Study Now

LaTeXwebinar

Benjamin Rotendahl — Benjamin@Rotendahl.dk

October 27, 2021

Contents

1 Introduction

Dette dokument fungerer som et *cheatsheet*. Det gennemgår de forskellige former for opsætning og funktioner i LaTeX. Når du selv sidder og skriver, kan du enten bruge den færdige PDF som opslagsværk, eller kigge på kildekoden. Husk, at du selv kan udvide dokumentet med yderligere sektioner, hvis du finder nogle seje funktioner i din videre færd med LaTeX.

2 LaTeX baggrund

Før vi går i gang med syntaksen osv., gennemgår vi, hvordan LaTeX fungerer og laver kildekode om til en PDF. LaTeX er et skriftsstyringssprog, som bruges til at skrive dokumenter med. Det er særligt godt til at skrive dokumenter med matematik, kode og andre videnskablige figurer. LaTeXer en udvidelse af TeX, som blev udgivet af Donald Knuth i 1989.

2.1 Fra kildekode til PDF

LaTeXer ikke et skriveprogam, men et sprog samt compiler[1]. Den tager kildekoden, som er skrevet i en *editor*, og laver det om til en PDF. En editor til lokalt brug kan f.eks. være *Visual studio code*[2]. Det er en editor, som kan bruges til at skrive alle slags sprog. Man installerer ekstra pakker, som udvider dens funktionalitet, f.eks. har den både pakker til at hjælpe med python-kode, javascript og LaTeX[3].

Efter koden er skrevet, skal den gives videre til en *compiler*, som tager kildekoden og laver det om til en PDF. Det er compileren, som inkluderer filer, figuerer osv. Er der en fejl i kildekoden, gør compileren sit bedste for at fastslå, hvad fejlen er, og hvordan den kan fixes. Der findes forskellige versioner af LaTeX compileren, som prinært varierer i, hvilke ekstra pakker og værktøjer de inkulderer. "The latex project TeX"[1] har links til distrubtioner for de de forskellige styresystemer. Med en compiler og editor installeret er man klar til at bygge sit første latex dokument, processen kan ses i figuer 1.

2.1.1 Overleaf

Sektion 2.1 forklarer, hvordan du opsætter dit eget miljø lokalt. Der findes tjenester som Overleaf[2], der fungerer på samme måde som google docs. Man arbejder i en webrowser, hvor de giver en et fuldt opsat miljø. Man slipper altså for at installere noget lokalt og har nemmere ved at dele sit arbejde med eventuelle gruppemedlemmer. Ulempen er, at man ikke har samme konfigurationsmuligheder som lokalt, og det kræver internetadgang at tilgå.

2.2 LaTeX tankegang & syntax

LaTeX adskiller sig fra programmer som word, google docs, osv. ved ikke at være en så såkaldt *WYSIWYG*[3] editor. I stedet for at formattere teksten, mens man skriver, og bruge knapper i det grafiske interface til at ændre udseendet, så gør man det via kommandoer. Tanken bagved er, at man skal koncenterere sig om sit indhold og lade kompileren om formatteringen via kommandoerne.

En latex-fil består af to dele: den såkaldte *preamble* og dokumentet. Preamblen består af kommandoer, som definerer hvilken type dokument, vi arbejder med, laver global opsætning og inkluderer de pakker, dokumenetet kræver.

2.2.1 Kommando syntax

En kommando består af en "backslash" efterfulgt af et kommandonavn. Tager kommandoen parametre, skal de skrives i krølleparanteser efter. F.eks. kan vi skrive LaTeX ved at skrive `\LaTeX` . Der findes genveje til de mest gængse kommandoer.

Linjeskift LaTeX sørger selv for at lave linjeskift i ens tekst. Laver man et manuelt linjeskift, ignorer programmet det. Det er på den måde, at man har bedre mulighed for selv at formattere ens

[1] Det, man kalder en *oversætter* på dansk
[2] `overleaf.com`
[3] What you see is what you get

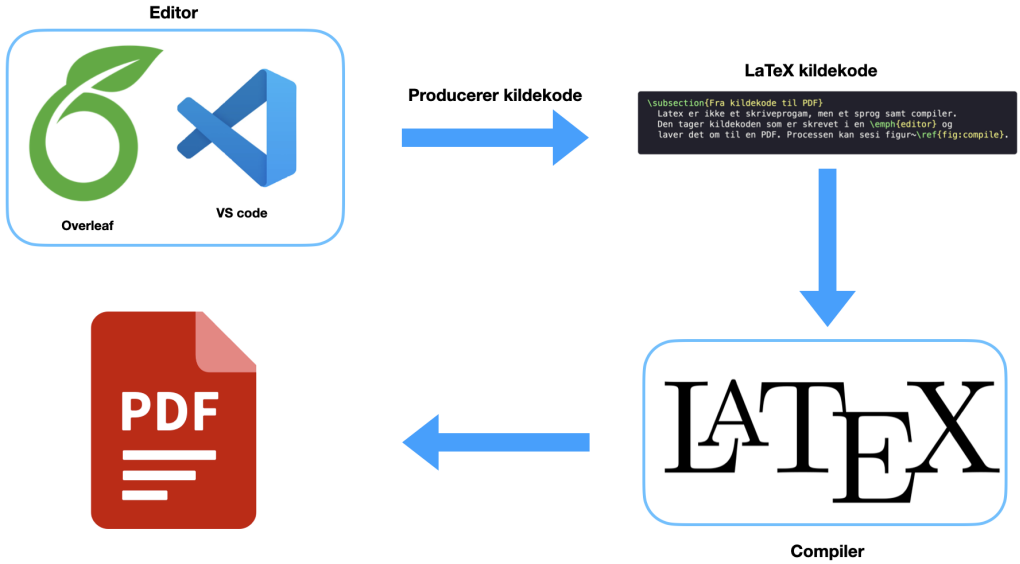

Figure 1: Hvordan man kommer fra kildekode til pdf

kildekode, uden det påvirker den endelige pdf. Ønsker man selv at bestemme et linjeskift, kan man enten indsætte to linjeskift i sin egen kode, skrive `\linebreak` eller to backslashes. `\\`

Tvunget mellemrum Ligesom med linjeskift, fjerner LaTeX også nogle mellemrum. Hvis man har man sat flere end et, bliver de slået sammen til et enkelt. Ønsker man et tvunget mellemrum, skriver man `hej ~~~ verden`

Ny side Ønsker man at tvinge LaTeXtil at skifte til en ny side, skriver man `\newpage`

Anførselstagn Ønsker man at sætte tekst i "anførselstegn", gøres det ved at skrive ``` ``anførselstegn'' ```

Kommentar Ønsker man at inkuldere noget tekst i sin kildekode, men ikke i den færdige pdf, kan man lave en *Kommentar*. Det er brugbart for at lave påmindelser til sig selv eller gruppemedlemmer. Sætter man et % tegn, vil resten af linjen ikke blive inkluderet. For at skrive %, sætter man så en backslash foran `\%`

Matematik Der findes to måder at skrive matematik på, den ene er *inline*, hvor matematikken står inde i teksten, f.eks. $x^2 + 4$, hvilket gøres ved at skrive `\(x^2 +4\)` . Den anden er *display* mode, hvor matematikken står udenfor teksten, det skrives sådan her: `\[x^2 +4\]` . f.eks

$$x^2 + 4$$

Miljøer Ved større kommandoer kan det blive svært at skrive det hele i krølleparanterser bagefter. Til sådane kommandoer kan man skrive det med miljønotation. Ønkser vi f.eks. at have en linje i midten, kunne vi skrive `\centering {\\ Jeg er i midten \\}` .

Jeg er i midten

Var min tekst længere, ville det være nemmere at skrive

Der findes mange miløjer til at lave alt fra lister, matricer, tabeller, figurer osv.

Listing 1 viser en simpel latex-fil, hvor de overstående LaTeX kommandoer bliver brugt til at lave en simpel pdf.

2.3 Referencemateriale og problemløsning

At lære LaTeXer en løbende proces. Efter at have lært det basale, skal man bare i gang med at skrive. Der er for mange kommandoer til at sætte sig ned og lære det hele på en gang. En bedre proces er at kaste sig ud i det og løse problemer og lære nye kommandoer undervejs. Hvis man f.eks. ønsker at inkludere to figurer ved siden af hinanden og ikke ved det endnu, er google den bedste løsning. Sådan en søgning vil typisk give et eksmpel fra *tex.stackexchange.com* eller ligende, og her kan man så kopiere eksemplet fra. Ønsker man en større guide, findes der "The Not So Short Introduction to LaTeX[4]", som går mere i dybden med LaTeX.

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

```
\documentclass{article}
\usepackage{amsmath} % pakke til ekstra matematiksymboler
\usepackage{graphicx} % pakke til figurer og billeder

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

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

\begin{document}
  \maketitle
  \section{Introduction}
   Dette dokument viser hvordan \(\dots\)
\end{document}
```

Listing 1: Ekspempel på simplet latex dokument

Når man laver fejl i sin latex-kode, kan ens dokument ikke kompileres, og der vil ikke blive lavet en pdf. Compileren laver en log, hvori den prøver at beskrive, hvorfor den fejlede. Det er lidt en kunst at læse de logs og finde ud af, hvad der egentlig er i vejen. Skriver vi f.eks. `\newpge` i stedet for `\newpage` , giver loggen os følgende linjer begravet i omkring 300 linjer.

```
/Users/rotendahl/Documents/projects/study_now/latex/cheetsheet.tex:241: Undefined control sequence.
l.241 ...ewpage} vi f.eks \newpge
```

Den fortæller os, hvilken linje der fejlde, og at det sker, fordi den ikke ved, hvad `\newpge` betyder. Når man først har lært at lede i loggen efter det vigtige, kan man hurtigt fixe sine fejl eller google log-beskeden og lære, hvorfor den gik i stykker.

3 Matematik i LaTeX

Latex' største styrke er dens evne til at formatere matematisk notation. For at skrive matematik skal man fortælle compileren, at den skal gå i *math mode* for så at kunne skrive matematik specifikke kommandoer. Prøver man f.eks. at skrive `\pi` uden at være i math mode, fejler den. Man kan som nævnt gå i inline math mode ved at skrive `\(\pi \)` , som giver π. Der findes også miljøer, som sætter en i math mode, f.eks. `\equation` miløjet. Det går det også muligt at lave referencer til sine formler, f.eks. viser ligningen (1) formlen for prikproduktet af to vectorer.

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

Koden, der producerer overstående, kan ses i ekspempel 2

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

Listing 2: Ligning for prikproduktet

Ønsker vi at have en flere trin i en udledning, kan vi bruge flalign-miløjet. Der kan man have flere

linjer, som bliver centreret efter placeringen af `&` symbolet.

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

Koden, der skaber overstående, kan ses i listing 3. Ønsker man at fjerne tallene ude i højre side, kan det gøres ved at erstatte `flaling` med `flaling*` .

```
\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: Eksempel på formler centreret over flere linjer

For at lave paranteser i sine formler skal man bare skrive som normalt, men ønsker man, at de skalerer efter indholdet, så kræver det, at man skriver `\left ( x^2 \right )` rundt om. Typen af parenteser efter `\left , \right` kommandoen bestemmer, hvordan det ser ud. Listing 4 viser, hvordan følgende er lavet.

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

3.1 Matematikoperationer og notationer

Følgende er et opslagsværk med de mest gængse matematikoperationer og notationer. Først er der en liste over syntaxen, som bliver efterfulgt af et eksemplet på, hvordan det bliver formateret, og til sidst står koden for eksemplet. Eksemplerne i listen antager, at man allerede er i math mode, f.eks. `\begin {flalign} x + 1 \end {flalign}` eller f.eks. i display mode `\[ x + 1 \]`

Standardoperationer for at skrive $+, -$ gør man bare som normalt. Ønsker man et gangetegn kan det gøres med enten `a \cdot b, a \times b` , som bliver til $a \cdot b, a \times b$

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

Brøker for at lave en brøk bruger man `\frac {a}{b}` kommandoen. Indholdet af den første krølleparantes sættes i tælleren og, indholdet af den næste sættes i nævneren. Man kan også have brøker i brøker.

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

Potenser og subscript Ønsker man at skrive f.eks. x^2, x_i, gøres det ved at skrive `\(x^2, x_i\)` . Vil man have mere end et tegn inkluderet, skal det sættes i krølleparenteser, f.eks. `\(x_{i+1}\)` , som bliver til x_{i+1}.

Sum og integraler Ønsker vi at lave et summationstegn, kan det gøres ved at skrive sum-kommandoen efterfulgt af en øvre og nedre potens. `\sum _{i=1}^{n} a_i b_i` eller `\int _{i=1}^{n} a_i b_i` Et integrale kan skrives på samme måde, men med *int* i stedet for `\int {i=1}^{n} a_i b_i`

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

```
\begin{flalign*}
 f(x) &= (\frac{1}{x})^2  \\ % Ikke-skaleret parantes
 f(x) &= \left ( \frac{1}{x} \right )^2  \\ % skaleret parantes
 x  \in \left [ \frac{1}{4},  \frac{1}{2} \right ] \\ % Firkantede paranteser
 x  \in \left \{\frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \dots, \frac{n}{n} \right \} % krølleparan
\end{flalign*}
```

Listing 4: Eksempel på forskellige typer paranteser

3.2 Matricer og vektorer

Latex gør det nemt at skrive matricer og vektorer. Det gøres med miljøkommandoerne `\pmatrix` , `\bmatrix` , `\bMatrix` , som laver en matrice med henholsvis (, [, {. Inde i milljøet skriver man elementerne i hver række adskilt af `&` , og rækkerne adskilles enkeltvis af `\\` . Eksemplet herunder kan findes i listing 6. Læg mærke til, at kommandoen `\quad` bruges til at lave afstand mellem matricerne, og hvordan `\dots` , `\vdots` , `\ddots` bruges til at lave den sidste matrice.

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

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

Listing 5: Eksempel på forskellige typer matricer

4 Figurerer

Latex understøtter også figuerer, som er lavet med `figure` miljøet. En figur er en såkaldt *float*, det vil sige, at LaTeXselv finder ud af at indsætte indholdet et sted på siden. En figur kan indeholde tekst, tabeller, billeder, osv. At lave figur gøres typisk ved: I overstående starter vi først et figurmiljø. På første linje betyder `[h]` , at figueren skal være *here*, som fortæller latex, at vi prioriterer, at figueren er så tæt på naboteksten som muligt. Sætter vi en af `[t], [b]` ind i stedet, vil compileren enten

```
\begin{figure}[h]
 \centering\includegraphics[width=0.9\textwidth]{assets/compile.png}
 \caption{Hvordan man kommer fra kildekode til pdf}\label{fig:compile}
\end{figure}
```

Listing 6: Eksempel på indsættlese af figur

sætte det i toppen eller bunden af siden. Vi bruger `\centering` til at sætte vores billede i midten af siden, `\includegraphics` indsætter billedet og `\width =.9` sætter billedets bredde til 90% af sidens bredde. Den sidste del af `\includegraphics` kommandoer skal være i krølleparanteser og er stien til den fil, der skal inkluderes. Ved at bruge `\caption` og `\label` vil vi få et label til vores figur og en figurbeskrivelse.

4.1 TikZ

Man kan bruge udvidelsen TikZ til at lave grafer, plots, og andre figurer. TiKZ kan være rigtig smart, men det kræver en god del at lære hvordan man bruger det. Alternativt kan man fint lave sine grafer i python, excel eller andet og importere dem som billeder. Figur 2 viser et plot lavet i TikZ and eksempel 7 viser koden bag.

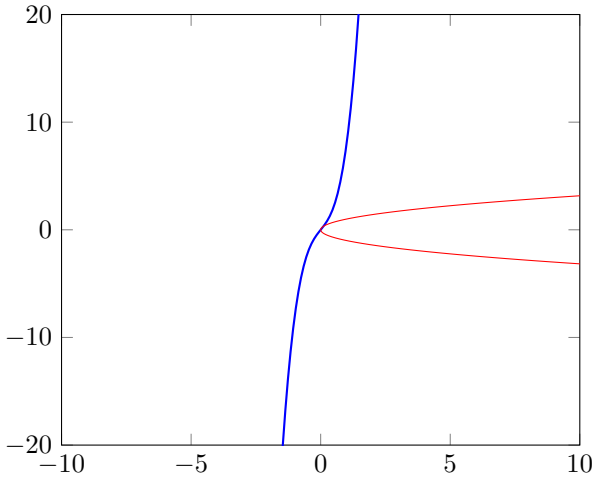

Figure 2: Eksempel på en tikz figur

```
  \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

5 Lister

I latex bruger man primært tre typer miljøer til lister. Alle typer skrives med et
`\begin {liste_type} \item ting 1 \item ting 2 \end {liste_type}` .
Alle listetyper kan indeholde lister, Latex sørger selv for, at listerne formateres korrekt. Koden, der laver listerne herunder, kan ses i listing 8

`itemize` En punktliste med prikker for at markere listeenheder

- Et punkt
- Et andet punkt
- Punkt med underliste
  - Et underlistepunkt
  - Et andet underlistepunkt

Enumerate En liste med tal for at markere listeenheder. Tallene bliver automatisk indsat. Man kan enda have en Enumerate inde i en Enumerate, hvor den selv ændrer tæller.

1. Et punkt
2. Et andet punkt
3. Punkt med underliste
   (a) Et underlistepunkt
   (b) Et andet underlistepunkt

`description` En liste med en beskrivelse af listepunkter, ud for hvert item skal der være en punktbeskrivelse `\item [beskrivelse]`

det første Et punkt
det andet Et andet punkt
det trejde Punkt med underliste
første underpunkt Et underlistepunkt
andet underpunkt Et andet underlistepunkt

```
\begin{itemize}
 \item Et punkt
 \item Et andet punkt
 \item Punkt med underliste  \begin{itemize}
     \item Et underlistepunkt
     \item Et andet underlistepunkt
    \end{itemize}
\end{itemize}

\begin{enumerate}
 \item Et punkt
 \item Et andet punkt
 \item Punkt med underliste \begin{enumerate}
     \item Et underlistepunkt
     \item Et andet underlistepunkt
    \end{enumerate}
\end{enumerate}

\begin{description}
 \item[det første] Et punkt
 \item[det andet] Et andet punkt
 \item[det trejde] Punkt med underliste \begin{description}
   \item[første underpunkt] Et underlistepunkt
   \item[andet underpunkt] Et andet underlistepunkt
  \end{description}
\end{description}
\end{description}
```

Listing 8: Forskellige typer af lister.

6 Tabeller

For at lave tabeller bruger man de to miljøer `\table` , `\tabular` , der fungerer som `\figure` , `\includegraphics` . Tabel 1 er lavet med koden i listing **??**. Vi starter ligesom i figure med at skrive `[h]` for at markere, at tabellen skal være her. `\centering` sætter tabellen i midten af dens *float*. Miljøet `\tabular` starter tabellen, i krølleparentersene, der står bagefter, fortæller vi kompileren hvor mange rækker, vores tabel fylder, og hvordan de skal adskilldes med | symbolet. Skrev vi `\tabular` |l|c r|, vil

tabellen have tre søjler, hvor teksten i den første står til venstre, i miden vil teksten være centreret, og den sidste vil stå til højre. Der vil også kun være en skillelinje mellem de to første søjler. I tabellen skriver vi elementerne ligesom i en matrice. Vi bruger `&` for at adskille elementer og `\\` for at adskille rækker. Vi kan bruge `\hline` til at lave en skillelinje mellem rækkerne.

søjle1	søjle2	søjle4	søjle4
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Table 1: Eksempel på manuelt skrevet tabel

```
\begin{table}[h]
\centering
\begin{tabular}{||c c | c c||}
 \hline
 søjle1 & søjle2 & søjle4 & søjle4 \\  \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{Eksempel på manuelt skrevet tabel}\label{table:data}
\end{table}
```

Listing 9: Eksempel på tabel

6.1 Automatiske tabeller

Har man en tabel i csv-form, f.eks. fra python eller excel, kan den inkluderes ved at sætte pakken `\usepackage {csvsimple}` i sin preamble. Det kan gøres ved at skrive:

	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 csc

```
\begin{table}[h]
  \centering\csvautotabular{assets/dices.csv}
  \caption{Automatic table from csc}\label{table:dices}
\end{table}
\caption{Eksempel på automatisk tabel }\label{lst:dices}
```

Listing 10: Code to generate automatic listing

7 Kodetekst

For at inkludere f.eks. pyton-kode i sit program kan det også enten gøres manuelt eller automatisk fra en fil. Ønkser vi f.eks. at inkludere en bid python i vores kode som vist i tabel: Koden kan ses her:

```
  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{Eksempel på automatisk tabel }\label{lst:dices}
```

Listing 11: Code to generate automatic listing

Ønkser man at læse kode fra en fil eller få bedre syntax highligting som ved eksemplernen på LaTeXi dette dokument, skal man bruge pakken *minted*. Det kræver en python-installation på systemet. Følg guiden på deres github: `https://github.com/gpoore/minted`

8 Referencer

I ens latex-kode kan man referre til sine egen figurer og tabeller ved at skrive `~\ref {ref:name}` , hvor *ref:name* er det, man har skrevet i dens label. Man kan bruge labels til sektioner, tabeller, figuerer og lignigner. Udover interne referencer kan man lave en bibliograpfifil, som indeholder referencer til videnskablige arktiler, hjemmesider, bøger, osv. En bibliograpfifil kan f.eks. hedde *references.bib* og have indhold som vist i listing **??** Når den fil er lavet kan man inkludere den i sin latex via: Ønsker jeg

```
 @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

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

Listing 13: Code to generate automatic listing

så at referere til artiklen med titlen *Attention Is All You Need*[4] skriver man `\cite {attention}` .

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.