A Running Introduction to LaTeX

Dr. Ken Rietz

Chapter 1

Getting Started

This is a brief introduction to LaTeX for technical mathematics. This is a running introduction because this is the first chapter of a document in process. If you have any suggestions, especially about things that are (a) confusing, (b) wrong, (c) incomplete, or (d) missing, please contact Dr. Coulliette who will relay the information back to me. Thanks!

1.1 Introduction/motivation

1.1.1 Advantages

What benefits are there in learning LaTeX? Oh, there are lots. Here are a few:

- LaTeX is free, forever, by design.
- LaTeX runs on more computers and more operating systems than you have ever heard of.
- LaTeX allows you to create documents exactly the way you want them to look. Anyone who has struggled with Microsoft® Word® knows that WYSIWWW ("What you see is what we want") is sometimes infuriating. With LaTeX, you can literally place each pixel where you want it. (That is a horribly long way to create a document, but is handy for graphs and figures.)
- LaTeX is professional. It produces documents that meet the very highest standards, without you having to do anything exceptional.

- LaTeX is a standard. If you ever want to submit a paper to a mathematical journal, you will find LaTeX is required.
- LaTeX is unbelievably flexible. There are extensions of LaTeX that can create recipes, crossword puzzles, chess problems, nicely formatted computer program code, presentations, musical lyrics, and knitting charts, for example.
- LaTeX uses standard ASCII text. That means it will always be usable. Ask someone who used WordStar to type their thesis how accessible it is today. That also means that it never contains any proprietary codes.
- LaTeX does a lot for you. It can create an index, a table of contents, a list of tables, and a list of figures automatically. OK, you have to tell it what you want in the index; LaTeX does not read your mind. The others are created for you atuomatically if you ask for them.
- LaTeX will also hyphenate for you, entirely under your control. (I usually just let LaTeX decide, since it does a really good job.)
- LaTeX is international. It "knows" more languages than you will ever need to use. This includes a huge variety of characters in non-alphabetic languages. It even knows how to hyphenate correctly words in many alphabetic languages.
- Dr. Coulliette is going to require you to use it.

1.1.2 Disadvantages

Is LaTeX perfect? No, there are a few hassles, and it is only fair to warn you about them up front.

- LaTeX is not WYSIWYG. Instead, it is a text formatting program. You provide codes telling the LaTeX processor what you want it to do. The formatting is very fast (It takes only 11 seconds to process the entire Calculus I text), so that is not a great difficulty.
- LaTeX has a learning curve. But so does every other processor. The advantage of LaTeX here is that once you learn it, you don't have to learn a new set of commands for the next version to come out. There are things that I know how to do in Excel®2007 that I haven't been able to find in Excel®10.

So, now that I have convinced you to use LaTeX, let's get a few background items.

1.1.3 Pronunciations

First, there was TeX, developed by Dr. Donald Knuth (pronounced kuh-NOOTH). The pronunciation of TeX is a bit tricky. The first two letters are not hard (with the "E" being short), but the "X" is harder. It is technically a guttural fricative, similar to the "ch" in Swiss German that sounds like you are clearing your throat. According to Knuth, if you are saying it properly to your computer, the screen should get slightly moist. Most English speakers simply pronounce TeX exactly like "tech". Pronouncing it "teks" is definitely wrong. A TeX guru is a TeXspert, not a TeXpert.

Leslie Lamport, the developer of LaTeX, decided that the pronunciation of LaTeX should be determined by usage and not by fiat. That means that there are at least three officially sanctioned pronunciations of LaTeX: LAH-tech, lah-TECH, and LAY-tech. You will hear any of these at different times, even from the same person.

1.2 Getting LaTeX

1.2.1 What is the color of your computer?

If you are running a Mac or a Linux machine, you already have LaTeX. Microsoft® Windows® machines need to download it.

For those of you so unfortunate as to be using Microsoft® Windows®, please note that you will have to install some version of LaTeX before you can get any LaTeX items to work. Fortunately for you, though, some nice people have created free versions of LaTeX that you can use. One very common version is `MiKTeX`, and another that I have used is `emTeX`, but that seems to have stopped being developed. Both can be downloaded from the Internet, and must be installed before you can get an IDE running. Note, however, that both of these are installations of TeX, not specifically LaTeX, but both include LaTeX.

1.2.2 TeX versus LaTeX

There **is** is big difference between TeX and LaTeX, much like the difference between computer programming in assembler versus computer programming in C++. What the LaTeX processor does is convert user-friendly (though at times you will debate that term!) commands into TeX commands, and then run a TeX processor on it. Think of LaTeX as a front-end for TeX.

TEX was written by Donald Knuth in order to typeset his computer books. LATEX was written by Leslie Lamport in order to make TEX usable for non-geniuses, like us.

1.2.3 Getting a decent IDE

LATEX can be run very effectively using nothing more than a text editor (my favorite being Vim) and a command line (terminal screen with a prompt that looks like $). I still tend to do that when I have a significant amount of work to do. But for this handout, I decided that it was only fair to use the IDE (Integrated Development Environment) I was going to recommend.

Having looked over a variety of them, I have settled on `texmaker`, which is available in identical interfaces for Mac, Linux and Windows®. Very little will depend on the IDE, but when necessary, I will refer to `texmaker` commands.

You can download this over the Internet for your particular system quite simply.

Before you can use `texmaker` effectively, you really do need to make a few customizations. There are directions with `texmaker` for doing this. I will assume that you have actually done those.

1.3 Getting used to LATEX

LATEX is not hard to use, but it has the general attitude that makes you want to scream "Do what I *mean* and not what I *say*!" In other words, it is a computer program.

1.3.1 Document basics

Parts of a LATEX document

There are different pieces to a LATEX document, and they have different purposes. We will discuss what ought to go in the different sections.

Preamble The basic type of document and information that is to be used throughout the document appears at the very top of the document, called the preamble. Various formatting options are commonly put here. More things will

appear later. The preamble is everything between the command `\documentclass` and the command `\begin{document}`.

Body Everything that appears between the `\begin{document}` command and the `\end{document}` command is the body (or text) of the document. Filling this in is what this document is all about.

The rest Anything after `\end{document}` is completely ignored by LaTeX. This can be useful, such as when you want to format just the first part of the document, without deleting the rest of the document first. Just put the `\end{document}` where you want LaTeX to stop. It is also a handy place to put things that aren't ready for inclusion in the document yet.

Creating a simple LaTeX document

Here is a complete LaTeX document in four lines:

```
\documentclass[12pt]{article}
\begin{document}
Hello, world!
\end{document}
```

As you might guess, all it does is create a PDF document that contains "Hello, world!" in 12-point font. We can expand this slightly.

```
\documentclass[12pt]{article}
\begin{document}
Hello, world!
Today is \today.

Have fun using \LaTeX!
\end{document}
```

This adds a phrase giving the date (August 22. 2011, for example), followed by "Have fun using LaTeX!" in a new paragraph. However, the new paragraph is very hard to see, since there aren't enough characters in the first line to make the indentations visible. You can try adding more characters (gobble gobble gobble gobble gobble ... or something more creative) after the period to fill out the first line, and you will then see the paragraph appear. But then you will need to know something really important:

1.3.2 How Do I Run LaTeX?

That's a good question. Each IDE has its own way, but `texmaker` uses two main ways:

1. Click on `Tools | Quick Build` on the menu bar.
2. `F1` on Microsoft®Windows® or `fn-F1` on a Mac, the shortcut key for this.

If the space directly below the LaTeX stuff you typed in is blank or only blue lines, you are fine. When red letters appear, that signals a LaTeX error that needs to be fixed. For now, make sure you have typed everything in correctly.

1.3.3 A few preliminaries

There are some observations that are useful.

- All LaTeX commands begin with a backslash (the \ character). It will be followed by one or more alphabetic characters or a single non-alphabetical character.
- All mandatory options to a LaTeX command are enclosed in braces, the { and } characters. Not all commands require options.
- All optional arguments (most LaTeX commands have optional arguments) are enclosed in square brackets, [and], and come between the command and mandatory arguments (if any).
- Some commands don't need any options at all.
- The first command must be `\documentclass`. The only thing that can come before it are comments, discussed later.
- Everything in between `\begin{document}` and `\end{document}` is formatted.
- LaTeX will put in its own line breaks. In fact, the purpose of LaTeX is to let it do all the formatting for you.

There are a lot more things to learn, but overall, you just type things into the body and let LaTeX loose on it.

1.3.4 Comments

A comment is put in a LaTeX document by using a percent (`%`) sign, and runs from the percent sign up to and including the end of the line containing it. Comments can go pretty much anywhere, except that they can't break apart LaTeX commands.

1.3.5 Verbatim text

That presents an interesting point. How could I put a `%` in this document? Wouldn't it immediately trigger a comment, and delete itself and everything that follows? And, if you think about it, how do I print the LaTeX commands here without causing them to be run?

There are two ways, both of which allow you to type almost anything. The font automatically is changed to typewriter (monospaced) font, and nothing is interpreted. The short command, used for inline material (like `\LaTex\`) is started with a command `\verb`. The next character is automatically set to be the beginning and ending delimiter. That is, if I type `\verb!`, then everything up until the next `!` will be typeset with no unterpretation. The one exception is that the `\verb` command argument (the text between delimiters) can't include a line feed, or you get an error.

The delimiter I tend to use the most is `!`, with `+` as an alternative when I have to include an exclamation point in the text (which is rare). The reason is that `!` has no other meaning in LaTeX, so it is safe.

If you want to include line feeds in verbatim text, you need a different command. Actually, it is a larger concept, called an environment. It starts with `\begin{verbatim}` and will end with the exact sequence of fourteen characters `\end{verbatim}`. An environment always starts with a `\begin{....}` and ends with a matching `\end{....}`.. We will encounter many more of these.

1.4 Mathematics in LaTeX

The place that LaTeX really shines is in formatting mathematical formulas. It is designed to make entering equations simple, and then typesets them beautifully.

1.4.1 Inline versus display mode

There are two places that formulas can appear: inline (as part of the running text) or in display mode (which is on a separate line, with the formula centered). The difference is clear. Here is an inline version of the Pythagorean theorem: $a^2 + b^2 = c^2$, and here is the display version of the same:

$$a^2 + b^2 = c^2.$$

Note that LaTeX will increase the line spacing to accomodate an inline formula when that is necessary, as in $\int_1^\infty \frac{1}{x^2}\,dx = 1$. There is a bit of extra space around the line with that formula in it.

Math mode

LaTeX typesets mathematics when it is in what is called math mode. Inline mode and display mode are both varieties of math mode, and you enter formulas the same for both. (I will show you how to get into inline mode and display mode in a moment.)

If you are used to entering formulas into Maple, some of this will be familiar.

Once you are in math mode, various things change:

- All letters are italicized automatically, unless they are part of a function. That is, in $y = \cos x$, the y and x are italicized, but cos is not. This is standard practice.
- You create superscripts using `^` but it will only raise the character following the `^` to the exponent. You get around that by enclosing larger exponents in braces. For example, `x^12` will appear as x^12, while `x^{12}` will appear as x^{12}.
- You create subscripts using `_`, with the same condition as exponents: only the character following the `_` is lowered, or you can lower a collection of characters by enclosing them in braces. For example, `x_ij` will appear as x_ij, while `x_{ij}` will appear as x_{ij}.
- Integral signs are obtained by the command `\int`. You put limits on an integral by treating the upper limit as a superscript and a lower limit as a subscript. The integral `\int_1^\infty dx/x^2 = 1` becomes $\int_1^\infty dx/x^2 = 1$. Note that the infinity sign (∞) is obtained using `\infty`. Also note that LaTeX knows to use different sizes of symbols depending on the situation. This happens automatically.

- LaTeX knows a whole lot of functions (like cos for cosine in the first item of this list). You can probably figure out most of them, like `cos` for cosine. The obvious command works just about every time. You can even use `lim` and `det` and other abbreviations. Feel free to explore. Later I will show you how to create your own.
- In math mode, all spaces and line breaks are ignored. You can put them in to help you reading, but LaTeX will space things out its own way.

Fractions and square (and other) roots

One less-than-obvious operation that LaTeX does is to create nice-looking fractions. Of course, you can write two-thirds as $2/3$, but it looks so much classier to write it as $\frac{2}{3}$.

It is actually easy to make fractions, once you know the trick. You have to be in math mode first (instructions follow). Once there, the operation is `\frac{<numerator>}{<denominator>}`, where you fill in <numerator> and <denominator>. So, for example, $\frac{2}{3}$ is obtained by `\frac{2}{3}`.

Square roots are even easier. Again, you have to be in math mode, but you simply use this: `\sqrt{<argument>}`, where you again fill in with the thing you want the square root of.

If you want cube roots, or other varieties of roots, you can add an optional argument, meaning it is enclosed in square brackets, and appears between the `\sqrt` and the `{<argument>}` this way. For the cube root of 7, for example, you would type `\sqrt[3]{7}`, and you would get $\sqrt[3]{7}$.

Combining roots and fractions is not at all uncommon:

$$\int \frac{1}{\sqrt{1-x^2}}\,dx = \text{Arcsin } x + C.$$

Inline formulas

Inline math mode is entered and exited by using a single $. That is, in order to typeset the Pythagorean theorem in inline mode earlier, I typed `$a^2 + b^2 = c^2$`. Remember that spaces are ignored, and you would get exactly the same output by typing `$ a ^ 2 +b^2= c^ 2$`. But be nice to yourself, and don't do that.

Note that this means you can't use a regular $ in text mode, since it is deleted

and puts LaTeX into display math mode. You can get LaTeX to typset a $ (obviously), but you have to use the sequence `\$`.

Display formulas

Any material starting after `\[` up until `!\]!`is typeset in display mode. So the display math mode for the Pythagorean theorem earlier way obtained using `\[a^2 + b^2 = c^2\]`.

1.4.2 Lists and tables

LaTeX is very useful for organizing material. One general idea is bulleted lists, as in the list of things that change in math mode on the previous page. The other idea is the table. Both of these are very flexible, but I am only going to go over the basics here. More information will come later.

Lists

There are two main lists: bulleted and numbered. Their formats are very similar; it is only the beginning and ending statements that change.

A bulleted list starts with `\begin{itemize}` and ends with `\end{itemize}`. A numbered list starts with `\begin{enumerate}` and ends with `\end{enumerate}`. These is called, as you might expect, list environments.

In each case, you have zero or items that each begin with `\item`.

That's about all you need. The text

```
\begin{itemize}
\item First thing to know
\item Another thing to no
\item Last thing to gno
\end{itemize}
```

produces the output

- First thing to know

- Another thing to no
- Last thing to gno

Similarly, the text

```
\begin{enumerate}
\item First thing to know
\item Another thing to no
\item Last thing to gno
\end{enumerate}
```

produces the output

1. First thing to know
2. Another thing to no
3. Last thing to gno

Lists always begin on a new line, and the text following a list starts on a new line. If you want to start a new paragraph after the list, leave an extra blank line, as this paragraph illustrates.

Tables

Tables are somewhat more complicated, because there are so many more things you can adjust with a table.

Tables always begin with the `\begin{tabular}` command and end with the `\end{tabular}` command, creating the tabular environment.

The `\begin{tabular}` command has a required parameter, indicating how many columns the table has, and how text in each of the cells in a column should be justified. The basic indicators of justification are `r, c, and l`, which will right-justify, center, and left-justify the text in that column, respectively. The most common approach is to left-justify text and right-justify numbers, but that is hardly required. And sometimes centering text just look really nice.

Another character is handy to put into the required parameter of `\begin{tabular}` is |, called a vertical line or a pipe (in computer lingo, anyway). It does not create a column, but rather allows you to put vertical lines in between corresponding columns.

Finally, you need to know how to enter the rows of the table. You enter the table one row at a time, with the different cells of the table separated by &. You end the row by putting in a \\. (The \\ is handy in general. It creates a line break in the body.) You are allowed to put most anything in the cell of a table, including another table, but that is just abusive.

Finally, it is handy on occasion to draw a line across all the rows of the table. This is done using the command `\hline` *after* the \\ that ends the line.

Here is an example. Suppose you want to start a table of powers of the integers from 1 to 5, say the second, third, fourth, and fifth powers. You would have five columns, and they would most likely be right justified. That would mean starting the table with `\begin{tabular}{r|rrrr}`, where the pipe would be used to separate off the numbers that are being raised to powers. However, to show what happens with different characters in the required argument, I will use `\begin{tabular}[r|crlc}`. Next, the rows will fill in. The end result will be this:

```
\begin{tabular}{r|crlc}
$n$ & $n^2$ & $n^3$ & $n^4$ & $n^5$ \\ \hline
1 & 1 & 1 & 1 & 1 \\
2 & 4 & 8 & 16 & 32\\
3 & 9 & 27 & 81 & 243\\
4 & 16 & 64 & 256 & 1024\\
5 & 25 & 125 & 625 & 3125
\end{tabular}
```

which produces the result

n	n^2	n^3	n^4	n^5
1	1	1	1	1
2	4	8	16	32
3	9	27	81	243
4	16	64	256	1024
5	25	125	625	3125

There is a common catch. A table does *not* automatically begin on a new line, nor does the material after it begin on a new line. Typically, you want to separate a table from surrounding text by blank lines, or at least \\ before and after.

1.4.3 Other mathematical symbols

LaTeX can typset just about anything mathematical you might ever encounter. Several of them (such as matrices) will require more work, but there are a few other things that are simple and common enough to include here.

Here is a table of common things you might include in formulas. Note that all of these operate only in math mode.

$\pm$	`\pm`	$\times$	`\times`	$^\circ$	`{}^\circ` (degree sign)
$\cdot$	`\cdot`	$\cap$	`\cap`	$\cup$	`\cup`
$\leq$	`\leq`	$\geq$	`\geq`	$\subset$	`\subset`
$\subseteq$	`\subseteq`	$\supset$	`\supset`	$\supseteq$	`\supseteq`
$\in$	`\in`	$\approx$	`\approx`	$\sim$	`\sim`
$=$	`=`	$\neq$	`\neq`	$'$	`'` (used for prime or derivative)
$\forall$	`\forall`	$\exists$	`\exists`	$\sum$	`\sum`

Then there are the functions that you might want. (I will show you how to define your own later.) The meanings of each are obvious, if you have had the relevant courses.

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

1.5 Sectioning commands

You use sectioning commands to split up the document into related pieces, using section titles, in a way that would allow it to collapse into an organized outline form if you wanted.

Thinking through sectioning commands is a great way to have to think about the overall structure of what you are dealing with. It pulls you out of tunnel vision, and makes you think more "big picture" which is a huge benefit. I recommend most highly using sectioning commands in your work!

Each of the sections has its own display style, but the syntax is the same. The easiest way to see it is to look at it:

A section

Text that follows `\section*{A section}`.

A subsection

Text that follows `\subsection*{A subsection}`.

A subsubsection

Text that follows `\subsubsection*{A subsubsection}`.

A paragraph Text that follows `\paragraph*{A paragraph}`.

A subparagraph Text that follows `\subparagraph*{A subparagraph}`

Note the * that follows each of those commands, called the starred form of the command. That generally has the effect in most commands of preventing the printing of some default text. In this case, normally, the section through subsection commands normally are numbered or labeled. You can see it here by removing the star:

```
\section{Dummy section}
\subsection{Dummy subsection}
\subsubsection{Dummy subsubsection}
```

produces

1.6 Dummy section

1.6.1 Dummy subsection

Dummy subsubsection

Using the starred form of these commands also prohibits that sectioning label from appearing in the table of contents, if you decide to create one, and those commands don't affect the numbering of non-starred sectioning commands.

As with just abour everything else in LaTeX the formats of the section numbering can be changed, and often is. But that is for later, or maybe never.

1.7 For more information

There are a host of resources available for LaTeX, many of them online. Google is your friend here.

If you want actual books (what is sometimes referred to disparagingly as "dead trees"), there are only two that you really need, despite an overwhelming number of the ones that have been published. The two you want are both published by Addison Wesley:

- *LaTeX: a document preparation system*, 2nd edition, by Leslie Lamport. This is the guy that built LaTeX. (Donald Knuth built TeX, and Lamport built on that.)
- *The LaTeX Companion*, by Michel Goosens, Frank Mittlebach, and Alexander Samarin. This is the definitive guide to all the packages that are out there, with detailed instructions on how to use them.

There are others, such as *The LaTeX Graphics Companion* that are useful when you are dealing with such things. Stick with Addison Wesley publisher stuff. They have wrapped up the best of the LaTeX authors.

Chapter 2

Maple and LaTeX

Maple output into LaTeX

It might surprise you to find out that Maple use LaTeX in its default math output. That is, Maple uses LaTeX to format what it prints back to you. This makes it natural to be able to export LaTeX from Maple and into your LaTeX document. It is not as easy as it sounds, because the Maple people have very distinct ideas about how math formulas should display, and regularly jigger with it.

There are two varieties of things that you might want to get from Maple into your document: the results of computations (including the commands that gave the results, so Dr. Coulliette can check you did them right), or you might create a graphic that you want to include, such as a nifty heart-shaped graph (that could show up on a Math Modeling T-shirt). I will deal with the text first.

Porting a Maple output into LaTeX

As the first step in importing LaTeX into your document, you have to use what are called style files created by Maplesoft. The style files create commands that the exported Maple LaTeX will use. And that becomes the first hurdle.

Maple will install those style files, but not where LaTeX expects to find such things. That means you have to find them, and then copy them to a usable place. But you also have to be careful, since the style files are different for different versions of Maple. You have to use the style files for the Maple version

you are using.

On Microsoft® Windows® you can find the files in the `\etc\` subdirectory of your installation. On a Mac, you can find the files in the `/Library/Frameworks/Maple.framework/Versions/Current/etc/` directory.

Files? Yes, you will need more than one. There are actually eight of them. And you are likely to need all eight. There are methods of moving the files (or pointers to them in real operating systems, like Mac OS X) to where LaTeX will know they are there. In Microsoft® Windows®, this is harder. You are probably best off copying the files to a folder on your desktop and moving those files to whatever directory you keep the LaTeX file in. The Microsoft® Windows® location for Maple style files to go is `C:\Program Files\MiKTeX\tex\latex\base`.

However, that is not the end of the situation. You can move the files all to the place they belong, run LaTeX, and discover that LaTeX tells you that it can't find the files. This is even more puzzling when you notice that other files in that same place *can* be found by LaTeX.

The reason is simple, once you know what it is. (Until then, it can drive you nuts.) When LaTeX wants to locate an internally-used file (which `\usepackage` requires), it doesn't go search the directory, which is what you would expect. It goes to a file that is a directory listing of all files in the LaTeX system. This turns out to be much faster, and is the reason that it is used. But here is the problem: just moving the `maplestd2e.sty` file into the right directory does not update that listing file for the directory. That is, the file can be in exactly the right place, but LaTeX will not find it unless the listing file indicates that it is there. The solution is simple: update the listing file.

In Mac and Linux systems, the file is called `ls-R`, named after the command that generated the file. (`ls` is the "list" command in Mac and Linux. The `-R` tells it to `R`ecursively list all the files in all subdirectories.) That makes reconstructing the listing file easy on Mac and Linux. All you do is go to the directory with that file in it and type the command `ls -R >ls-R` which runs the command `ls -R` (with a space in it) and sends the output to (that's the `>`) the file `ls-R`. Whatever used to be in the file is overwritten, but that is what you want to do in this case.

On Windows systems, there is a similar thing, since a LaTeX file with `\usepackage{maplestd2e}` will fail to find `maplestd2e.sty`. I am still trying to track down where the file that lists all currently-known files is.

Including Maple sessions

That is the first of several hard parts. The easy part is getting Maple to output the LaTeX version of its worksheet. Click on `File` in the menu bar, then click on `Export as` , and select `LaTeX` from the drop-down list (the second option). You will then be presented with a dialog box asking for the name of the new file. Maple will add `.tex` at the end of the file name for you. Select the destination directory, click on `Save`, and the rest is done for you.

After you have verified that you have the output that you want (even if it is a bit odd; see the example that follows), you have to edit portions of the Maple output into your document. Specifically, there will be preamble commands (things that start with `\usepackage` or `\def`) from the preamble of the Maple document that you will want to copy directly into your preamble. There are also a long list of `DefineParaStyle` and `DefineCharStyle` commands at the beginning of body. They also need to be put into your document. I usually put them in the preamble also, since it cleans up the body of the document. (Some things can go either place.) You do *not* want to include `\pagestyle{empty}` command in your document, at least at this point.

You then want to take things in between the `\begin{maplegroup}` and `\end{maplegroup}` commands (near the top and bottom of the body). If you don't want all of that, you can eliminate pieces between (and including) `\begin{mapleinput}` and `\end{mapleinput}`, or between (and including) `\begin{maplelatex}` and `\end{maplelatex}`. It is usually easy to see what it is. As more of a hint, the `mapleinput` sections are the commands you typed (sort of, see later), while the `maplelatex` parts are the output sections, what Maple types back as answers.

Let me give you an example. First, here is a rough example of what I typed into Maple 15 using the classic worksheet, and the corresponding output (less the colors, since this will probably be printed off in black and white):

> *#This is a comment.*
> *diff(cos(x), x)*

$$-\sin(x)$$

> $Int\left(\frac{1}{1+x^2}, x = 0 \,..\, infinity\right)$

$$\int_0^{\infty} \frac{1}{1+x^2}\, dx$$

$> \% = value(\%)$

$$\int_0^\infty \frac{1}{1+x^2}\,dx = \frac{1}{2}\,\pi$$

I figured that this would at least illustrate some of the things that Maple might do. One slick trick from Maple is the `Int` command, called the inert form. It simply displays the integral without trying at all to evaluate it. The command `value` causes Maple to evaluate it. So, the combination `% = value(%)` has the nifty property of displaying both the integral and the value it has.

I was expecting something that looked like this when I exported the Maple sheet. Wrong. Here is what I got:

$\frac{d}{dx}\cos(x)$

$$-\sin(x)$$

$\int_0^\infty \left(x^2+1\right)^{-1} dx$

$$\int_0^\infty \left(x^2+1\right)^{-1} dx$$

$\int_0^\infty \left(x^2+1\right)^{-1} dx = 1/2\,\pi = value\left(\int_0^\infty \left(x^2+1\right)^{-1} dx = 1/2\,\pi\right)$

$$\int_0^\infty \left(x^2+1\right)^{-1} dx = 1/2\,\pi$$

Ick. Maple clearly did not do what I expected. I will be investigating this some more. Hopefully, I can figure out how to get the Maple LaTeX to mimic the inputs better.

Including Maple graphics

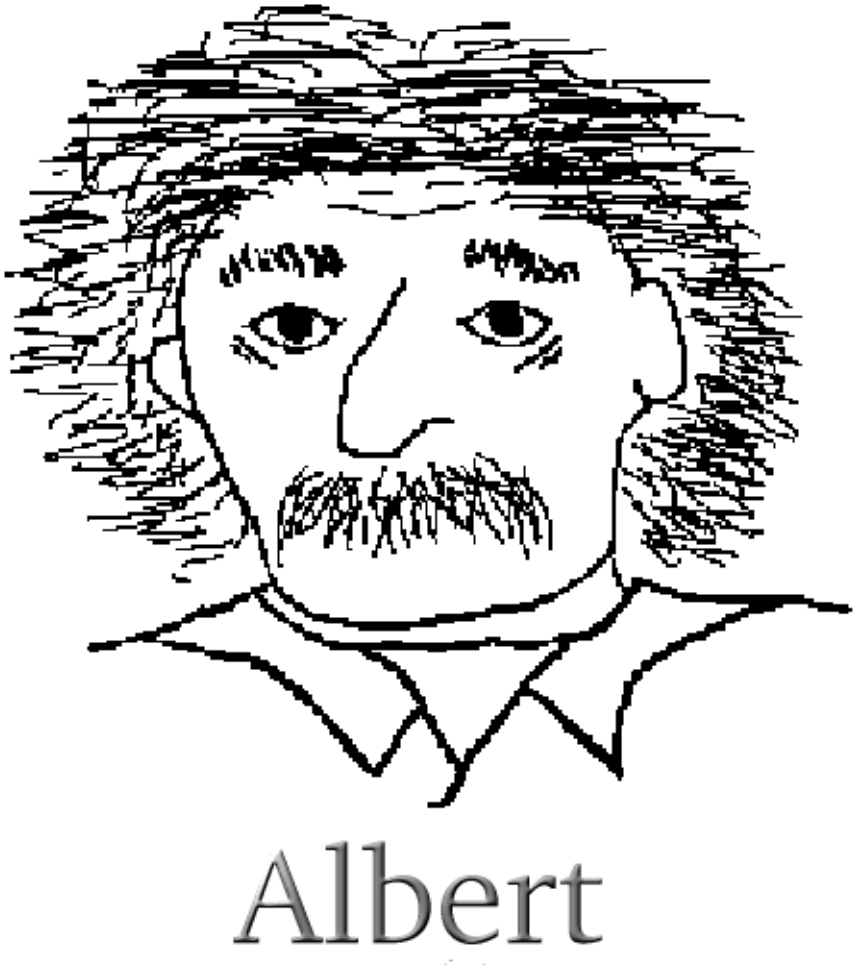
Okay, in the previous incarnation of this section, I made things far more complicated than necessary, mainly because I had used the wrong syntax for the simple way to do things, and my remedy then was worse than the disease.

Incorporating graphics, when done correctly, isn't all that bad, after all, if you follow a few fairly simple guidelines.

First, you need to put into the preamble (the section before `\begin{document}`) the package that will enable the interpretation of the graphics file and convert it into a format that LaTeX can cope with. There are a number of them, but the right one is `graphics`. Therefore, the right command is

```
\usepackage{graphicx}
```

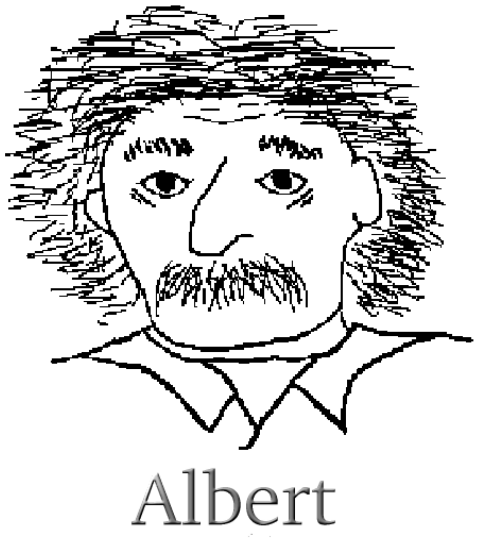
Once you have done that, you need only put in the command to include the graphic. The command is `\includegraphics`. Here is an example, taken out of the calculus I text:



The command that did that was `\includegraphics{albert.png}`. The name `albert.png` is what I had saved the drawing under.

There are clearly problems here. The graphic is a tad large. Well, there are ways of fixing that. The easiest is with an optional parameter to the `includegraphics` command. Remember that optional parameters are included

between the command and the required parameters (the ones enclosed with { and }). In this case, the easiest one to use specifies the width of the output graphic. Here is the command to specify the size:
`\includegraphics[width=200pt]{albert.png}`, which produces the output



It is useful to note that the default width of a LaTeX page is 350 points (put the 350pt inside the square brackets), making this exactly match the width of the text.

There are a variety of parameters you can use. For example, you can do things like

with the parameters `[totalheight=80pt,angle=75]`. Note that the angle is specified in degrees.

Note that I had to have the graphic in the same directory as this file, or it would not have worked. That is not a requirement. In fact, if there are a lot of graphics, then it is handy to have a separate directory for them. If I create

a `Graphics` subdirectory (which I have done), and move a copy of the cartoon there, then the new command is `\includegraphics{Graphics//albert.png}`. Note that it is safer to use `//` instead of just `/` to specify a subdirectory.

The next problem is that not all graphic formats are supported by `graphics`. Fortunately, the best ones are: `.png` and `.pdf`. But that doesn't mean that these are the most common ones; they aren't. However, there is help. A spectacular set of programs called `ImageMagick` is out there and will let you translate between any two formats you have ever heard of (and a lot that you have not heard of!). So, if you have `albert.eps` and you want `albert.png`, you download the `ImageMagick` package from the Internet (It is FREE!), and issue the command `convert albert.eps albert.png`, wait about a second, and the new file appears, in the format you want!

Maple graphics frequently are `.eps`, so this is more likely to be needed than you might think.

Chapter 3

More math tricks

Ok, so now you can make LaTeX formulas, but they don't look so good. This chapter is about how to polish them up. It is not essential, but the point of LaTeX is to make beautiful mathematical papers, so we really do need to look at these.

3.1 Tricks with delimiters

Some delimiters, like `\sqrt{}` automatically adjust their size to fit the insides, as well as center themselves themselves vertically when possible:

$$\sqrt{2} = \sqrt{\frac{4}{2}} = \frac{\sqrt{4}}{\sqrt{2}} = \frac{2}{\sqrt{2}}.$$

But most delimiters, like simple parentheses, don't. That makes for really funny-looking fractions, for example:

$$\frac{(\frac{a}{b})}{(\frac{c}{d})} = (\frac{a}{b})\,(\frac{d}{c}) = \frac{a\,d}{b\,c}.$$

What you really want is a version of parentheses (and brackets, and braces, and all the others) that automatically adjusted size. There is a way to do that.

Scaled delimiters must come in pairs, the first one introduced by a `\left`, and the last one introduced by a `\right`. The commands are followed immediately by the delimiter you want to scale. Here is an example comparing the two

approaches:

$$\left(\frac{a}{b}\right) = (\frac{a}{b}),$$

obtained using the LaTeX code: `\[ \left(\frac{a}{b}\right) = (\frac{a}{b}), \]`

You do have to be careful to pair up `\left` and `\right`. On occasion, it seems necessary that they both be on the same input line. I haven't quite figured that one out yet, because sometimes it doesn't have to.

3.2 Tricks with integrals

Integrals are an important part of advanced mathematics. LaTeX handles them really well.

3.2.1 Integrals in calculus

Indefinite integrals are simple in LaTeX. You just put in the `\int` command, and the integral appears. If only integration were always that easy!

Definite integral evaluation

Suppose you want to illustrate the evaluation of a definite integral, such as integrating x^2 from $x = 2$ to $x = 5$. Setting it up isn't hard: `\int_2^5 x^2\,dx`. (The `\,` adds a bit of space between the x^2 and the dx. I talk about spacing at the end of this chapter.) It looks like this: $\int_2^5 x^2\,dx$. To figure out the integral, you integrate, getting $\frac{1}{3}\,x^3$, which you evaluate between $x = 2$ and $x = 5$. But writing that out is not clear in LaTeX. Here's how it goes. You use auto-sizing delimiters. The thing is that the evaluation bar that follows the $\frac{1}{3}\,x^3$ is only after the expression, and not before. It looks like using auto-sized delimiters won't work. Well, there is a special case to deal with that situation. If you use either `\left.` or `\right.` (that is, you seem to be scaling a period as a delimiter), LaTeX treats it as a scaled delimiter for pairing purposes, but also treats it zero-sized delimiter. (That's the secret here: Pairing delimiters only means pairing the `\left` and `\right` commands; you don't have to use the same delimiter in both commands.) So, here is a complete typesetting of evaluating that integral:

$$\int_2^5 x^2\,dx = \frac{1}{3}\,x^3\Big|_2^5 = \frac{125}{3} - \frac{8}{3} = \frac{117}{3} = 39.$$

This is coded as `\[ \int_2^5 x^2\,dx = \left.\frac{1}{3}\,x^3\right|_2^5 = \frac{125}{3} - \frac{8}{3} = \frac{117}{3} = 39. \]`

Multiple integrals

There are two different varieties of integrals in higher dimensions. One is called the multiple integral, as in $\iint_R f$. The other is called an iterated integral, as in $\int_0^1 \int_{-1}^1 f(x,y)\,dy\,dx$. You evaluate a multiple integral by setting up and evaluating the corresponding iterated integral.

Double and triple integrals are easily obtained in either of two ways. The obvious one is to use `\int\int` or `\int\int\int`. But the spacing between the integral signs doesn't look right that way. To fix that, the American Mathematical Society (the highest level academic mathematical society, usually abbreviated AMS) came up with a set of LaTeX symbols covering this and oodles of other situations. You can get them by loading the right package: `\usepackage{amsmath}`. Put this in the preamble of your document. Then instead of `\int\int` $= \int\int$, you have `\iint` $= \iint$ and instead of `\int\int\int` $= \int\int\int$, you have `\iiint` $= \iiint$. You even get `\idotsint` $= \int\cdots\int$ for when you are integrating of an unknown or large number of dimensions.

3.2.2 Integrals in complex analysis

In complex analysis, you have two more varieties of integral notations. One of them is the contour integral, `\int_\gamma f(z)\,dz` $= \int_\gamma f(z)\,dz$. That isn't bad; you just use a subscript.

But there is another possibility, and that is the closed contour integral, which is `\oint_\gamma f(z)\,dz` $= \oint_\gamma f(z)\,dz$.

3.3 Tricks with dots

As with a lot of what we had before, there are multiple ways of accomplishing the same thing. However, not all the same things look professional. And LaTeX really does want to help you produce professional-looking papers.

3.3.1 Horizontal dots

Lowered dots

Probably the simplest, and most common, use of dots is called an ellipsis, usually written .... That is *not* obtained by putting a bunch of periods together: .... produces .... The right way to do that is with the command `\ldots`, which produces |...| (where I put in beginning and ending | characters so you could see how it actually is).

The command `\ldots` is one of the few that actually works in either text or math modes, which is fortunate since it is useful in both modes. See the next section.

Centered horizontal dots

There are two forms of an ellipsis, at least in mathematical usage. If you are simply omitting an unknown, or large, number of terms in a sequence, you would use lower dots, as in $x_1, x_2, \ldots, x_N$. But if you were adding those terms together, you would use dots that are centered vertically: $x_1 + x_2 + \cdots + x_N$. The code that does that is `\cdots`, and only is valid in math modes.

When do you use which one? It is a matter of style, but I will use `\ldots` when the punctuation between terms is on the bottom level of the line, as in the commas between $x_1, x_2, \ldots, x_N$. And I will use `\cdots` when the punctuation between terms is centered, as in the addition sign between $x_1 + x_2 + \cdots + x_N$. At least, that makes sense to me. If there is no punctuation between the terms, as in $\int \cdots \int$, they should generally be centered.

3.3.2 Vertical and diagonal dots

There are times that your lists are vertical, as in matrices (which we will get to next), and you might wish to have dots that go up and down. Well, LaTeX is here to fulfill all your desires (well, except the social ones; Dr. C deals with that personally). There is a command that you can probably guess the name of (`\vdots`) that does exactly that: $\vdots$. Again, it is defined only in math mode.

Similarly in matrices, it is also handy to have diagonal dots, which are (again as you might have guessed), `\ddots` = $\ddots$.

3.4 Tricks with arrays and matrices

There specific environments of tables that are used only in math mode, either arrays or matrices. The two are similar. Array environments require (as in the tabular environment) you to specify the alignment (`c, l, r`) of each column at the top of the array, as in `\begin{array}{rlcr}`, which would be followed by the rows (maximum of four columns in each row) separated by `&`, and ended with `\\`. You do not need to specify the number of rows in advance. You end the array with `\end{array}`. For example, here is an array with the code that gives it::

$$\begin{array}{rlcr} a+b+c & d-e/f & g^h & i_{jk} \\ z-y-x-w & v*u & t\,s\,r & o^{p^q} \end{array}$$

```
$\begin{array}{rlcr}
a+b+c & d-e/f & g^h &i_{jk}\\
z-y-x-w & v*u & t\,s\,r & o^{p^q}
\end{array}$
```

Now, as you look at that, you might feel that you want more space between the lines, so you can add a `\medskip` before the first `\\` and get this:

$$\begin{array}{rlcr} a+b+c & d-e/f & g^h & i_{jk} \\ \medskip z-y-x-w & v*u & t\,s\,r & o^{p^q} \end{array}$$

Much better!

3.4.1 Choosing your delimiters

If you want matrices to look like what you use in linear algebra (or applied math!), you want enclosing parentheses or brackets. The array environment does not provide them for you (sometimes you don't want any, which makes this handy), so you have to put them in yourself. For arrays, you will almost always want the auto-sizing delimiters. To put parentheses around the matrix above, you would put `\left(` before the `\begin{array}` and `\right)` after the `\end{array}`. Here is what you would then get:

$$\left(\begin{array}{rlcr} a+b+c & d-e/f & g^h & i_{jk} \\ z-y-x-w & v*u & t\,s\,r & o^{p^q} \end{array} \right).$$

Of course, you could use `\left[` and `\right]` to get square brackets around the array.

3.4.2 Matrix environments

If you put the `\usepackage{amsmath}` command in the preamble of your document (which I recommend, for the large number of things that I tend to assume

nowadays), then you get access to the `matrix` environment. It is much like the `array` environment, except you don't have to specify the number of columns or their horizontal orientation. Instead, you can have up to 10 columns, centered. (You normally ought to use centered columns anyway, so this is not a big problem.)

There are a variety of `matrix` environments, with a built-in first character that tells you want delimiter it automatically provides. Here is what you can get:

Command	Delimiter
`matrix`	None
`pmatrix`	Parentheses
`bmatrix`	Square brackets
`vmatrix`	Vertical lines (as in absolute values)
`Vmatrix`	Pairs of vertical lines (as in norms)

That makes it easy to get a variety of matrix forms with moderately little hassle:

$$\begin{matrix} 0 & -i \\ i & 0 \end{matrix} \qquad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \begin{vmatrix} 0 & -i \\ i & 0 \end{vmatrix} \qquad \begin{Vmatrix} 0 & -i \\ i & 0 \end{Vmatrix}$$

There is also an version of the `matrix` environment designed for inline formulas. It is called `smallmatrix`. The code
`$\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$`
produces $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ inline. It is handy for (obviously) small matrices and saves a lot of space.

3.5 Tricks with sequences of equations

Math notes are not much of anything without loads of equations, often line after line of them. LaTeX is very good at doing things with them. We have already covered inline and display formats, but there are a number of environments for dealing with bunches of them, depending on what you want.

3.5.1 Running equations

If you have a string of equations, such as what you would get if you were to work out "the long way" the derivative of $f(x) = x^4$, protocol says that the equal signs should all line up. The `align` environment does that for you. You get to determine where the alignment point is, by placing a `&` there, meaning that you can line up equal signs, or inequalities, or approximations all with this

environment. You end lines with the usual \\.

$$
\begin{aligned}
f'(x) &= \lim_{\Delta x\to 0}\,\frac{(x+\Delta x)^4 - x^4}{\Delta x}\\
&= \lim_{\Delta x\to 0}\,\frac{(x^4 + 4\,x^3\,\Delta x + 6\,x^2\,(\Delta x)^2 + 4\,x\,(\Delta x)^3 + (\Delta x)^4) - x^4}{\Delta x}\\
&= \lim_{\Delta x\to 0}\,\frac{4\,x^3\,\Delta x + 6\,x^2\,(\Delta x)^2 + 4\,x\,(\Delta x)^3 + (\Delta x)^4}{\Delta x}\\
&= \lim_{\Delta x\to 0}\,\frac{\Delta x\,(4\,x^3 + 6\,x^2\,\Delta x + 4\,x\,(\Delta x)^2 + (\Delta x)^3)}{\Delta x}\\
&= \lim_{\Delta x\to 0}\,4\,x^3 + 6\,x^2\,\Delta x + 4\,x\,(\Delta x)^2 + (\Delta x)^3\\
&= 4\,x^3
\end{aligned}
$$

This was produced with the LaTeX code:

```
\begin{align*}
f'(x) &= \lim_{\Delta x\to 0}\,\frac{(x+\Delta x)^4 - x^4}{\Delta x}\\
 &= \lim_{\Delta x\to 0}\,\frac{(x^4 + 4\,x^3\,\Delta x + 6\,x^2\,(\Delta x)^2 +
 4\,x\,(\Delta x)^3 + (\Delta x)^4) - x^4}{\Delta x} \\
 &= \lim_{\Delta x\to 0}\,\frac{4\,x^3\,\Delta x + 6\,x^2\,(\Delta x)^2 +
 4\,x\,(\Delta x)^3 + (\Delta x)^4}{\Delta x}\\
 &= \lim_{\Delta x\to 0}\,\frac{\Delta x\,(4\,x^3 + 6\,x^2\,\Delta x + 4\,x\,(\Delta x)^2
 + (\Delta x)^3)}{\Delta x}\\
 &= \lim_{\Delta x\to 0}\,4\,x^3 + 6\,x^2\,\Delta x + 4\,x\,(\Delta x)^2 + (\Delta x)^3\\
 &= 4\,x^3
\end{align*}
```

Note that you don't need to put the `align` environment into math mode. It does that automatically.

Also note the asterisk (star) after the `align`. It is there to prevent LaTeX from numbering all the equations, which it will automatically do if you leave off the asterisk. This is preferred behavior in most cases, but not in this document (and in other places as well). And, the `\end` must also contain `align*`. The environment names must always match precisely.

3.5.2 Really long equations

There are times (like when you are writing a calculus text :) when you have to type equations that simply don't fit on a single line. The solution to that depends on the situation.

If you are writing a single equation, and it just happens to be long, the remedies are (in order of preference):

1. Break the line (that is, put in a \\) right before the equation, which will cause it to move to the left margin. Hopefully, that will allow the whole equation to fit on one line.

2. If that doesn't work, remove the line break, and try to figure out how much of the equation will fit on the line, and insert a line break within the equation itself to prevent it from overflowing the right margin. Note, however, that if you use \\ within a math environment, it is completely ignored. That means you have to finish the math mode, issue the line break, and then resume the math mode for the rest of the equation. This might need to be done several times.

3. If for some reason you can't do that (which can happen if you have, for example, a very long quotient rule that you can't break), see if you can rewrite the equation into a form that is more open to line breaks, and put the breaks in at those points.

4. Figure out something else to do. (Crying is not one of them. :)

There is also the possibility of getting some really long expressions in the `align` environment. (Think Taylor series; you will see them a lot in Applied Math!) That is trickier, but LaTeX is up to it! You can use another environment *within* the `align` environment, called `split`.

Suppose you want the Maclaurin series of $(1 - e^x \log(1 + x))^{-1}$ up to order 20. (Dr. Coulliette could easily ask this!) Of course, Maple is the only way to do that, and the command
`series(1/(1-exp(x)*log(1+x)), x=0, 20);` gives

```
1+x+(3/2)*x^2+(7/3)*x^3+(41/12)*x^4+(619/120)*x^5+(1841/240)*x^6+
(3622/315)*x^7+(961/56)*x^8+(3105559/120960)*x^9+(27838361/725760)*x^10+
(71542483/1247400)*x^11+(10264397621/119750400)*x^12+
(797921076263/6227020800)*x^13+(340753100459/1779148800)*x^14+
(23399185417913/81729648000)*x^15+(6217902153049/14529715200)*x^16+
(227529713319579049/355687428096000)*x^17+
(185510675194873849/194011324416000)*x^18+(
21733279474963789151/15205637551104000)*x^19+O(x^20)
```

No joy there. How would you write that in LaTeX?

$$\begin{aligned}\frac{1}{1-e^x\,\log(1+x)} = 1 &+ x + (3/2)*x^2 + (7/3)*x^3 + (41/12)*x^4 + (619/120)*x^5 + (1841/240)*x^6 + \\ &(3622/315)*x^7 + (961/56)*x^8 + (3105559/120960)*x^9 + (27838361/725760)*x^{10} + \\ &(71542483/1247400)*x^{11} + (10264397621/119750400)*x^{12} + \\ &(797921076263/6227020800)*x^{13} + (340753100459/1779148800)*x^{14} + \\ &(23399185417913/81729648000)*x^{15} + (6217902153049/14529715200)*x^{16} + \\ &(227529713319579049/355687428096000)*x^{17} + \\ &(185510675194873849/194011324416000)*x^{18} + \\ &(21733279474963789151/15205637551104000)*x^{19} + O(x^{20})\end{aligned}$$

The code for this is

```
\begin{align*}
\begin{split}
\frac{1}{1-e^x\,\log(1+x)} &=
1+x+(3/2)*x^2+(7/3)*x^3+(41/12)*x^4+(619/120)*x^5+(1841/240)*x^6+\\
&\qquad (3622/315)*x^7+(961/56)*x^8+(3105559/120960)*x^9+(27838361/725760)*x^{10}+\\
&\qquad (71542483/1247400)*x^{11}+(10264397621/119750400)*x^{12}+\\
&\qquad (797921076263/6227020800)*x^{13}+(340753100459/1779148800)*x^{14}+\\
&\qquad (23399185417913/81729648000)*x^{15}+(6217902153049/14529715200)*x^{16}+\\
&\qquad (227529713319579049/355687428096000)*x^{17}+\\
&\qquad (185510675194873849/194011324416000)*x^{18}+\\
&\qquad (21733279474963789151/15205637551104000)*x^{19}+O(x^{20})\\
\end{split}
\end{align*}
```

Note that I had to insert `{` and `}` around the exponents before running it through LaTeX. Try leaving them off and see what happens!

3.5.3 Cases

One more really slick trick that LaTeX can do is cases. For example, in your table of integrals, there is the integral $\int x^n\,dx$, which usually will be $(x^{n+1})/(n+1)+C$, except when $n=-1$, in which case it is $\ln x + C$. Here's how to do that with LaTeX using the `cases` environment:

$$\int x^n\,dx = \begin{cases} \dfrac{x^{n+1}}{n+1} + C & \text{if } n \neq -1 \\ \ln x + C & \text{if } n = -1 \end{cases}$$

Here is the code that generated this:

```
begin{align*}
\int x^n\, dx &=
\begin{cases}
{\displaystyle\frac{x^{n+1}}{n+1}+C} & \mbox{if $n\neq -1$}\medskip\\
\ln x+C & \mbox{if $n=-1$}
\end{cases}
\end{align*}
```

Note that I put several tricks here. The `\mbox` command allows you to enter regular text (with the option of math mode text inside) while you are in math mode. Also, the equations seemed a bit too close, so I spread them out a bit using the `\medskip\\` command at the end of the first line of the `cases` stuff. Finally, the integrand in the case $n \neq -1$ was too small for my liking, so I enlarged it using the command `\displaystyle`, which turned it into the size used in display math mode.

3.6 Tricks with vectors

LaTeX is really handy with vectors also! But there are some stylistic features that need to be observed.

3.6.1 Why you shouldn't use $\vec{i}$

The general way of putting a vector sign over a letter is with the `\vec` command, so $\vec{R}$ is written as `$\vec{R}$`. Note that `\vec` requires math mode.

However, you should not use `\vec{i}` or `\vec{j}` for the unit vectors along the x and y axes. The reason is the dot over those letters. It is better to use a "dotless" form, available only in math mode: `\imath` and `\jmath`, which look like $\imath$ and $\jmath$ respectively. Putting the little vector arrow over those gives the right look: $\vec{\imath}$ and $\vec{\jmath}$.

3.6.2 Making your own commands

But one thing about LaTeX is that it is supposed to help you. After all, $\vec{\imath}$ is a lot of typing for something that will be used a lot. You are allowed to create your own commands. Think of them as shortcuts. Here is the syntax:

```
\newcommand{\ivec}{\vec{\imath}}
```

You are telling LaTeX that you want it to replace the thing in the first set of

braces (in this case, `\ivec`) with the thing inside the second set of braces (in this case, `\vec{\imath}`).

Then, with that in place, I can write $\vec{R} = 3\,\vec{\imath}$ by using `$\vec{R} = 3\,\ivec$`. Now, while we are at it, we might as well define `\jvec` and `\kvec`:

```
\newcommand{\jvec}{\vec{\jmath}}
\newcommand{\kvec}{\vec{k}}
```

The `\kvec` is not really too helpful, except that it maintains a symmetry with `\ivec` and `\jvec`. With that, you can write such nifty things as $\vec{R} = a\,\cos(\omega\,t)\,\vec{\imath} + a\,\sin(\omega\,t)\,\vec{\jmath} + b\,t\,\vec{k}$ using the LaTeX code
`$\vec{R} = a\,\cos(\omega\,t)\,\ivec + a\,\sin(\omega\,t)\,\jvec + b\,t\,\kvec$`.

All `newcommand` commands belong in the preamble of a document, just to make sure that you know where to find them. That isn't a LaTeX requirement, but is very good practice.

3.6.3 Dot products

It might seem that defining new commands is useful for making your typing documents easier, and it is. But there are times when it becomes crucially useful, and dot products are one of those times.

You see, mathematicians have two very different notations for dot products, and physicists have a third. Mathematicians with write the dot product of $\vec{v}$ and $\vec{w}$ as either $\vec{v}\cdot\vec{w}$ or $\langle\vec{v},\vec{w}\rangle$. A physicist (especially in quantum mechanics) will often write this as $\langle\vec{v}|\vec{w}\rangle$. (The LaTeX code for these three are, respectively, `$\vec{v}\cdot \vec{w}$`, `$\langle \vec{v}, \vec{w}\rangle$`, and `$\langle \vec{v} | \vec{w}\rangle$`. Note that in the first one, we use `\cdot` rather than the similar `\cdots`, which would produce three dots in a row.)

Finding your own way of writing dot products is fine, but you will most likely encounter a situation where you actually need to write dot products a different way. If you have a Word® document, you would have to go through and change each occurence. and that would be horrible if the paper is of any significant length. Instead, what you can do with LaTeX is to createa command that writes your preferred form of the dot product, and put it in the preamble. Then if you want to change the form, you make one change in the preamble, run LaTeX on it again, and in seconds, you are done! And you know that you kaven't missed any!

So far, the commands we defined did not depend on anything else, but a dot product requires that you write different things inside it. That can be handled

using parameters, in the exact form of required arguments. You will be able to write `\dotprod{\vec{v}}\\vec{w}}` and get the right form of the dot product.

The command for the first form of the dot product would be
`\newcommand{\dotprod}[2]{#1 \cdot #2}`
Let's look at this carefully. The `\newcommand{\dotprod}` part is exactly as before. But then there is the thing that looks like an optional argument, `[2]`. It actually is required, but only when you want to pass parameters to the new command. In this case, you will have to pass two parameters. Then there is the part in the second set of braces, the thing that replaces `dotprod` and its two parameters. You can refer to the parameters inside the new command as `#1` and `#2`. They get put in just as you wrote them as arguments.

This gets more interesting with the second form of the dot product, the one that is used in more advanced mathematics. The command for it would be
`\newcommand{\dotprod}[2]{\langle #1, #2 \rangle}`
After you have done this, you can type the LaTeX command `\dotprod{\vec{v}}{\vec{w}}` and get $\langle \vec{v}, \vec{w} \rangle$. But, later on in Applied Math, you will take the dot product of two *functions*, and you just as well write `\dotprod{f}{g}`.

What happens if you try to define it both ways? LaTeX gets grumpy, since it doesn't know which you want. Don't do that.

3.7 Spacing my way

I have my own idea of spacing in math formulas. I tend to want to spread things out more than most people. For example, I would rather see $3\,x^4\,y^3$ than $3x^4y^3$. So, I am very generous with my use of `\,`, which adds just a bit of extra horizontal space. If you look back at the examples I gave earlier, I simply put that space in almost by reflex.

Vertical spacing is, for me, similar. I don't like to see things mashed together too closely, So, I will normally put `\medskip\\` around quite a bit as well when others don't.

This is just me. Feel free to find the spacing that you like. Just don't expect me to like it. But that's all right; you are just as welcome not to like mine.

Chapter 4

Fonts and other stuff

4.1 A few general ideas

Fonts are complicated, and LaTeX lets to get at all of them. That means that LaTeX fonts, in their full generality, are complicated. As you understand more about LaTeX, they are worth looking into, but not here. One of the better descriptions of what you can do with fonts is in The LaTeX Companion.

Even reasonably simple things degenerate rapidly into a maze of terminology. Let me try to demystify some of it for you.

4.2 Fonts, in brief

4.2.1 Easy-to-get-to fonts

LaTeX generally is about making things easy. And the large majority of things that you will ever want or need are, in fact, simple.

One thing to keep in mind is that when you change the font for a part of a document—a word, a sentence, or even a paragraph—you will want to end the changes. If you just put in a font-changing command (like `\tt`, which changes the font to a monospaced typewriter look), you will almost certainly want to enclose the part you change in braces, `{ }`. So, you might do something like `this for a few words`, which came from the command `{\tt this for a few words}`.

What you probably call fonts are actually *font families*. There are three font families that you will commonly use: roman, `typewriter`, and sans serif. Each one can be accessed in one of two ways: command or declaration. The command form looks like `\texttt{typewriter}`, which is how I created the listing for the typewriter font family. The idea is to use the command form with a required argument that is set in that font family. The declaration form might look like `{\sffamily sans serif}`, which is how I produced the sans serif font above. A declaration changes all the text from that point until the next set of close braces or then end of an environment.

Each of these requires some extra comments. The roman (or serifed) font is the default. Yes, this can be changed, but normally in the preamble. You probably don't want to do that yet. Maybe once you get used to fonts. The commands for this are `\textnormal`, `\textrm`. The declaration is `\rmfamily` or just `\rm`.

Technically, that is the way to put text into math mode expressions. But there are other ways of doing it. I tend to use `\mbox{...}`, but LaTeX also supplies a `\text{...}` command for doing the same thing. You have seen this in the `\cases` envrionment we did earlier.

The `typewriter font` is useful in several places. It tends to look typewritten (for obvious reasons), meaning that it appears a little more personal. It is also a *monospaced* font, meaning that each letter is given the same horizontal space. The default font is *proportional spaced*, meaning letters are only given the horizontal space that they need. Proportional spacing allows things to be typeset closer together, so they tend to typeset to fewer pages, and often look more professional. But there are times that monospaced fonts are the best things to use. Specifically, when you want to align things in columns, such as when you are printing out computer program sections, you want to preserve indentations and column placements. That requires monospaced fonts.

There is one caution, though. If you set a section of text in monospaced font, you will want to turn off right justification, the property that adjusts spaces to make the right-hand side of the text aligned. You can do that with one of two commands: `\raggedright` or `\obeylines`. They do different things. The `\raggedright` will wrap lines when they are full, but will not attempt to stretch those lines out to align the right margins. This is mostly what you will want. The `\obeylines` command forces LaTeX to put in line breaks where the input file puts them. This is very handy in a few situations, such as shopping lists, where you don't want line filling and wrapping.

Typesetting the paragraph two above using the `typewriter font` and ragged right would produce this:

```
The typewriter font is useful in several places.  It tends to look
typewritten (for obvious reasons), meaning that it appears a little
```

```
more personal.  It is also a monospaced font, meaning that each letter
is given the same horizontal space.  The default font is proportional
spaced, meaning letters are only given the horizontal space that they
need.  Proportional spacing allows things to be typeset closer together,
so they tend to typeset to fewer pages, and often look more professional.
But there are times that monospaced fonts are the best things to use.
Specifically, when you want to align things in columns, such as when
you are printing out computer program sections, you want to preserve
indentations and column placements.  That requires monospaced fonts.
```

Note finally that verbatim text is always in typewriter font. This can almost certainly be changed, but I have never seen how, nor have I ever wanted to do it.

The command for typewriter font is `\texttt`, and the declarations are either `\ttfamily` or just `\tt`.

The sans serf font family tends to look more modern. The serifs on a font are the little projections at the ends of letters. It used to be thought that serifs provided more visual clues to letters and therefore were easier to read. This has since been disproven; seriffed and sans serif fonts are read equally fast.

4.2.2 Modifying font families

There are varieties of things that you can do even within a font family, such as making it italic, or bold faced, or different sizes. Tackling this in LaTeX is not as easy as you might think, due to the way that LaTeX grew up.

Bold face fonts

First, let's tackle bold faced fonts. Consider the following lines:

```
{\sf\bf sans serif bold face}\\
{\bf\sf bold face sans serif}\\
{\sf\it sans serf italic}\\
{\it\sf italic sans serif}
```

which produces the following output:

sans serif bold face
bold face sans serif
sans serf italic
italic sans serif

It certainly looks like, when you put two commands together, only the final one has any effect. That is true, if you use those commands. Consider now the commands:

```
{\sffamily\bf sans serif family bold face}\\
{\bf\sffamily bold face sans serif family}
```

which produce the following output:

sans serif family bold face
bold face sans serif family

The last one is right! What is going on? A lot, mostly beyond comprehension. LaTeX used to have a very limited number of fonts, and the declarations `\bf` or `\sf` actually shifted what font was displayed, rather than modifying the font. To an extent, this has continued. But you want to be able to *modify* fonts. The last example gives you some hope that it can be done. And it can. You just have to use the right commands.

While it is correct that both `\sf` and `\sffamily` declarations will change to a sans serif font, the declarations are not equivalent in other aspects. And while `\bf` does produce bold faced fonts, it is not the only way to do that, either. Note the following two lines:

```
{\bfseries\sffamily bold face series sans serif family}\\
{\sffamily\bfseries sans serif family bold face series}
```

which produces the output:

bold face series sans serif family
sans serif family bold face series

At last! Declarations that don't clobber each other!

What is the thing you need to remember here? While you will rapidly get used to `\bf` to make the default text bold faced, you have to be more careful when you want to do the same with other font families. And remember that the `\..series` and `\..family` (and `\..shape`, which I didn't cover here) are the most friendly, but longest-to-type versions of these commands.

Italicizing fonts

Italic fonts are not just fonts that have been tilted over. They actually have a different shape. Consider the difference between "a" (default) "*a*" (slanted) and "*a*" (italic).

So, how do you italicize fonts? Rather than going over all of the same hassles, let me give you the final result, almost.

If you want to italicize default font, the easiest way is to either use the command `\textit`, as in `\textit{italicized!}` which gives *italicized!* or you can use the declaration `{\it ...}`, as in `{\it italics!}`, which produces *italics!*.

There is another, useful command and declaration combination. The normal use of italics is to emphasize text, as in "Well, don't *do* that!" If you are emphasizing a number of lines of text, sometimes there is an extra emphasis in those lines. In that case, standard protocol says that you shift back to the default (called *upright*) text. This makes life painful, but LaTeX has a way out, which I will often use. There is a command `\emph` and a declaration `\em` which italicizes upright text, and reverts to default style text that is in italics.

Size

There are a collection of size-changing declarations:

Command	Example
`\tiny`	Size
`\scriptsize`	Size
`\footnotesize`	Size
`\small`	Size
`\normalsize`	Size
`\large`	Size
`\Large`	Size
`\LARGE`	Size
`\huge`	Size
`\Huge`	Size

Remember that a declaration operates until the environment it is in finishes, or the ending brace for the grouping it is in occurs. That means if you want to make something really small, such as this, you have to use `{\scriptsize this}`. If, instead, you use `\scriptsize{this}` (which is the proper syntax, if `\scriptsize` were a command), you will discover that everything from that word to the end of the file will be really small. Your prof would not be amused.

One limitation of the current version of LaTeX is that there is no way to specify a font size relative to another, that is, to make one font that is 2 points larger than another.

Other modifications

There are two other things you can do to a font family: make them slanted and make them small caps.

The command for slanted fonts is `\textsl` and the declaration is `{\slshape ...}`. Thus, `\textsl{slanted}` and `{\slshape slanted}` produce *slanted*. You can use slanted fonts much the same way that you use italic fonts.

You have probably seen the small cap font. It is used extensively in the Bible (particularly the Old Testament), in the word LORD. What the small cap shape does is replace all lower-case letters with smaller versions of the upper-case letters. Upper-case letters are left alone. The command is `\textsc` and the declaration is `{\scshape ...}`. Thus, both `\textsc{Lord God}` and `{\scshape Lord God}` produce LORD GOD. Now you can impress your OT profs.

4.2.3 More fonts, this time for math

If you remember back in Logic & Sets, one topic that was covered was the power set of a set, the set of all subsets of that set. So, if A is a set, the power set was denoted $\mathcal{P}(A)$.

There are also the standard symbols for the integers, real numbers, and the complex numbers: $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$.

The two extra varieties of fonts are, unfortunately, restricted to dealing only with capital letters. The first is called the math calligraphic font family, the second is called the outline font family. The power set symbol is `$\mathcal{P}$`, and the three sets of numbers are, respectively, `$\mathbb{Z}$`, `$\mathbb{R}$`, and `$\mathbb{C}$`. Note that both fonts can only be used in math mode.

However, note that you can add subscripts or superscripts to these, giving things like the set of positive integers as `$\mathbb{Z}_{+}$`, producing $\mathbb{Z}_{+}$, or 11-dimensional real space as `$\mathbb{R}^{11}$` giving $\mathbb{R}^{11}$. (Note the braces in the exponent. Explain why they are necessary!)

4.3 More font-like things

Okay, here is a really cool thing to do, but only once: $\mathfrak{Fraktur}$, (Fraktur), $\mathfrak{Gothisch}$, $\mathfrak{also\ called\ Textur}$ (Gothisch, also called Textur), and $\mathfrak{Schwabacher}$ (Schwabacher).

Don't use this on anything you intend to turn in to a professor, unless the prof really enjoys reading ancient German texts. (Default answer: No.)

Other things, like the full set of Zapf dingbats are available (probably for a fee), and a huge range of other stuff (that is free): ♠, ♡, ♢, ♣, or lots of other things.

But probably the only other thing that you would actually use on occasion is the "round up to the nearest integer" and "round down to the nearest integer" symbols. These are called the ceiling and floor functions, respectively. When applied to the real number x, you get $\lceil x \rceil$ and $\lfloor x \rfloor$, obtained as `$\lceil x \rceil$` and `$\lfloor x \rfloor$`.

4.4 Some "Real-life" fonts

LaTeX uses a default font called Computer Modern. It isn't bad at all. But there are other font families out there that are more common, like Times New Roman. You can get that (and I actually recommend using it!) by putting into the preamble the series of commands:

```
\usepackage[T1]{fontenc}
\usepackage{textcomp}
\usepackage{mathptmx}
```

Be warned, however! By doing this, you can lose access to various symbols that are part of the Computer Modern font family. If that happens, you might want to remove those lines. In general, the whole topic of fonts is *really complicated*, so I am not going to go into it in more detail here. Yes, you can get back to those symbols, but it is more hassle to explain here than you can imagine.

Chapter 5

Presentations

5.1 Introduction

You will, of course, be interested in using LaTeX in your presentations at various meetings. Since LaTeX does produce PDF outputs, and since Adobe Acrobat reader exists on just about every computer, you might consider projecting your paper while you read it to people. That, however, goes way beyond tacky. You want something with glitz, with sparkle, with sizzle. In short, you want the `Beamer` class.

The `Beamer` class is another possible `documentclass` parameter, which means that it defines and redefines a whole lot of things. But it acts just like another LaTeX document other than that.

5.2 Installation

If you are really lucky, `Beamer` is already installed. In fact, `TeXmaker` comes with `Beamer`. Consider yourself fortunate. I speak from experience.

5.3 Starting with Beamer

As with all presentation software, Beamer creates slides that are projected successively, except that they are called `frames` in `Beamer`. So, successive pages start and stop with the commands `\begin{frame}` and `\end{frame}`.

But there are also larger groupings available, called, as before, `sections` and `subsections`. You still have title pages and table of contents available.

In an appendix, I have included a full `Beamer` presentation that "the docs" put together on the material that we used for our middle-class project. Look it over! I would recommend copying the beginning parts into whatever document that you are preparing and then modifying it adapt it to your situation. There are numerous comments as part of the document (that were there when I got it) that explain a lot about what to do.

5.4 Running with Beamer

The output of Beamer is an Adobe Acrobat PDF file. You have to try it out to believe it. I still marvel at how well it does what it does. Pretty much anything you might want to do with Microsoft® PowerPoint®, you can do just as well with `Beamer`.

Chapter 6

What to do when LaTeX bombs

Maybe this should have been the first chapter, but I didn't want to scare you.

Actually, this is more about when the LaTeX processor quits early because of an error, or produces output that can only be viewed as bizarre. And since this handout focusses around `texmaker`, I will use that for my examples.

6.1 Ending early because of an error

When `texmaker` ends because it can't parse what you mean, it will always print some error message, in red letters at the bottom of the input fie area. (You have by now noticed it, and that what you want to see is all blue. When errors occur, they are put before the blue warnings, so they are easily visible.) There will be the line number on which the error occurred, and some indication of what went wrong. Sometimes, there is no line number (usually because the processor dropped off the end of the file while looking for something), and often the error message is even more incomprehensible than a math professor at his best. In short, you need help.

When `texmaker` does encounter an error, it will not produce any output file. The right side will either be the last successful run of LaTeX or a blank screen if the previous run was clobbered. This can be confusing at first, since it looks as though it worked, but you notice that nothing you just put in made into the output. But the appearance of red letters below the panel that you type in is a

key sign that what you just did had problems.

6.1.1 When you have a line number

When you are fortunate enough to have a line number given, you have a major help in figuring out what went wrong. Sometimes. But please note that `texmaker` lists the line numbers of the file, and a brief description of the error. To go to the error, double click on the error, and the IDE will put the cursor on the error line in the file.

When LaTeX discovers an error in an environment, it normally will only tell you that the error occurred at the exit point of the environment, and not the line number where it actually happened. In that case, you have to look through the environment for the error. Help for that will show up momentarily.

First, if you have a specific line number, you should read the error message, and see if you can make sens of it. Often you can, and correcting the problem is not at all hard. If you can't, your best shot is to go over the line carefully and see if you can spot any problems.

6.1.2 Commenting out your work

Far and away the most general way to locate problems is to comment out increasingly large chunks of your LaTeX code until the error disappears. Then start adding stuff back until the error reappears, commenting out smaller and smaller sections until you can isolate where the error is.

Well, that requires you knowing how to comment. If you go back to section 1.3.4 of these notes, I described comments as beginning with a % sign and running through the end of the line. So, the trick then involves putting in large numbers of % signs at the beginning of lines, which has the effect in `texmaker` of turning the line a medium grey color, to visually set it apart from regular text.

This is what you will need to do within an environment that contains an error. If you have added some new material to a working environment, comment out all that you just added, and see if it still works. (It usually does.) Then remove comments slowly until the error reappears. Again, that will show you where the problem is.

6.1.3 When you don't have a line number

This gets tricky to handle sometimes; `texmaker` just ends, with a nondescript (or no) error message, doesn't produce any output, and stares at you defiantly as if to say "Okay, buzzard breath. I dare you to figure out what happened!" Being insulted by `texmaker` can be really aggravating.

Again, commenting will sometimes help, especially if you had a functional LaTeX document that you have been editing. Comment out parts that you just put in, and see if that helps. If you have a very big file, you can copy large chunks of it to another file (remember to include all the preamble stuff) and run it separately. Which brings up a useful strategy.

6.1.4 Writing LaTeX files in pieces

Even this handout is broken up into chapters, in individual files. Here is the entire body of the handout, between the `\begin{document}` and `\end{document}`:

```
\begin{document}
\maketitle
\vspace*{-40pt}

\input{myguide-01}

\input{myguide-02}

\input{myguide-03}

\input{myguide-04}

\input{myguide-99}

\end{document}
```

I then created files myguide-**.tex (this one is 99, to make sure I remember to keep it last), and as I am writing this, I haven't gotten myguide-05.tex written yet.

Note a few things. I moved up the start of the text by 40 points by using `\vspace*{-40pt}`, where `\vspace` stands for vertical space. (There is also a `\hspace` command for horizontal space.)

The files that I include are `myguide-01.tex` through `myspace-99.tex`. I don't have to include the `.tex` part; LaTeX assumes that is what you are going to use.

In that case, I can help debug things by commenting out entire chapters until I locate the offending chapter. This narrows things down rapidly. It also makes it easier to maintain smaller files. It comes at a price, though. If you use the `F1` or `Tools | Quick Build` on the chapter files, you get errors, since the chapters don't contain the preamble stuff, just the text of the chapter. That means you have to save the file (Ctrl-S or Command-S on a Mac) and then go to the main document (`View` and either `Next Document` or `Previous Document`). Then build.

6.2 Weird output

The most likely source of weird output is a declaration that was not properly limited. For example, you could have put a `\LARGE` declaration in your file, thinking that it was a command rather than a declaration, you would end up with a whole lot more pages of text than you intended. Just go back to the place that it started, and you will rather easily tell what to do.

6.3 Here's where you can help!

I am sure I have run into other problems with LaTeX, but I can't remember all of them. When you encounter a problem, tell Dr. Coulliette, and he will send it along to me, and I will both reply to you, and put the solution (if I can come up with a clever way of stating it) in these notes.