Study Now

LaTeX webinar

Benjamin Rotendahl — Benjamin@Rotendahl.dk

October 13, 2021

Contents

1 Introduction

This document is as a *cheat sheet*. It covers how to setup a LaTeX document and showcases the most common LaTeX commands and functions. You have both the pdf and source code such that you can add other cool commands you find on your journey.

2 LaTeX background

Before diving into the syntax of LaTeX you should know a bit about the process of writing LaTeX documents. LaTeX is a typesetting program and language. The main advantage of LaTeX is its ability to typeset math, code and other scientific/technical figures. LaTeX in an extension of TeX, which released by Donald Knuth in the year 1981.

2.1 From source code to PDF

LaTeX is not a word processor, but a language and compiler which given a file containing source code, written in an *editor*, creates a PDF.

An editor for local use, could be *Visual studio code*[2], it can be used to write any programming/-markup language. Support for languages comes through packages that extends its functionality, there exists packages for most languages such as python, javascript and LaTeX[3].

After the editor has saved the source code, it must be passed to the compiler. This can be setup to happen automatically on save. There are different versions of the compiler, they differ in the amount of packages and features they come with. "The latex project TeX"[1] has links to various distributions for the most common operating systems. With an editor and compiler installed you are ready to write LaTeX. The processes is illustrated in figure 1.

2.1.1 Overleaf

Section 2.1 explains how to create a LaTeX setup on your own computer. There exists services such as Overleaf[1], they are to LaTeX what Google docs is to Microsoft Word. They live in the browser and give you a full environment, no installation required. This makes it easier to share documents with potential group members, but has the disadvantage that they require an internet connection to function. They are also more limited in the amount of configuration options and packages available.

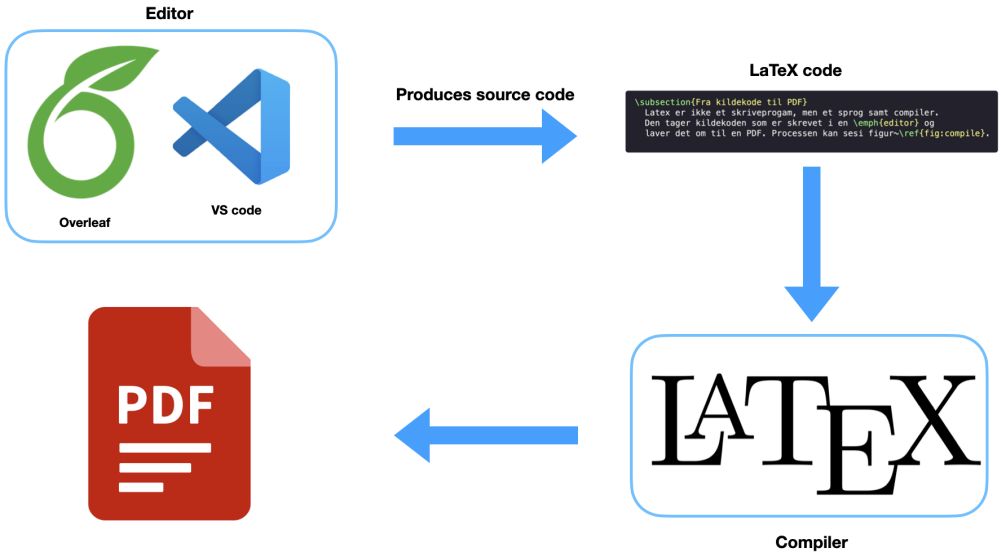

Figure 1: From source code to pdf

[1] overleaf.com

3 LaTeX syntax

In LaTeX we don't click buttons to change formatting, instead we tell the compiler what type of text we are writing. This leaves the appearance to our template/compiler leaving us, the writer, to focus on the content.

A LaTeX file consists of two parts, the *preamble* and the document. The preamble is where we define the LaTeX packages we are going to use, the style of our document, macros, and other configurations options. The document is where we write the content.

3.1 Command syntax

A command consists of a "backslash" followed by a command name. If the command takes parameters they are written after the command name in curly parenthesis. As an example if we want to write LaTeX we type `\LaTeX` . Below is a list of the most common syntax constructs.

New lines LaTeX handles newlines for us, if you write a single line break it will be ignored, so you can format your source code as you wish. If you want to force a line break you can write two line breaks, write `\linebreak` or type two backslashes `\\` .

Forced spaces As with line breaks, more than one space are ignored To force a space you can write `\space` or `~~` .

New page To force latex to start a new page we type `\newpage`

Qoutes To write quotes such as "Some thing", we type plings and apostrophes: ``` ``qouted text'' ```.

Comments If you want to type something in your source code but not include it in the final pdf, we can use comments. This is a useful way to leave notes to your future self or group mates. typing % will result latex in ignored the remainder of the line. To type %, prefix it with a backslash `\%` .

Math There are two ways to type math in LaTeX. The first is to *inline*, which places the expression in the flow of the text, i.e $x^2 + 4$, this is done by typing `\(x^2 +4\)` . The other way is *display* mode, where the expression is centered and larger, its written as `\[x^2 +4\]` .

$$x^2 + 4$$

Environemnts For commands with a large number of parameters we can use the *begin/end* construct. This is done by typing `\begin {environment}[options]\end {environment}` where environment is the name of the environment and options are potential configurations.

Listing 1 shows a simple latex-fil, using the above LaTeX commands.

```
\documentclass{article}
\usepackage{amsmath} % Package for maths
\usepackage{graphicx} % Package for graphics

\title{
 \large{Study Now} \\
 \Large{\LaTeX webinar} \\
}

\author{
 Benjamin Rotendahl --- Benjamin@Rotendahl.dk
}

\begin{document}
  \maketitle
  \section{Introduction}
   This documents gives you \(\dots\)
\end{document}
```

Listing 1: Example of simple Latex document

3.2 Reference martial and problem solving

Learning LaTeX is an iterative process, after having learned the basis you should start using it as much as possible, improving gradually. There are to many commands to learn all in one sitting. Once you start writing and run into something you don't know how to do yet, google it and remember it for next time. For instance if you want to know how to insert two figures side by side, a google search will most likely lead you to *tex.stackexchange.com* which has several examples. For more advanced features and a comprehensive guide, checkout "The Not So Short Introduction to LaTeX[2]".

You will often, especially in the beginning, run into errors where the compiler fails to produce a pdf. The Compiler outputs a log where it tries to explain what went wrong in a helpful way. As an example if we mistype and write `\newpge` in stead of `\newpage` , the compiler will write the following in the log.

```
[h]
.../study_now/latex/cheetsheet.tex:241: Undefined control sequence.
l.241 ...ewpage} we write \newpge
```

The log tells us where the error happened and why. Once you learned to read the logs you will be able to fix errors quickly.

4 Math in LaTeX

Writing beautifully formatted math is one of the biggest strengths of LaTeX. We start by telling the compiler to enter *math mode*, it will then parse the following text as math. It's the difference between y+x and $y + x$. Many commands only work in math mode, If you write `\pi` in a non math mode the compiler will fail. Instead we write `\(\pi \)` , resulting in π.

For larger math expressions we can use the *display* mode, which is invoked by typing `\[x+y\]` or by using the `\equation` environment. Using the second approach makes it possible to create references to formulas. As an example the equation (1) shows the dot product of two vectors.

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i \tag{1}$$

The code to create the above expression is shown in listing 2.

```
\begin{equation}\label{dot} % Label to reference
 \vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i
\end{equation}
```

Listing 2: Equation for the dot product

If we wish to have multiple steps in a derivation we can use the `flalign` environment. This is a way to align equations in a derivation, we type each line use `\\` to start a new line and set the `&` symbol to specify the symbol of each line that should be centered.

$$f(x, y) = 3x^2y + y^2 \tag{2}$$
$$\frac{\partial f}{\partial x} = 6xy \tag{3}$$
$$\frac{\partial f}{\partial y} = 3x^2 + 2y \tag{4}$$

The code to produce the above can be seen in 3. If you don't want line numbers on the right hand side type `flalign*` instead of `flalign` . To write parenthesis in formulas we can type them normally, but this will not scale them according to their contents. To scale them type `\left ( x^2 \right )`

[2] http://web.math.ku.dk/ holm/download/lshort.pdf

```
\begin{flalign}
 f(x, y)                        & = 3x^2y + y^2 \\
 \frac{\partial f}{\partial x} & = 6xy          \\
 \frac{\partial f}{\partial y} & = 3x^2 + 2y
\end{flalign}
```

Listing 3: Example to create multi line math

to use different types of parenthesis replace the symbol after `\left` , `\right` . Listing 4 shows the source for the following example

$$f(x) = (\frac{1}{x})^2$$
$$f(x) = \left(\frac{1}{x}\right)^2$$
$$x \in \left[\frac{1}{4}, \frac{1}{2}\right]$$
$$x \in \left\{\frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \dots, \frac{n}{n}\right\}$$

```
\begin{flalign*}
  f(x) &= (\frac{1}{x})^2  \\ % % Non scaled
  f(x) &= \left ( \frac{1}{x} \right )^2  \\ % Scaled
  x  \in \left [ \frac{1}{4},  \frac{1}{2} \right ] \\ % scaled brackets
  x  \in \left \{\frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \dots, \frac{n}{n} \right \}
\end{flalign*}
```

Listing 4: Code to scale parenthesis

4.1 Math operations and notation

The following is a list of the most common math notation.

Standard operations Writing $+, -$ is done normally, to do multiplication you can either use `a \cdot b`, `a \times b` , which becomes $a \cdot b, a \times b$.

$$1 + 1 - y \cdot x$$

Fractions To write fractions we use the `\frac {a}{b}` command. The content of the first argument is placed on the numerator and the second the denominator. Fractions can also contain fractions.

$$\frac{a}{b} + \frac{a}{\frac{b}{c}}$$

Super/Sub-scripts To write x^2, x_i, one types `\(x^2, x_i\)` . To include more than one symbol in a super/sub-script it has to be in a set of curly parenthesis, ie. `\(x_{i+1}\)` , resulting in x_{i+1}.

Sums and integrals To write sums and integrals we write the name of the command followed by super and sub scripts. `\sum _{i=1}^{n} a_i b_i` or `\int _{i=1}^{n} a_i b_i` Resulting in:

$$\sum_{i=1}^{n} a_i b_i + \int_{i=1}^{n} x$$

4.2 Matrices & Vectors

LaTeX makes it easy to write vectors and matrices. It's done by using the environments:
`\pmatrix` , `\bmatrix` , `\bMatrix` which respectively creates a matrix with $(, [, \{$. In the environemnt you write the rows using `&` to seperate the elements and `\\` to seprate the rows. The

following example used the code in listing 5. Note that the command `\quad` was used to create spacing between the matrices and that the last matrix used the `\dots` , `\vdots` , `\ddots` commands.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{Bmatrix}, \quad \begin{bmatrix} 1 & \dots & n \\ \vdots & \ddots & \vdots \\ n & \dots & n \end{bmatrix}$$

```
\[
 \begin{pmatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
 \end{pmatrix}
 ,\quad
 \begin{bmatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
 \end{bmatrix}
 ,\quad
 \begin{Bmatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
 \end{Bmatrix},
 ,\quad
 \begin{bmatrix}
  1      & \dots  & n      \\
  \vdots & \ddots & \vdots \\
  n      & \dots  & n
 \end{bmatrix}
\]
```

Listing 5: Examples of different types of matrices

5 Figures

To create figures with images and graphs we can use the `figure` environment. A figure is *floating* element, LaTeX will place the figure in the document to minimize blank space. We create a figure

```
\begin{figure}[h]
 \centering\includegraphics[width=0.9\textwidth]{assets/compile.png}
 \caption{From source coude to pdf}\label{fig:compile}
\end{figure}
```

Listing 6: Example of inserting a figure

environment, adding the `[h]` option to indicate that we prioritize having the figure close to the text. *h* could be replaced by `[t]`, `[b]` which would place the figure at the top or bottom of the page respectively. The `\centering` command places the figure in the middle, and `\includegraphics` inserts the image, the `\width =.9` specified the width of the image to 90% of the page width. The last part of the `\includegraphics` command in curly parenthesis is the to the file file. Using `\caption` and `\label` we create a description of the figure and a label to reference.

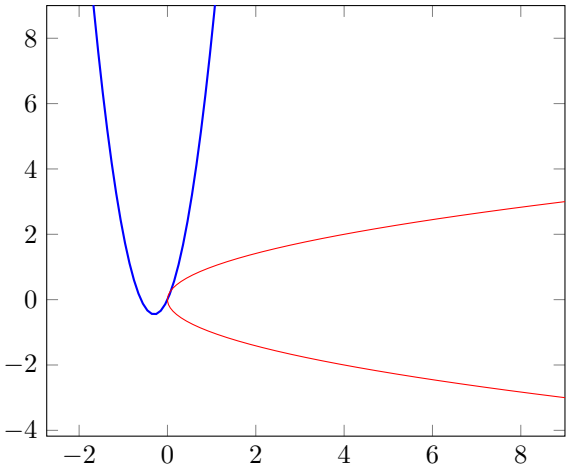

Figure 2: Example of a tikz graph

5.1 TikZ

Using the tikz extension we can create our own graphs and figure directly from latex. TikZ is as powerful as it complex, learning it takes time and effort. Creating the graphs in another tool and exporting a png is also a viable option. Figure **??** shows a graph created in TikZ and listing 7 shows how it was created.

```
    \begin{figure}
     \centering
     \begin{tikzpicture}
      \begin{axis}[xmax=9,ymax=9, samples=50]
      \addplot[blue, thick] (x,5*x^2+3*x);
      \addplot[red, thin] (x*x,x);
    \end{axis}
   \end{tikzpicture}
  \caption{Example of a tikz graph}\label{tiks:graph}
 \end{figure}
```

Listing 7: Example creating a figure in tikz

6 Lists

In latex we primerailliy use three different types of lists, in general their syntax is:
`\begin {list_type} \item thing 1 \item thing 2 \end {list_type}` . Lists can contain sub lists and handles bullets and enumerations automatically. The code to produce the following list can be seen in listing 8

`itemize` A list with bullets.

- Some Bullet
- Some other Bullet
- A bullet point and a sub list
  - a sub bullet
  - some other sub bullet

Enumerate A list with numbered points, with a sub list where each sub point also has a unique id.

1. Some point
2. Some other point
3. Some other, other point with a sub list
   (a) sub list 1
   (b) sub list 2

`description` A list with a heading for each list item written as `\item [header]`

First thing first

Second thing second thing
Third thing list with sub headings
Fist sub thing first sub thing
Second sub thing second sub thing

```
\begin{description}
  \item[\tex{itemize}] A list with bullets.
    \begin{itemize}
      \item Some Bullet
      \item Some other Bullet
      \item A bullet point and a sub list  \begin{itemize}
              \item a sub bullet
              \item some other sub bullet
            \end{itemize}
    \end{itemize}
  \item[Enumerate] A list with numbered points, with a sub list where each sub
  point also has a unique id.
    \begin{enumerate}
      \item Some point
      \item Some other point
      \item Some other, other point with a sub list  \begin{enumerate}
              \item sub list 1
              \item sub list 2
            \end{enumerate}
    \end{enumerate}
  \item[\tex{description}] A list with a heading for each list item written as
 \tex{\item[header]}
    \begin{description}
      \item[First thing] first
      \item[Second thing] second thing
      \item[Third thing] list with sub headings \begin{description}
          \item[Fist sub thing] first sub thing
          \item[Second sub thing] second sub thing
        \end{description}
    \end{description}
\end{description}
```

Listing 8: Different types of lists

7 Tables

Creating a table requires two environments `\table` , `\tabular` , they function just as `\figure` , `\includegraphics` , one creating a float and the other the contents. Tabel 1 is the results of the code in listing 9. As with the figure we add the `[h]` option to have the table close to our text. The `\tabular` environnement starts the actual table, the first pair of curly parenthesis indicate the number of elements in each row and where they justification. By typing `\tabular` |l|c r|, we create three columns, the first column is left justified, the second is centered and the third is right justified. The pipe symbol `|` creates a vertical bar between the columns. The elements of the table is written as a matrix using `&` to separate elements and `\\` to separate rows. The `\hline` command can be added after a line to create a horizontal separator.

7.1 Automatic tables

If you have a file in csv format you can add the table automatically by including `\usepackage {csvsimple}` in your preamble and typing:

column1	column2	column3	column4
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Table 1: Example of table

```
\begin{table}[h]
\centering
\begin{tabular}{||c c | c c||}
 \hline
    column1 & column2 & column3 & column4 \\  \hline
    1       & 6       & 87837   & 787     \\
    2       & 7       & 78      & 5415    \\
    3       & 545     & 778     & 7507    \\
    4       & 545     & 18744   & 7560    \\
    5       & 88      & 788     & 6344    \\
    \hline
\end{tabular}
\caption{Example of manual table}\label{table:data}
\end{table}
```

Listing 9: Example of manual table

```
  \begin{table}[h]
    \centering\csvautotabular{assets/dices.csv}
    \caption{Automatic table from csc}\label{table:dices}
  \end{table}
  \caption{Example of automatically generated table}\label{lst:dices}
```

Listing 10: Example of automatically generated table

8 Code listings

To include code listings in your document we can again either do as shown in the examples below: Which was produced by:

```
def fib(x):
 if x == 1 or x == 2:
  return 1
 else:
  return fib(x-1) + fib(x-2)
```

```
\begin{listing}[!h]
 \begin{lstlisting}[language=python, tabsize=1]
  def fib(x):
   if x == 1 or x == 2:
    return 1
   else:
    return fib(x-1) + fib(x-2)
  \end{lstlisting}
\caption{Manual code listing }\label{lst:dices}
```

Listing 11: Code to generate automatic listing

	2	3	4	5	6	7	8	9	10	11	12
counts	2833	5571	8282	11240	14000	16765	13722	11104	8290	5446	2747
percentage	2	5	8	11	14	16	13	11	8	5	2

Table 2: Automatic table from csv

To get prettier listings with syntax highlighting as in this example we must install the package *minted*. Follow the guide on their page: `https://github.com/gpoore/minted`

9 References

Adding labels to you contents allows you to easily reference them. Labels are written as `\label {thing:name}` and can be added to almost any element, including but not limed to `\section` , `\subsection` , `\figure` , `\table` , `\equation` . Once a label has been made we can refer to it by: `\ref {thing:name}` . Labels are used for internal references in the document, to refer to other articles and papers we use the extension *bibtex*. These are written in a file named along the lines *references.bib* and have contents similar to listing listing 12 One the file has been created we

```
@article{attention,
 author    = {Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit},
 eprint    = {1706.03762},
 journal   = {arXiv},
 keywords  = {Language Transformer},
 rating    = {5},
 title     = {{Attention Is All You Need}},
 year      = {2017}
}

@misc{latexPackage,
 howpublished = {\url{
  https://marketplace.visualstudio.com/items?itemName=James-Yu.latex-workshop
 }},
 title        = {VS Code Latex Package}
}
@misc{texLive,
 howpublished = {\url{https://www.latex-project.org/get/}},
 title        = {The LaTex Project}
}
@misc{vscode,
 howpublished = {\url{https://code.visualstudio.com}},
 title        = {Microsoft Visual Studio Code}
}
```

Listing 12: Code to generate automatic listing

add the following to the bottom of our document to generate the references. To reference an article

```
\bibliography{assets/references}{}
\bibliographystyle{plain}
```

Listing 13: Code to generate automatic listing

we type: `\cite {attention}` resulting in *Attention Is All You Need*[4]

References

[1] The latex project. https://www.latex-project.org/get/.

[2] Microsoft visual studio code. https://code.visualstudio.com.

[3] Vs code latex package. https://marketplace.visualstudio.com/items?itemName=James-Yu.latex-workshop.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv*, 2017.