativity postulates that
spacetime is a four-dimensional Minkowskian
manifold \cite{Szekeres:2004}.
```

This will produce an inline citation, automatically formatted by the style. Using the chicago style, this would print as (Szekeres, 2004). Most styles will allow you to pass more information to \cite, such as the page number, i.e., \cite[536]{Szekeres:2004} to cite page 536. This input gives (Szekeres, 2004, 536) in chicago.

You can force the inclusion of a document that you do not explicitly cite with \nocite{<key>}. This essentially creates an invisible citation. For styles that do not default to parenthetical citations, you can usually use \citep{<key>} to make one. The footbib package introduces the \footcite{<key>} macro that will place the full citation in the page footer, although it only works with some styles. This is also possible with the jurabib package. Finally, if you just want to list every single entry in your source file in the bibliography, use \nocite{*}.

Appendices

Nearly all the software you need is freely distributed, with some frontends being the notable exception. As a consequence, LaTeX runs quite well on just about any modern platform.

The software necessary for using LaTeX mostly falls into three categories: LaTeX distributions, editors, and frontends. LaTeX distributions include the LaTeX compiler, and generally come with a DVI viewer, and some other useful programs for working with LaTeX documents.

Editors allow you to manipulate plain text files, such as LaTeX source files. A basic editor is included with most modern operating systems. For the Mac, SimpleText was the included basic editor for a very long time, and more recent versions of the Mac OS (Mac OS X) include a more sophisticated editor, TextEdit. Windows offers Notepad, which is a bit limited, as well as WordPad in all current versions. Lastly, Unix operating systems typically have many editors. These include vi, its cousin vim, Emacs, Pico, and Pico's GNU counterpart Nano.

Even the best editor is still focused on the task of editing text files. While a text editor is the most basic requirement to use LaTeX, there is another class of programs called frontends, which seek to simplify the task of creating LaTeX documents. Frontends typically include editor capabilities that are specifically geared towards LaTeX. This means that source code is often coded for clarity, the pairing of delimiters is sometimes tracked, and some have auto-complete functions based on what is currently being typed. The most important feature in most frontends, however, is some degree of integration with a compiler. This allows you to edit, compile, and view your document all from within a single program. Some frontends even go so far as to let you view the compiled code in real-time, or near real-time, and edit the underlying source code graphically from the compiled version.

A.1 Installing a LaTeX Distribution

The first and most important thing to have before creating LaTeX documents is a LaTeX distribution. This section will describe some of the most common LaTeX distributions for different operating systems.

Windows 95/98/98SE/ME/2000/XP/Vista

MiKTEX is a very popular LATEX distribution for Windows. Before installing, though, you have decide whether to go for a barebones MiKTEX installation and then download the packages as you need them or a complete installation of MiKTEX with all packages and other optional files installed. Regardless of the type of installation you want, to install it, download the appropriate installer. Once the download is finished, run the installer. It will then connect to the Internet and download the files needed to install MiKTEX. After the installer finishes downloading everything, close the installer and run it again. This time, let it install the distribution from a local directory (the temporary directory the installer saved all the files to). After a few minutes, MiKTEX should be installed.

Mac OS X and Unix

TEX Live is the recommended LATEX distribution for Mac OS X and Unix. You can go to their web site at http://www.tug.org/texlive/ for more information and to download the installer.

A.2 Frontends

At this point, you have a working LATEX installation. While you could use a regular text editor to make your LATEX files (discussed in the next section), using a dedicated LATEX frontend is recommended. Here are a couple of LATEX frontends for Windows and Macs.

Windows

TEXnicCenter is an open source and freely distributable frontend to LATEX. It integrates the LATEX and BIBTEX compilers as well as a menu-driven interface to insert common LATEX commands. You can download TEXnicCenter at http://www.toolscenter.org/.

Mac OS X

TEXShop is an open source and freely distributable frontend for LATEX on the Mac OS X. Like TEXnicCenter for Windows, it integrates the LATEX compilers together and offers a user-friendly interface for LATEX beginners. You can download TEXShop at http://www.uoregon.edu/~koch/texshop/.

Multiplatform

AUCTEX is a multiplatform LATEX frontend that interfaces with Emacs to provide a comprehensive editing environment for TEX documents. It has many features that are too numerous to mention in here, but you can find out more at http://www.gnu.org/software/auctex/.

A.3 Text Editors

While using a frontend is recommended, there are times when a simple text editor is the best option (and in some cases, the only option). Some examples that have been mentioned previously are Emacs, vi (and its cousin vim), Pico, and Nano.

Before using these math commands, it is a good idea to load the `amsmath`, `amssymb`, and `amsthm` packages to ensure that the commands work as shown.

Table B.1: Math Commands

Symbol	Command
$\sum$	`\sum`
∞	`\infty`
$\int$	`\int`
$\cap$	`\cap`
$\cup$	`\cup`
$\oplus$	`\oplus`
$\otimes$	`\otimes`
$\pm$	`\pm`
$\in$	`\in`
$\subset$	`\subset`
$\subseteq$	`\subseteq`
$\leqslant$	`\leq`
$\geqslant$	`\geq`
$\ldots$	`\ldots`
$\cdots$	`\cdots`
$\vdots$	`\vdots`
$\ddots$	`\ddots`

Table B.2: Lowercase Greek letters

Letter	Command	Letter	Command
α	`\alpha`	ν	`\nu`
β	`\beta`	ξ	`\xi`
γ	`\gamma`	π	`\pi`
δ	`\delta`	ρ	`\rho`
ϵ	`\epsilon`	σ	`\sigma`
ζ	`\zeta`	τ	`\tau`
η	`\eta`	υ	`\upsilon`
θ	`\theta`	ϕ	`\phi`
ι	`\iota`	χ	`\chi`
κ	`\kappa`	ψ	`\psi`
λ	`\lambda`	ω	`\omega`
μ	`\mu`		

Table B.3: Lowercase Greek letter variations

Letter	Command
ε	`\varepsilon`
ϑ	`\vartheta`
ϖ	`\varpi`
ϱ	`\varrho`
ς	`\varsigma`
φ	`\varphi`

Table B.4: Capital Greek letters

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

Table B.5: Arrows

Arrow	Command
$\to$	`\to`
$\mapsto$	`\mapsto`
$\rightarrow$	`\rightarrow`
$\Rightarrow$	`\Rightarrow`
$\longrightarrow$	`\longrightarrow`
$\Longrightarrow$	`\Longrightarrow`
$\leftarrow$	`\leftarrow`
$\Leftarrow$	`\Leftarrow`
$\longleftarrow$	`\longleftarrow`
$\Longleftarrow$	`\Longleftarrow`
$\leftrightarrow$	`\leftrightarrow`
$\Leftrightarrow$	`\Leftrightarrow`
$\longleftrightarrow$	`\longleftrightarrow`
$\Longleftrightarrow$	`\Longleftrightarrow`

Table B.6: Math fonts

Font	Command
$\mathbb{NQRZ}$	`\mathbb{}`
text	`\textrm{}`
text	`\text{}`
math	`\mathrm{}`
bold	`\mathbf{}`
ABCXYZ	`\mathsf{}`
italics	`\mathit{}`
$\mathcal{ABCXYZ}$	`\mathcal{}`
$\mathfrak{ABCXYZ}$	`\mathfrak{}`

While this list is by no means exhaustive, it does contain the names of many of the most commonly used and most useful packages.

Table C.1: General utility packages

Package	Description
acronym	Automates the use of acronyms.
appendix	Easy way to make appendices.
calc	If you are very lazy, you can do arithmetic in your document.
comma	Spaces large numbers with commas.
endnotes	Makes endnotes out of footnotes.
fancybox	Annoy your readers by putting boxes around everything.
fancyhdr	More control over document and page headers.
fullpage	Sets all margins to one inch.
geometry	Easily set margins and borders.
longtable	Support for tables that span multiple pages.
lscape	A landscaped layout for pages.
microtype	Allows you to adjust the microtypographic properties of your document.
setspace	An easy way to set line spacing in your document.
titling	Control the uncooperative `maketitle` command.
verbatim	A better `verbatim` environment.
vruler	Allows you to add line numbers.

Table C.2: Special document packages

Package	Description
assignment	A simple template for typing up homework.
bizcard	In case you need a business card.
booklet	Booklets to hand out on Sproul.
calendar	It makes calendars.
cdcover	Makes a CD cover for the Coldplay album you just got from a friend.
cv	Type up your curriculum vitae (a résumé on steroids).
fax	If you should ever find yourself in possession of a fax machine.
memoir	A general purpose book document class with some nice features.
newsletr	A newsletter document class.
refman	A reference manual document class.
slides	Slides for a presentation.
beamer	A computer slideshow document class.

Table C.3: Graphics, color, and drawing packages

Package	Description
bardiag	Make bar diagrams without Excel.
caption	Better support for making captions to figures and tables.
color	Change text and page colors.
colortbl	Color cells in a table.
graphics	The typical LaTeX package for including graphics files.
graphicx	Improved version of the graphics package.
graphpap	Print your own graph paper.
histogr	Histograms.
rotating	Rotate stuff. Paragraphs, figures, pages, whatever.
texdraw	Draw stuff.
xcolor	Extension of the color package.

Table C.4: Bibliography and citation packages

Package	Description
bibunits	Allows multiple BibTeX bibliographies in the same document.
footbib	Make footnotes out of your textual citations.
jurabib	A BibTeX package of styles useful in law and the humanities.
natbib	Lots of useful bibliography styles.
splitbib	Organize a bibliography by category.

Table C.5: Science packages

Package	Description
bpchem	Package for typesetting chemical formulae and names.
chemsym	Another package to simplify typing chemical symbols.
feyn	Why Feynman diagrams? Because I am Richard Feynman.
ochem	Macros for drawing organic compounds.
sistyle	Easily type numbers with SI dimensions.
unitsdef	Similar to the sistyle package.

Table C.6: Computer science packages

Package	Description
algorithm2e	A package for typesetting symbolic algorithms.
algorithmcx	Similar to the algorithm2e package.
binhex	Will convert decimal numbers to binary, octal, or hexadecimal.
bytefield	Makes bit field diagrams in case you are designing a microprocessor.
c-pascal	Macros to help typeset code written in C or Pascal.
circ	Draw your own circuit diagrams wtihout a template and graphing paper.
listings	Display a source code document in LaTeX.
timing	Timing diagrams.

Table C.7: Language packages

Package	Description
babel	Support for forty-one languages and multilingual documents.
babelbib	. . . and now with multilingual bibliography support.

Table C.8: Mathematics packages

Package	Description
amsmath	The core of the AMS-LaTeX system.
amssymb	More math symbols provided by AMS-LaTeX.
amsthm	An improved version of the `newtheorem` macro.
amscd	Commutative diagrams!
breqn	Automatic line breaking in equation environments (very useful).
fp	Like the calc package, but with support for real numbers.
polynom	Manipulate polynomials in your document. For the very lazy.
xy	Introduces the `xymatrix` environment for drawing crazy figures.

Table C.9: Oddball packages

Package	Description
backgammon	Apparently, some people still take backgammon seriously.
chess	For the aspiring chess genius.
crossword	For the aspiring senior citizen.
musictex	Type your own sheet music... but why not use a real music program?
movie15	You can include media files in PDF documents to annoy people.
othello	Does anyone still play this?

If you have every compiled a LaTeX document, chances are high you have received a few error messages. Sometimes they come from something as stupid and as easy to fix as forgetting a parenthesis or forgetting to end an environment. There are also a lot more cases where you have no idea what you have done wrong and it takes you a long time to find or even understand your error.

The purpose of this is to explain some of the common errors that may happen when compiling a LaTeX document and suggestions for what is probably going on and how to debug your document.

D.1 The Form of an Error

There are two forms of errors: LaTeX errors and TeX errors. In both types of errors, the part after the error message will tell you where the error occurred. An example:

```
l.15 <offending text>
```

The `l.15` tells you what line the error occurred on and the text will tell you the text that caused the error.

LaTeX Errors

The general form of an error in LaTeX is shown below:

```
! LaTeX error: <error message>

See the LaTeX manual or LaTeX Companion for
explanation.
Type H <return>   for immediate help.
...
```

The `!` lets you know that the error has occurred. The error message will tell you what type of error you have committed. After the ellipses, you will find the line at which the error occurred and the text that caused the error (or at least the text where LaTeX found the error).

TEX Errors

Errors may also have the following form:

```
! <error message>
```

These errors are formatted differently because they are error messages that came from TEX instead of LATEX. After the error, you will still find the line that the error occurred in and the text of the error.

D.2 Warnings

There are some error messages that are just warnings and will not stop or change the compilation of the document. Chances are you have seen them many times.

Underfull

The following error results when a line does not extend the width of the page, something LATEX always tries to accomplish:

```
Underfull \hbox (badness 10000) in paragraph at lines
104--107
```

This error message is just a warning and is not something to worry about. For the most part, when a line does not span the width of the page, it is because you have written something that you want to only cover part of the page.

Overfull

The following error results when a line extends beyond the width of the page:

```
Overfull \hbox (16.04988pt too wide) in paragraph at
lines 30--31 [] [] \OT1/cmtt/m/n/12 I'm trying to put
way too much text into a line in my document.
```

Usually this error comes form when you are using the `verbatim` package because it will not move to the next line if your text does not go to the next line. The easiest way to fix this is to find the place in your document where this is occurring and change the text so that it fits to the page.

This error will still show up if the text is still on the page but outside of the width of text that LATEX has set. In this case, you are welcome to fix things so that the error does not show up or you can leave the text as it is.

References

The following warnings occur when references are changed when LaTeX was compiled:

```
LaTeX Warning: Label(s) may have changed. Rerun to get
cross-references right.

LaTeX Warning: There were undefined references.

LaTeX Warning: Reference ‘name’ on page 1 undefined on
input line 15.
```

The way to fix these errors is to recompile the document again to correct the page numbers. Sometimes it is necessary to recompile the document twice to fix this error. You also may have defined a reference wrong, so you should check to make sure your label is correct.

D.3 Beginning and Ending

Begin Ended by End

This type of error occurs when each environment is not correctly started and ended. When you are missing an `\end` command, the following error will show up:

```
! LaTeX Error: \begin{enumerate} on input line 23
ended by \end{document}.
```

To fix this, you need to end the environment mentioned in the error with the appropriate command.

When you are missing a `\begin` command, the following will appear:

```
! LaTeX Error: \begin{document} ended by
\end{itemize}.
```

To fix this, you basically do the same thing as before, correctly beginning the environment mentioned in the error with the appropriate command.

End Occurred Inside a Group

The following error message will show up at the end of compiling a file if an environment is begun that is not ended:

```
(\end occurred inside a group at level <n>)
```

To fix this error, make sure you end the environment that was begun. The previous error is more helpful in finding the `\begin` statement.

Ended by End of Line

The following error will occur when you try to place a command inside a section heading:

```
! LaTeX Error: \verb ended by end of line.

See the LaTeX manual or LaTeX Companion for
explanation.
Type H <return>   for immediate help.
...
```

There will be many errors of the same type for this mistake. In order to find where you put the command, look in the output file and find the last heading that shows up.

Missing Begin Document

This error is self-explanatory:

```
! LaTeX Error: Missing \begin{document}
```

D.4 Errors Usually Caused by Bad Spelling

Unknown Control Sequence

This error results when you use a command (something that starts with a \) that is not recognized by LaTeX:

```
! Undefined control sequence.
```

Usually this error results from spelling a command incorrectly. Go to the line that is indicated by the error and fix the command.

Environment Undefined

This error results when you begin an environment with a `\begin` command that is not recognized:

```
! LaTeX Error: Environment verbatin undefined.
```

Usually you have just spelled your environment incorrectly, so you just need to fix it.

Bad File Name

This error results when you have mistyped the command `latex` or do not have LaTeX installed on your computer:

```
Bad command or file name
```

To fix this, correctly spell the command to compile your file or make sure that LaTeX is correctly installed on your computer.

Cannot Find File Name

This error occurs when you try to compile a file that the computer cannot find:

```
! I can't find file 'sample'.
<*> sample

Please type another input file name:
```

To fix this error, make sure you have spelled the file name correctly. You also may be in the wrong directory to compile the file, so check to make sure you are in the same directory as your file.

D.5 Fatal Errors

Runaway Argument

This error happens when a paragraph ends before a command's argument is done (i.e., LaTeX thinks that there is a missing }):

```
Runaway argument?
```

To fix this, you should use a different command to accomplish what you are trying to do. An example of this is to use `\bfseries` instead of `\bftext` to make bold text in more than one paragraph.

This error can also be caused by a missing mandatory argument to a command.

Just an *

This error normally occurs when you do not end your document with `\end{document}`:

```
*
```

If you are prompted to enter something in, it is best to enter

```
\end{document}
```

and hope it works. Be sure to end your document with the appropriate command.

Emergency Stop

This error happens when LaTeX will stop trying to compile your document due to a serious error:

```
! Emergency stop.
```

To fix this error, you will need to figure out what caused it to stop compiling. Chance are you forgot to end your document with `\end{document}`, but there might also be another reason for the emergency stop.

Please Type a Command or Say End

This error happens when your file has ended prematurely:

```
(Please type a command or say `\end')
```

The best way to deal with this type of error is to type

```
\end
```

or

```
\end{document}
```

in the case that the absence of that command caused the error. Usually if you have ended your document correctly, the error will result from a missing } or forgetting to end a verbatim environment.

D.6 Graphics Errors

Too Many Unprocessed Floats

This error occurs when figures or tables (i.e., floats) have not been typeset:

```
! LaTeX Error: Too many unprocessed floats.
```

LaTeX can only have so many floats waiting to be typeset. In order to fix this error, make sure that you are placing your floats where you want them (with a `[h]` option) and not wanting too many on one page in sequence. Using the command `\clearpage` can be very useful in distributing floats correctly.

Unknown Graphics Extension

The following error occurs when you try to use a type of graphic that is not supported by the type of file that you are producing:

```
! LaTeX Error: Unknown graphics extension: .gif
```

In order to fix this error, you should change your graphics to the types that are supported by the type of file you are outputting or you will need to include the correct package to deal with that type of graphic. Sometimes you may have named the graphic poorly so that LaTeX will not recognize it as a graphic file.

Division by Zero

The following error occurs when the height of a graphic object is zero:

```
! Package graphics Error: Division by 0.
```

This is usually caused when you rotate an object with zero depth so that its height becomes zero. The best way to fix this is to use the keyword `totalheight` instead of `height`.

D.7 Math Errors

Display Math Should End With $$

This error occurs when the displaymath or equation mode is ended incorrectly:

```
! Display math should end with $$
```

To fix this error, make sure that you end the displaymath or equation mode correctly (ending them with a $ is not acceptable).

Bad Math Environment Delimiter

This error occurs when you do not have your delimiters correct in math mode:

```
! LaTeX Error: Bad math environment delimiter.
```

Usually this occurs when you forget to match a right delimiter with every left delimiter. This error may also happen when you forget to end an array.

Missing Right

This error occurs when you have a missing right parenthesis:

```
! Extra \right.
```

To fix this, you either need to add a `\right` command or you need to end an array.

Missing Delimiter

This error message occurs when a delimiter is missing:

```
! Missing delimiter (. inserted).
```

To fix this error, you need to make sure that you have a right delimiter for every left delimiter. If you do not want a right delimiter matching a left delimiter, you need to use "." to not have an error message show up.

Missing $ Inserted

The following error occurs when you try to use a character that can only be used in math mode, like _ or ^:

```
! Missing $ inserted
```

To fix this error, make sure you change the character to what it should be in text mode.

D.8 Tabular Environment Errors

Misplaced Alignment Tab Character &

This error occurs when you use `&` and when you are not in a tabular environment:

```
Misplaced alignment tab character &
```

To fix this error, you need to use `\&` to make a &.

Extra Alignment Tab

This error occurs when you use too many tabs for the number of columns in a table:

```
! Extra alignment tab has been changed to \cr
```

The result of this error is that a new row is formed where the extra tab was. You should go back and fix your table so that the correct number of items in each row would show up.

Argument Has an Extra }

These errors happen when an incorrect number of arguments to a tabular environment have been specified:

```
! Argument of \cline has an extra }.

! Argument of \multicolumn has an extra }.
```

To fix this error, make sure your arguments to the tabular environment are correct.

D.9 Errors With Lists

Missing Item

This error occurs when there is plain text in an environment that takes items:

```
! LaTeX Error: Something's wrong--perhaps a missing
\item.
```

To fix this error, make sure the plain text is changed into an item.

Too Deeply Nested

This error occurs when there are too many lists for LaTeX to handle:

```
! LaTeX Error: Too deeply nested
```

LaTeX can only handle four levels of one type of list and six levels of different types of lists. To fix this, you need to use less levels of lists or define your own list environment.

D.10 Miscellaneous Errors

Only Used in the Preamble

This error occurs when you place a command in the body of a LaTeX document that should be placed in the preamble:

```
! LaTeX Error: Can be used only in the preamble.
```

To fix this error, just move the command to the preamble.

There Is No Line/Page Here to End

This error occurs when you incorrectly use the commands that make a new line or a new page:

```
! LaTeX Error: There's a no line here to end.
```

You may just leave the command that is making a new line in place or you can take it out. Here, LaTeX is just trying to make sure that everything looks nice.

Command Already Defined

This error occurs when you try to define a command that already exists:

```
! LaTeX Error: Command ... already defined.
```

To fix this, you need to define your command differently.

Missing Number

This error is made when a number is expected as an argument and one is not provided:

```
! Missing number, treated as zero.
```

To fix this error, you need to find where a number is expected so that you can provide the correct one.