LaTeX is Radical

A detailed introduction to the basics.

Part 1

Version 1.5.3

David Goulette

April, 23 2014

Abstract

This is where a brief summary of your paper would go. You should write something that is direct and to the point so that an expert in the field could read it in about a minute and know exactly what the article covers and what the key results are. When I say "expert," I mean, say, a university professor with research level experience in the area. Abstracts are often hard to fully understand if you are not an expert. But this abstract that you are reading now is the worst ever! It has nothing to do with this document... o.k. I mostly just threw this in to show you how to make an abstract in LaTeX. It uses the `abstract` environment, which does all of the formatting for you automatically! By-the-way, if you don't know what an "environment" is, don't worry. I will explain what a LaTeX environment is in section 4. After you finish that section, come back and look at the code that created this abstract in the .tex file again. It will make sense.

Contents

1 Before you start reading! 3

2 The most basic LaTeX document possible 3

3 Basic stuff in normal text mode 4
- 3.1 Comments on commenting . . . 4
- 3.2 Basic Spacing and Indentation . . . 4
- 3.3 Accents and quotation marks . . . 6
- 3.4 Text sizes and the scope of a command . . . 6
- 3.5 Special characters and surprising results . . . 6
- 3.6 Script variations . . . 7

4 Environments 8

5 Packages 9

6 How to type basic math in LaTeX 10
- 6.1 In-line math mode vs. display math mode . . . 10
- 6.2 How to do in-line math . . . 11
- 6.3 How to do display math . . . 11
- 6.4 How to do display style math in-line . . . 12
- 6.5 A.M.S. packages . . . 13
- 6.6 The `align` environment . . . 13
- 6.7 Common math mistakes and how to avoid them . . . 14
- 6.8 Miscellaneous Math Examples . . . 15

6.8.1 Unexplained Math Examples . . . 16
6.9 Issues with common math symbols, scripts, and math variations. . . . 18
6.10 Examples of some basic math script styles and symbols . . . 19

7 More advanced spacing in both math and text mode 20
7.1 Horizontal spacing . . . 20
7.1.1 Basic fixed-length positive horizontal space . . . 21
7.1.2 Basic negative space . . . 21
7.1.3 Choose your own width using \hspace{} . . . 22
7.1.4 Flexible horizontal space with \hfill . . . 23
7.1.5 The function . . . 23
7.1.6 Examples of all types . . . 24
7.2 Vertical Spacing . . . 27
7.2.1 Changing the global line spacing . . . 27
7.2.2 Variable newline space with \\[length] . . . 27
7.2.3 \smallskip, \medskip, and \bigskip . . . 27
7.2.4 \vspace{} and \vspace*{} . . . 29
7.2.5 \vfill . . . 31

8 More mathematics 32
8.1 Variable sized math grouping symbols . . . 32
8.1.1 \left and \right . . . 33
8.1.2 \big, \Big, \bigg, and \Bigg . . . 34
8.2 Fun ways to use invisible stuff with {} or . . . 36
8.3 Inserting text inside of math . . . 37
8.4 Under-stuff and over-stuff . . . 39
8.5 Matrices . . . 42

9 Labeling and referencing things 46
9.1 How to label and reference a section . . . 47
9.2 Referencing Equations . . . 48
9.3 Labels in large documents: the good and the bad . . . 50

10 Future additions to this document 51

11 Stuff I will teach in separate documents outside of this one 51

Notice that LaTeX automatically indented this paragraph and it isn't in the abstract environment any more. That is the default behavior for normal paragraphs. But you will see below that the first line after a section heading is not indented. That is just common style in technical articles so that is the default behavior when you choose `\documentclass{article}`. There are always ways to override these defaults if you really want to do that.

1 Before you start reading!

Let me tell you right from the start, that there are some *fantastic* resources on the internet for learning LaTeX. For example, I can't hope to do better than the "Not So Short Introduction to LaTeX 2_ε" by Tobias Oetiker,

http://tobi.oetiker.ch/lshort/lshort-letter.pdf

which is extremely well written and I highly recommend you read. And my humble introduction to LaTeX will never be as comprehensive as the wikibook,

http://en.wikibooks.org/wiki/Latex

which will be a fantastic resource for you once you know the basics. But my introduction differs from others in one important way. I have essentially written this as two parallel introductions to LaTeX. I have made available the .tex file AND the .pdf file that it generates so you can read them side-by-side. In fact, if you don't have both, you will miss out on a lot of what I am doing here. I have heavily commented the .tex file with explanations which you won't see in the final document. And I have done some things in the .tex file that will look confusing until you see the final .pdf document and then it will make sense. And I have done a few fancy things that you might have to look back and forth a few times to figure it out. I feel the best way to learn LaTeX (and any programming language for that matter) is to see good examples and start coding your own stuff as soon as possible. So if you are currently reading the .pdf version of this right now and you don't have the .tex file that created this, stop right now and go get it![1]

http://www.sjsu.edu/people/david.goulette/courses/latex/

Also double check that the version number of both files is the same (if not, re-download both of them). I will update the version number if I make any changes. I suggest you watch my first three video lessons on YouTube that explain what LaTeX is, how compiling works and also what files are created. So here goes!

2 The most basic LaTeX document possible

Every LaTeX document needs at least the following:

```
\documentclass{article}
\begin{document}
Hello World!
\end{document}
```

O.k... so you don't need "Hello World!" You can type whatever you want. And also there are other choices other than `article`. The function `\documentclass{article}` declares what type of a document it is. The pair of functions `\begin{document}` and `\end{document}` are the beginning and end of the document. Now that you have the most basic document lets discuss what you can put in the document.

[1] If you are reading an old version of this and the link to my website is dead, then I may have moved on to a new place in life. Just Google my name, throw in the word `LaTeX`, and see what happens.

3 Basic stuff in normal text mode

3.1 Comments on commenting

If you read the `.tex` file you will see many lines in the code that begin with a percent character, `%`. Any text you type in a `.tex` file that comes after a `%` will be ignored by LaTeX. For example, if you type this in your `.tex` file:

```
I made 10% more money this year.
```

then, when you compile your document, you will see this in the final output:
I made 10

Note that the percent sign and everything after it were ignored by LaTeX.[2] Of course there will be times when you want an actual percent sign in your final document! Don't worry, I will show you how to do this in section 3.5 below.

As I mentioned above, I have many comments in the `.tex` file that you should read. I especially have a lot towards the beginning of the `.tex` file. Don't skip over these comments because they are a part of the lesson here. For example, I explain a lot about how to add packages in the `.tex` file because it makes more sense to explain it there. If you don't know what a package is don't worry. I am going to explain what they are later in section **??**. You will find, for example, that my discussion of packages that you see in the .pdf file is more basic and introductory in nature, but the comments in the `.tex` file are more specific and detailed. So read both.

3.2 Basic Spacing and Indentation

Notice that LaTeX did not automatically indent this first line of the section. That is a default behavior because this document is using the `article` class. LaTeX does a lot of automatic spacing like this (if you don't like the spacing LaTeX does, don't worry, I will show you how to manually space things in just a minute). Note also that LaTeX is adjusting the spacing between words to make the text fit the column with publishing quality. Long words like antidisestablishmentarianism, will be hyphenated for you if they don't fit the column. If you don't like the automatic hyphenation you can override it and fix it. Basically any "automatic" thing that LaTeX does can be overridden if you want. LaTeX will completely ignore multiple spaces and single carriage returns in a .tex file. So what you are reading looks completely normal in the final document even though the `.tex` file looks pretty messed up.

You should look at the `.tex` file to see what I mean with this.

This is on a new line with no indent because I used \\ to end the line.
Some
words
on
new
lines
so
you
can
see the result.

Let me mention here that a line of code in your .tex file is not the same as a line in your resulting document. And typing on a newline in your .tex file will not result in a newline in your document. The command \\ gives you a new line in your *document* but pressing `<enter>` on a PC (or `<return>` on a Mac) gives you a new line in your .tex file. *Big difference!*

[2] By-the-way, commenting lines can be very useful for debugging your code. If you have a strange error when you compile and you are not sure what line is causing the problem, try commenting out various lines you suspect might be causing the problem and see if it compiles then. This will help you narrow down your search for the error.

Two carriage returns gives you a new line that's indented. ("Carriage return" means a new line in the .tex file, that is, `<enter><enter>`.)

This is on a new line with a space and indented.

Use `\noindent` to force LaTeX to not indent like this line. Note that anything in your `.tex` file that starts with a `\` will be interpreted as a command by LaTeX. So typing `\noindent` in your `.tex` file is a command to LaTeX that tells it not to indent the line that follows.

To force a page break use: `\newpage`

3.3 Accents and quotation marks

There are lots of examples of accent marks. Here are a few: San José State, la forêt enchantée, Möbius, and Peña. Be sure to use `{}` around the letter that you are accenting. For example, Hungarian has the letter o with a double acute accent over the top like this: ő. You get this accent mark over the o by typing `\H{o}`. So if you type, `Paul Erd\H{o}s`, you will get: Paul Erdős. If you leave out the `{}` and only type `Erd\Hos`, you will get an error because LaTeX will be confused about how to interpret it; is the function `\H`? or `\Ho`? or `\Hos`? When you use `{}` you make it clear where the name of the function ends and what the input to the function is (and LaTeX likes that).

If you want to use quotes you can't do `"word"` because you will get: ”word”
Note the wrong direction for the first quotation mark. Instead you need to do press the single quotes that point in the opposite direction like this: ``` ``word'' ```, to get this: “word.” Here is a case where you really have to look at the .tex file to see what I mean because it is hard to explain in words but easy to see in the code. ☺

3.4 Text sizes and the scope of a command

Some commands in LaTeX have the effect of changing the behavior of LaTeX from that point forward. If I use the `\tiny` command, then all text after that command will be tiny. This can cause errors because you might have other environments (or math) later on in the document that cannot be “tiny.” So you usually limit the scope of the command with `{}`.
If you type this in your .tex file:

```
This is normal size, {\tiny this is tiny,} and now we are back to normal.
```

You get this:

This is normal size, ^{this is tiny,} and now we are back to normal.
Notice how the {} limited the scope of the command. Other sizes:

^{Alice} Alice Alice Alice Alice Alice Alice Alice Alice
^{Really} cOOl, right? ^{I think so.}

3.5 Special characters and surprising results

The following characters are special in LaTeX: `# $ % ^ & _ { } \ ~`
When I say “special” I mean that they are reserved for functions and commands. So LaTeX interprets them as coding instructions. If you want these characters in your text you need to type the following commands: `\# \$ \% \^{} \& \_ \{ \} \textbackslash` to get # $ % ^ & _ { } \
Notice that `\\` already means newline!, which was why we needed `\textbackslash`.

Now we can fix the example from section 3.1 above. If you add the backslash before the % sign we will get what we intended. So if we type this in our .tex file:

```
I made 10\% more money this year.
```

we will get this in the final document:

I made 10% more money this year.

The % is no longer interpreted as the beginning of a comment in the code so we get what we intended to get.

Tilde is a complicated case because it can be done a few ways with different results: ~ and ˜, are two choices. (But how often do you want a floating tilde anyway?)

Here is a fun example. If you type `<` or `>` in your .tex file you will get ¡ and ¿ by default.

¿Cuándo es tu cumpleaños?... *¡Hoy es mi cumpleaños!*

There are *MUCH* better ways to handle characters from languages other than English using the `babel` package, but this is a hack for some quick Spanish. Note that `<` and `>` aren't really special function characters. (Well,... they are not special when you are in text mode like we are now.) But I am pointing out the fact that they don't create the output you might expect. (Later in section 6, we will be discussing math mode where `<` and `>` will mean "less than" and "greater than"... but that is only in math mode and we are not there yet.) When you are in normal text mode, if you want to get < and > you need to type the commands `\textless` and `\textgreater`.

Finally if you type the `|` character in your code you will get a small horizontal line — like that. And you can do a bunch of them in a row to create a line. Like this:

This is useful for adding a horizontal dividing line. (Obviously, David...) I have some later in this document to divide up consecutive examples.

3.6 Script variations

Let me start this short section by saying what this section is *not* about. This section is not about changing fonts. The following examples are different script styles within one font family. The word "font" in LaTeX refers to the style of the text you are reading now, along with all the different styles seen below. The default font in LaTeX is called Computer Modern. Other fonts are available. (It is a complicated topic to be left for another day.) Here are the script variations available with the default Computer modern font:
This is italic
This is Roman script
This is sans serif
THIS IS SMALL CAPITALS
`This is typewriter-like`
This is slanted (different that italic)
This is underlined

You can combine these script styles with sizing commands like, say, very large small caps: LARGE SMALL CAPS. (That seems contradictory! But it isn't. "Large" is the size of the script and "small caps" is the name of the script style.)

And you can combine *some* of the script styles like:
This is bold underlined italics!
But some won't combine. You can't have italic small caps, for example.

Let me emphasize that all of the script variations you see above are variations on the Computer Modern font. We will see the math script style for Computer Modern below.

The function `\emph{}` has different behavior than `\textit{}`. `\emph{}` is used to emphasize something and it changes regular font to italics or italics to regular font depending on what is the current style (`\textit{}` just forces italics). Compare this:

I want to *emphasize* this point.
to this:
I want to emphasize *this point.*

Here is a case where you really need to look in the .tex file to fully understand this example.

If you want to type a short bit of text that comes out exactly like you type it including any special characters, then you use the function `\verb|***|`. In the verbatim environment, LaTeX will ignore the meaning of special characters and just give you exactly what you type in a typewriter script. I use this for typesetting a short block of computer programming code and I also use the verbatim function for my students when I am typing up instructions on how to use a calculator or learn LaTeX![3] Now, `\verb` is a unique function because the delimiters for

[3] For long sections of code I actually prefer to use the `verbatim` environment. I will explain environments later in this document.

the argument are not {} in this case. The reason is because, well... what if you also want the characters { or } in your verbatim text?... LaTeX would get confused. With \verb, you can actually use any normal character as a delimiter. I use the symbol | in general just because I like it. That is, unless I want a | in my verbatim text (like the four I already have in this paragraph!) which means I would need to use + or something else. This is a case where examples are clearer than explanations.

If you type this in your .tex file:

```
\verb|\LaTeX\ is cool.\newpage \begin{document} $ % ^|
```

Here is a case where looking at the .tex file might actually be confusing at first! But it would be a good exercise for you to figure out what I wanted and how I accomplished it.

Then you will get this in your document:

```
\LaTeX\ is cool.\newpage \begin{document} $ % ^
```

Notice that we got everything between the | | verbatim, and LaTeX didn't try to interpret the special characters as commands.

If you type:

```
\verb+ <> | } # \noindent+
```

Then you will get:

```
<> | } # \noindent
```

Notice we got everything between the + + this time. I had to switch to + because I had a | in my text.

Everything inside of \verb|stuff| has to be on the same line. You will get errors if you press <enter> in the middle of your verbatim text and try to compile it.

```
\emph{Note that when you have a long verbatim line, then you have to split
the line manually and force a new line in your document.  \LaTeX\ will not do
any formatting with verbatim text. Notice how this will run right into the margin if I let it!!!}
```

Always remember that LaTeX will do *exactly* what it is told to do! And sometimes that is not what you *want* to happen. So you have to fix it.

4 Environments

Environments are basically anything that looks like this:

```
\begin{environment name}
junk in the middle
\end{environment name}
```

(There is actually one important exception to this that you will see below in math mode that I will show you in section 6.3.) So this document itself is an environment because it starts with \begin{document} and ends with \end{document}. I used the verbatim environment to make the example before this paragraph. The verbatim environment is an alternative to the \verb function I explained earlier but it has essentially the same results. The abstract at the beginning of this document is an environment. Here are just a few more useful examples (there are lots more than what I have here):

This is the center environment.

This is the flushright environment.

Here is a long quote using the `quote` environment. It does the formatting for you! This is from a really good book:

> Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice, 'without pictures or conversations?'
>
> So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

Don't you want to find out what happens to Alice next?
I'll tell you this much, she falls down a hole.
O.k. that was me being silly. I just wanted to show you the quote environment.

Here is an example of the `itemize` environment nested inside of the `itemize` environment nested inside the `itemize` environment:

- Bread
- Cheese
- Fruit
  - Oranges
  - Apples
    - Fuji
    - Granny Smith
  - Kiwi
- Coffee

As you can see it creates multi-level bullet points. My example here is just the default itemized list. There is a very cool package called `enumerate` that you can add on to your LaTeX document which makes bullet point lists and enumerated lists very customizable. But wait!..., now that I mention it, I realize that I just mentioned something called a "package" which I haven't explained to you yet! So here goes.

5 Packages

A package is an add-on to the standard LaTeX. New packages are added in the preamble to the document. The preamble is everything between `\documentclass{classname}` and `\begin{document}`. ⇐ Oops!
So, when you look at a .tex file, this is where the preamble is:

```
\documentclass{book}
**PREAMBLE STARTS HERE**
     *
     *
     *
**PREAMBLE ENDS HERE**
\begin{document}
Call me Ishmael.... etc.
```

The preamble is the place where you set global parameters for the document (like title, author, and date), create new commands, set the options you want (like pagination style, header and footer style), etc. etc.

An example of a package that I have used in this document is the `geometry` package, which makes changing the margins much simpler than with the standard LaTeX methods. The general syntax for adding a package is: `\usepackage{package name goes here}` (See the preamble to the .tex file for my extensive commentary on packages.) I have also added the `marginnote` package which I really like for making notes in the margin (obviously). Margin notes can be useful when working on a single document with a group of people or when you are proofreading someone's paper and you want to make comments. I am also using a package that lets me use color in this document as well. See the comments in the preamble to this .tex document to read about the awesome package called `hyperref`. It is the reason why all of the section references in this file are live links. Also, if you click on a url in this document it should open up your favorite browser and take you straight to the web site. And `hyperref` is also why you can see a table of contents on the left hand side of the window if you are using Adobe Reader or any similar viewer to read this .pdf file. If you are reading this document in the .pdf viewer included with your .tex editor, then you probably won't see the table of contents on the left. So open this file up in Adobe Reader and check it out!

Here is a margin note where you can put text! I set the `marginpar` *width to be wider than normal in the preamble using the* `geometry` *package so that I had room for this. See my comments in the preamble to the .tex for details.*

The truth is that, *technically*, many of the things that packages help you do are actually possible without the packages. But the packages make things MUCH easier. There are thousands of packages that people have made. Most are free. If you are using MiXTeX in Windows, you have a package manager that will show you a list of possible packages and the ones you have installed. You can probably find it under:

`Start ==> All Programs ==> MiXTeX ==> Maintenance ==> Package Manager`

If you are on a Mac or using Linux, you are on your own. ☺ Oh yeah... one last thing here, if you add a package in your preamble that you don't have currently installed, NO PROBLEM! MiKTeX will install it for you on the fly when you compile. So you don't actually have to use the package manager very often.

6 How to type basic math in LaTeX

The most impressive part of LaTeX is on display when you create great looking mathematics and/or technical symbols. This is an endless topic that I can only begin to explain here. But it is important for you to know the most important basics of math mode.

There are two main modes in LaTeX: text mode and math mode. Everything we have seen so far in this document has been in normal text mode. When you are in text mode, LaTeX interprets functions a certain specific way. All of the functions and environments that I have mentioned or used up until now were for text mode. Once you switch to math mode the game changes.

6.1 In-line math mode vs. display math mode

The first thing to know about math mode is that math mode itself has two modes! *Sheesh!* Don't worry it isn't that hard. The two math modes are "in-line math" and "display math." Sometimes you want a bit of math that goes right in the line. The equation $2x^2 + 3x - 7 = 0$ shows a little bit of in-line math. Or you might want $\sum_\alpha x_\alpha$, $\forall \alpha \in \mathcal{I}$. Note that even though this is a little more complicated, this is still pretty readable in-line. Also note that the script style changes a bit to look sort of "mathey." This script style is different than any that were mentioned in section 3.6 above. Good LaTeX fonts blend the text with the math very nicely so they look different but not so different that it is jarring (like in Microsoft Word... *Yuk*). But the limitation of in-line math is clear when you want bigger constructs that can get scrunched if you try to do them in-line. Consider this: $\Phi(x, y) = \frac{x^2 - y}{\int_\zeta^\omega f(x) dx}$. That looks horrible and it is hard to read the letter zeta. And notice that LaTeX had to push this line down a bit to fit it in! In this case you should use display mode instead of in-line which makes the equation big and

also centers it. With display mode you get this:

$$\Phi(x,y) = \frac{x^2 - y}{\int_\zeta^\omega f(x)dx}.$$

Or you might prefer the look of this:

$$\Phi(x,y) = \frac{x^2 - y}{\displaystyle\int_\zeta^\omega f(x)dx}, \qquad \text{or maybe you prefer} \qquad \Phi(x,y) = \frac{x^2 - y}{\left(\displaystyle\int_\zeta^\omega f(x)dx\right)}.$$

There are many options and you are in control of the final appearance.

In the displayed versions you can actually read the Greek letter zeta as the lower limit of the integral. So, in general, larger or longer equations should be done in display mode. But you have the control so it is ultimately your choice.

6.2 How to do in-line math

Doing in-line math is easy. Just put your math between two dollar signs. So if you type `$3x-5=y$` in your .tex file, then you will get $3x-5=y$ in your final document. LaTeX just switches to math mode when you type the first `$`, and it switches back to text mode when you type the second `$`. Note that $3x-5=y$ does not look the same as 3x-5=y. The first example used dollar signs and the second did not, so LaTeX just stayed in text mode. When you are mentioning mathematical variables in your text you should put them in dollar signs so they look like x, y, and z instead of x, y, and z. So make sure you do math in math mode so it looks like math and not text (and vice versa). You will know when you mess up and do the opposite. If you accidentally type text in math mode you will get $totalgarbagelikethis$. In math mode, every character is considered to be a mathematical letter. Therefore, LaTeX will not put any space between letters even if you press space bar between them. All spaces are completely ignored in math mode.

Another new thing to learn is that when you are inside of math mode you have to use math commands to get what you want and these math commands *only* work in math mode. For example, to get an exponent you need to use the `^` character. Typing `$f(x)=x^2$` gives you $f(x)=x^2$. From now on I won't always explicitly explain how to get every single mathematical symbol or construct. Instead, I will show you a bunch of examples and if you want to learn how I did it, you need to look at the .tex file to see how! Here are some in-line math examples. Fractions like $\frac{2x}{z}$ or $2x/z$ are easy. So are roots like $\sqrt{\Delta - y_1}$ or $\sqrt[3]{\varphi}$. Matrices like $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ are usually too big for in-line mode but you can do it and LaTeX will adjust things.[4] As we have seen, large fractions like $f(z) = \frac{az+b}{cz+d}$ don't look the best in in-line mode. But you can often rewrite them with parenthesis and you get $f(z) = (az+b)/(cz+d)$ which is fine if the equation is simple.

6.3 How to do display math

Display math has more options and possibilities. To use display math you will always need some sort of an environment similar to how we did a bullet point list in section 4 above. However, the difference is that the environments I'm about to use all send you into math mode instead of text mode.

In this section I will only show you the two basic options for display math. They are also the most common. First, if you want displayed equations that are *not* numbered you enclose your math inside `\[ \]`. So if you type this:

```
\[
f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}
\]
```

Notice I do not use dollar signs in this example. Dollar signs are only for in-line math.

[4] To do this matrix I used the `amsmath` package which I explain below in section 6.5. And by-the-way, if you want to know how to do a basic footnote like this one, it's easy! Just do `\footnote{and type whatever you want here}`. LaTeX will stick the footnote reference right where you put the `\footnote{}` and nicely format the footnote at the bottom of the page.

you will get the displayed equation with no automatic numbering:

$$f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}.$$

Using `\[ \]` to have displayed equations is an example of an environment that doesn't use `\begin{}` and `\end{}`. Now, a word of warning: you will see old .tex files that will display unnumbered equations with double dollar signs `$$ $$` instead of `\[ \]`. This practice is out of date and not advised. It is a relic of old versions of TEX and I have read that it will sometimes cause errors with recent packages. So just don't do it. But in case you see it in other people's work, you will know why.

If you want LATEX to number the equation for you, just use this instead:

```
\begin{equation}
f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}
\end{equation}
```

and you will get the same thing we got before but numbered so you can reference it later:

$$f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}. \tag{1}$$

6.4 How to do display style math in-line

There are times when you will actually need to do display style math but you will want it in-line. This is a very simple topic to learn but it is probably something that you won't need very much when you are just doing basic stuff. However, when you start to use LATEX more you will need this occasionally. I thought I would put this section here for the sake of completeness and you can refer back to it when you need it someday. I have used it at least a dozen times in this document already. To force display math when you are doing in-line math all you need is the command `\displaystyle` inside of the `$ $`. This function overrides the in-line math behavior that shrinks your math. Here is an example of what you type in your code followed by the result:

```
Compare $\frac{ax+b}{cx+d}$ to $\displaystyle \frac{ax+b}{cx+d}$
```

Compare $\frac{ax+b}{cx+d}$ to $\displaystyle \frac{ax+b}{cx+d}$

The problem with normal display math environments is that they usually do a lot of formatting and sometimes you don't want that. They often force the equation to be centered on the page and they sometimes add extra spacing above and below the equation as well. Sometimes you want display math style but you want it in a very particular spot. It comes up a lot when you are formatting a table where some regular text needs to be lined up with some display math side by side. See for example the table that I made on page 35. The second row of that table has display math that is lined up with regular text. Using the `\displaystyle` command is the easiest way to handle that situation. Let's have an example in this section. Let's say you are really into electromagnetism and you want a sentence like this:

My favorite equation in the whole world is: $\displaystyle \oiint_{\partial\Omega} \mathbf{E}\cdot \mathrm{d}\mathbf{S} = \frac{1}{\varepsilon_0}\iiint_{\Omega}\rho\,\mathrm{d}V$

Clearly you want the equation to be placed right after the colon in this situation. And if that really was your favorite equation, it would be a shame to make it small. The `\displaystyle` function is just what you need here.

Just in case that is your favorite equation and you want to see the code for that last example, here it is but try not to be intimidated by it if it looks confusing. This example requires the AMS packages to be included in the preamble which I will explain in section 6.5 and you also need the `esint` package to get the cool double integral. I also use math bold, math roman, and some symbols which I will explain later in section 6.9 as well as a spacing command that I will explain in section 7. But here it is anyway:

```
$\displaystyle
\oiint_{\partial \Omega}\mathrm{\mathbf{E}\cdot d\mathbf{S}}
=
\frac{1}{\varepsilon_{0}}\iiint_{\Omega}\rho\, \mathrm{d}V$
```

In case you are wondering, my favorite equation is: $e^{i\pi} + 1 = 0$. Mind. Blown. (How is it possible that a transcendental number, e, can be raised to the power of another transcendental number, π, multiplied by the complex imaginary unit, i, and somehow that equals the integer -1! I know proofs of this equation... but still...) This beautiful equation happens to look just fine in normal in-line mode. ☺

The 5 most important numbers in all of mathematics are contained in this one truly amazing equation.

6.5 A.M.S. packages

Before I go on to show you a bunch of math examples I have to mention the American Mathematical Society packages. These are important expansions of the basic LaTeX math capability. They add many symbols, constructs, and environments that are not available with basic LaTeX. The in-line matrix I made earlier used the `bmatrix` environment from the `amsmath` package. See the preamble to the .tex file for more information. Many of the examples below require these extra packages.

6.6 The `align` environment

One environment I use a lot is `align` and this is a good place for me to teach you the use of the ampersand character &. We use & to align things. The ampersand is used in both text mode environments and math mode environments. If you use `\begin{align}` then it will number every line in the output like this silly example:

```
\begin{align}
ax+b &= \mu-\theta\\
&\leq \eta+42\\
&< \epsilon.
\end{align}
```

which gives you

$$\begin{aligned} ax + b &= \mu - \theta && (2)\\ &\leq \eta + 42 && (3)\\ &< \epsilon. && (4) \end{aligned}$$

You need to use the `\\` command to get each line of math on a new line. But more importantly, notice the use of `&` in the code for this example and the results. LaTeX will align the lines based on where I placed the `&`, so I chose to put them just to the left of $=$, $\leq$, and $<$ so they line up perfectly. I have found that putting them to the left of the relation symbols works well.

I often use align more like the following example where I am showing steps and I don't want to number everything. To get the unnumbered version of most environments just add a star (a.k.a. asterisk) to the command like this:

```
\begin{align*}
\frac{d}{dx}(x^2+2x+1)^2 &= 2\cdot(x^2+2x+1)\cdot(2x+2)\\
&= (2x^2+4x+2)\cdot(2x+2)\\
&= 4x^3+8x^2+4x+4x^2+8x+4\\
&= 4x^3+12x^2+12x+4.
\end{align*}
```

And you will get this:

$$\begin{aligned}\frac{d}{dx}(x^2+2x+1)^2 &= 2\cdot(x^2+2x+1)\cdot(2x+2)\\ &= (2x^2+4x+2)\cdot(2x+2)\\ &= 4x^3+8x^2+4x+4x^2+8x+4\\ &= 4x^3+12x^2+12x+4.\end{aligned}$$

Here is one last example:

$$\begin{aligned}2(x+4) &= 4x-2\\ 2x+8 &= 4x-2\\ 10 &= 2x\\ 5 &= x.\end{aligned}$$

6.7 Common math mistakes and how to avoid them

There are many little mistakes that are easy to make when you are first learning LaTeX. This section discusses a few common ones. First, make sure to group things with {} whenever necessary. It is common to make mistakes with exponents and subscripts in this regard. For example, if you type

`$x^23+y_12+z_a^2$` you will get $x^23+y_12+z_a^2$.

You have to put {} around your exponents and subscripts if they are longer than one character. So if you type

`$x^{23}+y_{12}+z_a^2$` then you will get $x^{23}+y_{12}+z_a^2$.

Notice that I don't need the brackets around the subscript and superscript on z because they are only one character long. It turns out that if you want a subscript that is, say, a Greek letter, then you don't need to use brackets even though the command for a Greek letter is more than one character long. LaTeX treats the `\alpha` in the next example as one object:

`$F_\alpha$` will give you this: F_α

But I recommend you get in the habit of using brackets like this:

`$F_{\alpha}$` which will still give you this: F_α

This "good habit" will help you avoid errors in other circumstances.

Using brackets is also important when a LaTeX function has an alternate subscript-superscript behavior. For example, the `\sum` function will give you an alternate behavior when you do subscripts and superscripts. This is because we tend to write the indices of a summation directly above and below the summation symbol. But the syntax is the same as the examples in the previous paragraph. So if you type

```
\[
\sum_{i=0}^n i=\frac{n(n+1)}{2}
\]
```

you get

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}.$$

Notice how important it is that I put {} around the $i=0$ in the code so that all three characters went below the summation sign.

Compare to the behavior here:

`$F_{i=0}^n$` which gives you this: $F_{i=0}^n$

Examples of other symbols with behavior similar to `\sum`:
`\int_a^b` `\int\limits_a^b` `\bigcup_a^b` `\bigcap_a^b`
`\prod_a^b` `\coprod_a^b` `\bigsqcup_a^b` `\bigoplus_a^b` `\bigotimes_a^b`

$$\int_a^b \quad \int\limits_a^b \quad \bigcup_a^b \quad \bigcap_a^b \quad \prod_a^b \quad \coprod_a^b \quad \bigsqcup_a^b \quad \bigoplus_a^b \quad \bigotimes_a^b$$

By-the-way, note that `\sum` and `\Sigma` are not the same thing! Sure, they are the same capital Greek letter, but the first will look and behave differently than the second because the first one is a LaTeX math command and the second one is a math text symbol (which I will explain in more detail in section 6.9). Here we have `\sum_i^n` on the left and `\Sigma_i^n` on the right.

$$\sum_i^n \neq \Sigma_i^n$$

LaTeX has many already built-in functions to make the most common math constructs. You have already seen some but I want to emphasize here what can happen when you don't use them. For example, if you type this

`$sin(x)=tan^{-1}(x)$` then you will get $sin(x) = tan^{-1}(x)$,

which looks bad because LaTeX writes the letters in *sin* and *tan* like they are variables. You see, LaTeX doesn't know that you *want* the sine and tangent function. It just knows that you are in math mode and you typed some letters. So it treats *sin* and *tan* like variables that are being multiplied. Instead you want to use the built in functions for sine, `\sin`, and tangent, `\tan`. These functions make the letters in Roman (as they should be) in order to differentiate them from variables. Also these built in functions tend to space things better. So if you type this

`$\sin(x)=\tan^{-1}(x)$` then you will get $\sin(x) = \tan^{-1}(x)$,

which looks correct. All of the trig and inverse trig functions have corresponding LaTeX functions, as well as log, mod, max, min, and many other common constructs. Make sure you use the built in math commands that are available in LaTeX.

6.8 Miscellaneous Math Examples

This is just stuff I am throwing in for no specific reason at all except to show you some examples of what you can do in math mode. Even though this section of the document is titled "How to type basic math in LaTeX," many of the examples that follow are a little more advanced than "basic." Note that there are a few things I have below that I have not explicitly explained (especially more advanced spacing) but they are explained later in this document. I am just showing you some things so you can see some of what LaTeX can do. Hopefully it will motivate you to learn more!

One fun example is that there is actually a solution to a problem that I mentioned earlier. The problem was that matrices like $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ do not very look good in-line because they are so big and for LaTeX to squeeze it in between as best it can. There is actually an environment called `smallmatrix` that will allow you to make a matrix that looks better in-line like this: $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$. It doesn't look too bad and if you are really trying to save space then this is your best option that I know of.

6.8.1 Unexplained Math Examples

$$f(x,y) = \int_{\alpha}^{\beta} \frac{x^2 + \sin y}{\ln x} dx dy + \sum_{i=0}^{n} \sqrt{x^{2i} + y}$$

$$f : X \xrightarrow[T]{\text{injective}} Y$$

$$\int_a^b f(x)dx = F(a) - F(b)$$

For, $x \in A$ and $y_1 \in B$, $\delta \pm \sqrt{x^2 - y_1} = \log_2(\phi)$.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \not< \aleph_0$$

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$$

$$\frac{\partial}{\partial y}\left[x^2 \cos(y)\right] = -x^2 \sin(y)$$

$$\left(\frac{2x}{z}\right) \cdot \left(\frac{5z}{2w}\right) \approx 2.73 \times 10^{-5}$$

$$12 \equiv 2 \pmod 5$$

You can have the limits of integration on the right side of the integral sign like this:

$$\int_0^{\pi} \int_0^{2\pi} \int_0^{1} \rho^2 \sin(\phi)\, d\rho\, d\theta\, d\phi = \frac{4}{3}\pi r^3$$

Or you can have the limits above and below the integral sign like this:

$$\int_{0}^{2\pi} \int_{\pi/6}^{\pi} g(\theta, \phi)\, \mathrm{d}\theta\, \mathrm{d}\phi$$

←Some people prefer to have the differential operators in roman like this instead of slanted math style.

$$\iiint f(x,y,z)\, dx\, dy\, dz$$

$$\int_{n} \cdots \int f(y_1, y_2, \ldots, y_n)\, dV$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} a_{1i}\, x_i \\ \sum_{i=1}^{n} a_{2i}\, x_i \\ \vdots \\ \sum_{i=1}^{n} a_{ni}\, x_i \end{pmatrix}$$

$$P_{r-j} = \begin{cases} 0 & \text{if } r-j \text{ is odd,} \\ r!\,(-1)^{(r-j)/2} & \text{if } r-j \text{ is even.} \end{cases} \tag{5}$$

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}}$$

$$\sum_{\substack{0\le i\le m\\ 0\le j\le n}} {}_nC_m$$

$A = \{x + iy \in \mathbb{C} : x \in \mathbb{Z} \text{ and } y \geq 0\}$

Cool equations from complex analysis.
$$\begin{cases} e^{i\theta} = \cos(\theta) + i\sin(\theta) \\ f(a) = \dfrac{1}{2\pi i}\oint_\gamma \dfrac{f(z)}{z-a}dz \end{cases}$$

$$d|n := \text{“}d \text{ is a divisor of } n\text{”}$$

$$\left\{\begin{matrix}\text{All true}\\ \text{mathematical}\\ \text{statements.}\end{matrix}\right\} \subsetneq \left\{\begin{matrix}\text{The set of all}\\ \text{interesting true}\\ \text{statements.}\end{matrix}\right\}$$

Let's solve $x^2 + 5x + 4 = 0$ by factoring.

$$\begin{aligned} x^2 + 5x + 4 &= 0 \\ (x+4)(x+1) &= 0 \end{aligned}$$

which clearly implies that either

$$x + 4 = 0 \qquad \text{or} \qquad x + 1 = 0,$$

so

$$x = -4 \qquad \text{and} \qquad x = -1$$

are solutions.

For the chemistry fans, here are some isotopes:

$$\ce{^{14}_{6}C} \quad \ce{^{238}_{92}U} \quad \ce{^{16}_{8}O^{2-}}$$

6.9 Issues with common math symbols, scripts, and math variations.

As we saw earlier in section 3.6, there are a variety of script styles available when you are in text mode. But it turns out that `\underline{}` is the only one of those that *also* works in math mode. The rest will not work in math mode... well, I should say, it *might work* but it probably will not have the effect that you expect. I should clarify what I mean with some examples.

If I type,

```
$2x + \mu- \underline{3y^2 -\sqrt{\zeta} + \int_a^b f(x)dx}$
```

then I get,

$$2x + \mu - \underline{3y^2 - \sqrt{\zeta} + \int_a^b f(x)dx}$$

So notice that I was able to use `underline{}` inside of math mode and I got what I expected: underlined math. But watch what happens when you try to make some bold math with the `\textbf{}` function inside of math mode. Typing this:

```
$2x^2 -z + \textbf{3x - 5y} + z^3$
```

will give me,

$$2x^2 - z + \textbf{3x - 5y} + z^3.$$

Technically it compiled... so it "worked," but notice what happened. Everything I put inside of `\textbf{}` was converted to text mode. So I didn't get bold math as I had hoped. I got bold text surrounded by regular math. When you type `\textbf{}`, LaTeX will go back to text mode after the `{`, but then switch back to math mode after the `}`. This means that you will get compilation errors if you try to put this in your document:

```
$2x^3 + \textbf{x^2} -7$
```

You will get errors because we have `x^2` inside of the `textbf{}` command. The exponent command is only allowed in math mode but LaTeX is in text mode. If you want math bold you have to use the command `\mathbf{**}` but unfortunately there are limitations on which characters and symbols can be bolded with basic LaTeX. Notice that the pi in the following example isn't bolded. Neither are the + or − operators. Also notice that the bolded math is upright instead of slanted.

$$\mathbf{2x^2} + \pi - \mathbf{\Sigma} = \mathbf{f(x)}$$

Because of the limitations of `\mathbf{}`, the AMS packages added `\boldsymbol{}` and `\pmb{}` to *somewhat* correct for these deficiencies (but they have limitations too). You will have to experiment with each version of bold to see if you like it. Here is an example that I borrowed directly from the AMS documentation[5]. to show the differences:

$$A_\infty + \pi A_0 \sim \mathbf{A_\infty + \pi A_0} \sim \boldsymbol{A_\infty + \pi A_0}$$

You have likely noticed that I have used Greek letters many times in this document. And, if you are reading along in the .tex file (as you should be!) you might have noticed that I have only used Greek inside of math mode. It turns out that you can't use the basic Greek letters in text mode.[6] With standard LaTeX, Greek letters are treated as mathematical symbols only. So

[5] Here is the AMS-LaTeX website: http://www.ams.org/publications/authors/tex/amslatex
Here is the direct link to the short guide: ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf

[6] If you want to actually type text in Greek, Hebrew, Cyrillic, Hangul, or some other script type, you have to use an add-on package like `babel` to do the job. There are packages supporting all the major non-English types of script.

if you type `\delta` in text mode, you will get an error. Thus, if you do want a δelta or a piece of πie in the middle of your text, then you need to put it inside `$ $`. But keep in mind that LaTeX is treating these Greek characters as *math* and not text.

So the main lesson to learn in this section is that there are some things that only work in text mode, some things that only work in math mode, and some things that work in both. If you are not sure what will work, then give it a try and see what happens!

6.10 Examples of some basic math script styles and symbols

I will not even try to show you the hundreds and hundreds of symbols that are available for math. Instead visit my website where I will refer you to sources that can help you find the symbols you want.[7] But I will show you the basic script types that are available in math mode as well as a few extremely common math symbols.[8] Let me emphasize that the scripts you see in this section are all *in math mode* and not normal text. Some of them look similar to what we saw in section 3.6 but everything in that section was for normal text.

Capital Greek:	$\Gamma\Delta\Lambda\Phi\Pi\Psi\Sigma\Theta\Upsilon\Xi\Omega$
Small Greek:	$\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega$
Variations on basic Greek:	$\varepsilon\vartheta\varphi\varpi\varrho\Delta\Theta\Lambda\Xi\Sigma\Upsilon\Phi\Psi\Omega$
Math Calligraphy:	$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$
Math Sans Serif:	$\mathsf{abcdefghijklmnopqrstuvwxyz}$ $\mathsf{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$
Math Typewriter:	$\mathtt{abcdefghijklmnopqrstuvwxyz}$ $\mathtt{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$
Math Blackboard:	$\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$
Math Fraktur:	$\mathfrak{abcdefghijklmnopqrstuvwxyz}$ $\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$
A sample of common stuff:	$\emptyset\varnothing\infty\times\ \leq\geq\neq\approx\equiv\sim\simeq\ \pm\partial\ \perp\ \cup\cap\ \subset\supset\subseteq\supseteq\ \setminus\ \to\ \nabla\neg\forall\exists\ \in$
If elliptic functions are your thing, then you need:	$\wp$
And a bit of Hebrew for the set theorists:	$\aleph$
Here are some extra symbols you might use occasionally.	$*\cdot\div\ldots\bullet\cdots\ltimes\rtimes\otimes\oplus\odot\dagger\therefore\blacksquare\square\diamond\lozenge\heartsuit\sharp\flat\natural\spadesuit\clubsuit\checkmark\S\ \nearrow\searrow\nwarrow\swarrow$

Many capital Greek letters are the same as Roman letters (like A or Z), so there is no LaTeX version of these letters because they are not needed.

←The first two symbols in this line are two different versions of the empty set that are available. Please note that it is not the Greek letter phi. The empty set is a stylized version of the letter Ø from the Norwegian alphabet... I'm sorry... that is a pet peeve of mine.

With these letters and symbols available in math mode, it makes it easier to have a nice variety of symbols in your text. This is especially helpful when you are typesetting more advanced mathematics that has a lot of variables. The variety makes your document much more readable. And many of these have become relatively standard for certain sets, constants, and variables, like: $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{C}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathrm{GL}(n,\mathbb{R})$, π, $\aleph_\circ$, and $\wp$. Some things are context dependent but

[7] http://www.sjsu.edu/people/david.goulette/courses/latex/

[8] By-the-way, you should look at the .tex file to see how I align the math text examples here. Also, read my comments right before the example for an explanation of the syntax. I used the `tabular` environment and I used the `&` character to line up things just like I did with the align environment earlier.

common like: x, y and z for real variables, z, w, ζ, and ω for complex variables, θ, ϕ, and ρ for spherical coordinates, the list goes on and on.

7 More advanced spacing in both math and text mode

(From this section on, it would be better to completely read in the .pdf document first, especially if you are more of a beginner. Then, after you are done, you can read the .tex file if you are interested in how I did something that I didn't explain. From here on out I have far fewer comments in the .tex file and I show the code for most of the examples in the .pdf document The reason is that the level of complexity will increase slightly and I use more advanced things like tables that help align the examples nicely. Many of these examples are hard to read in the .tex file when you are new to the game. Eventually you should learn how to do things like make a table though![9])

In section 3.2 we covered some of the basics of spacing and indentation. But sometimes you need more control over spacing in your document. This can happen when you don't like a default LaTeX behavior or when you are trying to do something outside the basics. Fortunately we have some more advanced commands at our disposal. This section will not be exhaustive of all possibilities but it will give you all of the tools you will likely ever need in order to space things however you want. Also, let me mention that all of the functions I introduce here work in both text mode *and* math mode, but might not *always* work in all situations... it depends on how you use it. You will have to play with them and see what happens, and if one method gives you errors or doesn't work the way you want it to, there are other options you can try instead. Off the top of my head I can't remember getting errors doing spacing... but it might happen...

A word of warning. This is the first section in this document where I am giving you the tools to manually override the behavior of LaTeX at the local level. It is true that packages add-on new behavior to LaTeX but it is a global change. And it is also true that environments change behavior locally, but in an environment, you are relying on the environment to do the changes for you; you are not manually forcing the changes at a low level. In this section you are learning commands that override LaTeX directly. You are commanding LaTeX to do very precise things. So the results may look great or they may look terrible. And it is also possible to give LaTeX a spacing command that causes errors. So try to use these commands judiciously. It will take some practice to know when and how to use them.

7.1 Horizontal spacing

First, let me remind you that in text mode, when you put spaces between letters in your .tex file, then LaTeX will put *one* space in between. So if you type this in your .tex document:

```
aa a      a              a                        a
```

the output will look like this:

aa a a a a

LaTeX ignores all of the extra spaces after the first one in text mode. But when you are in math mode, LaTeX ignores *all* of the spaces *including* the first one. So typing this:

```
$xx x      x              x                        x$
```

will produce this:

[9] Making tables in latex really isn't very hard to do but it is a big topic with a lot of options and details. Once you learn the `align` environment and how to make matrices below, you will have the requisite knowledge to take on tables no problem. A matrix really is just a tiny table! If you are interested in learning how to use the `tabular` environment, (which is slightly more involved than what I am covering in this introduction) see http://en.wikibooks.org/wiki/LaTeX/Tables

xxxxxx

But what I didn't emphasize is that LaTeX adjusts the spacing between letters and symbols based on what the letters or symbols are. This is part of why LaTeX looks so good. It adjusts horizontal spacing to make the text look better depending on the context.

7.1.1 Basic fixed-length positive horizontal space

If you want to override the automatic horizontal spacing, then a basic way to do this is to use any one of the following seven *horizontal* spacing commands:

```
\,   \:   \;   \   \enspace   \quad   \qquad
```

The first three, `\,` `\:` and `\;`, create a small, medium, and large space that is fixed in length. The fourth is a backslash character followed by a space which creates a "control space" that tends to be a little wider than `\;` but might depend on your document settings (I'm not sure about this one...). The command `\enspace` `\quad` and `\qquad` create slightly larger spaces[10]. All six of these work in both math mode and text mode. Here is an example with various spaces in text mode and mathmode along with the resulting output. The first two letters in each line have no space between them for comparison.[11]

```
aa\,a\:a\;a\ a\enspace a\quad a\qquad a
mm\,m\:m\;m\ m\enspace m\quad m\qquad m
MM\,M\:M\;M\ M\enspace M\quad M\qquad M
$MM\,M\:M\;M\ M\enspace M\quad M\qquad M$
$xx\,x\:x\;x\ x\enspace x\quad x\qquad x$
$ii\,i\:i\;i\ i\enspace i\quad i\qquad i$
$\Pi\Pi\,\Pi\:\Pi\;\Pi\ \Pi\enspace \Pi\quad \Pi\qquad \Pi$
```

aa a a a a a a a a
mm m m m m m m m m
MM M M M M M M M M
$MM M M M M M M M M$
$xx x x x x x x x x$
$ii i i i i i i i i$
$\Pi\Pi\,\Pi\:\Pi\;\Pi\ \Pi\enspace \Pi\quad \Pi\qquad \Pi$

Note that the various lines of text in the results on the right-hand side have different lengths. This is caused by the *width of the letters I chose*, not the spacing. The space widths are in fact consistent in each line.

7.1.2 Basic negative space

The command `\!` creates a negative space, meaning in reduces the default amount of space between letters or symbols. In fact, it can cause letters or symbols to be so squished together that they overlap slightly. Here is an example in text mode and in math mode side-by-side followed by the result:

`xx\!x \quad $\Sigma\Sigma\!\Sigma$` ⇒ xxx $\Sigma\Sigma\Sigma$

Notice how the third x and Σ get scrunched up with the one before it. You will likely never use `\!` the way I just did. What I find it useful for is smashing together math symbols that you think are too spaced out by default. The following example shows how I like to use negative space *and* a little extra positive space to change the default spacing in an integral. Actually I will use *two* negative space commands back-to-back here because I want to double the horizontal "squishing." Compare the following two examples where the first version is the default and the second is my adjusted version. I show the code for both versions first then the results below. Since I want to number these I will use the `equation` environment instead of `\[ \]`.

[10] The space created by `\quad` is roughly the width of a capital M, `\enspace` is half of a `\quad` and `\qquad` is a double `\quad`.

[11] Notice how the following example flowed over into the right hand margin. This is an example of LaTeX trying to do exactly what I told it to do. I happen to be using the `tabular` environment here to line things up nicely and LaTeX tries to keep this environment together in one block without word-wrapping. This is an example of how you have far more power to force what you want with LaTeX than you do with any other word processing software. I happen to be o.k. with this example stretching past the main column into the right hand margin, but I am technically "breaking the rules" here.

```
\begin{equation}
\int_{a}^{b}\int_{\alpha}^{\beta}f(r,\theta) dr d\theta
\end{equation}

\begin{equation}
\int_{a}^{b}\!\!\int_{\alpha}^{\beta}\!f(r,\theta)\: drd\theta
\end{equation}
```

$$\int_a^b \int_\alpha^\beta f(r,\theta) dr d\theta \tag{6}$$

$$\int_a^b \int_\alpha^\beta f(r,\theta)\, dr d\theta. \tag{7}$$

In my opinion the default version, equation (6), has a little too much space between the two integral signs and a bit too much space between the second integral sign and the $f(r,\theta)$. And I think there should be a little *more* space between the $f(r,\theta)$ and the differential $drd\theta$. The second version, equation (7), makes these adjustments with the `\!` and `\:` commands. It is a subtle difference. I like the second one better but you might not care either way. This was just a simple example of how you might use negative space.

7.1.3 Choose your own width using \hspace{}

If none of the previous options work for what you are trying to do, you always have the following option which is more powerful than all of the previous methods and works in both math mode and text mode: `\hspace{}`. This function creates your own user defined horizontal spacing with a variety of units. If you type `\hspace{1in}` then you will add one inch of horizontal space like this:

```
``Space,\hspace{1in}the final frontier...''
```

"Space, the final frontier..."

There is one limitation to `\hspace{}` though. It turns out that you cannot use `\hspace{}` directly after a newline command (the reasons for this are technical and I don't fully understand them myself). For this reason there is an alternate version of this command: `\hspace*{}` You *can* use `\hspace*{}` right after a line break. Here is an example where `\hspace{}` doesn't work and you have to use the starred version instead. The code is followed by the result:

```
\noindent
Here is a non-indented line\\
\hspace{4cm}This did not get shifted to the right.\\
\hspace*{4cm}This \emph{did} get shifted to the right by four centimeters.
```

Here is a non-indented line
This did not get shifted to the right.
This *did* get shifted to the right by four centimeters.

You can also use negative values for the lengths to shift things to the left like we did above with `\!`, but this allows you to shift as far to the left as you want. Here is a silly example of shifting a word to the left:

```
\hspace*{15mm}Goulette\hspace{-2.4cm}David
```

David Goulette

The first `\hspace*` pushes my last name 2 cm to the right and then my first name "comes next" in the code but is shifted 2.4 cm to the left so it actually comes before my last name in the final document. So you can type stuff right over other things which might be a fun effect in some

instances. If you type this in your code:

```
This message\hspace{-2cm} is scrambled.
```

you will get this:

This message is scrambled.

There are six common units of length that people use in LaTeX: `in, cm, mm, pt, ex,` and `em`. The first three, inches, centimeters and millimeters, are self explanatory and I have already used them in the examples above. The unit `pt` is a "point" and is actually the standard unit of measurement in typography (think "12 point font"). The unit `ex` is approximately the height of a letter x in the current font and size. And the unit `em` is the width of an upper-case M.[12] Here are the six common units from shortest to longest:

```
A\hspace{1pt}A
A\hspace{1mm}A
A\hspace{1ex}A
A\hspace{1em}A
A\hspace{1cm}A
A\hspace{1in}A
```

A A
A A
A A
A A
A A
A A

These units can be used in any LaTeX function that takes a length as input (not just `\hspace{}`). For example, the `marginnote` command, which I have used many times in this document, has an optional input that shifts the margin note up or down by whatever length you want. There are lots of commands and environments that use lengths.

7.1.4 Flexible horizontal space with \hfill

An occasionally useful function is the rubber band spacing command `\hfill`. This function fills in the space between text objects to fill the horizontal space of the column. An example is best here. Suppose you want three text blocks in a line that has a block that is left justified, one that is centered and one that is right justified on the same line. Then you can do this in your code to get that result:

```
\noindent David Goulette \hfill Assignment 1 \hfill \today
```

David Goulette Assignment 1 April 23, 2014

So, if I were doing homework, this is an example of how I might put my name, title and date on one line. The `\hfill` function fills the space in the middle to make things balanced. So you could do this:

```
This\hfill is\hfill a simple\hfill example.
```

This is a simple example.

Note that because I did not use `\noindent` this time, the line is still indented. But since I used four text blocks with three `\hfill` commands in between, the blocks are evenly spaced between the indent and the right margin.

7.1.5 The function

I am introducing the `\phantom{}` function as a method of creating horizontal space but it turns out that `\phantom{}` has a lot of other uses. But let me first show you how to use it to create

[12] There actually are more units than this. For more on this subject see: http://en.wikibooks.org/wiki/LaTeX/Lengths

horizontal space. The `\phantom{}` function creates invisible text that takes up the space of "real" letters but you can't see them in the final document! Here is an example:

`abcdefg`	abcdefg
`ab\phantom{cde}fg`	ab fg

So the `\phantom{cde}` created an "invisible" cde that takes up exactly the amount of space the letters cde would take up, but you can't see them in the final document. So you can clearly use this as a method for creating horizontal space that matches a certain sequence of characters.

7.1.6 Examples of all types

The next two pages have a selection of horizontal spacing of various types shown vertically for easy comparison:

This space is intentionally left blank...
except, of course, for the fact that this stupid message
is in the space, which makes this a false statement.

←Note that I centered this horizontally with the `center` *environment. But I also centered it vertically in the remaining blank space on the page. I did this using* `\vfill`*, which is a vertical fill that works just like* `\hfill`*. I will explain this later in section 7.2.5 where we will discuss vertical spacing.*

Horizontal spacing examples in text mode

`xy`	xy
`x\!y`	xy
`x\,y`	x y
`x\:y`	x y
`x\;y`	x y
`x\enspace y`	x y
`x\quad y`	x y
`x\qquad y`	x y
`x\ y`	x y
`x\hspace{1pt}y`	xy
`x\hspace{1mm}y`	x y
`x\hspace{1ex}y`	x y
`x\hspace{1em}y`	x y
`x\hspace{1cm}y`	x y
`x\hspace{1in}y`	x y
`x\hspace{1.1in}y`	x y
`x\hspace{3.7cm}y`	x y
`x\hspace{37mm}y`	x y
`x\hspace{12pt}y`	x y
`x\hspace{7em}y`	x y
`x\hspace{5.3ex}y`	x y
`x\hspace{-1pt}y`	xy
`x\hspace{-2mm}y`	xy
`x\hspace{-4mm}y`	yx
`xabcdy`	xabcdy
`x\phantom{abcd}y`	x y
`xINVISIBLEy`	xINVISIBLEy
`x\phantom{INVISIBLE}y`	x y
`x\hfill y`	x y
`x\hfill y\hfill z`	x y z
`w\hfill x\hfill y\hfill z`	w x y z
`A rose\hfill is a\hfill rose.`	A rose is a rose.

Note that all of these are the same as in text mode. I only changed the last example.

Horizontal spacing examples in math mode

`$xy$`	xy
`$x\!y$`	xy
`$x\,y$`	$x\,y$
`$x\:y$`	$x\:y$
`$x\;y$`	$x\;y$
`$x\enspace y$`	$x\enspace y$
`$x\quad y$`	$x\quad y$
`$x\qquad y$`	$x\qquad y$
`$x\ y$`	$x\ y$
`$x\hspace{1pt}y$`	$x\hspace{1pt}y$
`$x\hspace{1mm}y$`	$x\hspace{1mm}y$
`$x\hspace{1ex}y$`	$x\hspace{1ex}y$
`$x\hspace{1em}y$`	$x\hspace{1em}y$
`$x\hspace{1cm}y$`	$x\hspace{1cm}y$
`$x\hspace{1in}y$`	$x\hspace{1in}y$
`$x\hspace{1.1in}y$`	$x\hspace{1.1in}y$
`$x\hspace{3.7cm}y$`	$x\hspace{3.7cm}y$
`$x\hspace{37mm}y$`	$x\hspace{37mm}y$
`$x\hspace{12pt}y$`	$x\hspace{12pt}y$
`$x\hspace{7em}y$`	$x\hspace{7em}y$
`$x\hspace{5.3ex}y$`	$x\hspace{5.3ex}y$
`$x\hspace{-1pt}y$`	$x\hspace{-1pt}y$
`$x\hspace{-2mm}y$`	$x\hspace{-2mm}y$
`$x\hspace{-4mm}y$`	$x\hspace{-4mm}y$
`$xabcdy$`	$xabcdy$
`$x\phantom{abcd}y$`	xy
`$xINVISIBLEy$`	$xINVISIBLEy$
`$x\phantom{INVISIBLE}y$`	xy
`$x\hfill y$`	$x\hfill y$
`$x\hfill y\hfill z$`	$x\hfill y\hfill z$
`$w\hfill x\hfill y\hfill z$`	$w\hfill x\hfill y\hfill z$
`$2x^2\hfill y_2\hfill \Psi$`	$2x^2\hfill y_2\hfill \Psi$

7.2 Vertical Spacing

You usually don't need to do vertical spacing quite as much as you do horizontal spacing so I will just give you some basics.

7.2.1 Changing the global line spacing

First, if you want to globally change the vertical line spacing then you can use the `\linespread{}` function to change the default line spacing. You have to use this function in the preamble to your document (*i.e.* it has to go before `\begin{document}`). If you put either of the following commands in your preamble it should change the global line spacing.

```
\linespread{1.3} % <== roughly one and a half line space
\linespread{1.6} % <== roughly double space
```

The default value is 1. Play around with different values to see what you like. Another (and probably better) option you have for either global OR local line spacing changes is to use the `setspace` package.[13]

7.2.2 Variable newline space with \\[length]

This is a simple and handy way to create a newline with a prescribed vertical space that follows the newline command. Instead of typing `\\` to get a regular new line, you can add an optional argument in square brackets like this `\\[length]`. The optional length will be the amount of space between the line the precedes the `\\` and the line that follows. The effect of the optional `[length]` command local. It only effects the single newline command that it is attached to.

```
\noindent
This is on a line,\\
and this is on the next line with the usual space.\\[1cm]
But this is on a new line after 1 cm of space. Pretty simple, right?
```

This is on a line,
and this is on the next line with the usual space.

But this is on a new line after 1 cm of space. Pretty simple, right?

Basically, all of the measurement rules you learned regarding horizontal spacing in section 7.1 apply when you are doing vertical spacing except for one important thing: since this is a *vertical* spacing command, positive lengths go in the *downward* direction and negative lengths go in the *upward* direction. So:

```
This message \\[-4.2mm] is scrambled.
```

This message
is scrambled.

The words "is scrambled" are on a new line but that new line is shifted *upward* by 4.2 mm from where it would normally be, which puts it right over the top of the previous line.

7.2.3 \smallskip, \medskip, and \bigskip

To force some vertical space between paragraphs you can use `\smallskip`, `\medskip`, or `\bigskip`. These commands need to be put in the right place and I don't completely understand why certain things work here and why some things don't but I will show you what works and

[13] For more details see: http://en.wikibooks.org/wiki/LaTeX/Text_Formatting

what doesn't. Basically, the skip commands need to come between two paragraphs. Remember that hitting the <enter> key (or <return> on a Mac) one time does not get you a new paragraph. If you hit <enter><enter> then you will get a new paragraph. Anyway, none of the following examples work correctly because the skip command is not between two paragraphs. I show each one followed by the result.

```
This\bigskip doesn't \medskip work.
```

Thisdoesn't work.

```
This \bigskip
doesn't \medskip
work\smallskip
either.
```

This doesn't workeither.

The way to make sure that these work is to either have an empty line of code directly following the skip function, or place the skip function right before the \\ command, or you do both (each option will have a different result). If you have a \\ preceding the empty line it will have the effect of adding an extra line *and then also* tacking on the skip space. The rules about indentation that we covered in section 3.2 still apply. Examples that work:

```
\noindent
This does work! \bigskip

This is indented after a big skip. \medskip

This is indented after a medium skip!
```

This does work!

This is indented after a big skip.

This is indented after a medium skip!

```
\noindent
This works too! \bigskip\\
This is not indented after a big skip.\smallskip

Indented after a small skip.
```

This works too!

This is not indented after a big skip.

Indented after a small skip.

```
\noindent
This is a different example. \smallskip\\

Indented after a space and a small skip. \bigskip\\

\noindent
This is not indented after a space and a big skip.\\
```

This is a different example.

Indented after a space and a small skip.

This is not indented after a space and a big skip.

In this last example note that the `\smallskip` made a small space and the `\\` combined with `<enter> <enter>` added an additional space. The same thing happened with the `\bigskip` but the skip is larger and I suppressed the indent manually. Compare this last example with the earlier examples and you will see the differences.

7.2.4 \vspace{} and \vspace*{}

If you want to force some vertical space locally with variable height then your best option is to use the `\vspace{}` function. It basically works just like `\hspace{}` except that now, positive lengths go downward and negative lengths go upward. But the one issue with `\vspace{}` is that you have to put it in the correct places to get it to work. Just like with `\smallskip`, `\medskip`, and `\bigskip`, you have to put `\hspace{}` between paragraphs. Examples:

```
\noindent First sentence.\vspace{1in}

Second sentence indented after 1 inch of space.
```

First sentence.

Second sentence indented after 1 inch of space.

```
\noindent First sentence.\vspace{1.3cm}\\
Second sentence not indented after 1.3 cm of space.
```

First sentence.

Second sentence not indented after 1.3 cm of space.

```
\noindent
First sentence.\vspace{-3mm}\\
Second sentence shifted UP 3 mm.
```

First sentence.
Second sentence shifted UP 3 mm.

That last example shows how negative lengths shift the next line up in the negative vertical direction.

One limitation to `\vspace{}` is that it will not work if you use it directly after a `\pagebreak`. If you want vertical space at the beginning of a page you need to use `\vspace*{}` instead, then it will work.[14]. The vertical space it creates comes after the top margin, meaning the vertical space will be added to the length of the top margin. Here some example code followed by the result:

```
This is on this page.

\pagebreak

\vspace*{1in}
This is on the next page, indented, after one inch of blank space below the top margin.
```

This is on this page.

[14]If you ever want to make nice multiple columns I suggest you use the `multicol` package. This great package allows you to do a `\columnbreak` to break to the next column (just like a page break). In this case, if you want vertical spacing right after the `\columnbreak` command, you need to use `\vspace*{}`. For more on the `multicol` package see the official documentation here: http://www.ctan.org/pkg/multicol

This is on the next page, indented, after one inch of blank space below the top margin.

7.2.5 \vfill

The command `\vfill` works just like `\hfill` but it does a rubber band spacing in the vertical direction. Just like with `\bigskip` and `\vspace{}`, you have to be careful where you stick the `\vfill` command. I'm not really sure why but lines always seem to be indented after a `\vfill`. Anyway, I have two examples that show how I use it. The first is a nice example of how to use `\vfill` along with the `center` environment on page 24. Here is a more basic example where I have 3 lines of text that are spread out to fill the remaining space on this page. The first line is at the top of the remaining space on this page, the third line is at the bottom, and the second line is right in the middle. All indentation rules still apply. The `\newline` command at the end is required to make sure that the block of text that follows is pushed to the next page:

```
This is a line of text at the top of the remaining space.\\
\vfill
This indented line is half way between the previous line and the bottom of the page.\\
\vfill
\noindent
This is not indented and is forced to the very bottom of the page.
\newpage
```

This is a line of text at the top of the remaining space.

This indented line is half way between the previous line and the bottom of the page.

This is not indented and is forced to the very bottom of the page.

8 More mathematics

Now we will return to cover a hodgepodge of mathematical topics. Some are very important and are used all of the time. Some only come up once in a while. For example, section 8.1 is extremely important for mathematics. But section 8.2 covers more uses of the `\phantom{}` function that might only come in handy occasionally.

8.1 Variable sized math grouping symbols

There are many different types of mathematical grouping symbols such as parentheses or brackets. The following are some examples where I show you on the left what to type in the .tex file and the output on the right. These examples (like every example in this section) are done in math mode, so these would have to be inside `$ $` for in-line math or inside a math environment for display math.

`$(x)$`	$\rightsquigarrow$	(x)
`$\{x\}$`	$\dashrightarrow$	$\{x\}$
`$[x]$`	$\Longrightarrow$	$[x]$
`$\langle x\rangle$`	$\rightarrow$	$\langle x\rangle$
`$\|x\|$`	$\longmapsto$	$\|\|x\|\|$
`$|x|$`	$\hookrightarrow$	$\lvert x\rvert$

The arrows here are meaningless except to say that what is on the left turns into what is on the right. I just used various arrows here to show you some variety.

Also there are some cases where an operator itself is as a pair of grouping symbols like the floor and ceiling functions.

`$\lfloor x \rfloor$`	$\rightarrowtail$	$\lfloor x\rfloor$
`$\lceil x \rceil$`	$\twoheadrightarrow$	$\lceil x\rceil$

Now, when you are doing most in-line math or display math that is not very tall, then these basic grouping symbols work just fine. So you can have something like $(2x+3)^2 = \lfloor x/7 \rfloor$ which looks great. Display math looks great in these basic situations as well:

$$\|x\|^2 = (x_1)^2 + (x_2)^2 + \cdots + (x_n)^2 = \langle x, x\rangle.$$

But suppose you want a large display fraction and you would like it to be in parentheses. If you type this:

```
\begin{equation}
(\frac{z^2}{\arccos(w)})^{2\pi i}
\end{equation}
```

then you will get this

$$(\frac{z^2}{\arccos(w)})^{2\pi i}, \tag{8}$$

which clearly doesn't look good. We really want the outer parentheses surrounding the fraction to be larger. Or, for another example, suppose you have a set containing various objects like this:

```
\begin{equation}
S=\{
    \{\Sigma,b\}, \{a,[x;y;z]\}, \mu,Z
  \}
\end{equation}
```

←Note how I put lines of code on new lines and use a few spaces to indent the nested grouping symbols. This is just to make the code more readable and easier to edit. Remember that LaTeX ignores spaces in math mode so it comes out right.

which gives you this:

$$S = \{\{\Sigma, b\}, \{a, [x; y; z]\}, \mu, Z\} \tag{9}$$

In truth, this example doesn't look too bad as it is. But it would be a little more readable (in my opinion) if we vary the size of the nested grouping symbols. So let's learn how to do it.

There are two basic options you have that will adjust the size of the grouping symbols in example equations (8) and (9) above. It turns out that the first option works well to fix equation (8) and the second option is better for (9).

8.1.1 `\left` and `\right`

The first option is to let LaTeX automatically adjust the size for you. You do this by using the pair of functions `\left` and `\right`. The `\left` and `\right` functions must directly precede the grouping symbol that you would like LaTeX to resize for you, like this:

```
\left( ***junk in the middle*** \right)
```

Also, you must have a `\left` paired with a `\right`. If you do a `\left` without a matching `\right` you will get errors. With this new tool in hand we can fix the parentheses in equation (8) by adding `\left` and `\right`:

I switched to using `\[ \]` here because I didn't want to number these equations like I did earlier with equation (8).

```
\[
\left(\frac{z^2}{\arccos(w)}\right)^{2\pi i}
\]
```

Now it looks better:

$$\left(\frac{z^2}{\arccos(w)}\right)^{2\pi i}.$$

These functions are great when you have tall constructs inside of grouping symbols. Even if you really stack some massive things up, then `\left` and `\right` will stretch things to make it work:

In my TeX editor, when I view large math in the .pdf viewer, it doesn't look as good as when I view the document in Adobe Reader. Sometimes the corners of root symbols don't quite line up, sometimes I see a few white dots in the middle of the large parenthesis, etc.. Don't worry. When you view and print with Adobe Reader it will look great.

$$\left(\frac{\oint_{\alpha}^{\beta} g(z)dz}{\sum_{i=0}^{n}\varphi(n)}\right)\cdot\left(\frac{\prod_{j=1}^{\infty}\Phi(a_j)}{\frac{d}{dx}\psi(x)\Big|_{x=42}}\right)^{p/q} = \nabla f(p_1,p_2,p_3).$$

Here are some examples of grouping symbols that you can use with `\left` and `\right`:

`\left[ \frac{x}{y} \right]`	$\rightsquigarrow$	$\left[\frac{x}{y}\right]$
`\left\lbrace \frac{x}{y} \right\rbrace`	$\dashrightarrow$	$\left\{\frac{x}{y}\right\}$
`\left\langle \frac{x}{y} \right\rangle`	$\Longrightarrow$	$\left\langle\frac{x}{y}\right\rangle$
`\left| \frac{x}{y} \right|`	$\rightarrow$	$\left\|\frac{x}{y}\right\|$
`\left\\| \frac{x}{y} \right\\|`	$\longmapsto$	$\left\|\frac{x}{y}\right\|\|$
`\left\lfloor \frac{x}{y} \right\rfloor`	$\longmapsto$	$\left\lfloor\frac{x}{y}\right\rfloor$
`\left\lceil \frac{x}{y} \right\rceil`	$\longmapsto$	$\left\lceil\frac{x}{y}\right\rceil$

For the sake of completeness, I will mention one more option you have with `\left` and `\right`. You can use a period to suppress one of the `\left` or `\right` grouping symbols. You just type `\left.` or `\right.` in place of one of the regular grouping symbols. Remember, that you must pair a `\left` and a `\right`. But one of the pairs can have a period causing it not to appear. Since I can't think of an example that is simple and useful (sorry!), here is a contrived example to understand the syntax:

```
\[
x_1+x_2+\cdots+x_n \Rightarrow
\left\{\;
\sum_{i=0}^n x_n
\right.
\]
```

$$x_1 + x_2 + \cdots + x_n \Rightarrow \left\{ \sum_{i=0}^{n} x_n \right.$$

The reason I can't come up with a simple example is that many of the situations where you would use a singe bracket have better solutions than using this method.[15] And the one place where I would use `\left.` or `\right.` unfortunately uses too many things that I haven't taught. So I won't explicitly show the code here (you can see it in the .tex file of course!). The `\left.` or `\right.` commands can be useful if you want a grouping symbol on only one side of some math with explanatory text on the other side. This example is silly but this is useful in serious situations sometimes.

This is not the only way to do this. This example has a few things that are somewhat more advanced.

$$\text{Cool equations from complex analysis.} \Longrightarrow \begin{cases} e^{i\theta} = \cos(\theta) + i\sin(\theta) \\ f(a) = \dfrac{1}{2\pi i} \oint_\gamma \dfrac{f(z)}{z-a} dz \end{cases}$$

Now the great part about `\left` and `\right` is that LaTeX does the resizing for you automatically. But the bad part (sometimes) is that the automatic results might not look good. And there are also cases where `\left` and `\right` won't change the default sizing at all (even when you are hoping it will). But fortunately you have a second option whenever you don't like the automatic results. Well, actually... you have four options. *And those options are...*

8.1.2 `\big`, `\Big`, `\bigg`, and `\Bigg`

You can manually force four different sizes of grouping symbols using the four functions found in the title of this sub-section. These sizes are fixed and they do not adjust to what is inside of them. LaTeX won't adjust things to make the results so it is up to you to make the results look good (which is subjective...). By using these functions you are forcing things and overriding automatic LaTeX behavior. So you are in control.

Here is an example comparing the default sizes to some forces sizes. This:

```
\[
(\frac{x^2}{17})^2  \:
\bigg(\frac{x^2}{17}\bigg)^2 \:
\Bigg(\frac{x^2}{17}\Bigg)^2 \qquad
\|\aleph\|\:
\Big\|\aleph\Big\|\:
\Bigg\|\aleph\Bigg\| \qquad
[x^2] \:
\big[x^2\big] \:
\Big[x^2\Big]
\]
```

gives you this:

$$(\frac{x^2}{17})^2 \bigg(\frac{x^2}{17}\bigg)^2 \Bigg(\frac{x^2}{17}\Bigg)^2 \qquad \|\aleph\| \Big\|\aleph\Big\| \Bigg\|\aleph\Bigg\| \qquad [x^2] \: \big[x^2\big] \: \Big[x^2\Big]$$

[15] A good example is the `cases` environment that is good for functions like the equation (5) on page 16.

Just like with `\left` and `\right`, you have to put these functions directly before the grouping symbol that you want to resize manually. But unlike `\left` and `\right` you *do not* have to worry about matching left and right bracket. So you could use these with a single unpaired grouping symbol if you want.

Here are all the sizes side by side with the same curly bracket. If you apply these sizes to left and right brackets, like this:

```
\Bigg\{ \bigg\{ \Big\{ \big\{ \{ \varnothing \} \big\} \Big\} \bigg\} \Bigg\}
```

then you will get something that resembles a Russian matryoshka doll with nothing inside:

$$\Bigg\{\bigg\{\Big\{\big\{\{\varnothing\}\big\}\Big\}\bigg\}\Bigg\}$$

This is an extremely empty set! Actually, this set is not empty at all... it is a set in a set in a set in a set in a set which happens to contain nothing.

But let me emphasize again that you will get these sizes no matter what. LaTeX isn't doing any automatic adjusting for you. So it might or might not look good. You just have to play around to see what you like. To let you see the size differences in display mode here is a table with all of them side by side. The first one is the default size and the second entry uses `\left` and `\right` so you can compare the cases. Note that `\left` and `\right` adjust to the size of the object it encloses but the rest have fixed height no matter what is in the middle. Here is a table to show you all of the options we have covered side-by-side.

Sizes:	Normal	`\left` `\right`	`\big`	`\Big`	`\bigg`	`\Bigg`
Examples	$(x)(\frac{m}{n})$	$(x)\left(\frac{m}{n}\right)$	$(x)(\frac{m}{n})$	$\big(x\big)\big(\frac{m}{n}\big)$	$\bigg(x\bigg)\bigg(\frac{m}{n}\bigg)$	$\Bigg(x\Bigg)\Bigg(\frac{m}{n}\Bigg)$

Now that we know both methods, lets go back and fix how equation (9) looks. It turns out that if you try to use `\left` and `\right` they won't change the look at all. So typing this:

```
\[
S=\left\{
    \left\{\Sigma,b\right\}, \left\{a,[x;y;z]\right\}, \mu,Z
  \right\}
\]
```

will give you this:

$$S=\left\{\left\{\Sigma,b\right\},\left\{a,[x;y;z]\right\},\mu,Z\right\}$$

which looks *exactly* the same as (9). That is because there are no stacked constructs (like a fraction) or very tall operators (like an integral sign). But to improve readability I prefer to vary the bracket sizing like this:

```
\[
S=\bigg\{
    \big\{\Sigma,b \big\}, \big\{a,[x;y;z] \big\}, \mu,Z
  \bigg\}
\]
```

which gives you this:

$$S=\bigg\{\big\{\Sigma,b\big\},\big\{a,[x;y;z]\big\},\mu,Z\bigg\}$$

Notice that I kept the innermost square brackets at normal size and increased the sizing outward. It is a matter of taste but I think this final version is easier to read. You may not agree (or care one way or the other) and that is fine. *Vive la différence!*

Here is one last case where `\left` and `\right` actually do not look nearly as good (in my opinion) as forcing the size you want (this is a modified version of an example found in the AMS-math documentation). Consider this:

```
\[
\left[ \sum_i a_i
   \left\lvert \sum_j b_{ij}
   \right\rvert^q
\right]^{1/3}
\]
```

which gives you this:

$$\left[\sum_i a_i \left\lvert \sum_j b_{ij} \right\rvert^q \right]^{1/3}$$

Forcing the sizing using `\Bigg` and `\bigg` looks better:

```
\[
\Bigg[ \sum_i a_i
   \bigg\lvert \sum_j b_{ij}
   \bigg\rvert^q
\Bigg]^{1/3}
\]
```

This is the result:

$$\Bigg[\sum_i a_i \bigg\lvert \sum_j b_{ij} \bigg\rvert^q \Bigg]^{1/3}$$

I think the second version looks much better because the brackets and the absolute value aren't so tall. Also, the q in the exponent is tucked under the right bracket better in the second version.

Now, I only fuss over these things when I am finishing up a document that really matters. You shouldn't waste time worrying about minutia when you are drafting your documents. These are the things that are easy to go back and fix after you have your ideas down and you want to really make your work look professional. In general I try to make the document as readable as possible and there are often a variety of choices. Sometimes the differences are subtle. I will close this section with a final side-by-side comparison. Here I have three options of parentheses side-by-side:

$$(\sqrt{2}-\sqrt{x})^2 = \big(\sqrt{2}-\sqrt{x}\big)^2 = \Big(\sqrt{2}-\sqrt{x}\Big)^2.$$

The first of the three is the default size and the parentheses are a little too small. Either the second one, using `\big`, or the third one, using `\left` and `\right`, look better to my eye. Take your pick.

8.2 Fun ways to use invisible stuff with {} or

We have seen the use of `\phantom{}` earlier in the horizontal spacing section but there are occasionally times when you might want to use a `\phantom{}` to help you do other things as well. Also, you can use empty curly braces `{}` in math mode as well, which is more or less equivalent to using `\phantom{}` without any inputs at all! If you use `\phantom{}` with no inputs you get a character with no width. But why would you ever want to do that? Well the nice thing is that

you can give an invisible `\phantom{}` character *visible* subscripts and/or superscripts. So you can do things like this:

`${}_a^b$` will give you this: ${}_a^b$

Or equivalently
`$\phantom{}_a^b$` will give you this: ${}_a^b$

They are essentially the same. In this example we are using an invisible letter (with no width since the brackets are empty) to force a subscript and a superscript that just seem to float in space. Now it may not seem like this would be very useful, but what if you want a subscript and superscript that *precedes* a letter or symbol? This is common in chemistry but also hypergeometric functions and combinatorics use preceding subscripts occasionally. The easy way to do this is with `{}` and/or `\phantom{}`. Here is an example of one common notation for "5 choose 3," meaning all possible combinations of 3 things chosen from a set with 5 things:

`${}_5 C_3$` gives you this result: ${}_5C_3$

So the phantom character lets you create the subscript 5 which follows directly before the letter C. Since LaTeX ignores the space, the 5 and the C get squished together just like you would want them to be.

By-the-way, let me digress since I mentioned combinations. You should know that the binomial coefficient notation is also available via the AMS packages. The function is `\binom{}{}`:

```
$\displaystyle {}_n C_k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$.
```

$$ {}_nC_k = \binom{n}{k} = \frac{n!}{k!(n-k)!}. $$

The following is an example from chemistry that you might like to recreate:

$$ {}^{238}_{92}\mathrm{U} $$

Based on the example above, you might try to do this:

`${}^{238}_{92}\mathrm{U}$` but that will give you this ${}^{238}_{92}\mathrm{U}$

which isn't quite get the right result. The preceding 238 looks correct but the problem is that we want the 92 to be shifted to the right by the same width as the 2. But this is exactly what `\phantom{}` does for you! We just need to insert a `\phantom{2}` in front of the 92 like this:

`${}^{238}_{\phantom{2}92}\mathrm{U}\quad $` which gives you ${}^{238}_{92}\mathrm{U}$

and the problem is solved. Now there is a little invisible 2 that is helping line things up perfectly.

8.3 Inserting text inside of math

There are many circumstances where you want to insert regular text inside of a math environment. There are a lot of ways to do this but in this section I am just going to give you some basic easy methods. Inserting text inside of math is often not an issue when you are working with in-line math mode because you can just switch in and out of math mode with dollar signs. So it is easy do things like:

```
Let $f(x)=x^2$ and $g(x)=x+2$. Then $f$ composed with $g$ is
$(f \circ g)(x)=(x+2)^2$.
```

Let $f(x) = x^2$ and $g(x) = x + 2$. Then f composed with g is $(f \circ g)(x) = (x + 2)^2$.

But sometimes you want text inside of math where you can't easily leave math mode. Often you just want one or two words, like "and," "or," "such that," or "therefore" in the middle of your math. To do this use the `\text{}` function which is available via the AMS math package.

```
$A=\{x+iy\in\mathbb{C}\,|\,x\in\mathbb{Q}\ \text{and}\ y > 0\}$
```

$A = \{x + iy \in \mathbb{C} \mid x \in \mathbb{Q} \text{ and } y > 0\}$

← *Note that I am using some alternate script styles and two types of forced spacing here. Remember that you need to force horizontal spacing in math mode.*

Sometimes I use text inside of math as a pedagogical aid for my students. Here is something I would use for my statistics students who are learning to calculate Z-scores:

```
$\displaystyle Z =
\frac{\text{``sample statistic''} -\text{``sampling distribution mean''}}
{\text{``standard error''}}
=\frac{\widehat{p}-\mu_{\widehat{p}}}{\sigma_{\widehat{p}}}$
```

$$Z = \frac{\text{“sample statistic”} - \text{“sampling distribution mean”}}{\text{“standard error”}} = \frac{\widehat{p} - \mu_{\widehat{p}}}{\sigma_{\widehat{p}}}$$

Another common way you might want to enter text inside of math mode happens within the `align` environment. You often want to interject a quick comment in the middle of some aligned equations but the problem is that if you leave the `align` environment to type the text, then you lose your alignment settings. So then you can't align the equations that follow. The AMS math packages have a command `\intertext{}` that allows you to do this. Here is an example of how I might use this to explain something to a College Algebra class. Suppose I want to show the steps to solving $2(x - 3)^2 - 3 = 15$. Then I might do this:.

```
\begin{align*}
2(x-3)^2-3 &= 15\\
2(x-3)^2 &= 18\\
2(x-3)^2 &= 18\\
(x-3)^2 &= 9.\\
\intertext{Taking square roots of both sides gives us}
x-3 &= \pm\sqrt{9}\\
x-3 &= \pm 3,\\
\intertext{so}
x &= 3\pm 3,\\
\intertext{therefore}
x = 0\quad&\text{and}\quad x = 6
\end{align*}
are the solutions.
```

$$\begin{aligned}
2(x-3)^2-3 &= 15\\
2(x-3)^2 &= 18\\
2(x-3)^2 &= 18\\
(x-3)^2 &= 9.\\
\intertext{Taking square roots of both sides gives us}
x-3 &= \pm\sqrt{9}\\
x-3 &= \pm 3,\\
\intertext{so}
x &= 3\pm 3,\\
\intertext{therefore}
x = 0\quad&\text{and}\quad x = 6
\end{aligned}$$
are the solutions.

so

$$x = 3 \pm 3,$$

therefore

$$x = 0 \quad \text{and} \quad x = 6$$

are the solutions.

Note how the alignment (using the & character) continues even after the use of `\intertext{}`. Also note how I used `\text{}` to put the word "and" in the middle of the last line. I also used some `\quad` space and an & to line the "and" up with the equals signs from above.

All of the examples I have shown you so far come up quite a bit so they will be useful. But unfortunately any other example of putting text inside of math that I ever need requires some things that are a little more advanced than I want to cover here. Specifically they require the use of boxes: `\parbox`, `\fbox`, `\framebox`, `\raisebox`, etc. I used `\raisebox` to make my silly Alice example on page 9. But that didn't have math. I use a `\parbox` to do this:[16]

There is a 1-1 correspondence between

$$\left\{ \begin{array}{c} \text{isomorphism classes of} \\ \text{connected coverings} \\ F : U \to V \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{c} \text{conjugacy classes} \\ \text{of subgroups} \\ H \subseteq \pi_1(V, q) \end{array} \right\}$$

Using a `parbox` allowed me to put word-wrapped text inside of math braces with a mini paragraph (which is what the "par" means in `parbox`). For more on boxes see:
http://en.wikibooks.org/wiki/LaTeX/Boxes

8.4 Under-stuff and over-stuff

There are many times in mathematics where you want a line, hat, arrow or brace on top of your letter (or sometimes below). Like these simple examples:

```
\[\vec{x}\quad \dot{f}\quad  \ddot{g}\quad \widetilde{\eta}\quad  \hat{z}\quad
\widehat{p}\quad \overleftarrow{ab}\quad  \overrightarrow{pq}
\overline{A}\quad  \underline{B}\quad  \underrightarrow{\Gamma}\quad \]
```

$$\vec{x} \quad \dot{f} \quad \ddot{g} \quad \widetilde{\eta} \quad \hat{z} \quad \widehat{p} \quad \overleftarrow{ab} \quad \overrightarrow{pq} \quad \bar{\omega} \quad \overline{AB} \quad \underline{B} \quad \underrightarrow{\Gamma}$$

Some of these will extend to be as wide as the letters in the argument of the function but some are not extensible. For instance `\hat{}` will only cover one character but `\widehat{}` can cover as many as you want (well... there are limits to how wide you can go...). In fact these two commands look different even over one letter. So you might like the look of one over the other in different situations. Here are some examples comparing the two:

```
$\hat{p}\quad \widehat{p}\quad \hat{xyz}\quad \widehat{xyz}$
```

$\hat{p} \quad \widehat{p} \quad \hat{xyz} \quad \widehat{xyz}$

Here are some other extensible over/under braces:

```
\[\widetilde{ABCD}\quad\overleftrightarrow{wxyz}
\quad\overline{\omega \zeta \theta}\quad\overbrace{MNOP}
\quad\underbrace{QRSTUV}\quad\underbracket{AB\Gamma\Delta}\]
```

$$\widetilde{ABCD} \quad \overleftrightarrow{wxyz} \quad \overline{\omega\zeta\theta} \quad \overbrace{MNOP} \quad \underbrace{QRSTUV} \quad \overrightarrow{x_0x_1} \quad \underbracket{AB\Gamma\Delta}$$

[16] This clever use of words inside set braces came right from a book I happen to be reading today: *Algebraic Curves and Riemann Surfaces* by Rick Miranda. Good book by-the-way.

This is a case where the extended under-stuff and over-stuff do not look good in my LaTeX viewer. It looks much better in Adobe Reader, but it still isn't absolutely perfect even there (but it isn't too bad). There are packages that improve this if you *really* need it to look publishing perfect or it just bothers you.[17]

Having these variable length over and under things makes it easy to do some useful things. For example, an extensible bar is useful for complex variables (this also gives us a chance to use big parentheses too):

```
\[
\overline{(e^{-iz}+w)^2} =
\Big(\overline{e^{-iz}+w}\Big)^2 =
\Big(\overline{e^{-iz}}+\overline{w}\Big)^2 =
\Big(e^{\overline{-iz}}+\overline{w}\Big)^2 =
\Big(e^{i\overline{z}}+\overline{w}\Big)^2
\]
```

$$\overline{(e^{-iz}+w)^2} = \left(\overline{e^{-iz}+w}\right)^2 = \left(\overline{e^{-iz}}+\overline{w}\right)^2 = \left(e^{\overline{-iz}}+\overline{w}\right)^2 = \left(e^{i\overline{z}}+\overline{w}\right)^2$$

Sometimes, for more advanced documents, you want more complicated stacked constructs. Fortunately there are many ways to do this. But I will only show a few here. One way is to use the subscript command after the `\underbrace{}` command or use the superscript command after the `\overbrace{}`. These have the effect of putting the text above or below the brace. In this example I use a brace and a bracket, and I also use the `\text{}` command we learned in the last section, plus I use some horizontal spacing.

```
\[
\overbrace{f(x)=2x^3+3x^2+2x-2}^{\text{polynomial function}}
\hspace{1cm}
\underbracket{\Phi(x)= 2xe^x+13\cos(2x+35)+x^{\pi}}_{\text{transcendental function}}
\]
```

$$\overbrace{f(x)=2x^3+3x^2+2x-2}^{\text{polynomial function}} \qquad \underbracket{\Phi(x)= 2xe^x+13\cos(2x+35)+x^{\pi}}_{\text{transcendental function}}$$

When LaTeX tries to fit everything in, it can do some strange horizontal spacing. So here is another example where I will show the unedited and the edited versions next to each other. I improved the output with some horizontal spacing adjustment. Fortunately this sort of micro-editing doesn't come up too often. But when you want to do something special, you might need the skills to make it look good.

```
\[
f(z)=f(x+iy)=
\underbrace{u(x,y)}_{\text{Real part}} +
i\underbrace{v(x,y)}_{\text{Imaginary part}}
\]
```

$$f(z)=f(x+iy)= \underbrace{u(x,y)}_{\text{Real part}} + i\underbrace{v(x,y)}_{\text{Imaginary part}}$$

```
\[
f(z)=f(x+iy)=
\underbrace{u(x,y)}_{\text{Real part}} +\;
i\hspace{-7mm}\underbrace{v(x,y)}_{\hspace{5mm}\text{Imaginary part}}
\]
```

$$f(z)=f(x+iy)= \underbrace{u(x,y)}_{\text{Real part}} +\; i\hspace{-7mm}\underbrace{v(x,y)}_{\hspace{5mm}\text{Imaginary part}}$$

[17] The Math Time Pro 2 package, created by Michael Spivak, has really elegant looking braces (and lots of other features). Unfortunately the full package is not free. However the "Lite" version is free and has the nice braces. Both the free and pay versions are available here: http://www.pctex.com/mtpro2.html

$$f(z) = f(x+iy) = \underbrace{u(x,y)}_{\text{Real part}} + i \quad \underbrace{v(x,y)}_{\text{Imaginary part}}$$

$$f(z) = f(x+iy) = \underbrace{u(x,y)}_{\text{Real part}} + i\underbrace{v(x,y)}_{\text{Imaginary part}}$$

Another useful pair of commands is the `\underset` and `\overset` functions. These both take two inputs. You need, `\underset{under-stuff}{regular-stuff}` and the analogous inputs for `\overset`. There are times when you might want something under or over another thing that isn't standard. This happens all the time in advanced mathematics when you are making things up that have no standard notation. Maybe you want a math squiggle *below* your variable. You can do this with the command `\sim` which usually is used for things like `$p\sim q$`, which gives you $p \sim q$. But this is how to hack an under-squiggle:

`\underset{\sim}{x}` to get $\underset{\sim}{x}$

So the x is in-line in the normal place, and the twiddle is below the x. By-the-way, I don't love the amount of space between the x and the squiggle underneath so this is a perfect place to use a `\phantom{}` character! If we use `\widetilde` over a phantom character but put the whole thing in the `underset` *below* the x then it will push the squiggle up! (O.k. it's a hack... I know. But it is good to learn some hacks.) Now, another thing about `\underset` and `\overset` is that the script inside of the under/over set is the size of an exponent. So if you want it to be normal in-line size, then you have to force it with `\displaystyle`. Compare the following versions of putting a twiddle underneath the x:

```
$\underset{\sim}{x} \qquad
\underset{\widetilde{x}}{x} \qquad
\underset{\widetilde{\phantom{x}}}{x} \qquad
\underset{\displaystyle \widetilde{x}}{x} \qquad
\underset{\displaystyle \widetilde{\phantom{x}}}{x} \qquad
\widetilde{x}$
```

$\underset{\sim}{x} \qquad \underset{\widetilde{x}}{x} \qquad \underset{\widetilde{}}{x} \qquad \underset{\displaystyle \widetilde{x}}{x} \qquad \underset{\displaystyle \widetilde{}}{x} \qquad \widetilde{x}$

The first example is the one with `\sim` like before. I threw in the second example, with the visible x, just so you how the x in the underset was functioning. But then I made it vanish in the third one. But it is too small, so the fourth example shows the size fix, and the fifth one is the result I was looking for. Note that the fifth version matches the sixth example which is the twiddle that you get *over* the x in the normal way. You would would want the under-twiddle to match the over-twiddle.

Just recently I was documenting a graph theory algorithm that I was coding and it had two parts which found ascending and descending edges connecting vertices. Anyway, to make it clear which version of the algorithm I was using, I differentiated the results of the two methods with:

`$v_i\overset{+}{\rightarrow}v_j$` and `$v_i\overset{-}{\rightarrow}v_j$`

$v_i \overset{+}{\rightarrow} v_j$ and $v_i \overset{-}{\rightarrow} v_j$

This made my documentation much clearer than relying on the reader to know in context which version of the algorithm I was using. Good notation is often the key to clear technical writing. So the more you use LaTeX the more creative ways you will find to make your documents clear and expressive.

Here is a slightly more involved example which I might create for an algebra student to remind them of the steps required for completing the square of a quadratic expression. Again,

this is a chance for us to apply a variety of things we have learned. One thing I do here is use `\underset` inside of another `\underset` in order to get text below and arrow which is below one of the coefficients. Also I use some `\!` for negative space to squeeze things together a bit (this is not necessary). And I also needed variable sized parentheses:

```
\[
x^2+\!\underset{\underset{b=3}{\uparrow}}{3}\!\!x+2 =
x^2+3x+\overbrace{\underbrace{\left(\frac{3}{2}\right)^2}_{(b/2)^2}\!\!
-\left(\frac{3}{2}\right)^2
^{\text{\footnotesize add and subtract}}\!\!+\, 2 =
\overbrace{x^2+3x+\left(\frac{3}{2}\right)^2
^{\text{\footnotesize Perfect square}}\!- \frac{1}{4}=
\left(x+\frac{3}{2}\right)^2\!-\frac{1}{4}
\]
```

$$x^2+\underset{\underset{b=3}{\uparrow}}{3}x+2 = x^2+3x+\overbrace{\underbrace{\left(\frac{3}{2}\right)^2}_{(b/2)^2}-\left(\frac{3}{2}\right)^2}^{\text{add and subtract}}+2 = \overbrace{x^2+3x+\left(\frac{3}{2}\right)^2}^{\text{Perfect square}}-\frac{1}{4} = \left(x+\frac{3}{2}\right)^2-\frac{1}{4}$$

8.5 Matrices

Now that you know what an environment is, you know what display math is, you are aware of the AMS packages, *and* you have seen how the ampersand character works, you have all the building blocks for matrices! And it even helps that you know how to adjust the spacing if needed (since sometimes you need it with matrices). There are a variety of ways you can do matrices but the best way is to use any one of the following environments: `matrix`, `bmatrix`, `pmatrix`, `Bmatrix`, `vmatrix`, or `Vmatrix`. Note that I said these are environments, so you will need to do `\begin{pmatrix}` and end it with `\end{pmatrix}`. The entries of the matrix go in between. Unlike the `align` environment or the `equation` environment which are stand-alone math mode environments, you have to put a matrix inside $ $ or inside a display math environment like `\[ \]` or `\begin{equation} *** \end{equation}`. That is why my examples begin and end with commands that invoke the math mode. Here is the syntax for a matrix with brackets:

```
\[
\begin{bmatrix}
a & b\\
c & d
\end{bmatrix}
\]
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Note that the ampersand & is used to split the columns and line things up. Also you need a newline, `\\`, command to move to a new row. So if you want more columns, use more &, if you want more rows, use more `\\`. The brackets are created by the `bmatrix` environment itself.

The following example shows a rotation matrix which rotates a vector around the y-axis. The equation is show with parentheses matrices. It requires a matrix environment for each matrix:

```
\[
\begin{pmatrix}
\cos(\theta) & 0 & -\sin(\theta)\\
0 & 1 & 0\\
\sin(\theta) & 0 & \cos(\theta)
\end{pmatrix}
\begin{pmatrix}
u_1\\
u_2\\
u_3
\end{pmatrix}
=
\begin{pmatrix}
v_1\\
v_2\\
v_3
\end{pmatrix}
\]
```

$$\begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Notice that I had two ampersands in each row of the first matrix because there were 3 columns. Also note that you do not need the ampersands to line up. LaTeX will do it for you. Writing code for matrices is a little different that anything else because you have to think vertically even though it will be printed horizontally. To help make things clearer I put the equals sign on it's own line. Here are the other versions of matrices that are available. The only difference between the different versions is what type of grouping symbol they put around the entries (or nothing at all in the first case). The syntax for filling in the entires is exactly the same.

`\begin{matrix}`	$\begin{matrix} a & b \\ c & d \end{matrix}$
`\begin{Bmatrix}`	$\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$
`\begin{vmatrix}`	$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$
`\begin{Vmatrix}`	$\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$

You can have empty spots in a matrix and you can also use `\cdots`, `\vdots`, and `\ddots`, which will give you $\cdots$, $\vdots$, and $\ddots$ inside of your matrix. The next example is a general linear transformation $T : \mathbb{R}^n \to \mathbb{R}^m$. It is shown here as an $m \times n$ matrix multiplied by an n dimensional vector. The result is m dimensional. Admittedly, this is an involved example, but it comes up if you do any matrix algebra. And I wanted to show you that you can put more complicated things inside of a matrix like a summation. Note the way I space things in the code to line up the ampersands. This is only for readability and is not necessary. I also use some vertical spacing, blank spaces, and lots of dots.

```
\[
\begin{pmatrix}
a_{11}& a_{12} & \cdots & a_{1n}\\
a_{21}& a_{22} &        & \vdots \\
\vdots&        & \ddots & \vdots\\
a_{m1}& \cdots & \cdots & a_{mn}
\end{pmatrix}
\cdot
\begin{pmatrix}
x_1\vspace{1mm}\\
x_2\\
\vdots
\vspace{1mm}\\
x_n
\end{pmatrix}
=
\begin{pmatrix}
\sum_{i=1}^{n}a_{1i}\, x_i\vspace{2mm}\\
\sum_{i=1}^{n}a_{2i}\, x_i\\
\vdots\\
\sum_{i=1}^{n}a_{mi}\, x_i
\end{pmatrix}
\]
```

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} a_{1i}\, x_i \\ \sum_{i=1}^{n} a_{2i}\, x_i \\ \vdots \\ \sum_{i=1}^{n} a_{mi}\, x_i \end{pmatrix}$$

Sometimes there are just too many entries to line things up so you just have to be careful with your code. The default limit to the number of columns is 10.[18] A matrix in Jordan canonical form has the following form where any empty space is a zero:

```
\[
\begin{bmatrix}
\lambda_1 &&&&&&&&&\\
& \lambda_2  & 1 &&&&&&&\\
&& \lambda_2 &&&&&&&\\
&&& \ddots &&&&&&\\
&&&& \lambda_i & 1 &&&&\\
&&&&& \lambda_i  & 1 &&&\\
&&&&&& \lambda_i &&&\\
&&&&&&& \ddots &&\\
&&&&&&&& \lambda_{n-1} &\\
&&&&&&&&& \lambda_n \\
\end{bmatrix}
\]
```

[18] You can increase the default number of matrix columns allowed. See: http://www.latex-community.org/forum/viewtopic.php?f=46&t=11214

$$\begin{bmatrix}
\lambda_1 & & & & & & & & \\
& \lambda_2 & 1 & & & & & & \\
& & \lambda_2 & & & & & & \\
& & & \ddots & & & & & \\
& & & & \lambda_i & 1 & & & \\
& & & & & \lambda_i & 1 & & \\
& & & & & & \lambda_i & & \\
& & & & & & & \ddots & \\
& & & & & & & & \lambda_{n-1} & \\
& & & & & & & & & \lambda_n
\end{bmatrix}$$

What if you are doing some more advanced matrix theory with block matrices, in other words, you need matrices inside of matrices? No problem! Here is a 5×6 matrix represented as a 2×2 in block notation (by-the-way, I_2 is shorthand for a 2×2 identity matrix and $0_{3\times4}$ is shorthand for a 3×4 matrix filled with zeros):

$$\begin{bmatrix} A_{3\times2} & B_{3\times4} \\ C_{2\times2} & D_{2\times4} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix} & 0_{3\times4} \\ I_2 & \begin{pmatrix} \alpha & \beta & \gamma & \delta \\ \epsilon & \zeta & \eta & \theta \end{pmatrix} \end{bmatrix}$$

Here is the code for this example:

```
\[
\begin{bmatrix}
A_{3\times2}& B_{3\times4}\\
C_{2\times2} & D_{2\times4}
\end{bmatrix}
=
\begin{bmatrix}
\begin{pmatrix}
a & b\\
c & d\\
e & f
\end{pmatrix}
&
\mathit{0}_{\,3\times 4}
\\
I_2
&
\begin{pmatrix}
\alpha & \beta & \gamma & \delta\\
\epsilon & \zeta & \eta & \theta
\end{pmatrix}
\end{bmatrix}
\]
```

If you really scrutinize this code you will see that the `bmatrix` on the right hand side of the equals sign is a 2×2 matrix (there is only one &, meaning two columns, and only one `\\` meaning two rows). It just so happens that the upper left-hand entry is a `pmatrix`, and the lower right hand entry is also a `pmatrix`.

The final example in this section utilizes many things we have learned in an example that might look familiar to you (assuming you have taken a basic multivariable calculus course). This is a formal mnemonic to remember how to compute the cross product of two vectors $\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{v}} \in \mathbb{R}^3$. This example uses the align environment, two types of matrices, bold math letters, two types of over arrows (I put bigger ones on the variables and smaller arrows on the unit basis vectors), and also some spacing just to make things look nice. We have really covered a lot of material!

$$\begin{aligned}
\overrightarrow{\mathbf{u}} \times \overrightarrow{\mathbf{v}} &= [u_1 \;\; u_2 \;\; u_3] \times [v_1 \;\; v_2 \;\; v_3] \\
&= \det \begin{pmatrix} \vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{pmatrix} \\
&= \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} \vec{\mathbf{i}} - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix} \vec{\mathbf{j}} + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} \vec{\mathbf{k}} \\
&= (u_2 v_3 - v_2 u_3)\, \vec{\mathbf{i}} - (u_1 v_3 - v_1 u_3)\, \vec{\mathbf{j}} + (u_1 v_2 - v_1 u_2)\, \vec{\mathbf{k}}
\end{aligned}$$

Here is the code for the previous example:

```
\begin{align*}
\overrightarrow{\mathbf{u}} \times \overrightarrow{\mathbf{v}}
&= [u_1\;\; u_2\;\; u_3]\times [v_1\;\; v_2\;\; v_3]\\[7pt]
&= \det \begin{pmatrix}
\vec{\mathbf{i}} & \vec{\mathbf{j}} & \vec{\mathbf{k}}\\
u_1 & u_2 & u_3\\
v_1 & v_2 & v_3
\end{pmatrix}\\[7pt]
&=
\begin{vmatrix}
u_2 & u_3\\
v_2 & v_3
\end{vmatrix}\vec{\mathbf{i}}
-
\begin{vmatrix}
u_1 &  u_3\\
v_1 &  v_3
\end{vmatrix}\vec{\mathbf{j}}
+
\begin{vmatrix}
u_1 & u_2 \\
v_1 & v_2
\end{vmatrix}\vec{\mathbf{k}}\\[4pt]
&=
(u_2 v_3-v_2 u_3)\,\vec{\mathbf{i}}-
(u_1 v_3-v_1 u_3)\,\vec{\mathbf{j}}+
(u_1 v_2-v_1 u_2)\,\vec{\mathbf{k}}
\end{align*}
```

9 Labeling and referencing things

One of the really great things that LaTeX can do for you is help you reference things like sections, equations, theorems, definitions, figures, etc. etc. You know,... you might want to type something like, "Figure 3 below is a photograph of some dark matter..." Or maybe you want to reference a page number like, "Later, on page 56, we will prove that P = NP..." Or

maybe you want to reference an equation that you had earlier in your paper like, "If we substitute 42 for x in equation (7) on page 17, we will clearly prove that the Riemann Hypothesis is true." Stuff like that comes up all the time. Well, the numbers that I just typed in those examples were hand typed. I wasn't using dynamic referencing in the proper way. You do not have to create any of the numbering manually. LaTeX will do that for you just like it numbers equations for you. I will not cover how to reference a figure or a theorem here. I will just cover how to reference a section and an equation here.[19] All other referencing you might do is similar.

In order to reference something, you first have to label it. The way you label something in your code is by adding this: `\label{labelname}`. The "`labelname`" can be any name that you want to give it (you can't use special characters). You just have to put the `\label` right after whatever you are labeling. Then later you can reference this label with `\ref{labelname}`, `\eqref{labelname}` or `\pageref{labelname}`. But first, here is an example of how you label a section, just put it right after the section command like this:

```
\subsection{How to label and reference a section}
\label{SectionOnLabelingSections}
This section of the paper shows you how to add a label to a section so that
you can reference it.
```

This is what you will see in the document output:

9.1 How to label and reference a section

This section of the paper shows you how to add a label to a section so that you can reference it.

Note that you can't see the label in the final document. The label is just a marker in the code. So if you want to reference this section somewhere in the paper you just have to reference this label that we just created. To do the reference you just have do the following (I will explain my use of the ~ character in a parenthetical aside below the example... I have never explained what it does up to now):

```
You are currently reading section~\ref{SectionOnLabelingSections}.
```

LaTeX will take this code and convert the `\ref` command into the number of the object that is referenced inside the braces. So you will see this as a result:

You are currently reading section 9.1.

(Let me make a parenthetical side comment regarding the tilde character. The ~ character creates a non-breaking space. You do not *have to* have the ~ here in this reference; you could just do this:

```
You are currently reading section \ref{SectionOnLabelingSections}.
```

But I recommend adding the tilde. When you put a ~ between normal text characters in your code, LaTeX will create a space where the ~ is (just like it normally would if you put a space in the code with space bar), but LaTeX *will not* allow a line break where you have a ~. You wouldn't want a line break to split up the word "section" from the number, like this: section 9.1. That looks strange. One would prefer the 9.1 to come right after the word "section." The ~ character "glues" them together to make sure the awkward line break won't happen between them. If a word wrap is needed, the whole block will be sent to the new line.)

Now, with basic LaTeX, your section reference won't be colored blue like it is in the above example and it won't be a live link either. You might remember that in section 5 above, I mentioned that I am using the `hyperref` package in this document. The `hyperref` package

← Note how I referenced an earlier section here. Very useful!

[19] I will cover how to add figures in another document and I will mention how to label them and reference them in that document. It is similar to labeling equations.

makes your references live links and I chose to have them colored blue to make it obvious that they were links (see the preamble for more details). So if you make labels without using `hyperref`, then the numbers will be black.

Instead of referencing the section number, you can alternatively reference the page that the section label is on by doing this:

```
Section~\ref{SectionOnLabelingSections}, which can be found on
page~\pageref{SectionOnLabelingSections}, covers section labeling.
```

Section 9.1, which can be found on page 47, covers section labeling.

Notice that when you use `\ref` you get the number, but when you use `\pageref` you get the page that it is on instead.

9.2 Referencing Equations

Labeling an equation is similar but you can only attach a label to an equation that is tagged with a number. So you need to be labeling equations that are inside of some numbered equation environment. Here is an example where I create an equation label and I reference it in three different ways (the result directly follows the code):

```
\begin{equation}
f(x)=\sum_{i=0}^n a_i x^i \label{poly}
\end{equation}
Equation~\ref{poly} is a polynomial function.  It can be found on
page~\pageref{poly} (obviously since it is on this page!). I think
equation~\eqref{poly} looks better because \verb|\eqref| adds
parenthesis around the number.
```

$$f(x) = \sum_{i=0}^{n} a_i x^i \tag{10}$$

Equation 10 is a polynomial function. It can be found on page 48 (obviously since it is on this page!). I think equation (10) looks better because `\eqref` adds parenthesis around the number.

So using `\eqref` creates parentheses which matches the automatic numbering format, but using `\ref` just has the number. Also note that we needed to use the `equation` environment here, and not simply `\[ \]`, because we need the equation to be numbered in order to label it.

You can also label multiple lines within the `align` environment. Here is an example where I label each line. Note that I put the label right before the newline command, except for the last line where you don't need the newline command:

```
\begin{align}
2x+3 &= 0\label{line}\\
x^2+y^2+x+y-1 &= 0\label{circ}\\
x^2-y^2-4x+2y+4 &= 0\label{hyperb}
\end{align}
Equation~\eqref{line} is a line, \eqref{circ} is a circle, and
\eqref{hyperb} is a hyperbola.
```

$$2x + 3 = 0 \tag{11}$$

$$x^2 + y^2 + x + y - 1 = 0 \tag{12}$$

$$x^2 - y^2 - 4x + 2y + 4 = 0 \tag{13}$$

Equation (11) is a line, (12) is a circle, and (13) is a hyperbola.

Remember that the labels do not create the numbers in parentheses on the right; the `align` environment creates the numbering on each line. The labels just allow you to reference those numbers.

Now, what if you have four lines of equations in the `align` environment but you only want to label one of the four lines? Well, you need to add the command `\notag` to any line you *do not* want to number, and you label the line you *do* want numbered. Like this:

```
\begin{align}
e^z &= -2 \notag \\
(y-3)^2+5 &= 0 \notag\\
x^3-14x^2+2x-3 &= 0 \label{cubic}\\
\cos(\theta) &= 2 \notag
\end{align}
Equation~\eqref{cubic} is the only of the four equations that has a solution
in $\mathbb{R}$.  The rest have no real solutions.
```

$$\begin{aligned} e^z &= -2 \\ (y-3)^2+5 &= 0 \\ x^3-14x^2+2x-3 &= 0 \\ \cos(\theta) &= 2 \end{aligned} \tag{14}$$

Equation (14) is the only of the four equations that has a solution in $\mathbb{R}$. The rest have no real solutions.

You can also create a custom tag for an equation if you don't want it numbered. To do this you use the `\tag{label}` or `\tag*{label}`. The starred version does not have parenthesis, the non-starred version has them. Not that "`label`" can be whatever you want but it is important to note that when you are inside the brackets for the `\tag` command, you are in text mode. So if you want your "label" to be a math symbol, then you have to use dollar signs inside the brackets. Here are some examples:

```
\[
f(x)=x^2\label{func1}\tag{A}
\]
\[
g(x)=x+3\label{func2}\tag*{B}
\]
\[
h(x)=(x+3)^2\label{func3}\tag{$\star$}
\]
\[
e^{i\pi}=-1\label{love}\tag*{$\heartsuit$}
\]
If you plug equation~\ref{func2} into equation~\eqref{func1}, then you get
 equation~\eqref{func3}. I love equation~\ref{love}.
```

$$f(x) = x^2 \tag{A}$$

$$g(x) = x + 3 \tag{B}$$

$$h(x) = (x+3)^2 \tag{$\star$}$$

$$e^{i\pi} = -1 \tag{$\heartsuit$}$$

If you compose function B with function (A), then you get equation ($\star$). I love equation $\heartsuit$.

Notice the difference between the starred version and the regular version of `\tag`. Also note that the first two examples had regular text as a tag (just the letters A and B) but the third and the fourth example used the math-mode symbols `\star` and `\heartsuit` as the tag. So I needed to enter math mode inside the tag with dollar signs. Even though the `\tag{}` is inside of math mode, when you are inside of the `{}` of `\tag`, you are in text mode.

9.3 Labels in large documents: the good and the bad

When you start to write longer documents, using labels and automatic numbering will be extremely helpful. It would be a chore to number everything manually because there would be a lot of numbers to keep track of! But the really awesome thing about using labels is that you can add new labels wherever you want at any time, and all of the numbers and labels will adjust automatically. Similarly, if you use a `\pageref` somewhere and then later you add new sections to your paper, every time you recompile the document, it will change the page number accordingly. So once you set the label and the reference, you never have to worry about it again. You can add, subtract, cut and paste all you want.

So labels are great, but there are a few potential issues to keep in mind. First, after you create your labels, you may need to compile your document *twice* in order for the labels and references to work correctly.[20] You will know when there is an error with a label. When you compile your document you might expect to see this in your document: "equation (5)" but instead, your final document has: "equation ??" Whenever you see a double question mark, that means LaTeX got confused and didn't know what to do with a label reference. You should always try compiling the document a second or third time and see if that fixes the problem But if it doesn't, it might be because you have a typo in your code like this:

```
\begin{equation}
f(x)=k \label{mistake}
\end{equation}
Equation \eqref{mitsake} is a constant function.
```

$$f(x) = k \tag{15}$$

Equation (**??**) is a constant function.

Do you see the error? You won't necessarily get an error when you compile the document if you have a typo in the label like this, but you will see ?? where there should be a number.

The other issue with labels that can come up is that it can be hard to keep track of them when you have dozens of labels. For instance, at the moment, as I am typing this line I currently have 35 labels in this document! The way that I keep track of these is by naming the labels with a special prefix that reminds me of what the label means. Now I have not been using these prefixes in the examples in this section, just so you know. But here are examples of how I would label sections, equations, figures, theorems, lemmas, and corollaries:

```
\label{sec:SectionName}
\label{eqn:EquationName}
\label{fig:FigureName}
\label{thm:TheoremName}
\label{lem:LemmaName}
\label{cor:CorollaryName}
```

You get the idea. The nice thing about this is that when you use a more sophisticated LaTeX editor like TeXstudio, it will keep track of your labels for you and you can select your labels from a pull down menu. So the naming scheme with the prefixes will keep the various types of labels bunched together in alphabetical order.[21]

[20] I use the LaTeX editor called TeXstudio and I don't have to compile two times for the labels to be correct. But if you use a more basic TeX editor you might have to.

[21] You can get TeXstudio for free here: http://texstudio.sourceforge.net/

10 Future additions to this document

- script sizes inside of math-mode I explained displaystyle but there is also textstyle, scriptstyle and scriptscriptstyle.

11 Stuff I will teach in separate documents outside of this one

- How to make theorems, definitions etc.
- How to adjust the theorem styles
- How to add graphics (pictures, diagrams what have you)
- Bibliography stuff with BibTeX
- How to use Beamer which is like PowerPoint but LaTeX style.
- How to make your own basic commands and macros.
- Any requests? Email me and let me know.