INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES
DE MONTERREY

DEPARTMENT OF CHEMISTRY

THESIS TITLE IN ALL CAPS.

BY
STUDENT'S FULL NAME IN ALL CAPS

RESEARCH PROJECT PRESENTED AS PARTIAL REQUIREMENT FOR THE
DEGREE OF BACHELOR OF SCIENCE IN CHEMISTRY

MONTERREY, N.L. MONTH YEAR

THESIS TITLE IN ALL CAPS.

BY
STUDENT'S FULL NAME IN ALL CAPS

HAS BEEN APPROVED
MONTH YEAR

EVALUATING COMMITTEE:

Full name and title, Adviser	Full name and title, Adviser
Full name and title	Full name and title

Full name and title

ACCEPTED:

Full name and title
Head of the Bachelor's Degree in Chemistry

Full name and title
Head of the Department of Chemistry

Summary

Use a corrected, single column version of your evaluating committee summary. You will have made mistakes because you probably finished the summary mere hours before your defense.

ACKNOWLEDGEMENTS

Give thanks guys, you didn't get here alone. Make sure to include academic and personal mentors, friends, and family members who got you there.

In my case, I have to thank Professor Marcelo Videa Vargas—who was also an adviser in my actual thesis—for creating the original version of this template and introducing me to LaTeX via his fearsome exams.

DEDICATION

Keep it short, keep it sweet, dedicate it to someone who will not leave your side and/or mind any time soon.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Introduce us to your research topic.

Chapter 2

Methodology

Tell us how you went about doing science.

Chapter 3

Results

Tell us what you achieved.

Chapter 4

Conclusion

Tell us what you—and we—have learned by all this. Tell us what you achieved, what it means for science, and what that means for the uninitiated.

Appendix A

Introduction to LaTeX

A.1 Quick Description

LaTeX (written `\LaTeX`) is an open-source document typesetting language originally created as TeX by renowned mathematician and computer scientist, Donald E. Knuth, as a means of incorporating the best typesetting practices of academic—predominantly mathematical—journals, in an easy to use package.

TeX has now been expanded into LaTeX—which wraps around many primitives and makes the language more accessible. What started as an academic typesetting language for mathematicians has expanded its scope to include all three basic sciences, computer science, engineering, writing, and even music. It's an extremely flexible tool that can produce all kinds of documents from personal letters to beamer presentations, and everything in between.

The language is characterised by its modularity, flexibility, and ability to produce beautiful, standardised, and reproducible documents. Admittedly, the learning curve can be very steep—especially if one is pressed for time—but the end result is *very* worth it.

LaTeX is not a word processor, it's a typesetting languange. As such, every LaTeX document is comprised of two large sections:

1. Preamble: Where the user loads relevant modules, *packages*, which define a document's characteristics and provide functionality via *backslash commands* (commands for short) and *environments*.
2. Body: Contains the document's actual content.

A.2 Getting Started

There are several freely available distributions depending on your OS, as shown in table A.1.

Table A.1: LaTeX distributions.

Distribution	OS
TeX Live	Linux
MacTeX	OS X
MikTeX & ProTeXt	Windows

Since LaTeX is a language rather than a word processor, documents can be produced in any plain text editor and compilation carried out via console command. However, this is inefficient and there are better alternatives:

- VimLaTeX: a Vim module for writing LaTeX documents.
- Emacs/XEmacs + AUCTeX + RefTeX: modules for writing LaTeX documents on Emacs/XEmacs.
- Dedicated editor: there are a variety of dedicated editors, some of which come with different distributions. TeXworks for MikTeX, TeXShop for MacTeX and TeXStudio for ProTeXt. My personal favourite is TeXStudio, because it's *1*) lightweight, *2*) easy to use, *3*) compatible with Windows, Linux and OS X, and *4*) comes with a lot of quality of life and advanced features; such as automatic quotes, user-defined compilation trains, wizards, macros, folding, etc.

Being a non-interpreted programming language, LaTeX codes generate documents upon successful compilation. Compiling is done via console commands or keyboard shortcuts. All editors provide their own defaults, but they're also re-configurable. There is also something known as a *compilation engine*, of which there are a few. The two most modern are XeLaTeX (or XeTeX), and LuaLaTeX (or LuaTeX). The standard is still PDFLaTeX which does not provide full unicode encoding. All the distributions mentioned above come with all three (and more) engines. The default compilation engine can be specified within your editor of choice. This document was created with XeLaTeX, because it is more mature (less buggy, more features) than LuaLaTeX, and more competent than PDFLaTeX. Which means you have to use XeLaTeX if you want to compile the source code.

A file with a `.bib` bibliography must also be run through the BibTeX engine, which also comes in all aforementioned distributions and can be done within your editor of choice. Further details provided in subsection A.19.2.

A.3 Best Practices

Code can get ugly pretty quickly, especially when done wrong. So here is a checklist of things that will make your life easier.

- Indent sensibly. LaTeX does not interpret manual indents as spaces, that's what the spacing commands are for.
- Break apart troublesome lines. A single line break will not interrupt text flow unless a blank line is added between segments. This helps a lot when debugging and writing long equations.
- Divide and conquer. Use information theory ($\log_2 x$) to optimise your debugging. Divide things progressively in half (comment out segments), narrowing down your search by halving your search space with each iteration.
- Structure your approach to equation writing. Simplify equations as much as possible. Typeset small, independent sections and incorporate them where they are needed *after* they have been checked for mistakes. Repeat process until the complete equation has been typeset.
- Use sensible labels. I like to use "p"–part, "c"–chapter, "s"–section, "sb"–subsection, "ssb"–subsubsection, "f"–figure, "sf"–subfigure, "t"–table, "e"–equation, "se"–subequations. Followed by a colon and a short but descriptive name.
- Automate as much as possible. Define user commands for things you will use often.
- Customise your editor of choice. Take time to set up your theme, shortcuts, custom macros, auto-complete, compilation train, flags, dictionary, etc. Make sure you learn your most used shortcuts, macros, and commands.

A.4 Special Writing Characters

There exist special writing characters which have specific, non-trivial LaTeX characters. Table A.2 contains some of the most common ones, their LaTeX character, and usage.

Table A.2: Special writing characters.

Name	Print	Code	Usage
English Half-Quotation	‘’	`` `' ``	Nested quotation.
English Quotation	“”	`` ``'' ``	Speech or technical term.
Hyphen	-	`-`	Compound words.
En Dash	–	`--`	Ranges, scores, conflict/connection.
Em Dash	—	`---`	Cursory explanation, no pre/post-space.
Double Em Dash	——	`------`	Interrupted word, no pre-space.
Ellipses	…	`\ldots`	Pause, pre/post-space optional (format).

A.5 Reserved LaTeX Characters

Like many programming languages, LaTeX contains certain reserved characters used for different native processes. Table A.3 shows the symbol, its escape method (how to print them), and function.

Table A.3: Table of special LaTeX characters, their escape method, and function.

Symbol	Escape	Function
%	`\%`	Comment.
\	`\textbackslash`	Escape other characters, initiate macro.
&	`\&`	Align delimiter.
~	`\~{}` or `$ \sim $`	Non-breaking space. It's a character, not a space.
#	`\#`	Macro numbered argument.
$	`\$`	Maths environment (`$ maths $`).
$$	`\$\$`	Display maths environment (`$$ maths $$`).
^	`\char`\^`	Superscript (maths environment).
_	`\_`	Subscript (maths environment).
\\	`\textbackslash`(×2)	Line break.
{}	`\{\}`	Argument.

A.6 Environments and Commands

A big part of LaTeX's appeal is the fact that it uses explicit environments and commands which are loaded by the document class and packages declared in the preamble. Environments are sections of code with special rules, and may contain arguments and/or options. In contrast, commands[1] have a certain function, but do not change

[1] In a way, packages are commands which load other commands.

the rules of a section of code. They may also contain options and/or arguments. It's worth mentioning that not all environments and commands need or even use options and/or arguments. Options are always found within chevrons `[options]`, while arguments within braces `{arguments}`, options *always* go before arguments.

Environments are specified as:

```
\begin{environment}
    ...
\end{environment}
```

while commands are simply written as `\command`. Options and arguments—if needed—are added immediately after declaration. Commands also gobble one space immediately to their right, avoid this by adding a nonbreaking space, `~`, between it and the following text, e.g. `\command~text`. Punctuation can be placed immediately following a command without problem.

Some commands and environments have *starred* versions, where an asterisk, *, is placed immediately after their name `\command*`, `\begin{environment*}` and `\end{environment*}`. Starred versions change to an alternate behaviour, usually something to do with numbering or indexing.

A.6.1 Font Commands

There are a myriad of fonts and font types so only the most common ones will be included here.

The standard LaTeX font is the famed Computer Modern font family, which despite its superiority over plebian fonts, still has its minor flaws. Which the Latin Modern font family was designed to fix. Computer Modern is still the default font in most LaTeX distributions, so Latin Modern must be loaded with the `lmodern` package at the very end of the package declaration because it redesigns many fonts and symbols loaded by other packages.

There are also many font styles, the most common ones are found in table A.4. Some of them can be nested to produce a combination of effects, but only if the combination exists within the document's font family.

Table A.4: Common font styles.

Command	Result
`\emph{ABCdef123}`	*ABCdef123*
`\textsf{ABCdef123}`	ABCdef123
`\texttt{ABCdef123}`	ABCdef123
`\textit{ABCdef123}`	*ABCdef123*
`\textsl{ABCdef123}`	*ABCdef123*
`\textsc{ABCdef123}`	ABCDEF123
`\uppercase{ABCdef123}`	ABCDEF123
`\textbf{ABCdef123}`	**ABCdef123**

There are predefined font sizes whose absolute size varies depending on document class and default font size. They are declared as:

```
{\fontsizecommand <text or environment>}
```

Table A.5 contains these commands and what each does in a report with 12pt font size.

Table A.5: Pre-defined font sizes.

Command	Effect
`\tiny`	ABCdef123
`\scriptsize`	ABCdef123
`\footnotesize`	ABCdef123
`\small`	ABCdef123
`\normalsize`	ABCdef123
`\large`	ABCdef123
`\Large`	ABCdef123
`\LARGE`	ABCdef123
`\huge`	ABCdef123
`\Huge`	ABCdef123

Arbitrary font sizes can be selected by using:

```
{\fontsize{<size>}{<line space>}\selectfont <text or environment>}
```

where the size can be given as `pt`, or any properly abbreviated imperial or metric measure.

A.6.2 Include and Input Commands

`\input{}` and `\include{}` are two *very* special and useful commands. They take a `.tex` file as argument without extension.[2]

`\input{filename}` is very low level, and it simply inserts `filename.tex`'s content into the `.tex` file where it is called. It can be nested, i.e. input files can found be within other input and include files, and be used in the preamble—which is useful when documents share core packages and configurations.

`\include{filename}` is a bit more complicated and *much* more useful when it comes to sectioning. This command forces a `\clearpage` before and after inserting `filename.tex`'s content, so it's usually reserved for inserting chapters, as they already start on a new page. It also does some very convenient TEX-fu with `.aux` files, only compiling those whose corresponding `.tex` files have been modified, rather than the whole document's. This makes the first complete compilation noticeably slower than normal, but considerably hastens subsequent ones.

A.7 Spacing

Table A.6: Spacing commands.

Command	Function
`\indent`	Forces indentation.
`\noindent`	Removes automatic indentation.
`\hspace{}`	Add horizontal space in the desired units.
`\vspace{}`	Add vertical space in the desired units.
`\hfill`	Fill the page horizontally with space.
`\vfill`	Fill the page vertically with space.
`~`	Non-breaking space. Treated as a letter rather than a space.
`\,`	Thin space (maths and text).
`\;`	Thick space (maths).
`\:`	Medium space (maths).
`\!`	Negative space (maths).
`\quad`	Em dash of space.
`\qquad`	Two Em dashes of space.
`\\`	Line break.
`\\[]`	Line break with spacing in the desired units.

Note: Spacing units can be metric or imperial and use the standard abbreviated form. They can be negative and subtract spacing, or positive and add spacing.

[2] They also accept relative or absolute paths with the file.

One of the paramount things about LaTeX is the emphasis on proper spacing. LaTeX does not take Tab indents or single line breaks into account because they help in making code readable and easier to debug, so there are specific spacing commands for that. Table A.6 shows the most commonly used spacing commands and their function.

A.8 Counters

Counters are variables whose values increase whenever an event is triggered. There are standard counters for all sectioning commands, equations (all maths environments except in-line and display maths automatically increase it), floats (figure, table), paragraph, subparagraph, footnote, mpfootnote, and page. They are named after the object they keep track on—for example the figure counter is called `figure`, and the page counter, `page`. There are also counters for the `enumerate` and `itemize` environments. They cover the four default nesting levels, from first to last they are: *1*) `enumi`, *2*) `enumii`, *3*) `enumiii`, and *4*) `enumiv`.

Calling a counter is done with the command `\the<counter_name>`, which prints the counter as it is defined, and can be used to create custom formats with `\renewcommand{}{}` (more on renewing commands later). The command `\value{<counter_name>}` will return the counter's numerical value, which can be used in calculations and setting other counters to some related value.

Default counters are pseudo-numerical—except for footnote symbol counters. Table A.7 contains the default counter styles. Setting a counter's style is done by typing `\<counter_style>{<counter_name>}`, and has to be done within `\renewcommand{}{}` so the changes are global.

Table A.7: Counter styles.

Command	Function
`\arabic{}`	Arabic numbering.
`\alph{}`	Lower case alphabetic.
`\Alph{}`	Upper case alphabetic.
`\roman{}`	Lower case roman numerals.
`\Roman{}`	Upper case roman numerals.
`\fnsymbol`	Footnote symbol.

As an example, say we want our pages to be numbered as follows `<chapter_number>-<arabic_page_number>`, we would need to add

```
\renewcommand{\thepage}{\thechapter-\arabic{page}}
```

where we want this page numbering style to begin.

Counters can also be manually modified. Stepping a counter by one is done with `\stepcounter{<counter_name>}`. Adding or subtracting an arbitrary number to a counter is done with `\addtocounter{<counter_name>}{<number>}`. Setting a counter to a given number is done with `\setcounter{<counter_name>}{<number>}`. Setting a counter in relation to some other counter is done with

```
\setcounter{<counter_name>}{\number\value{<other_counter_name>}}
```

They can also be created with the `\newcounter{<counter_name>}` command. And counters which are reset to zero every time another counter is increased are created with `\newcounter{<counter_name>}[<trigger_counter>]`. New counters must be created in the preamble.

A.9 Paging

Page numbering and formatting are two of the most important, yet most overlooked things in document typesetting. There are a few paging commands and styles but the common ones are found in table A.8. The `\pagenumbering{<style_name>}` changes the page numbering to one of the default styles and resets the counter to its default value of 1.

Table A.8: Common paging commands.

Command	Function
`\pagebreak`	New page with spread out paragraphs.
`\newpage`	New page with normal paragraphs.
`\clearpage`	New page that cannot automatically have floats.
`\cleardoublepage`	Sale as `clearpage` but on two pages.

There are also paging styles which modify a single page's style. The most common default page styles and their function are found in table A.9. They are declared within the `\pagestyle{}` (all pages), or `\thispagestyle{}` (this page only) commands.

Table A.9: Common default page styles.

Style	Function
`empty`	Blank header and footer.
`plain`	Blank header, footer contains a centered page number.
`headings`	Blank footer, header contains page number & relevant info.

There are other pace styles defined in packages such as `fancyhdr`, whose manuals and downloads can be found on the net.

A.10 Table of Contents, Appendix, Index & Glossary

A.10.1 Table of Contents

Some of the most painful aspects of creating documents in word processors are this section's namesake. Fortunately, LaTeXhas us covered. In fact adding a complete tables of contents (toc) is as easy as adding the following code where the user wants them to appear.

```
\newpage
\tableofcontents    % Table of contents.
\cleardoublepage
\phantomsection
% Add table index to the table of contents (toc).
\addcontentsline{toc}{chapter}{List of Tables}
\listoftables       % Table index.
\cleardoublepage
\phantomsection
% Add figure index to the toc.
\addcontentsline{toc}{chapter}{List of Figures}
\listoffigures      % Figure index.
```

The `\tableofcontents` command prints the table of contents, `\listoftables` prints a list of tables, `listoffigures` prints one for figures, and `\addcontentsline{toc}{<section_type>}{<name>}` adds an entry to the table of contents as if it were a section type (part, chapter, section, subsection or subsubsection), under the desired name. The `\cleardoublepage` and `\phantomsection` commands ensure the correct page numbers are referenced. In the given example, the main table of contents prints the location of the lists of tables and figures as if they were chapters named "List of Tables" and "List of Figures".

Anything can be added to the table of contents with the `\addcontentsline{}{}{}` command, including appendices, indices, and unnumbered (starred) sectioning commands. Glossaries can be added with the `glossaries` package.

A.10.2 Appendix

Adding an appendix is as easy as writing `\appendix` where the appendix is to be printed. It only needs to be added once, all subsequent sectioning commands will be formatted accordingly.

A.10.3 Index

Creating an index requires the `makeidx` package, adding the `\makeindex` command in the preamble, and placing `\printindex` where the index is to be printed. Index entries are specified with the `\index{}` command. Table A.10 shows how advanced index entries are specified.

Table A.10: Sophisticate indexing.

Command	Function
`index{a}`	Plain entry.
`index{a!b}`	Sub entry under a.
`index{a}`	Plain entry.
`index{b@\<text_style>{b}}`	Formatted entry.
`index{a!b@\<text_style>{b}}`	Formatted sub entry under a.
`index{a|<text_style>}`	Formatted page number.
`index{a|see {b}}`	Cross-reference with *see*.
`\index{a|seealso{b}}`	Cross-reference with *see also*.

Accents cannot be added as unicode characters, and must instead be added as `\<accent>` followed by the word to be indexed. Advanced index entries can be nested as needed.

There is a way to create more than one index, but for most purposes, a single one will suffice. Regardless, the indexing page of the LaTeX wikibook[3] contains more information on the matter.

A.10.4 Glossary

The glossary requires the `glossaries` package, and adding the `\makeglossaries` in the preamble. Added functionality can be added by specifying the package options: `xindy`, which offers better indexing than `makeindex` (default); and `toc`, which adds the glossary to the table of contents. In order to make glossary entries hyperlinks, the package must be loaded *after* `hyperref`.

[3] https://en.wikibooks.org/wiki/LaTeX/Indexing

Defining a glossary term is done as follows:

```
\newglossaryentry{<label>}
    {
        name = <name>,
        symbol = <symbol>,
        description = {<description>},
        sort = <how to sort the entry>,
        plural = <plural>
    }
```

Where all entries but `name` are optional. If `name` and/or `symbol` require mathematical environments, they should be placed within `\ensuremath{}`. The description can have in-line maths. The `\longnewglossaryentry{}{}{<long_description>}` command should be used in cases where the description is over one paragraph long. It takes the same first and second arguments but the description is placed as the third, not as part of the second. Acronym definition is done with the `\newacronym{<label>}{<abbreviation>}{<full_word>}` command. If acronyms require a different list, add the `acronym` option to the `glossaries` package.

Referencing glossary entries is done with `\gls{<label>}`, by adding symbol, desc and pl between `gls` and the argument, it will reference the symbol, description and plural respectively. By capitalising `G` in the normal and plural versions, the printed name will have its first letter capitalised. An entry can also be referenced with alternate text by using the command `\glslink{<label>}{<alternate_text>}`.

There is much more information in the documentation for the `glossaries` package and the LaTeX wikibook's glossary entry[4].

A.11 Preamble

The preamble is where a document's overall characteristics are defined. Things such as required packages, page setup and layout, user-defined and user-edited macros, as well as titular information are all defined in the preamble.

A.11.1 Declaring a Document Class

The first line of the preamble is where the user declares the document type, default font size, and paper size.

[4] https://en.wikibooks.org/wiki/LaTeX/Glossary

It is well known that only barbarians utilise non-International Standard Paper Sizes (ISO 216 & 269)[5]. According to the all-knowing and less-wrong-than-encyclopaedia-britannica, wikipedia.org [1–3], Mexico is one such barbaric place. One can be an enlightened rebel—and use the A-series sizes—or a barbaric conformist and use non-standard sizes like so:

```
\documentclass[12pt,letterpaper]{article}
```

There are many document classes but the most common ones are: *1*) `article`, *2*) `report`, *3*) `book`, *4*) `letter`, and *5*) `beamer`. These provide the overall document format. In some cases, they also automatically load certain packages and provide custom commands and/or environments. The most obvious example of this is the `beamer` class, which automatically loads the `hyperref` package and provides custom commands and environments such as `\frametitle{}` and `frame`, respectively.

The following are common options of the `\documentclass` command, default values appear in **bold**:

- Equation numbering (left or right): `leqno`, **`reqno`**.
- Document language: `german`, `greek`, `spanish`, **`english`**. Can be overridden with the `babel` and `polyglossia` packages.
- Page orientation: `landscape`, **`portrait`**.
- Text columns: `twocolumn`, **`onecolumn`**.
- Font size: Any font size, **`12pt`**.
- Paper size: Any official ISO and non-ISO paper size as well as a custom paper size override, **`letterpaper`**.

Modern LaTeX editors have wizards that automatically declare document class and load essential packages via graphical user interface (GUI). They let the user decide which common options and packages they want to load without actually having to write all the code and memorise all relevant options.

A.11.2 Package Declaration

After declaring document class, users must declare all relevant packages. This is done as follows:

```
\usepackage[options]{name}
```

[5] https://en.wikipedia.org/wiki/Paper_size

The order in which some packages are loaded may affect their behaviour as dependencies need to be loaded before the packages that use them. A package's load order and dependencies are often specified either in its documentation and/or wikibook entry.

Some of the most used packages include:

- `babel`: sets punctuation (hyphenation), part, chapter, section, subsection, subsubsection, figure, subfigure, table, etc. to the language of your choice. `polyglossia` is the XeLaTeX alternative, but `babel` still has more regional options such as `[spanish,mexico]`.
- `fontspec`: loads utf-8 encoded fonts, natively supports accents—even in `.bib` files—and works on XeLaTeX and LuaTeX. It replaces PDFLaTeX's `inputenc` and `latin` packages; both of which require options in order to use unicode characters, and have trouble with accented unicode characters in `.bib` files.
- `xcolor`: provides tons of colour customisation and declaration, as well as a wide variety of predefined colours. Contains many options which expand the default colour palatte.
- `hypperref`: provides hyperlinks for references, footnotes, citations, and URLs. Contains various options which allow the user to customise their behaviour and appearance.
- `geometry`: provides page layout specifications for customisable margins and paper sizes.

This document contains more packages in `preamble.tex`, read the comments for more information. Further information on packages and LaTeX in general can be found on the web, particularly at the LaTeX wikibook[6], TeX Stack Exchange[7], and Sub-Reddit[8].

A.11.3 Page Format

The standard LaTeX page formatting is usually sufficient for everyday use. However there are times—such as when writing a thesis template and short but comprehensive LaTeX tutorial—when you need special formatting options. Such options follow the structure:

- `\setlength{\lengthcommand}{<len>}`: set a length command to a new value `<len>`.
- `\addtolength{\lengthcommand}{<len>}`: add a value, `<len>`, to a length parameter.

[6] https://en.wikibooks.org/wiki/LaTeX
[7] http://tex.stackexchange.com/
[8] https://www.reddit.com/r/latex

- `\linespread{<len>}`: set inter-line spacing to the value `<len>`.

Added lengths can be negative, so it is possible to subtract lengths. It's also possible to use factors of one length to define another by placing the factor before a specific `\lengthcommand`, e.g. `-0.5\textwidth` divides the text width by -2.

There are `\lengthcommand`s for all manner of things, from margin padding to paragraph skip length—and the need to modify them is niche at best, so most are excluded from this guide. More information on lengths can be found in the lengths entry of the LaTeX wikibook[9]. The most used by far are `\textwidth`—text width, which is often different from the page width due to text padding and multicolumns—and `\width`—page width with margins—because of how useful they are in scaling images.

A.11.4 User-Defined Commands

LaTeX, is in fact *Turing complete*, meaning it has the full capability of a programming language. Therefore, one can define and use programming language staples such as loops, ifthen statements, calculations, variables, etc. LaTeX's collective name for these is *macro*. In reality, only someone who truly hates himself/herself would actually seriously use LaTeX as a programming language. Most people just stop at defining new variables or redefining old ones. These variables are known to LaTeX as commands and environments. Redefining environments is reserved for package makers and *very* advanced users, so they are well outside this guide's scope.

All macros must be defined in the preamble. Commands can be defined as having standalone content; combinations of other standalone commands; as having arguments; and any imaginable combination. They are defined as follows:

```
\newcommand{\cmd1}{content}
\newcommand{\cmd2}[n]{content_1#1 content_2#2... content_n#n}
\newcommand{\cmd3}{\command_1 \command_2... \command_3}
\newcommand{\cmd4}[n]{\command_1{#1} \command_2{#2}... \command_1{#n}}
```

When using a command, its arguments will be specified within braces, {}, of which there will be as many as there are arguments in the command's definition. Arguments are denoted as, `#<n>`, and refer to the command's n^{th} argument left to right. For instance:

```
\newcommand{\helwo12}[2]{#1 #2}
\newcommand{\helwo21}[2]{#2 #1}
```

[9] https://en.wikibooks.org/wiki/LaTeX/Lengths

respectively produce the commands `\helwo12{}{}` and `\helwo21{}{}`. If one were to write `\helwo12{Hello!}{World!}`, the output would be "Hello! World!", but if one were to write `\helwo21{Hello!}{World!}` the output would be "World! Hello!".

Redefining existing commands is somewhat more complex, because it requires the user to know the command's internal parameters. Fortunately, these are found in each command's LaTeX wikibook page. As a general example, redefining existing commands is done in the following manner:

```
\renewcommand{\existingcommand}
    {
        \component_1 value_1
        \component_2 value_2
        ...
        \component_n value_n
    }
```

Commands do not need to be strictly renewed in the preamble, but it's generally advisable to do so. There are times when renewing commands in-document may be necessary—such as special page numbering.

For more information on LaTeX macros visit the wiki book's macro page[10].

A.12 Titular Information

Titular information varies greatly between document classes. There are however, key pieces of information which most documents must have, i.e. title, author and date; academic documents also require author affiliations, and some documents require subtitles. It's important to note that all these are added at the user's behest.[11] This information is added last in the preamble, just before the document proper.

```
\title{Document Title}
\subtitle{Document Subtitle}
\author{auth_1\thanks{affil_1}, auth_2\thanks{affil_2},...}
\date{<date>}
```

The date can be dynamically updated to the OS' by using `\today` as the `\date{}` command's argument. The affiliation is added as a footnote by the `\thanks{}` command.

The preferred way to add author affiliations is by using the `authblk` package:

[10] https://en.wikibooks.org/wiki/LaTeX/Macros

[11] Contrary to word processors, LaTeX will *not* automatically do anything unless the user expressly tells it to.

```
\author[x1]{author_1}
\affil[x1]{affil_1}
\author[x2]{author_2}
\affil[x2]{affil_2}
...
\author[xn]{author_n}
\affil[xn]{affil_n}
```

This package makes keeping track of authors and affiliations much easier because it relies on labels rather than a command's position. It also tidies and simplifies having multiple affiliations per author, and multiple authors per affiliation.

Printing the titular information requires the command `\maketitle` to be inside the document environment.

A.13 Document Environment

The `document` environment is where the document's actual content is found. It is the master environment within which all other environments must be called. Thus the first line after the preamble is *always* `\begin{document}` and the `.tex` file's very last line is *always* `\end{document}`.

A.13.1 Document Sectioning

A standard document can be sectioned into parts, chapters, sections, subsections and subsubsections. However, the user can add new and edit existing sectioning commands to change their format, or even add more arguments. It is quite well known—in *most* reputable moustache-twirling circles—that should you require more sectioning levels than the LaTeX default, you are most likely doing it *wrong*, and will forever be cursed with spaghetti code. Don't let it happen to you, structure your thoughts, documents, and code properly. The standard sectioning commands are declared as follows:

- `\part`: Parts. Primarily used in book writing.
- `\chapter{<chapter_title>}`: Chapters.
- `\section{<section_title>}`: Sections.
- `\subsection{<subsection_title}>`: Subsections.
- `\subsubsection{<subsubsection_title>}`: Subsubsections.

Their starred versions prevents them from being numbered, and stops them from being indexed unless they are manually added to the table of contents (see section A.10).

A.14 Normal Text Environments

There are many environments that respect the text flow and are not specifically designed for mathematical or special notation. This section contains descriptions for some of the most common ones.

A.14.1 List Structures

Both stratified listing environments have four standard nesting levels (`enum<x>` counters). Bullet shapes, enumeration characters, and list levels can be specified and customised in the environment's options, `\begin{environment}[options]`[12]. The `enumitem` package expands this capability.

Itemize

The `itemize` environment lists items as bullet-point lists. Bullet points can be redefined with `\renewcommand{}{}` or with functionality offered by the `enumitem` package. The environment is used as follows:

```
\begin{itemize}
    \item Nesting level 1.
    \begin{itemize}
        \item Nesting level 2.
        \begin{itemize}
            \item Nesting level 3.
            \begin{itemize}
                \item Nesting level 4.
            \end{itemize}
        \end{itemize}
    \end{itemize}
\end{itemize}
```

And outputs:

- Nesting level 1.

[12] https://en.wikibooks.org/wiki/LaTeX/List_Structures

- Nesting level 2.
  - Nesting level 3.
    - Nesting level 4.

Printed indentations are a result of the environment, not code indentation (see section A.10).

Enumerate Environment

The `enumerate` environment places items as numbered lists. The numbering can be redefined with `\renewcommand{}{}` or with functionality offered by the `enumitem` package. It's declared as follows:

```
\begin{enumerate}
    \item Nesting level 1.
    \begin{enumerate}
        \item Nesting level 2.
        \begin{enumerate}
            \item Nesting level 3.
            \begin{enumerate}
                \item Nesting level 4.
                \item Nesting level 4.
            \end{enumerate}
            \item Nesting level 3.
        \end{enumerate}
        \item Nesting level 2.
    \end{enumerate}
    \item Nesting level 1.
\end{enumerate}
```

And outputs:

1. Nesting level 1.
   (a) Nesting level 2.
      i. Nesting level 3.
         A. Nesting level 4.
         B. Nesting level 4.
      ii. Nesting level 3.
   (b) Nesting level 2.
2. Nesting level 1.

Printed indentations are a result of the environment, not code indentation (see section A.10).

In-line Lists

There are a couple of ways to do this, with either the `paralist` or `enumitem` package. For `paralist`, the counter tokens are A, a, I, i and 1; and respectively represent the `\Alph`, `\alph`, `\Roman`, `\roman`, and `\arabic` counter styles. Conversely, `enumitem` uses them explicitly. Both environments can take optional font shape switches and commands, as well as other symbols for item label customisation. In `paraenum` this is done like so:

```
\begin{inparaenum}[\itshape A\upshape)]
    \item Item 1.
    \item Item 2.
\end{inparaenum}
```

in `enumitem` the way to do this is:

```
\begin{enumitem*}[label=\itshape\Alph*\upshape)]
    \item Item 1.
    \item Item 2.
\end{enumitem*}
```

They both produce: *A*) Item 1. *B*) Item 2.

A.14.2 Verbatim Environments

Say you want to make a template, package, o LaTeX guide, so you need to textually print LaTeX commands so people can actually read them. Or you simply want to comment large sections of code. Then the `verbatim` package is essential.

Verbatim and Alltt

The `verbatim` environment *explicitly* prints whatever is written within it (spaces included), regardless of what it is. It's useful for printing LaTeX commands in independent lines. There is also a way to print in-line verbatim, `\verb|content|`, where the first character after `\verb` is the delimiter—which can be any character except for *. The `alltt` environment is somewhat different in that it accepts *some* commands within it[13].

[13] https://en.wikibooks.org/wiki/LaTeX/Paragraph_Formatting

Comment Environment

The `comment` environment is self-explanatory. Anything written within it will be treated as a comment and will therefore not be in the output file.

A.15 Float Environments

Floats are the names LaTeX gives environments which are uncoupled from normal text, i.e. can freely float around the document, namely figures and tables. LaTeX generally does a relatively good job at placing floats where they make sense. Unfortunately, it sometimes does an atrocious one—especially in documents with highly customised layouts—so the user has to intervene. Fortunately, there exist optional placement specifiers which give the user control over float placement. Table A.11[14] shows possible placement specifiers and their meaning.

Table A.11: Table with placement specifiers and their description.

Specifier	Description
h	Place *approximately* where it occurs in the source code.
t	Place at the *top* of the page.
b	Place at the *bottom* of the page.
p	Place in a *dedicated* page.
H	Place *exactly* where it occurs in the source code, needs `float` package.
!	Overrides LaTeXparameters for determining good float positions.
!htbp	Overrides LaTeXparameters and finds the best position out of all placings.

Along with these, there's the command, `\FloatBarrier`, which forces all previous floats to be rendered before it. And the `placeins` package, often used with the `section` option, which forces floats to be placed within the section they are called.

Generally, floats require a caption, which is added by the command `\caption{}`. As a rule, table captions in tables are found at the top, while figure captions at the bottom. There is also a starred version, `\caption*{}`, that eliminates the float number, useful when tables or figures need notes. In order to reference a float, it must be assigned a label, `\label{}`, after the caption. If a float has no caption but needs a label, an empty caption can be added before labeling, `\caption{}\label{label}`[15]. Floats are also generally centered on a page/column, so the `\centering` alignment switch is usually the first line in a float environment. In multicolumn documents, the normal environments may overlap with the text and each other. Starred versions,

[14] Edited from https://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions

[15] Do *not* use starred captions when labeling, otherwise there will be nothing—but emptiness and deep internal sadness—for the label to reference.

`{figure*}` and `{table*}` avoid this issue by reserving the whole page width (the value of `\width`) for the float. Unfortunately, not all placement options work with starred floats.

A.15.1 Figures

Documents often require the user to add images as figures or standalone inputs. As with many things, it is common knowledge that an angel looses its wings every time someone inserts a `.jpg` into a LaTeXdocument. So for the love of ceiling cat  , use vectorised images such as `.eps`, `.ps`, and `.pdf`. The only exceptions are scatter plots and heat maps, where it's advisable to use `.bmp` or `.png` instead—as they are more accurate in such cases.

The `figure` environment is generally used in the following way:

```
\begin{figure}[<placement_specifier>]
    \centering
    \includegraphics[options]{imagename.extension}
    \caption{caption}
    \label{f:label}
\end{figure}
```

`\includegraphics[]{}` can be used on its own to include in-line images. Its options and their descriptions are found in table A.12[16].

Table A.12: Options and their descriptions for the `\includegraphics` command.

Option	Description
`width=x`	Image width x.
`height=y`	Image height y.
`keepaspectratio`	`true`, `false`. Keep aspect ratio with max width x and height y.
`scale=x`	Scale by factor x.
`angle=x`	Rotate by $x°$ (standard rotation).
`trim=l b r t`	Trim left l, bott b, right r, top t (broken in XeLaTeX).
`clip`	Has to be `true` for trim to work.
`resolution=x`	Image resolution in x dpi.

[16]Edited from https://en.wikibooks.org/wiki/LaTeX/Importing_Graphics.

Subfigures

The `subfigure` environment can only be declared within the `figure` environment, and requires the `subcaption` package. Its usage is as follows:

```
\begin{figure}
    \centering
    \begin{subfigure}[t]{0.45\textwidth} % t = align subfigure on top.
    \includegraphics[width=\textwidth]{dank.jpg}
        \caption{A dank meme.}
        \label{f:dankmeme}
    \end{subfigure}
  ~ % add desired spacing between images, e. g. ~, \quad, \qquad, \hfill etc.
     % (or a blank line to move the subfigure to a next line.)
   \begin{subfigure}[b]{0.45\textwidth} % align subfigure at the bottom.
        \includegraphics[width=\textwidth]{ceiling-cat.jpg}
        \caption{Ceiling cat.}
        \label{f:ceilingcat}
    \end{subfigure}

   \begin{subfigure}[c]{0.45\textwidth} % b = align subfigure on its centre.
        \includegraphics[width=\textwidth]{watoge.jpg}
        \caption{Watoge}
        \label{f:watoge}
    \end{subfigure}
    ~
   \begin{subfigure}[c]{0.45\textwidth} % b = align subfigure on its centre.
        \includegraphics[width=\textwidth]{MLG.png}
        \caption{MLG.}
        \label{f:mlg}
    \end{subfigure}
    \caption{Pictures of memes.}\label{f:memes}
\end{figure}
```

Which produces the output found in figs. A.1, A.1a, A.1b, A.1c and A.1d:

(a) A dank meme.

(b) Ceiling cat.

(c) Watoge

(d) MLG.

Figure A.1: Pictures of memes.

A.15.2 Tables

Tables in LaTeX can either be the source of great pride or great pain. Done right, they look classy and slick…done wrong, they are tacky and I hate them[17]. Before designing a table, you must answer me these questions three[18]:

1. Readability: is it easily understandable?
2. Minimalism: does it contain the minimum required information—does it minimise redundancy?
3. Simplicity: how many borders does it have—does it really need all those vertical and/or horizontal borders?

The package `booktabs` provides improved versions of the horizontal line commands, `\hline` and `\cline{}`.[19] They are: `\toprule`, `\midrule`, `\cmidrule{}`, and `\bottomrule`. They all provide additional spacing to make tables more readable, and accept options to modify their trimming and thickness. A table is declared in the following way:

```
\begin{table}[<placement_specifier>]
    \centering
    \caption{caption}
    \label{t:label}
    \begin{tabular}{<n columns (with alignment), vertical borders>}
        \toprule
        heading 1 & ... & heading n
        \midrule
        a_11 & ... & a_1n \\
        .
                .
                        .
        a_m1 & ... & a_mn \\
        \bottomrule
    \end{tabular}
\end{table}
```

In line with the three principles of table design, one must avoid borders as much as possible without sacrificing readability. For most purposes, tables only need horizontal lines at the top, bottom, and for separating heading from the body. There is also

[17] Stickin' it to 'The Man'.

[18] You seek the holy grail.

[19] If there's a need for non-tabular horizontal lines—such as redefining footnote rules—you should still use `\hline`.

`\cmidrule{i-j}` which spans columns i to j. The `array` package extends table functionality by providing extra arguments for table customisation. The columns are represented by `l`, `c` and `r` for left, center and right alignment, they can be placed immediately after one another or separated by spaces. Vertical borders are denoted by `|` between columns. The actual columns are then separated by ampersands `&`. If the number of columns is n, the number of ampersands is $n - 1$.

To showcase what we have learned of tables up to now, table A.13 is an example of a very poorly formatted table. Its generating code is:

```
\begin{table}
    \centering
    \caption{Poorly designed and formatted table.}
    \label{t:example}
    \begin{tabular}{|cl|||r||}
        \cline{2-3}
        Measurement & Magnitude & Units \\
        \hline
        Speed & 15 & m/s \\
        Time & $\infty$ & s \\
        \hline
        \hline
    \end{tabular}
\end{table}
```

Table A.13: Poorly designed and formatted table.

Measurement	Magnitude	Units
Speed	15	m/s
Time	∞	s

The documentation for `booktabs` contains further information on table design, formatting, and package features.

Multicolumn and Multirow Tables

Sometimes tables have fields which span more than one row or column (sometimes both). When this is unavoidable, the `multirow` package provides the commands:

```
\verb|\multirow{num_rows}{width}{contents}|
\verb|\multicolumn{num_columns}{column_alignment and borders}{contents}|
```

An asterisk "*" as width in `\multirow` stands for the contents' natural width. `\multicolumn` gobbles all relevant columns, so there's no need to add the gobbled

ampersands. If `\multirow` is used, all subsequent rows in the same column(s) must be left empty and take up the same number of columns. Table A.14 was sourced directly from the wiki entry for tables[20] because it's an excellent non-trivial minimal working example:

```
\begin{table}
    \centering
    \caption{Multirow and multicolumn table.}
    \label{t:multirowcol}
    \begin{tabular}{cc|c|c|c|c|l}
        \cline{3-6}
        & & \multicolumn{4}{ c| }{Primes} \\ \cline{3-6}
        & & 2 & 3 & 5 & 7 \\ \cline{1-6}
        \multicolumn{1}{ |c  }{\multirow{2}{*}{Powers} } &
       \multicolumn{1}{ |c| }{504} & 3 & 2 & 0 & 1 &     \\ \cline{2-6}
        \multicolumn{1}{ |c  }{}                          &
       \multicolumn{1}{ |c| }{540} & 2 & 3 & 1 & 0 &     \\ \cline{1-6}
        \multicolumn{1}{ |c  }{\multirow{2}{*}{Powers} } &
       \multicolumn{1}{ |c| }{gcd} & 2 & 2 & 0 & 0 & min \\ \cline{2-6}
        \multicolumn{1}{ |c  }{}                          &
       \multicolumn{1}{ |c| }{lcm} & 3 & 3 & 1 & 1 & max \\ \cline{1-6}
    \end{tabular}
\end{table}
```

Table A.14: Multirow and multicolumn table.

		Primes				
		2	3	5	7	
Powers	504	3	2	0	1	
	540	2	3	1	0	
Powers	gcd	2	2	0	0	min
	lcm	3	3	1	1	max

Multipage Tables

Sometimes a project requires tables which span more than a single page. For cases such as these, the `longtable` package provides the `longtable` environment, which offers customisation commands relevant to multipage tables[21]. `longtable` does not need the `tabular` environment, it is declared as follows:

```
\begin{longtable}{<n columns (with alignment), vertical borders>}
```

[20] https://en.wikibooks.org/wiki/LaTeX/Tables
[21] http://mirrors.rit.edu/CTAN/macros/latex/required/tools/longtable.pdf

```
\caption{caption}
    \label{t:longtable}
    \toprule
    heading 1 & ... & heading n
    \midrule
    a_11 & ... & a_1n \\
    .
            .
                .
    a_m1 & ... & a_mn \\
    \bottomrule
\end{longtable}
```

`longtable` also works with the functionality offered by the `multirow` package. However, multiple compilations (4+) may be required to get long tables with multirows and/or multicolumns looking as they should.

A.16 Mathematical Environments

The primary reason why Don Knuth started developing TEX was to create an easy and powerful tool for typesetting academic mathematical documents. There are absurdly many mathematical tools and symbols in LATEX, but this document will only touch the most common ones non-mathematicians use. The packages `mathtools`, `bm`, `amssymb`, and `pxfonts` respectively provide mathematical packages and symbols, corrected bold maths fonts, other mathematical symbols, and more fonts and symbols. Mathematical symbols, functions and typefaces cannot exist outside a mathematical environment, though some have non-maths—or text—versions, usually `text<symbol_name>` or `\text<command_name>{}`.

A.16.1 Maths and Display Maths

The two most basic mathematical environments are the inline and display maths environments. The inline maths environment is delimited by single dollar signs `$ inline maths $`, which yields *inlinemaths*. Inline maths has its limitations because it is text-bound, which can cause problems when using mathematical notation that extends beyond the bounds of a normal line of text. For example $\frac{\sum_{i=0}^{n} i}{\sum_{i=0}^{n} 2i} = \frac{1}{2}$ generated with

```
\frac{\sum\limits_{i=0}^{n} i}{\sum\limits_{i=0}^{n} 2i} = \frac{1}{2}
```

Which pushes line spacing beyond acceptable levels. In cases such as these, when equation numbering is not an issue, one can use the display maths environment delimited by double dollar signs, `$$ display maths $$`, or the preferred method (better spacing) of using backslash followed by chevrons, `\[ display maths \]`. Display maths uncouples the notation from the text and places it *exactly* where the command is found the point it's called from. It also lets large symbols scale accordingly. Using the previous equation as an example,

$$\frac{\sum_{i=0}^{n} i}{\sum_{i=0}^{n} 2i} = \frac{1}{2},$$

it's easy to see the need for both. It's also important to note that all mathematical environments gobble spaces. Use `~`, `\,`, or any other spacing command to add spaces. If you need text to be printed within a mathematical environments use the `\text{}` command and make sure to explicitly add spaces at the beginning and end of the normal text to ensure proper spacing. For example: `$x\textrm{apples}+y\text{pears}$`, $x\text{apples} + y\text{pears}$ vs. `$x\textrm{ apples }+y\text{ pears}$`, $x\textrm{ apples } + y\text{ pears}$.

A.16.2 Common Symbols and Functions

There are a myriad of symbols and functions for virtually *all* areas of maths. This document touches on the ones non-mathematicians will generally use. If anyone is curious about more symbols make sure to check out Scott Pakin's 164-page list of LaTeX symbols [4].

Greek Maths Symbols

All maths greek letters are backslash commands of the letter's name. There are capitalised versions which are written the same, but start with a capital letter. Some letters also have variations which are denoted as `var<letter_name>`. Table A.15 contains an example of this.

Table A.15: Example of maths greek letters.

ϕ	φ	Φ
`\phi`	`\varphi`	`\Phi`

There are also bold versions of some symbols, which is specified by the `\boldsymbol{}` command. When a bold symbol does not exist there's the `\pmb{}` (poor man's bold) which prints slightly offset symbols to give the appearance of bold typeface.

Common Binary Operations

Table A.16: Common binary operations.

$+$	$-$	$\pm$	$\mp$	$\times$	$\div$	$\cdot$
`+`	`-`	`\pm`	`\mp`	`\times`	`\div`	`\cdot`

Common Relation Symbols

Table A.17: Common relation symbols.

$=$	$\neq$	$\equiv$	$\simeq$	$\approx$	$\sim$	$\cong$	$<$	$>$
`=`	`\neq`	`\equiv`	`\simeq`	`\approx`	`\sim`	`\cong`	`<`	`>`
$\leq$	$\geq$	$\ll$	$\gg$	$\in$	$\ni$	$\propto$	$\perp$	$\parallel$
`\leq`	`\geq`	`\ll`	`\gg`	`\in`	`\ni`	`\propto`	`\perp`	`\paralel`

Arrows

Arrows follow a few rules, capitalising the first letter makes the arrow have a two line shaft. The direction is specified first, if the arrow has two heads, left goes before right and up before down. There are two standard types, arrows (always stated as singular) and harpoons (singular except left right harpoons). Harpoons can only be horizontal and the orientation of the spike is specified last. Table A.18 contains some examples. Arrows can also be specified as long, which goes first in the command.

Table A.18: Arrows.

$\Leftrightarrow$	$\updownarrow$	$\leftrightharpoons$
`\Leftrightarrow`	`\updownarrow`	`\leftrightharpoons`
$\leftharpoonup$	$\Longleftrightarrow$	$\longrightarrow$
`\leftharpoonup`	`\Longleftrightarrow`	`\longrightarrow`

Common Miscellaneous Symbols

Table A.19: Common miscellaneous symbols.

$\ldots$	$\cdots$	$\vdots$	$\ddots$	∞	$\hbar$	$\emptyset$
`\ldots`	`\cdots`	`\vdots`	`\ddots`	`\infty`	`\hbar`	`\emptyset`
$\exists$	∇	$\neg$	$\Re$	$\Im$	$\angle$	∂
`\exists`	`\nabla`	`\neg`	`\Re`	`\Im`	`\angle`	`\partial`
			$\therefore$			
			`\therefore`			

Common Functions

LaTeX has commands for all basic functions. They are backslash commands defined as the function's shortened name without abuse of notation, e.g. inverse trigonometric functions are written as `\arc<function>`. In fact, most functions are defined exactly how they are written in rigorous pen & paper maths.

Common Variable-Sized Symbols

Some symbols can change size depending on where they are in-document. Sometimes however, they're not big enough for their argument. This is solved by the nestable `\mathlarger{}` command, provided by the `relsize` package. The number of integrals is defined by the number of i's in the command—up to 4 for normal integrals, 3 for cyclic ones[22]. Linear integrals can also have dots.

Table A.20: Common variable-sized symbols.

$\sum$	$\prod$	$\int$	$\oint$	$\int\cdots\int$	$\oiiint$	$\frac{a}{b}$	$\dfrac{a}{b}$
`\sum`	`\prod`	`\int`	`\oint`	`\idotsint`	`\oiiint`	`\frac{a}{b}`	`\dfrac{a}{b}`

Non-trivial Delimiters

Table A.21: Non-trivial delimiters.

$\|$	$\vert$	$\langle$	$\rangle$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$
`\\|`	`\vert`	`\langle`	`\rangle`	`\lfloor`	`\rfloor`	`\lceil`	`\rceil`

Common Mathematical Accents

There are many mathematical accents, however we will only mention the most common ones used in science. The majority of the most common ones are provided by the $\mathcal{AMS}$-TeX packages. The `mathdots` package redefines the spacing between dots for three and four dots. The number of dots is defined y the number of d's in the `\dots` command. There also exist dotless versions of $\imath$ and $\jmath$, provided by `\imath` and `\jmath` respectively—useful for properly adding hats to orthogonal unit vectors.

Some accents also have wide versions. Some bar accents (including arrows) are denoted as `\over<bar_accent>`, while other wide accents are denoted as `\wide<accent>`. Table A.22 contains some common mathematical accents and their respective commands.

[22] Triple cyclic integrals need the `pxfonts` or `txfonts` package

Table A.22: Common mathematical accents.

$\hat{a}$ `\hat{a}` $\check{a}$ `\check{a}` $\bar{a}$ `\bar{a}` $\vec{a}$ `\vec{a}` $\dot{a}$ `\dot{a}` $\ddot{a}$ `\ddot{a}` $\tilde{a}$ `\tilde{a}`

$\mathring{a}$ `\mathring{a}` $\widehat{abc}$ `\widehatabc` $\sqrt{abc}$ `\sqrt{abc}` $\sqrt[n]{abc}$ `\sqrt[n]abc`

$\underline{abc}$ `\underline{abc}` $\overbrace{\underbrace{abc}}$ `\overbrace{\underbrace{abc}}`

$\overleftarrow{\underleftrightarrow{abc}}$ `\overleftarrow{\underleftrightarrow{abc}}`

Furthermore, `\overbrace{}` and `\underbrace{}` may have sub/superscripts automatically placed directly below and above them without resorting to `\limits_{}^{}`. For example:

```
z = \overbrace{\underbrace{x}_{\text{Real}} +
  i \underbrace{y}_{\text{Imaginary}}}^{\text{Complex number.}}
```

$$z = \overbrace{\underbrace{x}_{\text{Real}} + i \underbrace{y}_{\text{Imaginary}}}^{\text{Complex number.}}$$

LaTeX will let the sub and superscripts take spacing priority, if this is a problem, placing the whole sub/superscript inside the `\mathclap{}` command will eliminate the extra spacing.

LaTeX Limits (not analysis)

Some symbols may have arguments and/or sub/supersripts in different places. Placing the `\limits` command immediately after a symbol, moves the immediately following sub/superscript below and above it, respectively—having both at the same time is not a requirement. Table A.23 shows this.

Table A.23: Effect of the `\limits` command.

No `\limits`	`\limits`	Command
$\sum_i$	$\sum\limits_i$	`$\sum\limits_{i}$`
$\sum^j$	$\sum\limits^j$	`$\sum\limits^{j}$`
$\sum_i^j$	$\sum\limits_i^j$	`$\sum\limits_{i}^{j}$`

Scaled Delimiters

It's very common to have large equations which require large delimiters, these are obtained by adding `\left`, `\right` and `\middle` (optional) before the actual delimiter, they also scale the delimiter according to the size of the equation. It's important to remember that braces still need a backslash for LaTeX to print them, `\left\{ \right\}`. Sometimes the delimiter is unilateral (only one is printed), but their scaled versions must be closed (i.e. have both left and right components), in such cases a single full stop, `.` becomes the closing delimiter. The following equations are good examples of this.

```
\begin{align}
    &\exp(-\sum\limits_{i}^{N}\frac{E_{i}}{k_{b}T}) \\
    &\exp\left(-\sum\limits_{i}^{N}\frac{E_{i}}{k_{b}T}\right) \\
   &\int\limits_{a}^{b} \exp\left(-\frac{E}{k_{b}T}\right)\,\mathrm{d}E
    = -k_{b}T\exp\left(-\frac{E}{k_{b}T}\right)|_{a}^{b} \\
   &\int\limits_{a}^{b} \exp\left(-\frac{E}{k_{b}T}\right)\,\mathrm{d}E
    = \left.-k_{b}T\exp\left(-\frac{E}{k_{b}T}\right)\right|_{a}^{b}
\end{align}
```

$$\exp(-\sum_{i}^{N}\frac{E_i}{k_bT}) \tag{A.1}$$

$$\exp\left(-\sum_{i}^{N}\frac{E_i}{k_bT}\right) \tag{A.2}$$

$$\int_{a}^{b} \exp\left(-\frac{E}{k_bT}\right)\,\mathrm{d}E = -k_bT\exp\left(-\frac{E}{k_bT}\right)|_{a}^{b} \tag{A.3}$$

$$\int_{a}^{b} \exp\left(-\frac{E}{k_bT}\right)\,\mathrm{d}E = -k_bT\exp\left(-\frac{E}{k_bT}\right)\Big|_{a}^{b} \tag{A.4}$$

Common Maths Typefaces

Oftentimes, different mathematical objects have specific notations. For most applications no packages are needed, but loading both `bm` and `mathtools` packages cover most other cases. Table A.24 provides some examples.

Table A.24: Common mathematical typefaces.

Command	Result
`\mathtt{ABCdef123}`	ABCdef123
`\mathbb{ABCdef123}`	$\mathbb{ABCdef123}$
`\mathbf{ABCdef123}`	**ABCdef123**
`\mathcal{ABCdef123}`	$\mathcal{ABCdef123}$
`\mathsf{ABCdef123}`	ABCdef123
`\mathit{ABCdef123}`	*ABCdef123*
`\mathrm{ABCdef123}`	ABCdef123
`\mathfrak{ABCdef123}`	$\mathfrak{ABCdef123}$
`\bm{ABCdef123}`	$\boldsymbol{ABCdef123}$

A.16.3 Align

The `align` environment is the one you should be using for all your equation typesetting needs. It does everything the `equation` and `eqnarray` packages do, but better. The starred variant removes all equation numbers. Its behaviour is exactly the same as `tabular`, but does not require the user to declare the number of columns or alignment within them. It also automatically defines all the lines it encloses as a mathematical environment, removing the need to add dollar signs. Indiviual equations can be labeled by placing `\label{<label>}` before a line break, and referenced with `\eqref{}`. Individual equation numbers can be removed with the `\nonumber` command before a line break. The following equations are aligned on the equal sign:

```
\begin{align}
    F(x) &= \int f(x)\,\mathrm{d}x + \mathrm{C}\\
    f(x) &= \frac{\mathrm{d}F(x)}{\mathrm{d}x} \nonumber\\
    g(x,y,z) &= 0
\end{align}
```

$$F(x) = \int f(x)\,\mathrm{d}x + \mathrm{C} \tag{A.5}$$

$$f(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}$$

$$g(x, y, z) = 0 \tag{A.6}$$

Subequations

Subequations are declared by the environment of the same name, which wraps around other equation environments such as, `align`, `eqnarray` and `equation`. They have

same behaviour as normal equations, including labeling and referencing. Their numbering can also be customised via `\renewcommand{}{}`. Using the aforementioned example within the `subequations` environment yields:

```
\begin{subequations}
    \begin{align}
        F(x) &= \int f(x)\,\mathrm{d}x + \mathrm{C} \label{e:int}\\
        f(x) &= \frac{\mathrm{d}F(x)}{\mathrm{d}x} \nonumber\\
        g(x,y,z) &= 0
    \end{align}
\end{subequations}
```

$$F(x) = \int f(x)\,\mathrm{d}x + \mathrm{C} \tag{A.7a}$$

$$f(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}$$

$$g(x, y, z) = 0 \tag{A.7b}$$

A.16.4 Matrices

Matrices can be represented in many ways depending on the context, or to denote certain matrix operations. Pre-defined matrices require the `mathtools` package. The default types are shown in table A.25, which is edited from the mathematics entry in the LaTeX wikibook[23]. Non-starred versions center the columns by default, starred variants allow user-defined alignments. The matrix environment must be declared within a maths environment. Matrices can also be created using the `array` environment, which is essentially the maths-mode version of the `tabular` environment.

Table A.25: Matrix types.

Command	Delimiter
`pmatrix`	( )
`bmatrix`	[ ]
`Bmatrix`	{ }
`vmatrix`	| |
`Vmatrix`	‖ ‖

Aligned and Array

There are times when annotating equations is useful for the reader, or desired by the author. In cases such as these, the `aligned` provides a quick solution, though it

[23] https://en.wikibooks.org/wiki/LaTeX/Mathematics

only provides a single ampersand for aligning items. In cases where more than one ampersand is needed use `array` instead, which is the maths equivalent of `tabular`.[24]

```
\begin{align}
    \left.
        \begin{aligned}
            f(x,y,z) &= \alpha x + \beta y + \gamma z + \delta \\
            f(x,y) &= \alpha x + \beta y + \gamma \\
            f(x) &= \alpha x + \beta
        \end{aligned}
    \right\}\text{Linear equations.}
\end{align}
```

$$\left.\begin{aligned} f(x,y,z) &= \alpha x + \beta y + \gamma z + \delta \\ f(x,y) &= \alpha x + \beta y + \gamma \\ f(x) &= \alpha x + \beta \end{aligned}\right\} \text{Linear equations.} \tag{A.8}$$

A.16.5 Cases

The `cases` environment allows the user to define piecewise functions without the pain of recurring to `\aligned` and delimiter scaling. Similarly to the `eqnarray` and `aligned` environments, it provides a single ampersand for aligning. Take eq. (A.9) for example. There also exists `dcases`, which provides a display version of the environment—it properly scales symbols—as shown in eqs. (A.10) and (A.11). It also has a starred version shown in Eq. (A.12), which sets everything to the right of the ampersand to normal text—therefore all maths in this region must be within single dollar signs.

```
\begin{align}
    |f(x) + b| + c &=  \begin{cases}
                           -f(x) - b + c & \textrm{if } x < 0 \\
                           f(x) + b + c & \textrm{otherwise}
                       \end{cases} \label{e:abs}\\
    g(x) &=  \begin{cases}
                 \int f(x)\,\mathrm{d}x & \textrm{if } x < 0\\
                 0 & \textrm{otherwise}
             \end{cases} \label{e:case} \\
    g(x) &=  \begin{dcases}
                 \int f(x)\,\mathrm{d}x & \textrm{if } x < 0\\
                 0 & \textrm{otherwise}
```

[24] They are used and declared in exactly the same way; only `tabular` is used within normal text, and `array` within maths environments.

```
            \end{dcases} \label{e:dcases} \\
    g(x) &= \begin{dcases*}
                \int f(x)\,\mathrm{d}x & if $x < 0$\\
                0 & otherwise
            \end{dcases*} \label{e:dcases*}
\end{align}
```

$$|f(x) + b| + c = \begin{cases} -f(x) - b + c & \text{if } x < 0 \\ f(x) + b + c & \text{otherwise} \end{cases} \tag{A.9}$$

$$g(x) = \begin{cases} \int f(x)\,\mathrm{d}x & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.10}$$

$$g(x) = \begin{cases} \displaystyle\int f(x)\,\mathrm{d}x & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.11}$$

$$g(x) = \begin{cases} \displaystyle\int f(x)\,\mathrm{d}x & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.12}$$

A.17 Referencing

Referencing in LaTeX is extremely easy to do, all it requires is a label—`\label{}`—and reference—`ref{}` for non-equations, and `\eqref{}` for equations—commands. Referencing floats also requires the label to come after a non-starred caption.

As long as an item can be *uniquely* identified, it can be labeled and referenced. Labels are placed immediately following sectioning, ToC, appendix, and indexing commands; before line breaks in equations; and after the caption in floats. The `hyperref` package loads further referencing capability, such as hyperlinks and colour-coded links depending on hyperlink type.

The package `cleveref` does not work on Overleaf, but adds incredibly useful functionality such as smart references, reference formats, reference ranges, compressed references, capitalisation, etc. It doesn't work when `amsmath` and `hyperref` are simultaneously loaded. Fortunately `mathtools` is vastly superior to `amsmath` in every respect, and does *not* present this bug; so use it in place of `amsmath` and enjoy the wonders of `cleveref`.

A.18 Non-English Documents

LaTeX is also capable of creating non-English documents. In fact, language packs are freely available in community packages and extensions. The `babel` and

`polyglossia`[25] packages provide non-english language capability. They change sectioning and float languages, hyphenation patterns, quotations, and preference for decimal dot vs. comma. There are also regional options for many languages: such as iberic and mexican spanish; iberic or brazilian portugues, etc. Unfortunately `polyglossia` still lacks some regional adjustments, including latin-american and mexican spanish, which means many users should stick to `babel` for the time being. Babel languages are declared as follows:

```
\usepackage[language1,language2...,languagen]{babel}
```

Where the last language will be the default one. Regional adjustment are placed immediately after the main language, for instance:

```
\usepackage[spanish,mexico]{babel}
```

It is also possible to create multilingual documents, so linguists and language academics are well catered for. More information on this can be found in `babel`'s user manual.

A.19 Bibliography

One of the appeals of LaTeX is bibliography management. There are two ways of doing so:

- Manually: better known as the hard way.
- BibTeX: better known as the only way. Google scholar also provides `.bib` files by clicking Cite under an article and selecting BibTeX at the bottom left of the popup window.

They can be referenced using the `\cite[]{}`, where the optional chevrons can be omitted, and the bibliographic label is placed inside the braces.

A.19.1 Manual

Doing things manually means the user has to hard-code the style into the bibliography. The user adds the `thebibliography` environment, which takes a numeric argument for the maximum number of bibliographic items:

[25] Preferred for XeLaTeX.

```
\begin{thebibliography}{99}
    \bibitem{<biblabel1_>} ...
    \bibitem{<biblabel2_>} ...
              ...
    \bibitem{<biblabel_n>}
\end{thebibliography}
```

A.19.2 BibTEX

The preferred way of handling bibliographies is by using the `natbib` package together with BibTEX. `natbib` offers further citing options and commands which are too numerous and readily available[26] to be placed here. Though I'd like to mention `\nocite{}`, because it adds the specified entry to the bibliography, but doesn't print the reference in-text. If the argument is an asterisk, *, all entries in the `.bib` file are added to the bibliography regardless of whether they were cited or not.

Using BibTEX requires a separate `.bib` file, where entries are defined according to specific formats depending on the type of document or file to be cited, the first string of which is the bibliographic label. The following commands are then placed after all the document's content and before `\end{document}`:

```
\bibliographystyle{<style>}
\bibliography{<filename>}
```

where the `<style>` is replaced by a bibliographic style, and `<filename>` is the `.bib` file's name without extension (also accepts relative and absolute paths). There are many bibliographic styles found in `natbib`'s manual or its wikibook entry[27]. They are not added here because their nuances are too subtle and involved for the purposes of this document. Other packages—especially journal-specific packages—add specialist styles.

In order to fully compile a file with a BibTEX bibliography, the following compilation train must be followed:

1. XeLaTeX(or engine of your choice).
2. BibTEX.
3. XeLaTeX(or engine of your choice).
4. XeLaTeX(or engine of your choice).

[26] https://en.wikibooks.org/wiki/LaTeX/More_Bibliographies
[27] https://en.wikibooks.org/wiki/LaTeX/Bibliography_Management#Natbib

This will ensure all references are parsed and the bibliography will be updated to show all the right entries with the correct style. Some editors can automate this process in a single shortcut.

A.20 Closing Remarks

Well, it's been quite a journey, hasn't it? I originally intended this to be ~ 10 pages long, but it seems I gravely underestimated. It contains all the stuff I've learned in my 2.5 years of using LaTeX—except what I know of `beamer` (that may come later) and `minted` (doesn't work on Overleaf). I have tried to include all the tips and techniques that have cost me so much time and late nights to learn—all in an effort to prevent it from happening to you. I hope my objectives are met when you read this document.

Finally, I'd like to ask something of you. The world is built one little brick at a time. But the process is often marred by greed and selfishness. Lets not be like that. I did this out of the kindness of my heart, expecting nothing in return but a promise you will make only to yourselves: that you will do your best to spontaneously help others whenever you can.

With that, I give you my thanks and wish you all the best.

Bibliography

[1] J. Giles, "Internet encyclopaedias go head to head," *Nature*, vol. 438, no. 7070, pp. 900–901, 2005.

[2] Encyclopædia Britannica Inc, "Fatally flawed," http://corporate.britannica.com/britannica_nature_response.pdf, March 2006.

[3] Nature, "Encyclopaedia britannica and nature: a response," http://www.nature.com/press_releases/Britannica_response.pdf, March 2006.

[4] S. Pakin, "The comprehensive latex symbol list," http://ctan.math.utah.edu/ctan/tex-archive/info/symbols/comprehensive/symbols-a4.pdf, November 2009.

[5] L. Communitu, "Latex wikibook," https://en.wikibooks.org/wiki/LaTeX.