Beginning LaTeX

Dan Parker* and David Schwein†

Fall 2016

Welcome to the first of the Brown Science Center's LaTeX workshops! By the end of it, you'll be able to write a basic LaTeX document containing mathematics. These notes outline the material we'll cover.

We assume that you have a working installation of LaTeX and a LaTeX editor. If you do not, please consult the installation guide.

1 Introduction

1.1 Questions and Answers

What is LaTeX?

LaTeX is a document preparation system, the de facto standard for mathematics, physics, and computer science.

More technically, LaTeX is a set of macros – extra commands – for the programming language TeX. Donald Knuth created TeX in the seventies because he was unhappy with the typography in one of his *Art of Computer Programming* books. He designed the program specifically for typesetting mathematics, and it was widely adopted in the years following its release. Later, in the eighties, Leslie Lamport wrote LaTeX to extend TeX's mostly low-level functionality. Today LaTeX has largely superseded TeX.

Why should I use LaTeX?

1. LaTeX documents are beautiful. Their typography is outstanding, especially compared to something like Microsoft Word's.

2. LaTeX can make almost any document you can imagine and many others besides.

3. LaTeX is free, fast, and has virtually no bugs.[1]

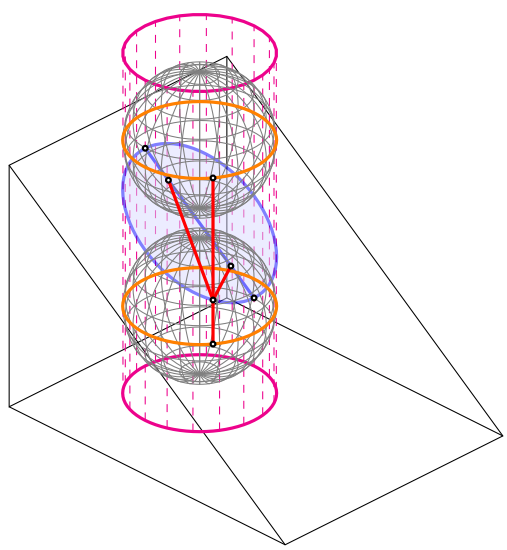
*danielericparker@gmail.com
†david_schwein@brown.edu

[1] The Bank of San Serriffe will pay $327.68 to anyone who can find a bug in TeX. See en.wikipedia.org/wiki/Knuth_reward_check.

LaTeX is not for everyone, even among academics. While designing these workshops we found, for instance, that the chemistry department at Brown does not care for LaTeX. But mathematicians should certainly use LaTeX, mainly because people will not take you seriously if you don't.[2] For the same reason, papers in other math-heavy scientific disciplines, such as physics and computer science, should also be prepared using LaTeX.

What can LaTeX do?

TeX is a Turing-complete programming language, which basically means that it can do anything a computer can do. Steve Hicks, for example, has written a program in TeX to control a Mars rover.[3] TeX supports equation typesetting for mathematics, physics, and chemistry and formats documents for publication. The Comprehensive TeX Archive Network (CTAN, at `ctan.org`) maintains a database of over 4,700 LaTeX packages, which add extra functionality to the program. For instance, the package `tikz`, can be used for creating technical diagrams. We'll discuss `tikz` in the final workshop.



Nevertheless, LaTeX is usually used only for typesetting documents.

I have no experience with computer programming. Can I learn LaTeX?

Yes! Do not be intimidated: LaTeX and TeX were designed for ease of use and most command have obvious names so that you can guess them. With that said, when you write a document in LaTeX you are programming a computer, and the process for making a document with LaTeX is slightly more complicated

[2] Scott Aaronson claims that you can catch 60% of wrong mathematical breakthroughs by checking if they were written in TeX. See `http://www.scottaaronson.com/blog/?p=304`

[3] See `http://sdh33b.blogspot.dk/2008/07/icfp-contest-2008.html`

than you are probably used to. There are more files for each document, and you can't immediately see the changes to your document.

How do you pronounce LaTeX?

There are two ways to pronounce LaTeX: LAY-tek or LAH-tek. We use the first pronunciation and Brown's computer science department tends to use the second. Everyone agrees, however, that you should not pronounce LaTeX like the word *latex*: this is not that kind of workshop. When you write the word *LaTeX*, remember to capitalize it as we have.

1.2 LaTeX's Philosophy

Document preparation systems come in two kinds: what-you-see-is-what-you-get (WYSIWYG) and what-you-see-is-what-you-mean (WYSIWYM). Microsoft word is a WYSIWYG editor: the file you edit looks the same as the document that comes off the printer. LaTeX is a WYSIWYM editor: the file you edit – a `.tex` file – stores information *about* the document, which is then typeset by the program to create the final document – by default, a `.pdf` file.

WYSIWYM editors nearly always produce superior documents because they reflect the natural demarcation in publication between composing a document and typesetting it, which was historically done by hand using movable type. WYSIWYG editors sacrifice typographic quality for speed: Microsoft Word can typeset your document as you type because it cuts typographic corners, On the other hand, TeX cannot synchronize changes in the `.tex` file and the `.pdf` file because the algorithm it uses to determine line justification and other typographic aspects of the document takes a little too long, around a second or less.[4]

The point is that using LaTeX forces you to focus on the content of your document instead of its form. You should not worry about choosing a font or sizing section headers, for instance: LaTeX makes all of those decisions for you, and it usually does an excellent job of it. Ceding stylistic control to LaTeX will be hard at first if you are accustomed to a WYSIWYG editor; do it anyway, and relearn old habits. With that said, it is possible, though not recommended, to customize any aspect of a LaTeX document. In this workshop we will spend little time discussing how to modify stylistic features of documents, such as typeface or section styles. Remember,

The number-one error LaTeX users make is focusing on form over content.

[4] I lied: there actually are WYSIWYG LaTeX editors. But why would you do that to yourself?

2 Starting Up the System

2.1 An Outline of How LaTeX Works

Before you make your first document in LaTeX, it will be helpful to have an idea of how the parts of the system fit together. To make a document with LaTeX, you edit a file ending in the `.tex` file extension. This file contains plain text – it is essentially a `.txt` file. When you tell LaTeX to typeset the `.tex` file it creates a `.pdf` file – your document – and a few auxiliary files we won't worry about in today's workshop. This means that whenever you modify the `.tex` file, you must typeset the document again to see the changes in the PDF file.

2.2 Your First LaTeX Document

For this beginner's workshop we will be using the online platform ShareLaTeX. There are a variety of other TeX editors that you can download on your computer. Once you've created an account and logged in, click on "New Project". Select the "Blank Project" option and enter a name for your document. The `.tex` file we be on the left side of the screen.

So that we can start from scratch, delete all the code in the `.tex` file. Copy and paste the following text into the file.

```
\documentclass{article}

\begin{document}
Hello world!
\end{document}
```

Press the button that says "Recompile" to compile your code into a `.pdf` file. (In other TeX editors it might be called "typeset") If you see a document appear, congratulations! You have made your first document.

If you have previous experience with LaTeX you may have expected the `.tex` file to contain more commands. Many new LaTeX users make documents using templates from the internet, which contain custom formatting commands and tend to be quite complicated. In the interest of simplicity we have tried to keep the documents we make in this workshop as simple as possible: you should understand every command.

In any case, every LaTeX document requires only two commands. The first of these is `\documentclass{article}`, which tells LaTeX that you will be creating an article. There are other document classes, such as `beamer` (presentations), `book`, but `article` is the default and works well for most documents. Every LaTeX command has the same syntax as `\documentclass{article}`: a command begins with a backslash and any arguments it takes are surrounded in curly braces. The `\documentclass` command takes a single argument, and other commands take no arguments or more than one.

The second requirement is the two lines containing `\begin{document}` and `\end{document}`. The first tells LaTeX to begin making the document; the second tells LaTeX that the document is finished. Consequently, any text appearing

between these two lines will be incorporated into the document and any text appearing after `\end{document}` will be ignored.

The space between `\documentclass{article}` and `\begin{document}` is called the *preamble* or the *topmatter*. It contains commands that modify global document parameters, such as the typeface or the appearance of headers and footers. If your document uses any external packages, you must include the name of the package in the preamble by typing`\usepackage{<package name>}`.

3 Document Structure

3.1 Sections

It's usually a good idea to divide a long document into sections, subsections, and even subsubsections, and LaTeX has nice built-in commands to do so. They take a single argument, the name of the section.

```
\section{...}
\subsection{...}
\subsubsection{...}
```

The commands above automatically number the parts of the document they create. Automatic numbering is one advantage of using LaTeX over other document preparation systems. Imagine typing a 100-page-long thesis in Microsoft Word and adding one section at the beginning: you would have to renumber all of the sections to compensate. If you had used LaTeX, the sections would have been renumbered automatically. To remove the numbers, use the starred versions of the commands. (An asterisk often denotes an unnumbered field, such as a section, equation, or quote.)

```
\section*{...}
\subsection*{...}
\subsubsection*{...}
```

If you type the command `\tableofcontents`, LaTeXwill automatically generate a table of contents based on the sections, subsections, etc. in your document. You may have to compile the file twice for this to work.

3.2 Titles

There are two steps to give your document a title: tell LaTeX what to put in the title, and tell LaTeX to typeset the title. In the preamble, use the commands `\title{...}`, `\author{...}`, and `\date{...}`. For example, the preamble to the `.tex` file for this document contains the lines

```
\title{Beginning \LaTeX}
\author{Dan Parker and David Schwein}
\date{\today}
```

To display the title, write `\maketitle` immediately below `\begin{document}`. The command `\today` prints today's date; the command `\LaTeX` prints LaTeX.

3.3 Headers

Professors often ask students to write their name on each sheet of a problem set. To do this, and to add other text to the header and footer of our LaTeXdocuments, we'll use the `fancyhdr` package. Add the following two lines to your preamble, before `\begin{document}`.

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

The second line tells LaTeX to use the new header, the default loaded with the `fancyhdr` package.

There are six commands that modify the header.

```
\lhead{Josiah Carberry}
\chead{}
\rhead{Psychoceramics}
\lfoot{}
\cfoot{\thepage}
\rfoot{}
```

The names of the commands are self-explanatory: `\cfoot`, for instance, indicates what goes in the center position of the footer. The command `\thepage` prints the current page number.

4 Formatting

4.1 Spaces and Line Breaks

The best way to understand how LaTeX deals with space is to experiment. With that said, here are a few guidelines. Multiple spaces in a row are treated as one space by LaTeX. A single line break is also interpreted as an interword space.

To make a block of text into a new paragraph, place an empty line before it. For example:

Input

```
I love
LaTeX

But not Microsoft Word
```

Output

I love LaTeX
But not Microsoft Word

4.2 Font Styles

Most typefaces – including Computer Modern, the default for LaTeX – comprise a variety of weights, italicizations, and other styles. You can access them with the following commands.

Input

```
\textit{italic}
\textbf{bold}
\textsc{small caps}
\texttt{typewriter}
```

Output

italic
bold
SMALL CAPS
`typewriter`

4.3 Special Characters

4.3.1 Hyphens and Dashes

Hyphens connect the parts of a compound word and are printed with a `-` character.

Input

```
a non-negotiable plan
```

Output

a non-negotiable plan

There are two types of dashes: en dashes – which are about as wide as the letter "N" – and em dashes—which are about as wide as the letter "M", or twice as wide as an en dash. An en dash is printed with `--`; an em dash is printed with `---`.

Input

```
pp. 126--128
Wait---come back!
```

Output

pp. 126–128
Wait—come back!

4.3.2 Quotation Marks

The quotation characters on your keyboard will not typeset the way you expect them to.

Input

```
"Smart quotes" look bad
```

Output

"Smart quotes" look bad

The proper way to quote is to use `` ` `` for the left marks and `'` for the right marks.

Input

```
``A `feisty' dog''
```

Output

“A ‘feisty’ dog”

On a QWERTY keyboard, the key for ` is left of 1 and the key for ' is right of ;.

4.3.3 Comments

To prevent LaTeX from typesetting text, place a % before it. The entirety of the line following % will be ignored.

Input

```
Where? % Here
```

Output

Where?

4.3.4 Ellipses

The technical term for three periods in succession is *ellipsis*. To print an ellipsis use the command `\dots`.

Input

```
This is ugly...
This is better\dots
```

Output

This is ugly...
This is better…

4.3.5 Miscellaneous Characters

Finally, LaTeX reserves a few special characters for its own purposes. You can access most of these characters by placing a \ before them.

Input

```
\{ \} \% \$ \& \_ \#
```

Output

{ } % $ & _ #

There are, of course, exceptions. To print \ use the command `\textbackslash`.

4.4 Lists

LaTeX has commands for making two types of lists: bulleted lists and numbered lists. To make either type of list, you must first tell LaTeX to enter the appropriate *environment*. Roughly speaking, an environment is a portion of your document that LaTeX treats differently from the text of the document. For instance, LaTeX includes environments for centering text, including figures, or making tables.

To begin an environment, use the command `\begin{<environment>}`; to end an environment, use the command `\end{<environment>}`. Does this terminology sound familiar? It should, because we use `\begin{document}` to begin every LaTeX document and `\end{document}` to end every document.

To make a bulleted list, use the `itemize` environment. To make a new bullet, use the command `\item`.

Input

```
\begin{itemize}
 \item We
 \item Ourselves
 \item Us
\end{itemize}
```

Output

- We
- Ourselves
- Us

To make a numbered list, use the `enumerate` environment. Again, the command `\item` adds a new item to the list.

Input

```
\begin{enumerate}
 \item Me
 \item Myself
 \item I
\end{enumerate}
```

Output

1. Me
2. Myself
3. I

To make a list with more than one level, nest list environments inside of each other.

Input

```
\begin{enumerate}
 \item Ends in \emph{gry}
  \begin{enumerate}
   \item \emph{angry}
   \item \emph{hungry}
   \item Anything else?
  \end{enumerate}
\end{enumerate}
```

Output

1. Ends in *gry*
   (a) *angry*
   (b) *hungry*
   (c) Anything else?

Itemize and enumerate also allow optional arguments that change the type of bullet or numbering.

5 Mathematics

LaTeX was designed specifically for writing math: there is a command to print virtually anything you can write. And because there are many mathematical symbols you can write, there are quite a few commands for printing mathematics. We don't want to bombard you with long lists of commands, so instead we're going to explain how to write mathematics in LaTeX by explaining the three most common mathematics environments. During next week's workshop we'll cover mathematics in more detail. Today is a gentle introduction.

The three types of mathematics environments differ only in the way they are typeset. Any command available in one math environment is available in any other.

Since Δ interchanges the two groups inside the parentheses it is enough to compute the first of them, i.e.,

$$H^1_{\text{unr in } \Sigma-\mathfrak{p}}(\mathbf{Q}_\Sigma/L,\ K/\mathcal{O}(\nu)). \tag{4.2}$$

The inflation-restriction sequence applied to this gives an exact sequence

$$\begin{aligned} 0 \quad &\to \quad H^1_{\text{unr in } \Sigma-\mathfrak{p}}(L(\nu)/L,\ (K/\mathcal{O})(\nu)) \\ &\to \quad H^1_{\text{unr in } \Sigma-\mathfrak{p}}(\mathbf{Q}_\Sigma/L,\ (K/\mathcal{O})(\nu)) \\ &\to \quad \mathrm{Hom}\left(\mathrm{Gal}(M_\infty/L(\nu)),\ (K/\mathcal{O})(\nu)\right)^{\mathrm{Gal}(L(\nu)/L)}. \end{aligned} \tag{4.3}$$

The first term is zero as one easily checks using the divisibility of $(K/\mathcal{O})(\nu)$. Next note that $H^2(L(\nu)/L,\ (K/\mathcal{O})(\nu))$ is trivial. If $\nu \not\equiv 1(\lambda)$ this is straight-

Figure 1: The three types of equations in their natural habitat.[5]

1. Inline formulas: formulas inside a sentence. Usually short and not containing any tall symbols, like sums or integrals.
2. Displayed formulas: formulas on their own line with spacing before and after. Used for long, tall, or important equations.
3. Aligned formulas: several formulas, each on its own line, with a certain symbol (often =) in the same place in each line. Used for expressions that are too long for a single line, or computations with multiple steps.

[5] This is an excerpt from Wiles' proof of Fermat's Theorem. Don't worry if it's completely unintelligible to you – we can't make sense of it either. The point is to see the three types of formulas in action.

5.1 Inline Math

Let's begin with inline formulas, the most commonly used of the three types.

Input

```
Let $x = r\cos\theta$
and $y = r\sin\theta$.
Then $r = \sqrt{x^2 + y^2}$
and $\theta = \arctan(y/x)$.
```

Output

Let $x = r\cos\theta$ and $y = r\sin\theta$. Then $r = \sqrt{x^2 + y^2}$ and $\theta = \arctan(y/x)$.

To make an inline formula, enclose the expression in `$`'s. As we promised, command names make sense: for instance, the command for the greek letter θ is `\theta`. Anything between the dollar signs is interpreted as math, not text, and TEX treats math differently.

- All letters are automatically italicized. This gives readers a visual cue that they're looking at a variable instead of a normal letter.
- Commands for mathematical symbols, such at fractions and greek letters, can be used. Most of these commands will not work outside of math mode.
- The spacing is tighter. For instance, spaces between letters are removed.
- Blank lines are not allowed.

There are thousands of math commands in LATEX, enough to make any symbol you could want and many more you probably don't. If you know what a symbol is called but don't know the command for it, you can find it in one of the many lists of LATEX math symbols available on the internet, including

- a short list of basic symbols, curated by Brown's CS department,[6] and
- the Comprehensive LATEX Symbol List (164 pages!).[7]

If you don't know what a symbol is called, try using the website Detexify, which has you draw the symbol and guesses the corresponding LATEX command.

Here are examples of some of the most common math commands.

`$a^b$`	`$a_b$`	`$\log(x)$`	`$\alpha$`	`$\sqrt{x}$`	`$\frac{a}{b}$`
a^b	a_b	$\log(x)$	α	$\sqrt{x}$	$\frac{a}{b}$

[6] cs.brown.edu/about/system/software/latex/doc/symbols.pdf
[7] mirror.hmc.edu/ctan/info/symbols/comprehensive/symbols-a4.pdf

It is sometimes necessary to nest expressions in {brackets}:

Input

```
a^bc
a^{bc}
```

Output

$a^b c$
a^{bc}

5.2 Displayed Math

Large symbols, such as sums or integrals, are usually too large to fit into a single line – $\sum_{n=0}^{\infty} \frac{1}{n^2}$ is so large that LaTeX adds space to compensate. Moreover, some equations are so important that they should be separated from the body of the text and placed on their own line. Displayed math mode is the mode of choice for expressions like these. Use `$$` to delimit displayed math code to leave it.

Input

```
$$
\int_0^1 f(x)\, dx =
\lim_{n\to\infty}
\sum_{i=0}^{n-1}
\frac{1}{n}
f\left(\frac{i}{n}\right)
$$
```

Output

$$\int_0^1 f(x)\, dx = \lim_{n\to\infty} \sum_{i=0}^{n-1} \frac{1}{n} f\left(\frac{i}{n}\right)$$

The `\,` command adds a little space.

5.3 Aligned Math

To use *aligned mode*, we need to make a modification to our document's preamble: add the line `\usepackage{amsmath}` anywhere before `\begin{document}`. This will load the `amsmath` package, which gives us access to more commands. Having done so, use the `align*` environment to enter aligned mode. In aligned mode, type equations as you would for inline or display mode, but with the following modifications:

- Place `&` before the symbols you want to have vertically aligned.
- Place `\\` to make a new line.

Input

```
\begin{align*}
(a+b)^2
&= (a+b)(a+b) \\
&= a(a+b) + b(a+b) \\
&= a^2 + ab + ba + b^2 \\
&= a^2 + 2ab + b^2.
\end{align*}
```

Output

$$\begin{aligned}
(a+b)^2 &= (a+b)(a+b) \\
&= a(a+b) + b(a+b) \\
&= a^2 + ab + ba + b^2 \\
&= a^2 + 2ab + b^2.
\end{aligned}$$

6 Cross References

The reason that we number pages, figures, and other parts of a document is so that we can refer to them later. LaTeX has a simple mechanism for making cross references.

Input

```
\section{Introduction}
\label{sec:introduction}
Welcome to my lab report.

\section{Results}
See Section
\ref{sec:introduction}
for a friendly welcome.
```

Output

1 Introduction

Welcome to my lab report.

2 Results

See Section 1 for a friendly welcome.

To refer to an object, you have to label it. The command for labeling is `\label{...}`, which takes the label as its argument and goes after the object you want to label. To refer back to a labeled object, use the command `\ref{...}`, which takes the label as its argument and prints the labeled object's number.

Here are a few of the objects that LaTeX can label:

1. Sections and subsections
2. Equations
3. Figures and tables

Use the command `\url{...}` from the package `hyperref` to create a hyperlink. To create a footnote, type `\footnote{...}` inline with your text.

7 Graphics

Including graphics in LaTeX is easy. Positioning them correctly is tricky because they take up lots of space. But before we talk about how to do either of these things, let's take a step back and talk about good graphics and bad graphics.

7.1 File Formats

Broadly speaking, computer graphics fall into two categories: raster and vector. *Raster graphics* tell the computer how to draw an image by listing the color of each pixel; *vector graphics* tell the computer how to draw an image by describing the image at a high level. A raster graphic would draw a red circle by giving it the positions of all the red pixels in the circle, while a vector graphic would do the same by making "a red disk with radius 1 cm in the center of the image".

The difference between the two file formats is that vector graphics are resolution-independent: you can magnify a vector graphic without any loss of resolution. Raster images do not scale well and look pixelated after enough magnification. For this reason, you should use vector graphics whenever possible.[8]

Raster File Formats	Vector File Formats
`.jpg`	`.eps`
`.gif`	`.svg`
`.png`	`.ps`
`.tiff`	`.pdf`

7.2 Figures

Including an image in a LaTeX document requires the `graphicx` package. After loading `graphicx`, move the image into the same directory as your `.tex` file.

The `includegraphics` command takes one argument, the path to the image file relative to your `.tex` file. This can be used to place images into a TeX file from other directories. For instance, if you had a directory called `images` in the same directory as your `.tex` file, then you can include an image inside the `images` folder with the command `\includegraphics{images/vector.eps}`.

To make a captioned figure, use the `figure` environment.

[8] Unfortunately, LaTeX cannot load `.svg` files. To include an `.svg` graphic, you must convert it into one of the other vector graphics formats.

Input

```
\usepackage{graphicx}
. . .
\begin{figure}[h]
\center
\includegraphics{logo.eps}
\caption{An example image}
\end{figure}
```

Output

Figure 2: An example image

The `figure` environment takes an optional command which controls where the figure is placed on the page. There are three possibilities: `[h]` for *here* on the page, `[b]` for the *bottom* of the page, or `[t]` for the *top* of the page.

Positioning images can be quite tricky because LaTeX automatically positions them for you, and it can be hard to predict or control where they will appear. If the `h`, `t`, or `b` option doesn't put your image where you want it to be, then try changing the size of the image or the position of the `figure` environment in your code. Using references, you can always direct the reader to a figure on another page. Finally, because changing the text of your document can change the positions of figures, remember to

Position figures only after the text of your document is complete.

You may want to try the `wrapfig` package, which places figures so that the text wraps around them. This functionality is often useful for small images that don't need a blank horizontal band, or figures that don't need much emphasis.

8 Additional Resources

This set of notes is deliberately brief, which is good for getting across the important ideas but not good for communicating subtleties. If you would like to know more, the internet has extensive resources for learning and mastering LaTeX; the following are a few of our favorites.

- The Wikibooks LaTeX page.[9] A compendium of basic and advanced information about LaTeX. If you want to know how to do something with LaTeX, this website can probably tell you how.

[9] `en.wikibooks.org/wiki/LaTeX`

- Detexify.[10] The LaTeX names for mathematical symbols can be hard to remember or look up because there are so many of them. Detexify lets you draw a symbol on the computer screen, then guesses which symbol you drew and tells you the LaTeX command for it.

- TeX Stack Exchange.[11] A question-and-answer site about TeX, LaTeX, and friends. The users of the site – among them people who designed LaTeX – are pros, and can answer about any question you have. Save it for the stumpers, though: Stack Exchange sites discourage users from asking a question whose answer can be easily looked up.

- The Comprehensive TeX Archive Network (CTAN).[12] Members of the LaTeX community have written over 4500 packages that add extra functionality to LaTeX. CTAN houses these packages and their documentation. When you have a question about a package, the documentation is often the best place to start.

[10] `detexify.kirelabs.org`
[11] `tex.stackexchange.com`
[12] `www.ctan.org`