

Example Project

How to write in Latex

A helpful guide to get started and to show some common use cases

Max Mustermann 1234567

Mira Musterfrau 9876543

~~01.01.2020~~

March 31, 2022

Professor: your Professor

Declaration of Authorship

We hereby certify that the work we are submitting is entirely of our own making except where otherwise indicated. We are aware of regulations concerning plagiarism, including disciplinary actions that may result from it. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Max Mustermann

Mira Musterfrau

Abstract

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Keywords: some, informative, keywords

Contents

Abstract	III
1 What is LaTeX	1
1.1 Following this document	1
1.2 Requirements to use LaTeX	1
1.3 Running LaTeX	2
1.4 LaTeX commands	2
1.5 Getting more information	2
2 Basic text formatting	3
2.1 Headings	3
2.2 Text paragraphs	3
2.3 Text spacing	4
2.4 Breaking pages	4
2.5 Text styling	4
2.6 Special characters	5
2.6.1 LaTeX command characters	5
2.6.2 Invisible characters	5
2.7 Cross-referencing	5
3 Examples	6
3.1 Using formulas	6
3.2 using Units	6
3.3 using images	7
3.4 using tables	9
3.5 lists and enumerations	9
3.6 CSV files	10
3.7 formating code	10
4 seperating the document	11
5 attachment	12
Messprotokoll	12
List of Figures	13
List of Tables	13

1 What is LaTeX

So you decided to get started with LaTeX. Great! So let's talk a bit about the basic concept and differences it comes with.

Up to this point you probably used a Word Processor like MS Word. The kind of workflow you know from there is often referred to as *What you see is what you get*. You see the exact final layout as you type it, press some colourful buttons to insert stuff and if it doesn't want to do something you need, you're screwed.

LaTeX on the other hand falls into the category of *What you see is what you mean*, which describes all forms of markup languages. This means you create your LaTeX document as a simple plain text file without any form of formatting and mix in a bunch of commands telling what you mean. For example: "This is supposed to be a chapter heading", "make this bold" or "insert an image here". This source file will then be passed to a document processor (the LaTeX program), which will, depending on its settings, create the document for you. The advantage is, that you can use the same markup with all sorts of formattings and target file types.

This is why working with LaTeX will require some getting used to and you will find yourself wanting to compile every five seconds to see the document update. Try to restrain yourself and concentrate on writing. You will find yourself working much faster.

1.1 Following this document

To see how the LaTeX source code and the resulting PDF correspond, I recommend you open this document's source code and PDF file next to each other and scroll through them simultaneously. If you already have a working LaTeX setup, most editors support *SyncTex*, which allows you to jump between source code and PDF file and vice versa. You have to compile yourself, which will create a file called `example.synctex.gz` in your project directory. Now you can `<CTRL>+Click` in the PDF and the corresponding line of source code will be highlighted.

The shortcut to jump from the source code into the PDF will depend on your Editor, but for VS Code it's right `Click→SyncTex` from cursor or `CTRL+ALT+J`.

1.2 Requirements to use LaTeX

As LaTeX files are just plain text files, you can edit them with any text editor (even Windows Notepad works, but that's just terrible). However, I would strongly recommend a more suitable editor. I use Visual Studio Code (which is a multi-purpose text editor that supports all major programming languages) but you could also use something like Texmaker, which is an editor specifically for LaTeX. There is also the online editor Overleaf, which saves you the trouble of setting up your own LaTeX installation and provides everything you need in the cloud.

As I have already mention above, you also need the LaTeX program. It comes bundled with packages and other additional software inside a Tex-distribution. There are two major ones, Texlive and MiKTeX. I recommend MiKTeX, but it essentially doesn't matter which one you choose.

Once you have the distribution installed, test it by running `pdflatex --version` in any terminal windows and it should return you some information about the installed version and setup.

1.3 Running LaTeX

To create a PDF file from your LaTeX source code, you can always navigate to the project folder in a terminal window and run `pdflatex filename.tex`. However, if you have a decent editor installed, it will provide you with a button and do this for you.

With these project files you also received a makefile, which demonstrates how to compile this example file successfully from the terminal. The README file also has some tips and information for you.

If you use VS Code, this project also contains settings for LaTeX and recommended extensions. If you open the folder for the first time you will be asked if you want to install them and should than be able to compile this file.

1.4 LaTeX commands

Now lets look at the LaTeX command. Every one will begin with a `\` followed by a letters only command name, like this: `\command`. Most commands also accept input, which is put after it into curly brackets: `\command{argument}`. They can accept multiple arguments either in multiple sets of curly brackets or as a comma separated list, depending on the command.

Some commands also accept optional arguments. These are passed inside square brackets between the command name and the curly brackets, like this: `\command[optional]{argument}`.

1.5 Getting more information

So what can you do if you get stuck or just want more information. The simple answer is: Google is your friend. Most questions have already been answered. For example on Tex Strackexchange. Also, Overleaf has a great section for learning LaTeX.

An of course you can always check the documentation, which you can find on CTAN.

2 Basic text formatting

To begin I want to show you the basics of how to get text onto the page and structure it. You can also see how I created this exact text as an example.

2.1 Headings

The exact commands available will vary depending on your documentclass, but they will always be a single command that expects any text inside curly brackets, for example `\section{text}`. The different commands form a hierarchy you can nest into each other, keeping track of its parent element. That means you don't have to worry about any formatting or numbering, LaTeX will handle that for you.

When using an *article* documentclass, the commands available are `\section{}`, `\subsection{}` and `\subsubsection{}`. Should you need more nesting levels, you are usually overcomplicating things, but you could additionally use the `\paragraph{}` command, which gives you a slightly bigger, bold first word for your paragraph.

The *report* documentclass adds the additional command `\chapter{}` as the highest heading level. You can still use the previous three commands for the nested headings. A chapter automatically starts on a new page, so it should be at least two pages long. You also get the command `\part{}`, which creates a separate page for the part's title. These should only be used in very long documents.

2.2 Text paragraphs

Latex will format any text that is not part of a command (so not prefixes with a `\` or inside `{}`) as a plain-text paragraph. So the layout in your source code does not influence the layout of the resulting PDF. If you wanted to you could put everything on a single line, and it would work. This wouldn't be very readable however, so you should format your source code at least a little. Putting every sentence on a new line is very common, or you can configure your editor to automatically break when the line gets too long (this is preconfigured if you open this project in VS Code).

LaTeX will also automatically format your text to fill up all the available space and break long words for you. Sometimes, if you use special words LaTeX doesn't know, you may need to tell it where to split those. For one off cases you can just put `\-` where you want a break and for words you use a lot you can declare the hyphenation in the preamble as a space separated list, for example like this: `\hyphenation{word list donau-dampf-schiff}`

Lots of online sources will list the double-backslash (`\\`) as the command for a line break. While this is not wrong, you should not use it to break your text. Instead, you should use `\par` to denote the end of a paragraph. As programmers are lazy, LaTeX will actually insert this command for you, if you just leave an empty line between blocks of text. Using the correct command allows LaTeX to better find automatic breakpoints, allows you to define the spacing between paragraphs and has some other benefits.

2.3 Text spacing

By default, these classes add no spacing between paragraph, but sometimes you want to visually enforce a breakpoint in your argumentation. For that you can add some space in between to paragraph by using one of the commands `\bigskip`, `\medskip` or `\smallskip`. How much space you want depends on your taste, but you should keep it consistent. Here is an example:

This text has a big space before it,

Here I used just some medium spacing

and this is a small space.

2.4 Breaking pages

Sometimes you will find yourself in situations, where you don't like where LaTeX splits your text to the next page. So first, take some advice: Don't worry about it for now. Your text will probably change a few times before its final. Just leave it.

If you are at the final stage, you can do a beautifying pass. Now you can use `\pagebreak` to tell LaTeX about better places to break the text.

Should you still not be happy (this happens especially with multiple images/tables in close proximity) you most likely have to little text and should redesign your document. But if you absolutely want to print it that way, you can use `\clearpage` to force all figures/tables to be put onto the page and then start a new page.

You might also need just a little more space only a page to just fit one more sentences. For that you can use the command `\enlargethispage{N\baselineskip}` with N being the number of lines you need. Use this sparingly however, as the bottom margin is there for a reason and you shouldn't intrude on the footer too much.

2.5 Text styling

When writing text, you will need to *emphasize* certain parts of the text. The easiest way is to use the `\emph{}` command around you text. You can also nest it *to emphasize even more*.

If you want to change to a specific font-type, you can do that like this:

<code>\underline{text}</code>	<u>Underlined</u>
<code>\textbf{text}</code>	Bold Font
<code>\textit{text}</code>	<i>Italic Font</i>
<code>\textrm{text}</code>	Roman Font
<code>\texttt{text}</code>	Typewriter Font
<code>\textsc{text}</code>	Small Caps Font

You might also want to change your text colour, which is what the `color` package is for. It provides the `textcolor{colour}{text}` command, **which allows you to change your text colour**.

2.6 Special characters

2.6.1 LaTeX command characters

As in most programming languages, some characters are used for LaTeXes commands and can't be used in text directly. Here is a table explaining them all:

<i>character</i>	<i>special meaning</i>	<i>how to get character</i>
<code>\</code>	beginning of a command	<code>\textbackslash</code>
<code>{</code> and <code>}</code>	denote a code block	<code>\{</code> and <code>\}</code>
<code>%</code>	beginning of a comment	<code>\%</code>
<code>#</code>	macro parameter character	<code>\#</code>
<code>\$</code>	beginning/end of math mode	<code>\\$</code>
<code>~</code>	non-breaking space	<code>\textasciitilde</code>
<i>only inside math mode:</i>		
<code>_</code>	subscript	<code>_</code>
<code>^</code>	superscript	<code>\textasciicircum</code>

2.6.2 Invisible characters

To properly typeset your text you may need a number of special characters under specific circumstances:

<i>explanation</i>	<i>command</i>	<i>example</i>
non-breaking space	<code>~</code>	Max Mustermann (Names shouldn't be broken)
3/4 non-breaking space	<code>\;</code>	10 000 (separate thousands)
medium non-breaking space	<code>\:</code>	z. B. (abbreviations)
1/2 non-breaking space	<code>\,</code>	1 V (number + unit)
normal space	<code>\space</code>	in case some command eats up all your space

2.7 Cross-referencing

When writing you will often have to reference something else in your document. Of course, you don't want to manually type the number you are referencing and have to redo it whenever you add/delete a chapter (and surely forgetting something). That's why LaTeX does this for you and you don't have to worry about it!

For this to work, you have to give unique names to everything you might want to reference. This is done with the `\label{name}` command, which you just put after whatever you are labelling (it doesn't matter if it's on the same line or the next). The name can be everything you like, but choosing something you will easily remember later makes sense. I also recommend prefixing the label names with a descriptor for what it's referring to, for example `\label{chap: ...}` for chapters, `\label{fig: ...}` for figures and so on. That way you can have a chapter and a figure with the same name while still having unique identifiers.

To later refer to a label, you use the `\ref{name}` command, which will create the number of the labelled item, for example, this is a reference to chapter 1. You can now click the "1" and be taken to the corresponding chapter.

To make this easier and create a bigger, better clickable link, this classes load the `hyperref` package, which introduces the `\autoref{name}` command. This automatically adds the name of whatever you're referencing before the number, like this: chapter 2.

For easy use, I recommend you work with a decent editor, which automatically detects all labels and suggest them to you whenever you type a reference.

3 Examples

red text and blue text

different subscripts: R_t R_t

using Units: $R = 200 \text{ m}\Omega + 345.675 \times 10^{-3} \text{ V/m} - 5 \text{ s/m}^2$

some information[**laboranleitung:physik**]

german number: 3,5 english number: 3.5

3.1 Using formulas

a numberd formula:

$$0,5 = \frac{1}{3} \quad (3.1)$$

Equation 3.1 is nice, but how about multiple lines:

$$\begin{aligned} x &= x^2 + 3 \\ \Leftrightarrow 0 &= x^2 - x + 3 \end{aligned} \quad (3.2)$$

and how could you align formulas?

$$x_1 = 6 \quad | \text{ mit } x \in \mathbb{N} \quad (3.3)$$

$$x_2 = 33 + \left| \frac{1}{4} \right| \quad | x_1 + 3 \quad (3.4)$$

$$\begin{aligned} &= 33,25 \quad | \text{ don't number everything} \\ x_3 &= 10^{22} \end{aligned} \quad (3.5)$$

3.2 using Units

For this the `siunitx` package is used. It provides Macros for all units.

$$200 \text{ kg} \quad (3.6)$$

The space between a number and it's unit should be a protected half-space, which can be created in latex using `\,`. In the classfile `siunits` is set up to use a separate macro for each subunit, even for size-modifiers:

$$200 \text{ mm} \cdot 2 \text{ V} \quad (3.7)$$

`Siunits` also allows for reformatting of numbers as well as units. Use the `\SI` and `\si` macros for that:

$$e = 160.218 \times 10^{-21} \text{ C} \quad (3.8)$$

$$1.000 \text{ }\mu\text{m} \quad (3.9)$$

$$124 \frac{\text{km}}{\text{s}^2} \quad (3.10)$$

$$400.000 \times 10^{-6} \text{ lm} \quad (3.11)$$

3.3 using images

Images can just be imported and used in a float environment with a caption and a label to reference it. (see Figure 3.1)

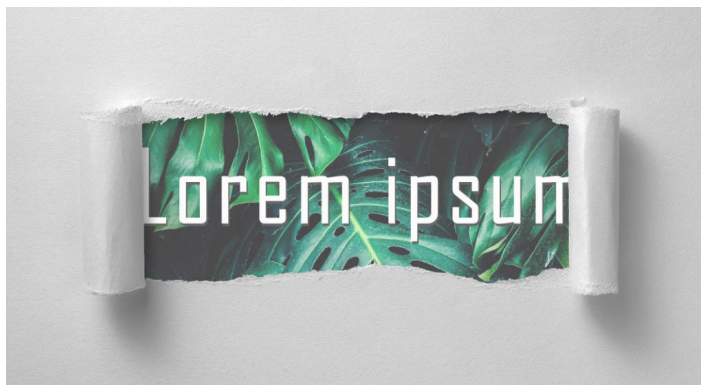
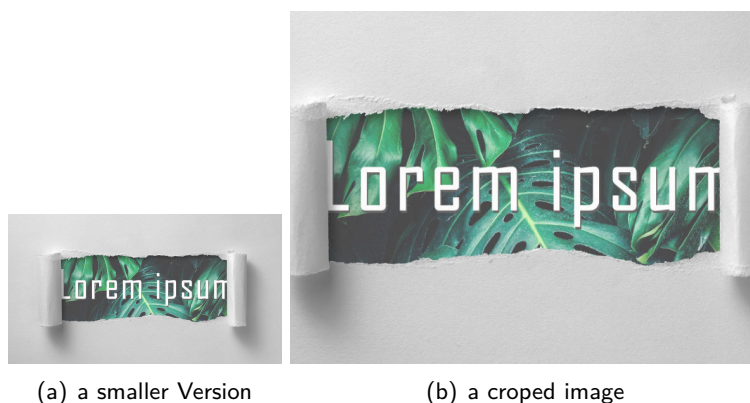


Fig. 3.1: just a random image

You can also display two or more images together, using the subfigure package. You can also resize or crop Images, as seen in Figure 3.2(a) and Figure 3.2(b)



(a) a smaller Version

(b) a cropped image

Fig. 3.2: some more images

Plots can be created directly with latex. It is recommended to do this in subfiles and just import the finished PDF pages. This speeds up compilation times by a lot. You should not change the size of precompiled images to keep font sizes consistent.

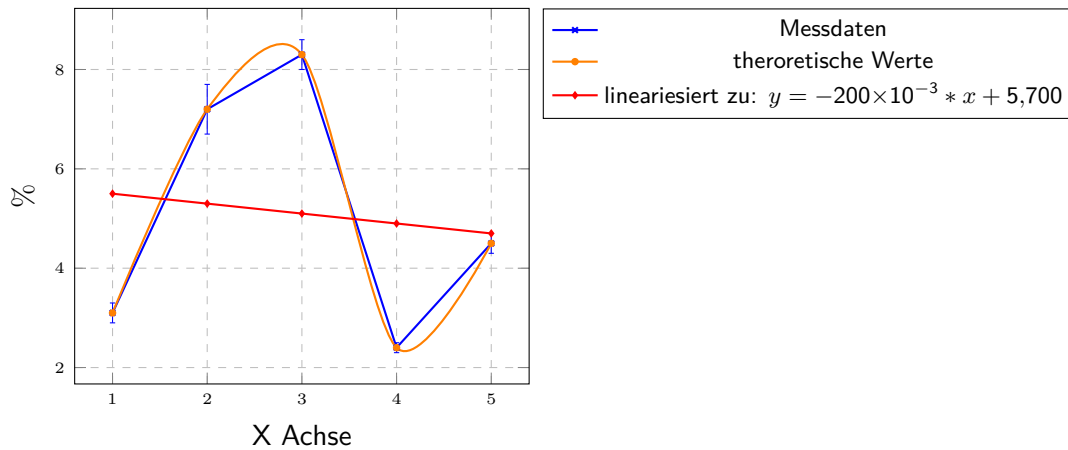


Fig. 3.3: a nice plot

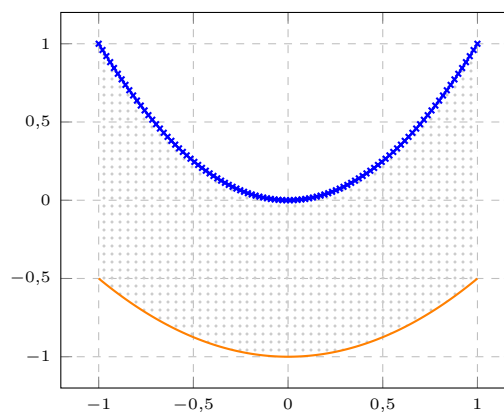


Fig. 3.4: a area plot

Circuit diagrams can also be created using a package called `circuitikz`. It is also recommended to get familiar with Inkscape which has a very good export to latex feature, as you can see in Figure 3.6. If you use Inkscape, there is a list of all electrical symbols here on wikipedia. You can download them as .svg files (not as png!) and just drag&drop them into Inkscape.

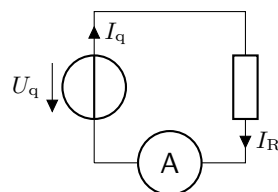


Fig. 3.5: a circuit diagramm

Using Inkscape, you can create SVG-vector graphics and import them easily into Latex.

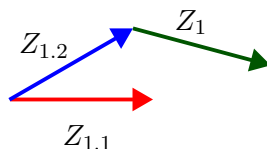


Fig. 3.6: A image created with Inkscape

3.4 using tables

Tables are a little bit complicated in LaTeX, but don't worry, here are some examples:

Tab. 3.1: a simple table

A	B
1	2
3	4

As you can see, tables are build using two nested environments. The `table` creates a floate just like a `figure` would. You can then just give it a caption and a lable.

The `tabular` environment creates the actual table. You need to devine the alignment for every column and give delimiters between lines. Each cell is ended by a `&` and a newline is created as always. Using `\hline` creates a vertical line after the row.

Here is a more complex example:

Tab. 3.2: a bigger table

ID	NAME	Price	Currency	Stock
1	Product A	10	EUR	20
2	Stuff	1	USD	200
A cool Teddy		50	EUR	1

3.5 lists and enumerations

This is a nested List:

- hallo
 - temp
 - temp
 - temp

And this is a nice checklist:

- ☐ first
- ☐ urgent
 - ☐ sub item
 - ☐ and another
- ☐ continue

3.6 CSV files

import a csv as table:

A	B	C	D
1	0	3,1	0,2
2	0	7,2	0,5
3	0	8,3	0,3
4	0	2,4	0,1
5	0	4,5	0,2

or do it manually to get more control:

Tab. 3.3: a nice list of numbers

first row	second row
number: 1 m	is not 3,1
number: 2 m	is not 7,2
number: 3 m	is not 8,3
number: 4 m	is not 2,4
number: 5 m	is not 4,5

3.7 formatting code

use the listings package:

```
#include <stdlib.h>
#include <sdtio.h>

int main(void) {
    printf("Hello World");
    return 0;
}
```

4 seperating the document

This was inputed from anothe file!!

It can be usefull to seperate yout document into chapterfiles. This allows to only compile the changed parts of the document or work with multiple people at the same time, but on different chapters.

If you use a more advanced text editor like VS-Code, the editor even compiles the hole document, even when you are editin a subfile.

5 attachment

Messprotokoll oder so As you can see its also possible to have some pages sideways. Just keep in mind you might need to adapt the margins

List of Figures

3.1	just a random image	7
3.2	some more images	7
3.3	centering	8
3.4	a area plot	8
3.5	a circuit diagramm	8
3.6	A image created with Inkscape	9

List of Tables

3.1	a simple table	9
3.2	a bigger table	9
3.3	a nice list of numbers	10