

A Short Introduction to L^AT_EX

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, L^AT_EX. This won't cover everything that can be done with L^AT_EX, but it will try to touch on most common areas used. The ease of L^AT_EX 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 L^AT_EX series on L^AT_EX, and the Overleaf/ShareLaTeX guides. The Not So Short Introduction to L^AT_EX2 ϵ is another useful resource to keep in mind.

For examples of L^AT_EX 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 L^AT_EX?

To use L^AT_EX, you need to have a valid L^AT_EX distribution installed on your computer. For more details on installing L^AT_EX for your Operating System, see <https://www.latex-project.org/get/>. All L^AT_EX 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 L^AT_EX from them if you prefer, e.g. I find Atom to be a nice text editor with many packages for running L^AT_EX.

Alternatively, you may find it simpler at first to sign up to Overleaf, an online L^AT_EX 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 T_EX 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 L^AT_EX- use it whichever way you most prefer.

2 Building your first document

Now that you have installed a T_EX distribution on your computer or set up Overleaf, it is time to build your first document. To create a new document for L^AT_EX, 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 L^AT_EX. 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  
\usepackage { tkz-diag } %custom sty for diagrams
```

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

```
\begin { document }  
\maketitle
```

Here, you can see packages I have called in order to customise the document, with descriptions 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 L^AT_EX, 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 Structure.

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 <i>proceedings</i>	\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 L^AT_EX 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 L^AT_EX is Computer Modern, a font designed specifically for use in T_EX packages. However if you want to use another font, this can be done through several methods. See for example the L^AT_EX 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 T_EX 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 L^AT_EX 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 L^AT_EX syntax due to how convenient it can be.

It is possible to make equations in the plain L^AT_EX 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 L^AT_EX. 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

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

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 L^AT_EX. You can see that some are simply keyboard symbols, such as `^` 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:

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

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

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 L^AT_EX are called 'display' environments - they remove themselves from the surrounding 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 on 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 L^AT_EX, 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 | <code>\begin{enumerate}</code> |
| 2. It's rough | <code>\item It's coarse</code> |
| 3. It's irritating | <code>\item It's rough</code> |
| | <code>\item It's irritating</code> |
| 4. It gets everywhere | <code>\item It gets everywhere</code> |
| | <code>\end{enumerate}</code> |

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:

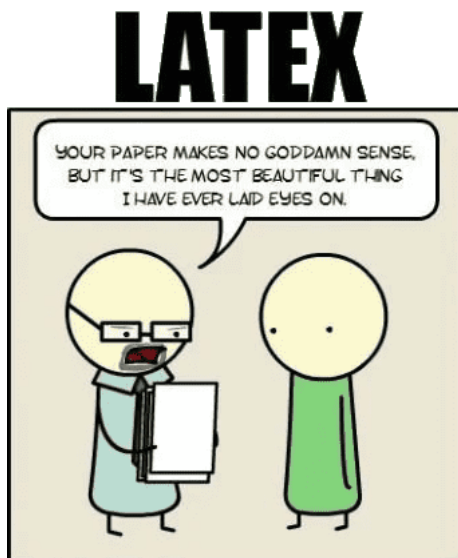


Figure 1: This can become a recurring theme in your life when part of the Church of \LaTeX .

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 specifiers can be found for example at Wikibooks/Floats. It is worth noting that there are other commands that you can put inside figure environments, but these are used much more sparingly.

Tables work similarly to Figures in terms of their placement and alignment within documents, so we will not dwell on this really, just refer to the above paragraphs for figures. The commands for making a table, however, are slightly different to those for introducing figures. Let's look at a simple table and see what we've got:

Name	Catalogue Number	Price	Avg. Customer Rating
Kragsta	904.525.87	£75	4.5★ (6)
Vittsjö	802.153.32	£45	4.9★ (21)
Arkelstorp	302.608.07	£110	4.8★ (11)
Tofteryd	401.974.86	£170	4.8★ (6)
Malmsta	602.611.84	£125	3.0★ (2)
Havsta	604.041.97	£85	4.7★ (7)
Gamlehult	104.343.09	£60	4.8★ (4)
Gualöv	703.403.79	£40	4.4★ (7)

Table 2: A table of my favourite Ikea tables.

Now we've got the table, let's look at its input again:

```

\begin{table}[H]
\centering
\begin{tabular}{l|c|c|c}
\hline\hline
\textbf{Name} & \textbf{Catalogue Number} & \textbf{Price} & \\
\textbf{Avg. Customer Rating} & \\\hline\hline
Kragsta & $904.525.87$ & $75$ & $4.5$ \ding{72} (6) \\\hline

```

```

Vittsj\{o} & $802.153.32$ & $45$ & $4.9$\ding{72} (21) \\
Arkelstorp & $302.608.07$ & $110$ & $4.8$\ding{72} (11) \\
Tofteryd & $401.974.86$ & $170$ & $4.8$\ding{72} (6) \\
Malmsta & $602.611.84$ & $125$ & $3.0$\ding{72} (2) \\
Havsta & $604.041.97$ & $85$ & $4.7$\ding{72} (7) \\
Gamlehult & $104.343.09$ & $60$ & $4.8$\ding{72} (4) \\
Gual\{o}v & $703.403.79$ & $40$ & $4.4$\ding{72} (7) \\
\hline\hline
\end{tabular}
\caption{\label{tab:tables} A table of my favourite Ikea tables.}
\end{table}

```

We can see some similarities to the figure environment, but the rest of it is certainly a lot different. So what is this input saying?

- The table environment with the H option is used the same way as the figure environment. You could use the figure environment here instead, but then the caption would read ‘Figure X’ instead of ‘Table X’.
- We call the `\centering` command again to align our table in the centre of the page, as with figure. We also call the caption for the table as with figure, so no surprises there.
- Now we have opened a new environment within the table environment, tabular. This is the environment for making a table itself; as with images, it can be used by itself, but it is good practice to wrap it in the table environment so it displays well.
 - ➔ Immediately after beginning tabular, we see the additional command `{l|c|c|c}`. Notice that this uses curly brackets, not square ones, which indicates this is not optional and must be called when using tabular. Each letter indicates a column for your table, and the `|` indicates a line between adjacent columns. So this command defines how many columns your table will have, and where, if any, there will be separating lines.
 - ➔ The letters themselves indicating the text alignment of the column: ‘l’ represents left alignment, as you can see in the first column of the table, so everything is aligned to the left; ‘c’ represents center alignment, as I’ve done with the other columns in the table. There are other letters, such as ‘r’ for right alignment, which can be used here. Further alignment letters can be found for example at Wikibooks/Tables.
 - ➔ Now we properly move into the tabular environment. The command `\hline` should be clear that it generates a horizontal line across the top of whichever row it is called on. Here I have called this twice, which is simply a style choice.
 - ➔ We then work our way across each row defining what goes in each column. The `&` symbol is again used as an ‘alignment tab’, here separating the columns on each row.
 - ➔ At the end of each row, we use `\\` to indicate we are moving to a new row, and then we can continue this pattern for as many rows as our table needs - we saw this before to make a new line in the align environment for maths. This command can be used in blocks of text as well to jump to a new line if needed.

That is the basics for creating figures and tables in L^AT_EX. These environments are incredibly powerful and there is much more you can do with them than shown here, but these examples should guide you through your early documents. The online resources referenced throughout the text all have more documentation on the more advanced implementations of these environments should you need them.

7 Useful packages

This section will contain a list of packages that are commonly used, with a short description of what they provide. Some may be almost necessary for use in every document you create, others may be needed on a case-by-case basis.

geometry provides easy and versatile ways of controlling document dimensions and geometry

amsmath,mathtools these allow for proper mathematical typesetting

siunitx provides a simple way of expressing SI units in formulae

multicol,multirow provides additional environments to mix single columns/rows with multiple

graphics,graphicx,epsfig accomodates the full functioning of graphics, figures, and images

float improves placement of figures and tables throughout the document

color,xcolor adds additional colour commands

hyperref hyperlink management

pifont call in extra symbols for style

titlesec,titletoc controls the appearance of section (etc) titles in the document and table of contents

fancyhdr further editing of headers and footers

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

As you develop your L^AT_EX documents, and add more packages and customisation to them, your preamble can become increasing long and difficult to manage alongside your document. You can instead create a ‘.sty’ file which can hold all of the input from your preamble. This file can be named anything as long as it has the file extension .sty. Once you have created this document, you can simply call it in to your largely reduced preamble as a package, e.g. for a lab report sty file, you may have labreport.sty, which you can call in to your .tex file as `\usepackage{labreport}` (if the sty file is not in the same folder as your .tex file, you will have to include the full path to the file in the usepackage command).

Creating a .sty file can be as simple as pasting your preamble into this new file, but with one change. For calling packages, where we once had `\usepackage{}` in our .tex file, we now must use `\RequirePackage{}` in our .sty file. Otherwise, we can write out our preamble in this file as before.

It is worth noting that creating a .sty file is not necessary for typesetting in L^AT_EX at all, but you may find it useful when you have a particularly long preamble for some documents, or when you are using a similar preamble for multiple documents, e.g. lecture notes for multiple lecture series. If you have similar documents, but do want slightly different options for some things in each one, you can still define things in your preamble as well as using your .sty file.

9 Labels and Hyperlinks

You will have noticed through some of the examples used so far that the command `\label{}` has been included but I have so far left it without an explanation. This command allows us to reference equations, figures, tables, etc throughout the document. If we are sure that we won’t be including any more above the one we want to reference (thereby changing its number), then we can just do this by hand, but it is always good practice to give each equation etc its own unique label that we can easily refer to regardless of where it appears in the text. For example, I can ask you to refer to Figure 1 or Equation 2, using `Figure \ref{fig:meme}` or `Equation \ref{eq:laurent}`. The `\ref{}` command finds the equation, figure, etc with the given label and returns its number. I have so far labelled every equation, table, and figure used, but it is not necessary to do so. If you have no need for referencing like this through your document, then you are able to create equations, figures, etc without the `\label{}` command included. You can also set labels for sections, chapters, etc in the same way as above to easily reference them even as your document changes. For further information on labelling, see for example Wikibooks/Labels.

Throughout the document, I have created many hyperlinks to useful resources. \LaTeX provides a simple way of creating these through the package `hyperref`. A good coverage of the `hyperref` package and hyperlinks can be found at Wikibooks/Hyperlinks. The hyperlink above was created using `\href{https://en.wikibooks.org/wiki/LaTeX/Hyperlinks}{Wikibooks/Hyperlink}`. So using the command `\href{...}`, I first call the URL I want to link to, then the text description to appear in the text. There are other commands you can use to create hyperlinks, which can be found at the above URL.

10 Drawing in Tikz and PGF

When creating notes, reports, etc, you will often need to include diagrams in your documents. This can be done by finding images in books or papers, or creating them in drawing tools like MS Paint. One of the brilliant things about \LaTeX is that it provides its own way of drawing graphs/diagrams such that they look native to the document. The packages `tikz` and `pgfplots` are brilliantly designed programs which allow you to create your graphics and diagrams to fit perfectly into the scope of your document. Using `tikz` and `pgfplots` takes a bit more learning than the native \LaTeX program, so we won't dwell on it here - it isn't necessary at all when learning to use \LaTeX , though some people (like me) will find it great fun to try out! There are some useful pages on Overleaf explaining the basics of these packages, such as TikZ package, A Tutorial for Beginners, and Pgfplots package. Here is an example of the power of `tikz` with its input, although we won't break down its elements here (keen readers can take it as a challenge to understand and look up how this was put together):

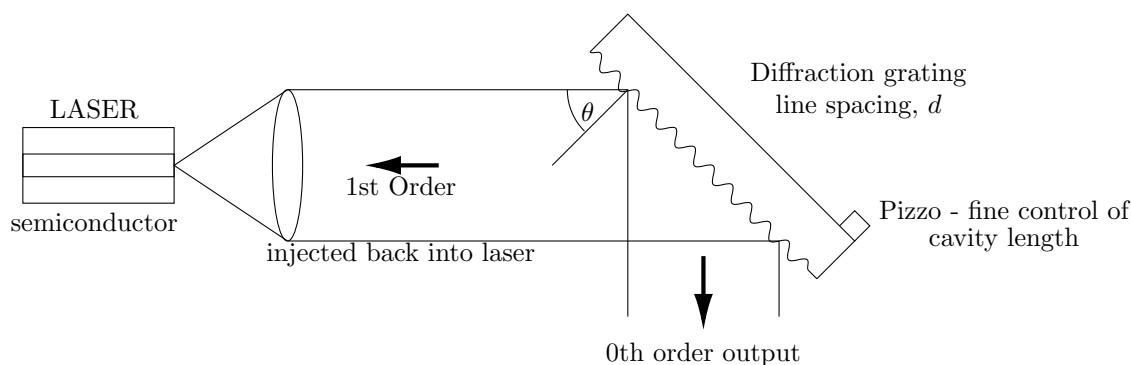


Figure 2: An external cavity diode laser.

```
\begin{figure}[H]
\centering
\begin{tikzpicture}
\draw (0,-0.5) rectangle (2,0.5) node[anchor=south,xshift=-30pt]
{LASER};
\draw (0,0.15) rectangle (2,-0.15) node[anchor=north,xshift=-30pt,
yshift=-10pt] {semiconductor};
\draw (3.5,1) -- (2,0) -- (3.5,-1);
\draw (3.5,0) ellipse (0.2cm and 1cm);
\draw (3.5,1) -- (8,1) -- (8,-2);
\draw (3.5,-1) -- (10,-1) -- (10,-2);
\draw[snake it] (7.5,1.5) -- (10.5,-1.5);
\draw (10.5,-1.5) -- (11,-1) -- (8,2) node[anchor=south west,
midway] {\shortstack{Diffraction grating \\ line spacing, \\ $d$}} -- (7.5,1.5);
\draw (11,-1) -- (11.2,-0.8) node[anchor=west] {\shortstack{Pizzo - \\ fine control of \\ cavity length}} -- (11,-0.6) -- (10.8,-0.8);
\draw (8,1) -- (7,0);
\draw (7.2,1) arc (180:225:22pt) node[anchor=west,midway]
```

```

        {\$\theta$};
\draw[{-\Latex[length=0.5cm,width=0.25cm]},ultra thick] (9,-1.2) —
        (9,-2.2) node[anchor=north] {0th order output};
\draw[{-\Latex[length=0.5cm,width=0.25cm]},ultra thick] (5.5,0) —
        (4.5,0) node[anchor=north,midway,yshift=-25pt] {injected
        back into laser} node[anchor=north,midway] {1st Order};
\end{tikzpicture}
\caption{\label{fig:tkz} An external cavity diode laser.}
\end{figure}

```

11 Referencing

It is important in any paper, report, thesis, etc to include a list of your references. There are many different ways of styling references/bibliographies in L^AT_EX, but we will only mention the natural system built-in here. A very popular method is using BiB_TE_X to create your bibliographies - for this, you create a separate file with all your references which BiB_TE_X can then import into your document and customise to your liking. For overviews of Bib_TE_X, see for example Overleaf or Wikibooks/Bibliography Management.

Below you will find an example bibliography using the built-in system, from which I can use `\cite{a}` to reference [1]. The `thebibliography` environment is designed to go at the very end of the document, although we will traditionally write it there anyway. Let's see how we made it:

```

\begin{thebibliography}{99}
\bibitem{a}
The Wikibooks \LaTeX\, pages, \href{https://en.wikibooks.org/wiki/LaTeX}
{https://en.wikibooks.org/wiki/LaTeX}.
\bibitem{b}
Overleaf's \LaTeX\, guides, \href{https://www.overleaf.com/learn}
{https://www.overleaf.com/learn}.
\bibitem{c}
My github page, \href{https://github.com/mbr-phys}{mbr-phys}.
\bibitem{d}
The Not So Short Introduction to \LaTeX$2\epsilon$,
\href{https://tobi.oetiker.ch/lshort/lshort.pdf}
{https://tobi.oetiker.ch/lshort/lshort.pdf}.
\bibitem{e}
The \LaTeX\, project, \href{https://www.latex-project.org/get/}
{https://www.latex-project.org/get/}.
\bibitem{f}
The Atom text editor, \href{atom.io}{atom.io}.
\bibitem{g}
The \LaTeX\, Font Catalogue, \href{https://tug.org/FontCatalogue/}
{https://tug.org/FontCatalogue/}.
\bibitem{h}
\TeX\, StackExchange, \href{https://tex.stackexchange.com/}
{https://tex.stackexchange.com/}.
\bibitem{i}
\TeX\, blog, \href{https://texblog.org/2008/10/16/lists-enumerate-itemize
-description-and-how-to-change-them/}{https://texblog.org/2008
/10/16/lists-enumerate-itemize-description-and-how-to-change-them/}.
\bibitem{j}
CTAN, \href{https://ctan.org/}{https://ctan.org/}.
\end{thebibliography}

```

After calling the `thebibliography` environment, we have passed the command `{99}` to tell L^AT_EX that we want to be able to create references up to number 99. Here, it is not the number itself that does this, but the number of digits it has; if we were to pass the number 42 here instead of 99, we would still be

able to generate up to 99 references. So if we had just put a single digit number in here, we would only be able to put up 9 references.

For each reference, we start off with the command `\bibitem{}`, which works like `\label{}`, although specifically for references. We then call a reference using its value in `\bibitem{}` with `\cite{}`, as shown above.

This is the simplest example of a bibliography in \LaTeX , which does come with the benefit of being quicker to utilise at first, but I would recommend taking a look at other methods referenced above, as they have a lot more power as you begin to reference more and more.

12 Final Word

\LaTeX can be a deeply rewarding and enjoyable typesetting tool which gives you much more power to control how your documents turn out than most traditional word processors. Learning the basics of this ingenious program can only take a few hours, though it takes a lifetime to master.

I hope that reading this short guide has given insight into the basics of \LaTeX , and that you continue to practice and expand your typesetting skills. This guide should have shown you how to create documents, customise them to your liking, and include several additional scopes such as equations or figures. The range of techniques and implementations covered here is by no way exhaustive, but you should find that most requirements in a document can be met from these pages and immediate references.

The most important skill you need if you are to use \LaTeX is the ability to use a search engine. Through the resources I have mentioned in this text, and so many more, you are likely to find answers, explanations, and solutions to anything you wish to do with this program. **Your most powerful technique is that of Copy and Paste.**

– Matthew Rosseter

References

- [1] The Wikibooks \LaTeX pages, <https://en.wikibooks.org/wiki/LaTeX>.
- [2] Overleaf's \LaTeX guides, <https://www.overleaf.com/learn>.
- [3] My github page, mbr-phys.
- [4] The Not So Short Introduction to \LaTeX 2 ϵ , <https://tobi.oetiker.ch/lshort/lshort.pdf>.
- [5] The \LaTeX project, <https://www.latex-project.org/get/>.
- [6] The Atom text editor, atom.io.
- [7] The \LaTeX Font Catalogue, <https://tug.org/FontCatalogue/>.
- [8] \TeX StackExchange, <https://tex.stackexchange.com/>.
- [9] \TeX blog, <https://texblog.org/2008/10/16/lists-enumerate-itemize-description-and-how-to-change-them/>.
- [10] CTAN, <https://ctan.org/>.