A Short Introduction to LaTeX

Matthew Rossetter

mbr-phys@protonmail.com

Foreword: This document is intended as a short introduction to compiling documents in the powerful and versatile typesetting program, LaTeX. This won't cover everything that can be done with LaTeX, but it will try to touch on most common areas used. The ease of LaTeX is that there are numerous online resources to help you if there are any issues you come across. Some that I find particularly useful are the Wikibooks series on LaTeX, and the Overleaf/ShareLaTeX guides. The Not So Short Introduction to LaTeX2ϵ is another useful resource to keep in mind.

For examples of LaTeX documents, both as source code and compiled pdfs, Overleaf is most useful. You can also find some specific examples, such as lecture notes and lab reports, on my personal github page, mbr-phys.

1 How do I run LaTeX?

To use LaTeX, you need to have a valid LaTeX distribution installed on your computer. For more details on installing LaTeX for your Operating System, see https://www.latex-project.org/get/. All LaTeX distributions will come with a text editor from which you write and compile your documents, although most flexible text editors used for multiple programming languages will have packages you can install to run LaTeX from them if you prefer, e.g. I find Atom to be a nice text editor with many packages for running LaTeX.

Alternatively, you may find it simpler at first to sign up to Overleaf, an online LaTeX compiler which contains everything you need to start typesetting, including extensive tutorials and guides as mentioned above. Using Overleaf has some obvious benefits, such as auto-saving your documents, collaborations of joint documents, and ease of use. There are also templates for all sorts of documents uploaded to Overleaf, from scientific journal entries to posters and presentations. However, it is not always as versatile as having a TeX distribution installed on your computer, as well of course as always needing an internet connection and some features being locked behind a pay wall.

In general, I find it best to have a working distribution installed as well as an Overleaf account (the free one, of course!) This allows you to access all the convenience found in Overleaf, but you can then download what you need and run on your computer to allow for more flexibility in how you compile. I would recommend working through an installed distribution foremost and then looking to Overleaf when you need its extra convenience, but there is no right way to use LaTeX- use it whichever way you most prefer.

2 Building your first document

Now that you have installed a TeX distribution on your computer or set up Overleaf, it is time to build your first document. To create a new document for LaTeX, you want to open a new file in whichever text editor you have chosen. The new file should have the extension .tex to identify it as for LaTeX. Some editors will do this for you automatically; others will require you to define it yourself.

There are three commands which set out the structure of your .tex file:

```
\documentclass[options]{class}

\begin{document}

\end{document}
```

Let's take a look at what all this means:

- \documentclass[options]{class} is always the very first command entered into a .tex document. This command defines what sort of *class* you want your document to be; report, article, and book are examples of common classes used. The class will change some properties of the documents, such as if you want to include chapters and parts as larger divisions than the sections you see separating this document. The [options] section allows you to add specifying options for each class you would use, e.g. paper/font size and one/two columns. For example, for this document, my first line reads \documentclass[a4paper]{article}.

  The *article* class tends to be the most commonly-used and versatile class to start from, but it will depend on what you aim to do with your document. The links mentioned in the Foreword all contain more detailed descriptions of classes and options you can choose.

- The part between \documentclass[options]{class} and \begin{document} is known as the *preamble*. This section is used for commands which will affect the entire document; here you will write any customisations you want for your document, e.g. margin size. You may need to call in packages not built in to the basic LaTeX environment in order to do certain things such as include graphics or customise colouring. My preamble is shown below:

  ```
  \documentclass[a4paper]{article}

  \usepackage[left=2.5cm,right=2.5cm,top=3cm,bottom=3cm]{geometry}
  %define margin sizes
  \usepackage{graphics,graphicx,float} %import figures etc
  \usepackage[table,dvipsnames]{xcolor} %extra color options
  \usepackage{multicol} %extra column customisation
  \usepackage{enumitem,pifont} %for list customisation
  \usepackage{mathtools,esint} %good maths environments for equations
  \usepackage{hyperref} %create hyperlinks inside the document
  \usepackage{listings} %type out source code

  \title{A Short Introduction to \LaTeX}
  \author{Matthew Rossetter \\ \\ mbr-phys@protonmail.com}
  \date{}

  \begin{document}
  \maketitle
  ```

  Here, you can see packages I have called in order to customise the document, with descriptioons of each one commented beside it. Notice how the comment symbol in LaTeX is %. Most packages you will need are installed with most TeX distributions; see Section 7 for more information on packages and how to install additional ones. I have also defined how the title of the document will look, which is then put into the document by the \maketitle command.

- \begin{document} defines the beginning of the document itself. It is after this command that you enter the text (and equations and figures, etc) that will become the compiled document. The final command of the document is then **end**{document}. This marks the end of your text for the document and signals the compiling program to stop.

So now you should be able to type out a basic .tex file with the above ingredients, but how do you compile this .tex file into a pdf document?

Compiling a document will usually be specific to the text editor you are using, but most follow the same pattern and make it obvious enough. If you are using Overleaf, this will automatically compile your document as you save the file. Many text editors will have clear buttons saying "Run" or "Compile" that will build your document for you, although some may work on a keyboard shortcut such as Ctrl−Shift−b. If it isn't clear how to compile from the text editor you are using, there should be clear guides on the editor's website and elsewhere on the internet.

If you are comfortable using Command Line/Terminal, then you can also compile your document by navigating to the folder containing your .tex file and entering the command 'pdflatex document.tex', where document.tex is the name your file.

3 Document Structure and General Typesetting

Once you are able to create and compile simple documents in LaTeX, you will want to begin customising your document to your image. The first point in customisation is setting your document class for your need. Different classes will provide different functions, with different document divisions as well, as mentioned previously. For class options, see for example Wikibooks.

Document Classes		**Document Divisions**	
Class	Description	Division	Comment
article	for short reports, general notes, etc	\part{}	not in letter
report	for longer reports, theses, lectures	\chapter{}	only books and reports
book	for real books	\section{}	not in letter
memoir	based off book, but with more flexibility	\subsection{}	not in letters
letter	writing letters	\subsection{}	not in letters
beamer	writing presentations, see online resources	\subsubsection{}	not in letters
slides	for slides, using big letters	\paragraph{}	not in letters
proc	similar to article, designed for *proceedings*	\subparagraph{}	not in letters

Table 1: A summary of popular classes and the document divisions ranked from largest to smallest.

If you want a table of contents for your document, this can be called by using \tableofcontents. Further customisation of the table of contents, e.g. how far down the rank of divisions it lists, is possible through packages such as titletoc, which can also customise how divisions look within the document as well. (In addition, see titlesec.)

A common customisation is changing the margin size. The default margins in LaTeX are very large, so it is quite common to loosen these when drafting certain documents. An example of how this is done using the geometry package is in Section 2.

The default font of LaTeX is Computer Modern, a font designed specifically for use in TeX packages. However if you want to use another font, this can be done through several methods. See for example the LaTeX Font Catalogue or this example for font options and instructions on setting them.

There are plenty other miscellaneous options for general formatting. Most likely, if there is something you wish to change about the structure/style of your document, there is a way to do it. The online resources mentioned in the Foreword will likely have many of them, and if not, searching your problem on the internet almost always brings a solution. Usually TeX StackExchange is the best place to find the modifications you're looking for (but not the droids).

4 Maths

Creating equations in Microsoft Word can be pretty tedious, so LaTeX comes to the rescue yet again with what is frequently a intuitive and simple way of typesetting equations as part of natural documents. In fact, Microsoft Word now supports writing equations using LaTeX syntax due to how convenient it can be.

It is possible to make equations in the plain LaTeX document, although only really if you're making the simplest equations. For most mathematical needs, it is much easier to load a package which introduces the full utility of typesetting maths in LaTeX. The traditional package to load is amsmath, but one can alternatively use mathtools which itself loads amsmath and then fixes some of the slight issues with the former package. In Section 2, you will have seen how I loaded in the mathtools package just as any other package (I also loaded esint alongside it, which is just for extra integral options I use below).

So let's look at an equation and how we would write this out. For a good example, I write out the time-independent Schrödinger equation, which reads

$$-\frac{\hbar^2}{2m}\nabla^2\Psi + V(\Psi)\Psi = E\Psi. \tag{1}$$

In my .tex file, this is written as

```
\begin{equation}\label{eq:schroeq}
    -\frac{\hbar^2}{2m}\nabla^2\Psi + V(\Psi)\Psi = E\Psi.
\end{equation}
```

Hopefully not too complicated. Every mathematical symbol you could think of has a relatively intuitive form in LaTeX. You can see that some are simply keyboard symbols, such as ˆ for raising to the power, and some are written, such as \Psi and similarly for all greek letters. The label command is for proper referencing through a document, which will be explained in Section 9.

The equation environment used automatically numbers the equations, as seen to the right. The numbering can be removed by using \begin{equation∗}...\end{equation∗} instead. If we want to group several equations together, there are different environments than equation to do this, such as align - which I show for the definition of a Laurent series:

$$f(z) = \sum_{n=-\infty}^{\infty} a_n(z-z_0)^n, \tag{2}$$

$$a_n = \frac{1}{2\pi i}\oint_C \frac{f(z)}{(z-z_0)^{n+1}}dz. \tag{3}$$

This was written as

```
\begin{align}
    \label{eq:laurent} f(z) &= \sum_{n=-\infty}^{\infty} a_n(z-z_0)^n, \\
    \label{eq:an}a_n &=\frac{1}{2\pi i}\oint_C\frac{f(z)}{(z-z_0)^{n+1}}dz.
\end{align}
```

Again, we can see simple ways of writing out complicated expressions and symbols, where the '&' symbol next to '=' on both lines works as an 'alignment tab', i.e. the two equations will be arranged such that their alignment is centered on these tabs.

The above examples of maths in LaTeX are called 'display' environments - they remove themselves from the surrouding block text to display their contents. We can also use 'inline' maths if we want to write small equations inline with the block text. For example, we could write Gauss' Law, $\Phi_E = \frac{Q}{\epsilon_0} = \oiint_S \mathbf{E}\cdot d\mathbf{A}$, using \$\Phi_E = \frac{Q}{\epsilon_0} = \oiint_S \mathbf{E}\cdot d\mathbf{A}\$. So to write equations inline with the text, we use \$...\$ around our equation.

For a full summary of maths environments, such as equation above, and symbols, I find the Wikibooks pages Mathematics and Advanced Mathematics particularly useful.

5 Lists

Lists can sometimes seem to have a mind of their own in Microsoft Word. The power of LaTeX, where you have seen already that we can put everything in clear environments with a beginning and an end, allows you to fully control your lists, and customise them to your liking. We add the package enumitem in the preamble for extra customisation, and pifont for new symbols below. Let's look at a few lists and their differences, based on my reasons not to like sand:

- It's coarse
- It's rough
- It's irritating
- It gets everywhere

```
\begin{itemize}
    \item It's coarse
    \item It's rough
    \item It's irritating
    \item It gets everywhere
\end{itemize}
```

This is just a straight-forward list with no modifications. For certain purposes, it can be fine, although you will notice that the separation between points is a bit large. Let's add some:

- It's coarse
- It's rough
- It's irritating
- It gets everywhere

```
\begin{itemize}[noitemsep]
    \item It's coarse
    \item It's rough
    \item It's irritating
    \item It gets everywhere
\end{itemize}
```

Now we have the same list, with its items much tighter together. In my opinion, that looks nicer, although it is quite close. If you want something inbetween the two above, you could instead use the option [itemsep=5mm] for example. We can also change from a standard bullet point as it suits as:

➤ It's coarse

➥ It's rough

➤ It's irritating

- It gets everywhere

```
\begin{itemize}[label=\ding{228}]
    \item It's coarse
    \item[\ding{229}] It's rough
    \item It's irritating
    \item[-] It gets everywhere
\end{itemize}
```

We have defined a common label for the whole environment in the square brackets, but we have also overwritten that by using square brackets on the \item itself. The term \ding{228} is provided by the pifont package. You can check yourself what other symbols you can load in from this. There are still many more options you can use to customise lists, but these are usually the most common. We can also include lists within lists:

- Sand:
  - It's coarse
  - It's rough
  - It's irritating
  - It gets everywhere
- Here:
  - Everything is soft
  - Everything is smooth

```
\begin{itemize}[nosep]
    \item Sand:
    \begin{itemize}[nosep]
        \item It's coarse
        \item It's rough
        \item It's irritating
        \item It gets everywhere
    \end{itemize}
    \item Here:
    \begin{itemize}[nosep]
        \item Everything is soft
        \item Everything is smooth
    \end{itemize}
\end{itemize}
```

We can keep putting lists inside other lists as far down as needed. In this example as well, we can see the more extreme version of noitemsep in nosep, which removes all separations to make the list very compressed.

What if we want a different type of list? There are other environments we can use instead of itemize to express our hatred for sand:

1. It's coarse
2. It's rough
3. It's irritating
4. It gets everywhere

```
\begin{enumerate}
    \item It's coarse
    \item It's rough
    \item It's irritating
    \item It gets everywhere
\end{enumerate}
```

So we now have the enumerate environment which will give us numbered lists. There are other list environments to suit your needs which can be found through the package documentations and resources listed throughout this paper.

All of the customisation options for lists can also be defined for the entire document through commands in the preamble; the same options can be used for enumerate and any others as for itemize. This website provides an overview of all this.

6 Tables and Figures

It is important when typesetting to be able to include tables and figures in a document, and also have these behave in the way you want. LaTeX is here to help yet again. Using its commands for figure management, you can fully specify where in a document an image goes, how big it is, how much it disrupts the flow of the text, and anything else you can think of. You will have seen again in my preamble in Section 2 that I called in packages for use with figures: graphics, graphicx, and float. There are many packages which handle different things for figures, but at minimum I always recommend importing these three to make sure pretty much everything handles exactly as you want it. Let's put an image into the document and take a look at how it was done:

LATEX
YOUR PAPER MAKES NO GODDAMN SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.

Figure 1: This can become a recurring theme in your life when part of the Church of LaTeX.

This figure was called in through:

```
\begin{figure}[H]
    \centering
    \includegraphics[scale=0.5]{latexmeme.png}
    \caption{\label{fig:meme} This can become a recurring theme in your
    life when part of the Church of \LaTeX.}
\end{figure}
```

Ok, looks nice and simple. We have created the environment figure in which we have imported an image using includegraphics. Here, I have only put the name of the image as it is in the same folder as my .tex file, but in general you need to put the full path to the image for the program to find it. I have passed the option scale=0.5 to scale down the size of the image such that it fits nicely into the document; there are many other options you can use here to customise figures you import. \caption{} should be quite clear as the command to pass a caption onto the image. It isn't needed if you don't want a caption, but for professional documents, it will always be advised. \centering moves everything inside the figure environment to wrap to the center of the text, otherwise it would naturally flush to the left of the page.

We could have used includegraphics to call an image without the use of the figure environment, but it would have treated the image then like a word (just a really big one) and can result in strange things in your document. It is always good practice to use the figure environment to make sure your images are nice and tidy.

However, figure won't always put an image where you want it. By itself, it will try to find the point in the document where it displaces the least amount of text: this is because images can't be broken up over a new page, but text can. To fix this, we can pass the *placement specifiers* options to figure to change how it places your images. There are several in-built specifiers which tell LaTeX roughly where to put the figure, but it can still override this if it is creating too much white space around text. I have used the option H, which is not built-in, but an additional option given the float package. This overrides LaTeX's desire to move your figures about and tells it to put it exactly where you have it in the text. The use of H over the natural placement specifiers is a matter of taste: I like to make sure figures are exactly where I want them to be, but it can sometimes be preferable to allow it to shuffle things about. A full description of placement specifers can be found for example at Wikibooks.

Tables

7 Useful packages

As you get more comfortable with the program, package documentation files become very useful for explaining how to better use each package. All these files are stored on CTAN, where you can also download and install new packages when needed.

8 .sty files

9 Labels and References

10 Drawing in Tikz and PGF

(Only small mention of this, let's not confuse them)