Pimp your thesis: a minimal introduction to LaTeX.

Maarten Bransen
IC/TC, U.S.S. Proton

March 20, 2018

Contents

1 What is LaTeX?

LaTeX, pronounced as "la-tech", with the Greek letter χ (chi), can be seen as a mark-up language for typesetting documents. It is in fact a set of user-friendly commands for a very powerful but complicated typesetting engine called TeX. Contrary to word-processing software you may be used to such as Microsoft Word, Google Drive or OpenOffice, LaTeX is not WYSIWYG (What you see is what you get), and LaTeX is NOT a program that comes with its own user interface. Instead, you create your document in a 'programming-code like' environment with a combination of plain text and *commands* for the typesetting. Then, you *compile* your document and create the final typeset document you want for viewing, which will be a pdf in almost all cases. This may sound unnecessarily complicated, but in practise it is not very complicated and certainly not unnecessary. It is thus a language, and not a program in the way that you may be used to! Usually, you use a separate program, an editor, that has a user interface to help you create and compile your document. With most editors, you no longer have to bother with the programs TeX and LaTeX directly since they are running in the background and controlled by the editor. You just create your document and click 'compile'. It is often said about LaTeX, that rather than WYSIWYG, LaTeX is WYWIWYG (what you want is what you get). The reason is that the program is very good at creating the type of document you probably want (or need) without telling it how to, but it can be highly customised when desired. In particular the scientific community has adopted its use.

2 Why should I use LaTeX?

LaTeX takes some time to learn, and initially a lot of things will seem counter-intuitive or unnecessarily complicated. One may therefore ask themselves, "why would I use LaTeX and not something "simple" like Word?" 'Simple' being between quotation marks, because anyone that ever tried to make a long document in Word with multiple figures, references and equations, knows that this is far from simple. Although big improvements were made over the last few years and many of the features that LaTeX has had since the 80's were added, they often remain buggy or not very flexible. We here give a list with (some of) the reasons why people use LaTeX.

1. It allows you to typeset good looking and consistent documents without much thinking about the layout. The outcome has a professional look and good readability because typesetting is automatically consistent throughout the document.

2. LaTeX is very good for separating the content from the appearance. When making the document, you can truly focus on the "what" rather than the "how". During writing you don't have to (and indeed shouldn't) worry over how it should end up looking. The layout of your document is defined separately, and can be changed afterwards without much work. In many cases you can change the entire look and feel of the document by changing only some style commands in the beginning of your document.

3. Unlike word processing software such as MS Word, LaTeX forces you to write well structured documents. Personally, I find it easiest to *start* with creating the chapters and sections into a logical structure before I do any writing. When the contents or plans

change, shifting the order around is as easy as copy-pasting the text and all the rest (numbering, references, ...) is taken care of automatically.

4. LaTeX is content aware. There are many algorithms working "behind the scenes" to present your content in a good way. There is automatic hyphenation (available in many languages), kerning (the spacing between words and letters), intelligent placement of figures, and many other features. For example, a new section will never begin on the last line of a page. These algorithms optimise a whole paragraph at once in a way that cannot be done 'on the fly' in WYSIWYG software.

5. Numbering of chapters, sections, references, figures, equations, etc. occurs automatic, and after you write the document. As a result, you can easily change the order of things afterwards without having to change any numbering: this is all automatic.

6. LaTeX is incredibly stable for (very) long documents. Bugs are rare after many years of public collaborative development, and when you open and compile a document made 20 years ago it will come out exactly the same as it did back then. LaTeX won't change anything unless *you* tell it to. This also means that creating and sharing templates is very easy, and that tips and advice you may find from years ago will be equally valid today.

7. The software is completely free and open source. There's a huge online community and an even larger amount of documentation, resources, examples, etc. If there's something you don't know, it is very easy to google your question and get an answer that is plain text, and not something cumbersome like: "if you have this version, then go to start, edit, preferences, etc." So far, I have never had a question that wasn't already asked and answered somewhere online.

8. Special characters are very well supported and available without the need to go through any tedious menu. Equations and math are among the most widely listed reasons that LaTeX is used so much in the natural sciences, because they work well and look good. There are even tools such as Detexify that let you draw a character and find the closest matches.

9. Using LaTeX you can do pretty much anything you can think of, in most cases via *packages*. These are created and maintained by users, and needed for anything more than the most basics utilities. If you want something, there's probably a package that does it for you. This also means that the program only loads and installs the functions that are needed for your document.

10. Historic reasons. While word processing software has finally been catching up with some of the listed features in the past few years, LaTeX could do these things already in the early 90's(!) which contributed to its success in science and publishing. As a result, it has become the standard in some fields.

One can of course wonder if there are any *disadvantages* of LaTeX compared to word processing software, and if there are cases where it is easier to use alternative software. One shouldn't use LaTeX:

1. if you don't want to learn it, or you don't have time to. There is a learning curve and, while the effort pays off, in the beginning the way of working takes getting used to. You will find that creating your first document in LaTeX will take more time than you may be used to and that you will not understand how anyone could think that this is what you want. It is advisable to practise long before you have to stress about something like a thesis deadline.

2. if your document is already written. Writing something in LaTeX works in a bit of a different way, and this works much better if you account for it from the start. Don't think: "I'll do it this way now because it is easier, and then I will put it in that neat LaTeX layout later". That defeats most of the advantages.

3. if you want to make an ugly/unclear/poorly structured document in a very short amount of time. For a short one or two page document the use of LaTeX isn't always the fastest way, especially if you don't care so much about the layout.

4. if graphics design *is* the purpose of your document. In principle, almost anything is possible and there are many packages for creating images, graphic elements and designing posters, presentations, etc. Nonetheless, if the design is the sole purpose (rather than the content), other programs such as Adobe Illustrator may be more useful.

5. if you already know you don't want to pursue (academic) research or a research-based master. While I would always recommend LaTeX for writing a thesis (and certainly a longer one), the unfortunate truth is that almost all companies including the chemical industry operate on MS Office. Those who already know that they prefer such a direction, may use their time more wisely by learning to use (tolerate) those programs instead.

3 Using LaTeX

Let's start by giving some examples of commonly used commands and how they are used. For examples, we show the LaTeX source side by side with the result of that code after compilation. On the left you have the source, this is the document as you create it while you are working on, say, your thesis. In other words, this is what *you* create. After you have made this, you can compile the source to create the document. On the right we show what the result generated by the software after you have compiled your source.

3.1 The basic structure

The most basic file would look something like this:

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

Hello world!

The file starts with the *preamble*, or the front matter, which is everything before the `\begin{document}`. This is where we tell LaTeX what type of document to use, what page-layout to use, which fontsize, and what packages to load. Here we specify the *global* settings, those that apply to the entire document. Everything between the `\begin{document}` and `\end{document}` forms the body of your text. At the very minimum, a *class* needs to be defined,

in this case we use `article`. The `article` class is what would typically be used for scientific articles and other short documents, such as the guide you are reading now. Other classes that you can use are for example `report` and `book`. A major difference with the `article` class is for example that `report` and `book` classes are subdivided into *chapters*, that start on a new page with a big chapter header. The chapters are then again subdivided into sections, subsections, etc., in the same way that articles are. The `beamer` class is used specifically for making presentation slides.

You might wonder how the document can be subdivided into chapters, sections, subsections and subsubsections. Because we use the `article` class, there are no chapters. The *body* of the text (the part between `\begin{document}` and `\end{document}`) can be subdivided as follows:

```
\section{The title of the section}
Which can be followed by anything else.

\subsection{a subsection}
Some text that is part of this section.

\subsubsection{a subsubsection}
More text here.
```

1 The title of the section

Which can be followed by anything else.

1.1 a subsection

Some text that is part of this section.

1.1.1 a subsubsection

More text here.

As you can see, we define the *function* of the text and not *layout*. The numbering is also not defined in the code, but taken care of automatically. In other words, switching around sections will not mess numbering up, and you never have to go through your document in the end to check if the numbering is correct. It is also possible to create unnumbered sections, by simply adding an asterix to the commands and typing `\section*{}` instead.

Often, you want a title at the start of your document. You can do this in an easy way with a set of commands for the title, author and date. A simple title such as the one at the beginning of this guide can be generated with the following code[1]:

```
\title{Pimp your thesis: an introduction to \LaTeX.}
\author{U.S.S. Proton}
\date{\today}
\maketitle
```

The `\today` command outputs the current date, but instead you can also type any date you like. If you want to omit the date entirely, you can simply use `\date{}`. Of course in many cases you will want something different, in which case you can manually design and format a custom title or complete title page. To do so you can use the `titlepage` environment.

3.2 Equations and maths

One of the things LaTeX is famous for, is its unrivalled support for typesetting equations and maths. Maths, like many things, is not typed out using normal letters and characters as you might in other programs. Equations are put within a special *environment* which is formatted in a different way, and which use their own set of commands. You will see that the concept of an environment, usually starting with the `\begin{...}` and `\end{...}` commands, is something that returns frequently in LaTeX documents. Generally speaking, there are two ways you can display maths. The first case is in-line maths that you put between your

[1] extra commands were used here to change the colour and font

text, such as this: $E = mc^2$. You can do this by putting your equation between dollar signs: `$E = mc^2$`. The second option is to use the `equation` environment, that gives a numbered and centred equation:

```
Using Pythagoras's theorem we know:
\begin{equation}
  b^2 = \sqrt{c^2 - a^2}
\end{equation}
where $a$, $b$ and $c$ are sides of a
  triangle.
```

Using Pythagoras's theorem we know:

$$b^2 = \sqrt{c^2 - a^2} \tag{1}$$

where a, b and c are sides of a triangle.

As you might have noticed, spaces in math mode are ignored and the correct spacing is dealt with automatically.

```
$2d\sin(\theta)=      n        \lambda$
```

$2d \sin(\theta) = n\lambda$

Most mathematical operators, like other commands in LaTeX are started with the backslash (\) and have arguments between curly brackets. Fractions can be made using `\frac{numerator}{denominator}`, superscripts and subscripts with `e^{ix}` and `E_{kin}` for e^{ix} and E_{kin}. More complicated examples, such as summations and integrations work in a very similar way:

```
The definition of the Taylor series for a
  function $f(x)$ around a point $a$ is given
  by:
\begin{equation}%
  f(x) = \sum\limits_{n=0}^{\infty}
  \frac{f^{n}(a)}{n!} (x-a)^n
\end{equation}
where $f^{n}(a)$ is the $n^{\text{th}}$
  derivative of $f(x)$ in $a$.
\begin{equation}
  \Omega = \int\limits_{a}^{b} \left(
  \frac{x}{2-x}-x^2 \right) d\theta
\end{equation}
```

The definition of the Taylor series for a function $f(x)$ around a point a is given by:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n \tag{2}$$

where $f^n(a)$ is the n^{th} derivative of $f(x)$ in a.

$$\Omega = \int_a^b \left(\frac{x}{2-x} - x^2\right) d\theta \tag{3}$$

While native LaTeX has a wide support for math and equations, packages such as `amsmath` can provide a wide variety of extra options with things like equations spanning multiple lines, alignment options and improved support for matrices. An example of a more complicated equation is given below, and demonstrates that equations in LaTeX can look good even when they are lengthy and contain many brackets or span multiple lines.

$$\begin{aligned} A_{(\mathrm{v})} = & \frac{\pi}{4} n_j \left[\int_0^{\pi/2} \frac{\cos\theta_2}{\cos\theta_1} \left| f_s - f_{p_b} \cos\theta_1 \right|^2 \sin\theta_1 \, d\theta_1 \right. \\ & \left. + \int_{\theta_c}^{\theta_{2max}} \left| \frac{\cos\theta_2}{\sinh\beta} \right|^2 \left(\frac{n_2}{n_1} \right)^2 \left| f_s^* - i f_{p_b}^* \sinh\beta \right|^2 \sin\theta_2 \, d\theta_2 \right] \end{aligned} \tag{4}$$

We will go into more detail about packages in sec. 4.

3.3 Special characters and whitespace

As you can see, some of the characters such as $ are used for a special purpose. Commands always start with the backslash \, arguments to the commands are enclosed in the curly

brackets { and }, and optional arguments are enclosed in square brackets [and]. This means not all of these characters can be used in your text, or TEX can get confused and think you want to use non-existent commands. If you want to use any of the protected characters in your document, you can do so as following:

```
\# \$ \^{} \& \_ \{ \} \~{} \textbackslash{}.
Accents on letters are achieved in a similar
  way: \'a \`e \~o \^i \"u. If you want
  quotation marks, you can use `this',
  ``this'' or "this".

Whitespace           (including tabs  spaces
  and enters) is
ignored by          default. However, a
  double enter starts a new paragraph.

Like this
\\
%anything after the percent sign is ignored
  in compilation
The double backslash forces a line break
  anywhere.
```

$ ^ & _ { } ~ \. Accents on letters are achieved in a similar way: á è õ î ü. If you want quotation marks, you can use ‘this’, “this” or "this".
Whitespace (including tabs spaces and enters) is ignored by default. However, a double enter starts a new paragraph.
Like this
The double backslash forces a line break anywhere.

3.4 Lists

You can make itemized or numbered lists with the `itemize` and `enumerate` environments. These are again enclosed in the `\begin{}` and `\end{}` commands like all environments.

```
some random bit of text, explained in bullet
  points:
\begin{itemize}
  \item this is an unnumbered item
  \item and so is this
\end{itemize}

\begin{enumerate}
  \item this item is numbered
  \item These lists can also be nested:
  \begin{enumerate}
    \item a sub-item
    \item another one
  \end{enumerate}
  \item a third item
\end{enumerate}
Some more text following the lists.
```

some random bit of text, explained in bullet points:

- this is an unnumbered item
- and so is this

1. this item is numbered
2. These lists can also be nested:
   (a) a sub-item
   (b) another one
3. a third item

Some more text following the lists.

Note that the indentation in the source is purely for easy readability of the source, it does not affect the compiled outcome.

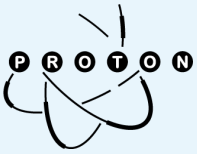
3.5 Figures and tables

To include images in your document, the `graphicx` package must be loaded. To use packages, let us expand the example document from sec. 3.1 by including the `graphicx` package with

the `\usepackage{...}` command. As you may remember, this goes into the *preamble* (the part before `\begin{document}`):

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
Hello world!
\includegraphics[width=2.5cm]{protonlogo}
\end{document}
```

Hello world!



When we load the package, we can use the `\includegraphics{}` command to include an image from a file, where the argument between the brackets is the filename of your image, it is not necessary to include the file extension. By default, the image file should be in the same folder as the .tex document you are working on. We can also see that we can use the *optional* arguments within square brackets to specify things such as the `length`, `width`, `scale` and `angle` of the image. If only the `width` or the `length` are specified, the aspect ratio of the image is kept equal to the original file. In our example we set the width to 2.5 cm, but many different units can be used for defining distances and length, such as millimetres (mm), inches (in) and printer points (pt). A very useful way to define lengths is relative to other things in your document, such as the width or height of the text. For this purpose, the `\textwidth` and `\textheight` units exist. If you for example want a proton logo with a width exactly half that of the lines of text, you can do this by typing `\includegraphics[width=0.5\textwidth]{protonlogo}`. Many internal sizes and default spacings are set with the units `em` and `ex` which correspond to the width of the letter 'm' and the height of the letter 'x' respectively, given the current font and size of the text.

More often than simply inserting an image, you will want to have a *figure*: an image with a caption to accompany a certain part of your text, rather than an separate image in a specific place. To do this, LaTeX makes use of something called *floating* environments. They 'float' through the text as their positioning is determined automatically by algorithms, based on the position and optional restrictions given by the user. For figures, the floating environment `figure` can be used:

```
\begin{figure}[h]
\centering
\includegraphics[width=25mm]{protonlogo}
\caption{the logo of U.S.S. Proton}
\end{figure}
```

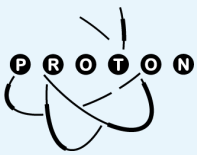


Figure 1: the logo of U.S.S. Proton

You see that inside of the `figure` environment the `\includegraphics{}` command is used in the same way as before. We also use `\centering` to centre the figure and caption on the page. The `\caption{}` command is used to insert a caption, and because we are inside of the `figure` environment the caption is automatically numbered. The `figure` environment takes optional arguments, here we use `h` that means something like: place the figure (approximately) here. We say approximately, because this doesn't happen when LaTeX decides that the image doesn't look good there. The following float specifiers can be used:

`h`	here	where it is placed in the source file
`t`	top	on the top of the page
`b`	bottom	on the bottom of the page
`p`	page	on a separate page with only floating environments
`!`	no, really!	combine with any of the above. Strongly urges LaTeX to use the specified option if the specifier isn't followed
`H`	exactly here	really, absolutely forces placement (needs the `float` package)

You can give multiple specifiers, the order doesn't matter. If no arguments are given, `[htbp]` is used, but generally you will find that figures tend to be placed on the top of the page. You can add the specifier `!` in combination with one or more of the other specifiers to override some of the internal settings in case you *really* want to place a float in a certain place. A common complaint by people new to LaTeX, is that it doesn't put the images where they want them to be. It helps to put the figure *before* the text it accompanies in the source file (due to the way the positioning algorithm works) such that it appears on the same page as the text. In my experience, the placement of LaTeX works well and often when it doesn't do what you want, it is worthwhile to consider if what you were doing was a good idea in the first place. For example, float placement can fail when many small figures or tables are all added separately. In such a case it works (and generally looks) better to combine multiple images into one `figure` consisting of a few subfigures. Of course, specific placement *can* always be forced if necessary by using the `H` specifier (to use this the `float` package must be loaded).

Tables with smart placement and a caption can be formatted in a similar way using the floating `table` environment. The table itself is made in something called the `tabular` environment, which can (but doesn't have to!) be placed inside of the floating `table` environment. An example of a simple table would look something like the following:

```
\begin{tabular}{c | r l}
sample  & amount & quantity     \\ \hline
pie     & 3.14   & surface area \\
honey   & 5      & viscosity    \\
chicken & 9001   & power level
\end{tabular}
```

sample	amount	quantity
pie	3.14	surface area
honey	5	viscosity
chicken	9001	power level

here, `\begin{tabular}{c r l}` means that we use three columns, where `c` is a column that is centred, `l` is a column with left alignment and `r` a column aligned on the right side. Horizontal lines are created with the `\hline` command at the beginning or end of a line. It is possible to place a vertical line between columns by placing a line (|) between the column specifiers as shown in the example, but from a stylistic point of view this should really be avoided at any possible time. The elements within one row are separated with the & sign while each new line is started with the double backslash (\\). Again, the spacing and alignment of the & separators is there purely for clarity in the source, and ignored in compilation. This is a very simple example and much more complicated tables can be created, a detailed description and many examples can be found in the LaTeX Wikibook (en.wikibooks.org/wiki/LaTeX/Tables).

3.6 Referencing and citations

A major argument for using LaTeX for scientific documents is the incorporation of powerful tools for internal referencing and managing citations. Say you want to refer back to a certain section, figure or equation. You can of course type something like: "Equation 3 describes the diffusion of spherical colloids through a liquid". But what if we later on decided to add or remove an equation? The numbering would be automatically updated, making the reference in our text useless. Instead, it is much safer (and, in many cases easier) to define the reference in relation to the equation itself, rather than the number it is assigned *ad hoc*. This is exactly how LaTeX deals with internal references: a custom label can be assigned to a certain environment or part of the document, and referred to using that specific label. The label is created by placing the `\label{mykey}` key within a (sub)section or environment, where `mykey` can be any (unique) label you want. This label is only visible in the source document (the .tex file) and does not appear in the final document after compiling. Once a label is created, it can be referred to using the `\ref{mykey}` command, where the key corresponds to the label you wish you refer to. The `\ref{mykey}` is converted into the appropriate number in the final document.

```
The diffusion of spherical colloids through a
  liquid is given by Equation
  \ref{stokes-einstein}:
\begin{equation}
\label{stokes-einstein}
D = \frac{k_B T}{6\pi\eta r}
\end{equation}
```

The diffusion of spherical colloids through a liquid is given by Equation 5:

$$D = \frac{k_B T}{6\pi\eta r} \tag{5}$$

The same methods apply to figures and tables, but be sure to place the `\label{}` command *after* the `\caption{}` command, and within the floating environment. Using the `\pageref{mykey}` command you can refer to the pagenumber of the labels and the `\autoref{}` can be used to automatically generate the label "Equation", "Figure", etc. They can be used as following:

```
the Stokes-Einstein equation is given on page
  \pageref{stokes-einstein} in
  \autoref{stokes-einstein}. You can also
  refer to a page like this:
  \autopageref{stokes-einstein}.
```

the Stokes-Einstein equation is given on page 10 in Eq. 5. You can also refer to a page like this: page 10.

It should be noted that it can be necessary for LaTeX to be compiled more than once before are references are shown correctly.

In the same spirit as cross-referencing within your document, it is much safer and easier to resolve the numbering of literature citations automatically. LaTeX has several methods to manage the literature references automatically, based on the `\cite{mycitekey}` where the `mycitekey` can again be any user defined label. The following is an example of a simple way to do the citation:

```
The stucture of DNA is a double helix \cite{Watson1953}. General relativity is much more
  complicated \cite{Einstein1922}.

\begin{thebibliography}{9}
\bibitem{Watson1953} Watson, James D., and Francis HC Crick. ''\emph{The structure of DNA.}''
  Cold Spring Harbor symposia on quantitative biology. Vol. 18. Cold Spring Harbor Laboratory
  Press, 1953.
\bibitem{Einstein1922} Einstein, Albert. ''\emph{The general theory of relativity.}'' The
  Meaning of Relativity. Springer Netherlands, 1922. 54-75.
\end{thebibliography}
```

The structure of DNA is a double helix [1]. General relativity is much more complicated [2].

References

[1] Watson, James D., and Francis HC Crick. "*The structure of DNA.*" Cold Spring Harbor symposia on quantitative biology. Vol. 18. Cold Spring Harbor Laboratory Press, 1953.

[2] Einstein, Albert. "*The general theory of relativity.*" The Meaning of Relativity. Springer Netherlands, 1922. 54-75.

A `\bibitem{mycitekey}` is created for each reference inside of the `thebibliography` environment, and used to cite the reference from the text. As you can see, LaTeX automatically replaces the `\cite{meycitekey}` command with the correct number.

While this is a very simple way of managing citations, it is also severely limited. The order of the entries in the bibliography is determined by the input order of the `bibitems`, not by the order of citing or on alphabetical order. For these (and many other) reasons, using this method is discouraged for anything more than a few references. A more advanced and automated way of handling citations in LaTeX is to a program called BibTeX. This program, which can be executed automatically during compilation in all common editors, automatically formats your citations and generates a `thebibliography` environment in the correct order and format. For BibTeX to be able to do so, you need to provide the necessary information about the literature you want to cite in a separate document, with the extension .bib and entries that look something like this:

```
@book{pimpyourthesis,
author    = {Proton, Utrechtse Scheikundige Studievereniging},
title     = {Pimp your Thesis: an introduction to \LaTeX},
publisher = {Proton},
year      = {2017},
pages     = {1--11}}

@article{debye1949,
title={Scattering by an inhomogeneous solid},
author={Debye, Peter and Bueche, Arthur M},
journal={Journal of Applied Physics},
volume={20},
number={6},
pages={518--525},
year={1949}}
```

Each entry is of a certain type, such as `article` and `book`, followed immediately by the citation key you want to use. All the information about the sources, such as the authors, journal and year are supplied in the necessary fields. This is all saved in a file `mybibliography.bib` in the folder of your LaTeX document. The bibliography is then created simply by calling the correct file using the `\bibliography{}` command:

```
A famous scientist from Utrecht is Peter Debye who worked on colloids \cite{debye1949}.
  Nowadays in Utrecht you can learn about \LaTeX\ in Proton's ''Pimp your thesis'' workshop \
  cite{pimpyourthesis}.

\bibliographystyle{ieeetr}
\bibliography{mybibliography}
```

A famous scientist from Utrecht is Peter Debye who worked on colloids [1]. Nowadays in Utrecht, you can learn about LaTeX in Proton's "Pimp your thesis" workshop [2].

References

[1] P. Debye and A. M. Bueche, "Scattering by an inhomogeneous solid," *Journal of Applied Physics*, vol. 20, no. 6, pp. 518–525, 1949.

[2] U. S. S. Proton, *Pimp your Thesis: an introduction to LaTeX*. Proton, 2017.

As you can see, the order of the entries in the `mybibliography.bib` file does not matter, as the order of the items in the bibliography is created automatically based on the `bibliographystyle` . In this case we use `ieeetr`, but if you choose a different style, the appearance of the bibliography, and possibly the citations themselves, changes. Using a different format for your citations, for example whether to abbreviate first names or whether to print the title, *after* you have already made the documents now only requires changing a single line in your document. If you decide to add or remove a section or to change the order, the citations are updated automatically which makes it very easy to change details of your thesis at the end, for example after you received feedback.

Note that during compiling you need to run LaTeX once, then BibTeX once and then LaTeX twice, before all the citations are correctly labelled and formatted (this sounds a lot more complicated than it is). Creating a `.bib` file with all of your references may seem like a tedious task, and it is if you do it manually. Fortunately, there are many methods and tools to help you do this semi or fully automatically. A very easy method is to use Google Scholar, that has a cite button beneath every link with several options, including a formatting for BibTeX which you can simply copy-paste. Another method that is very convenient, is to use Mendeley for keeping track of your literature. Mendeley has automatic options to export your entire bibliography in the correct format for BibTeX.

3.7 Layout and style

When you have written your thesis you may want to give it some 'flair' and add some degree of styling and customization. Overall, there are virtually no limitations on what is possible style-wise. That being said, of the many available options some are more straightforward than others. One of the most basic stylistic choices to make are the general page layout and that of the text itself. The most basic settings are those given to the `\documentclass[<options>]{<class>}` command. For example, when you use `\documentclass[a4paper,11pt,twoside]{article}` you get a document in the `article` class with a text size of 11 printer points for the normal text. The option `a4paper` tells the program that we want to use paper of the standard A4 size. Lastly, `twoside` makes the document a two-sided document which means that the appearance of odd and even pages will be slightly different. An example is that for a (printed copy of a) twosided document the page number may always appear on the outside of the page (i.e., left on the left pages and right on the right pages).

In some places you may want to use a different styling of the font, for example something larger than the main body of the text (such as is used in headers) or something smaller (for example in captions of figures or in footnotes). In LaTeX you can achieve this by using commands such as `\small` and `\large`. The nice thing is that these commands are defined

relative to the size of the main body of your text, rather than by explicitly giving an absolute font size in printer points. This means that they will scale accordingly with the size of your font if you choose to change it. The following size specifications can be used:

```
\tiny some tiny text

\scriptsize
  scriptsize text

\footnotesize
  footnotesize text

\small small text

\normalsize
  normalsize text
```

some tiny text
scriptsize text
footnotesize text
small text
normalsize text

```
%of course the text
  can also be larger
  than the default
\large large text

\Large Large text

\LARGE LARGE text

\huge huge text

\Huge Huge text
```

large text
Large text
LARGE text
huge text
Huge text

Other possibilities are of course to emphasise text by using italics or boldface, or to switch between serif or sans serif fonts. For these things usually two commands exist one to change it locally, which applies its effect on any thing given as the argument in curly brackets, or commands that apply for for as long as that piece of text continues without new specifications. Below you can see some examples of their use:

```
You can make text \textbf{bold},
  \textit{italic}, \textsl{slanted},
  \textsc{small capitals} or the default
  \textup{upright}.\newline

You can also use the analogous commands to
  switch until a new command is given, such as
  \bfseries for bold text or \itshape for
  italic sections\mdseries\upshape.\newline

You can \emph{emphasize} something in a way
  that depends on the surrounding
  context.\newline

Different font families are \sffamily sans
  serif, \ttfamily a monospaced (typewriter)
  font \rmfamily and the default roman (serif)
  font.
```

You can make text **bold**, *italic*, *slanted*, SMALL CAPITALS or the default upright.

You can also use the analogous commands to switch until a new command is given, such as **for bold text or *for italic sections***.

You can *emphasize* something in a way that depends on the surrounding context.

Different font families are sans serif, a `monospaced (typewriter) font` and the default roman (serif) font.

Lastly, in addition to changing the font family you may want to change the font entirely. This is usually done via packages that load the appropriate font and set it to the default for the document. For example, writing

```
\usepackage{libertine}
```

will set the documents serif font family to the Linux Libertine font. For a comprehensive list of available fonts and how to use them you can visit the LaTeX Font Catalogue.

4 Some other useful packages

In general, 'bare' LaTeX contains little functionality and relies on packages for anything but basic text. The philosophy behind this, in part, is that you only load what you need, thus keeping the program efficient. One should therefore view them as an essential component, not as some extra plugin or toolbar which adds some obscure functionality. In other words:

don't be afraid to use them! Virtually all packages and their documentation can be found on ctan.org, but most TeX distributions automatically contain all common packages, thus omitting the need for manual installation. More information on this is given in sec. 6.1. The following is a list (in no particular order) of some of the packages I find useful to know about and that I often use in my LaTeX documents.

`amsmath`	this package provides additional commands for mathematics. Often used in combination with `amssymb` for extra symbols and characters
`geometry`	A very useful package for changing the overall page-layout and margins of your document
`hyperref`	Automatically generates hyperlinking in your pdf. Click on the table of contents, references to the bibliography, sections, figures, equations, etc. to automatically scroll the pdf to the correct page. Also works for external URL's.
`siunitx`	A package to automatically typeset numbers, units and constants to the SI (or user-defined) standard in a way that prevents typos.
`mhchem`	a package for typesetting names of chemical compounds and simple reaction equations
`chemfig`	if you want to draw more advanced chemical structures, reaction mechanisms or complex equations, this package provides the tools to do so directly in your document without importing figures
`todonotes`	Provides commands to place very clearly visible to-do notes in the margins of your document, which is useful for any "work in progress". Using a single argument in the preamble, all to-do notes can be turned off for the final version so that no comments you wrote to yourself are left for your supervisors to see
`tikz/pgf`	a very powerful (but also complicated) package for drawing vector images directly in LaTeX. Can be used even for plotting data
`booktabs`	provides some new commands useful for typesetting good-looking tables that are easy to read
`tabularx`	automatic sizing of columns in a table to a specified table width
`caption`	a package that provides extra options for customising the way captions are displayed
`cite`	Provides a large amount of control over the way citations are formatted and how they appear in your text
`babel`	a package supporting many languages (including Dutch) that has improved hyphenation and automatically translates generated text such as "Figure 1:" and the table of contents. Mostly useful when not working in (UK) English
`bilatex`	widely considered the successor of BibTeX and even more comprehensive solution for managing references and citations with possibilities to define your own citation styles
`microtype`	enables extra options for font kerning and spacing. By intelligently varying spacing parameters it gives text a much nicer 'feel', even though the effects are subtle to see directly. It also reduces the number of overfull and underfull hbox errors
`fancyhdr`	for adding good-looking headers and footers to documents with a complete toolset for customizing the page style to your liking

5 Practise exercises

5.1 Introduction

For these exercises we'll use Overleaf, an online LaTeX-editor. This way, you won't have to install any programs on your computer for the compilation of your LaTeX-documents. Start by visiting overleaf.com and create an account. After you've created an account, log

in to it and create a new project. As you can see Overleaf offers quite a lot of ready-to-use templates for a variety of documents. In this case, we'll just use the 'Blank Paper' template under 'Basics'.

After you've created the new project, you'll see three panels, the left one gives you an overview of all files (this includes .tex files, images, bibliographies, etc.) in your project. The middle panel is used to write your actual LaTeX-code. This code will automatically be compiled by overleaf and the result will be shown in the right panel. Overleaf has automatically created the main tex file (main.tex), we'll use this document for our practice exercises.

5.2 The basics

After you've created a new project, make sure that you use the `article` class and a4 paper

```
\documentclass[a4paper]{article}
\begin{document}
(Type your content here.)
\end{document}
```

(Type your content here.)

Exercise 1 Setting the language

For correct hyphenation, you can include the package `babel`, you can give the language of your document as an option to the inclusion of this package. Add the package `babel` to the *preamble* of the document and add the option *english*.

Exercise 2 Adding the title

This page still looks a bit boring. Remove the default contents of the page (the whole body of the document) and add a *title*, *author* and *date* to the document and display them at the top of the document.

Exercise 3 Time to add some contents

Now you've created a title, it's time to add some contents. Try to reproduce the contents of the box below in LaTeX:

1 A section containing some difficult characters

In March 2006, Congress raised that ceiling an additional $0.79 trillion to $8.97 trillion, which is approximately 68% of GDP. As of October 4, 2008, the "Emergency Economic Stabilization Act of 2008" raised the current debt ceiling to $11.3 trillion.

A section without a number

As you might have noticed, this section doesn't have a number.

2 Here's another section

2.1 This section even has a subsection

And some more text

2.2 Another subsection

2.2.1 Groceries

A simple list of my groceries:

- 4 eggs
- 1L milk
- 1kg flour

Exercise 4 Adding the table of contents

To be able to quickly find the right section, it it handy to add a table of contents. Do this by adding the command `\tableofcontents` just below the command you used to display the title, author and date.

Exercise 5 Time for some mathematics

Start by including the `amsmath` package in the preamble of the document, you can add this package without any options (just omit the `[option]` part). Now, try to reproduce the following under a new section, called 'Mathematics', in your document. For the equation, use the `equation` environment, the second κ^{-1} should be displayed using in-line math within that sentence.

3 Mathematics

3.1 Debye length

$$\kappa^{-1} = \sqrt{\frac{\epsilon_0 \epsilon_r k_B T}{2e^2 n_0}} \tag{1}$$

The previous equation shows how the Debye length in an electrolyte can be calculated. κ^{-1} is usually expressed in nanometers (nm).

Exercise 6 Adding a table

It's time to add a table. Add a table such as Tab. 3 to your document. Can you add vertical lines on the sides to close the box? Does this look better?

Table 3: Ingredients for apple crumble

Amount	Ingredient
200 g	flour
100 g	butter
100 g	sugar
2	apples
2 tbsp.	cinnamon

Exercise 7 Adding an image

It's now time to add an image. Upload an image from your computer to your project in Overleaf. You will see that it is added to the panel with the files on the left side of the window. There you can also see the exact file name. Add the image you just uploaded to your document, including a caption and a label that you can use to refer to this image. Refer to the image using `\ref{yourlabel}`, this will output the number of the figure.

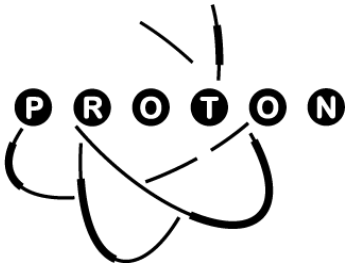


Figure 2: The logo of U.S.S. Proton

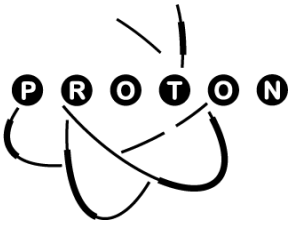


Figure 3: The logo if it was slightly smaller

5.3 Advanced exercises

These exercises are a little more advanced for those who already have experience with LaTeX or those who like a challenge right away. The exercises show some neat tricks and some possibilities for a little more than a 'standard' LaTeX document. The use of Google may be necessary and is encouraged, as the best way to learn something is to try and work it out yourself.

Exercise 8 More on images

Let's look at one of the things that makes LaTeX superior over many word processors: the ability to work with vector images. Make a simple graph in your software of choice (Mathematica, Excel, Origin, ...) and save it twice, as .jpg and as .pdf. Now include both images in your document, and place them side by side in a single `figure` environment, with both of the images occupying half of the width of the text. Hint: a line break between the images can be prevented by ending the line with a percent sign (%). Now zoom in and compare the image quality of both images, what do you see? What do you think happens when you print such a page on paper using a (inherently vector-based) laser printer or photocopier? What about text in the images?

Next, place a single caption beneath the images. Sometimes however you may want to put two unrelated images side by side in which case a single caption for both could be undesired. Try to give each of the images their own caption, such as in Fig. 2 and Fig. 3. Lastly, try and place a figure on the side of the page with the text flowing around it. Hint: look at the package `wrapfig`.

Exercise 9 More on tables

For basic tables the `tabular` environment provides the necessary tools, but many packages exist for more advanced options such as automatic column spacing. Load the `tabularx` and `booktabs` packages and try to recreate the following table below. Note that the table has columns of even width to a total table width equal to the width of the surrounding text. Furthermore, the top and bottom rules are slightly thicker as the rule separating the column headers from the data. For this example the table is filled with some random numeric values with error, but you can fill your cells with anything you like.

Table 4: measured transfer rates for nanoplatelets of increasing lateral size.

sample	**diameter** (nm)	A_{max}	E_{max} (eV)	γ^r ($K\,s^{-2}$)	Γ^{np} ($pJ\,s^{-2}$)
NP01	7.3 ± 0.2	0.80 ± 0.01	1.20 ± 0.12	2.3 ± 0.1	80.1 ± 0.1
NP02	8.1 ± 0.1	0.82 ± 0.01	1.18 ± 0.15	2.2 ± 0.1	72.1 ± 0.1
NP03	10.1 ± 1.2	0.81 ± 0.02	1.09 ± 0.27	2.1 ± 0.2	69.3 ± 0.1
NP04	12.3 ± 0.2	0.89 ± 0.01	0.99 ± 0.10	2.1 ± 0.1	50.7 ± 0.3
Xi *et al.*[3]	10.7	0.92 ± 0.05	-	-	48.2 ± 0.2

Exercise 10 Defining commands

Sometimes, you might want to do the same thing multiple times in LaTeX which can get a bit annoying if it consists of the same code over and over with only few changes each time. For this purpose, it is possible to define your own commands in LaTeX, starting with a backslash. Such commands can be defined with or without variable arguments, and can contain pretty much any other LaTeX commands or text. Commands can be defined in the preamble with the `\newcommand{\mycommand}[]{definition of my command}` where the optional argument specifies the number of arguments the function takes, and the `#1` in the definition is replaced by the first argument of your function, `#2` by the second, etc. An example of a function with and without optional arguments is would be something like the following. In the preamble of your document you define the functions:

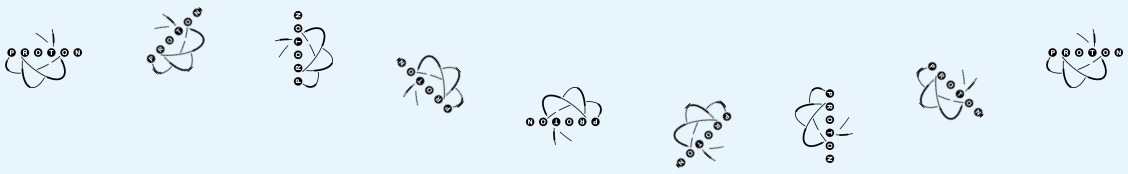
```
\newcommand{\cheese}{\large CHEESE\normalsize}
\newcommand{\food}[2]{The #1 food is #2}
```

You can then use these commands in the body of your document:

```
I really like \cheese, I think that \cheese\
  is the best. \food{worst}{broccoli}.
\food{greatest}{\cheese}!
```

I really like CHEESE, I think that CHEESE is the best. The worst food is broccoli. The greatest food is CHEESE!

Try and define a command that places a picture with a width of 1 cm and that takes only one argument: the rotation of the picture. Using your own picture and command, try and produce a line on your page with rotating pictures, evenly spaced over the line:



Hint: for the spacing look at the `\hspace` and `\hfill` commands.

Exercise 11 Making a title page

For a longer report you will typically use the `report` or `book` class rather than `article`. Create a new document, but this time use the `report` class. Instead of using the rather simple `\maketitle` command, create a fully custom title page in the `titlepage` environment that you can create with the `\begin{titlepage}` and `\end{titlepage}` commands.

Exercise 12 Using different page-numbers
It is common in long reports and books to use a different type of page-numbering for the abstract, preface and table of contents, than for the rest of the document. Start with an unnumbered chapter for the abstract, then the table of contents and then some chapters you could typically find in a thesis (Introduction, Methods, Results, etc.). Include a simple bibliography with your method of choice, and find a way to include it in the table of contents. Now try and use Roman numerals (I, II, ...) for the abstract and toc, while starting your normal page-numbers (1, 2, ...) on the first page of the introduction.

Exercise 13 Challenge yourself!
Think of things you see in theses of books that you like. Are there things you needed for an assignment or thesis but couldn't realise? Is there something you don't know how to do? This is your chance to ask and find out.

6 Concluding remarks

6.1 A few words on installation

For the exercises we suggested using the online TEX platform Overleaf because it does not require any installation and deals with compilation automatically. Platforms such as Overleaf and ShareLATEX therefore allow for a quick set-up, in addition to some nice features such as collaborative writing (convenient for assignments). For a longer document like a thesis, it is usually more efficient to install the TEX engines and packages on your computer directly and avoid the need to upload all your images and reference files all the time. To avoid having to manually install programs, styles and packages every time you need them, software distributions have been created that contain virtually all of the commonly used programs, compilers and packages. The two most notable TEX distributions are MiKTEX and TEX live and it is adviced to install one of these distributions. In addition to this you can use an editor of your choosing such as TEXmaker, TEXworks or TEXstudio.

6.2 Some tips

- Dive right in and get going. Maybe the beginning seems a bit daunting after reading this or after seeing examples, but when you start with the basics it is really not that bad. The best (and in my opinion only) way of learning this is by doing it.
- Google is your friend. If you have a question, the answer is probably already given on some message board or website. Approximately 100% of the knowledge in this document was learned through searching for the answer online and reading what came up.
- Print the LATEX 2_ε Cheat Sheet and hang it next to your computer to quickly look up commonly used commands.
- Don't use (complicated) templates that you don't understand. Templates may be very useful, but if they are full of commands and packages that you don't understand it may lead to error messages and unwanted behaviour. Using a step-by-step guide with annotated examples on the other hand is strongly encouraged.

- Lastly, if it's not for you, it's not. In my personal opinion it is totally worth putting in the effort to learn, and I would strongly urge anyone to not give up too quickly. That being said, some people like WYSIWYG more or may be already skilled with word processing software.

6.3 About this document

The full annotated `.tex` source for this file is available on request on m.bransen@uu.nl. The main font used is the `lmodern` font, a slightly altered version of the default `computer modern` font. Titles and headers are adjusted using the `titlesec` package and typeset with the corresponding sans serif family. Code snippets are typeset in a scaled down version of the `beramono` font with the use of the `listings` package. The headers were made using `fancyhdr`. I used MikTeX for installation and my editor of choice is TeXstudio.

I am by no means a graphic designer or (thesis) writing expert, nor am I particularly skilled or experienced with LaTeX. Corrections, ideas and suggestions are always welcome to improve this document.