Inhaltsverzeichnis

1 Grundlagen LaTeX

1.1 Einführung

Diese Übersicht soll das Schreiben nachträglich erleichtern. Für weitere Details schaue bitte in die .tex Datei hinein! LaTeX ist eine Sammlung von Makros die in **TeX** programmiert sind. Das zu Grunde liegende **TeX**-System ist eine Programmiersprache. Es gibt entsprechend für den Anwender viele Elemente einer Programmiersprache um den Textsatz zu gestalten. Mithilfe von *Variablen* können alle Format-Eigenschaften eines Dokuments programmatisch geändert werden. Hierbei gibt es verschiedene *Variablentypen*. Wichtig sind die *Längen-* und *Zählervariablen*. Diese Variablen können über LaTeX-Befehle verändert werden. Auf **TeX**-Ebene gibt es Kontrollstrukturen, also *if then* Anweisungen, *Schleifen* und noch vieles mehr.

1.2 Böse Sachen

Zuerst einmal Sachen die **NICHT** schön sind:

1. Manuelle Leerzeichen mit \\
2. Andauernd `\textbf{...}` verwenden. Lieber Betonungen (emph...)!

1.3 Struktur

Der Textkörper wird mit Abschnitts-Befehlen strukturiert. Diese unterscheiden sich nach Typ des Dokumentes (z.B. **article**). Jedoch gelten diese Befehle für alle:

- `\section`
- `\subsection`
- `\subsubsection`

Für **book** gilt wiederum:

- `\part`
- `\chapter`
- `\paragraph`
- `\subparagraph`

1.4 Zeichen

Folgende Zeichen können **nicht** einfach so eingetippt werden:

- & % $ # _ { } ~ ^

Diese können mit der Kombination aus dem \ Befehl und dem jeweiligen Zeichen erzeugt werden (*Escape-Befehl*). Um Leerzeichen zu forcieren kann `\@` benutzt werden. Ein Bindestrich kann mit `--` eingefügt werden. Mit dem Tilde Symbol kann ein gesperrtes Leerzeichen erzeugt werden (verhindert einen Zeilenumbruch).

1.5 Textformatierungen

LaTeX unterstützt folgende physische Textformatierungen:

- `\textbf{bold face}` – **bold face**
- `\textit{italics}` – *italics*
- `\textsl{slanted}` – *slanted*
- `\textsc{small caps}` – SMALL CAPS

1.6 Schriftgrößen

Folgende lokalen Schriftgrößenänderungen sind in LaTeX möglich:

1. `\tiny`
2. `\scriptsize`
3. `\footnotesize`
4. `\small`
5. `\normalsize`
6. `\large`
7. `\Large`
8. `\LARGE`
9. `\huge`
10. `\Huge`

Um die Schriftgröße für das gesamte Dokument zu ändern gibt es Präambel-Befehle.

1.7 Umgebungen

Eine Umgebung schließt einen Text mit `\begin{Umgebung}` und `\end{Umgebung}` ein. Innerhalb der Umgebung gelten eigene und spezielle Regeln. Wichtige Beispiele sind:

- enumerate (Aufzählung)
- itemize (Liste)
- equation (mathematische Gleichung)
- description (Lexikon ähnliche Aufzählung)

1.8 Listings

Listings enthalten nicht-formatierten Text, der wichtig zur Darstellung von Quellcode jeglicher Programmiersprachen ist.

```
int main(int argc, char* argv[])
{
    printf("Hello World!");
}
```

Mit den Optionen `\lstset{language=C}`, sowie `\begin{lstlisting}[frame=single]` können die Programmiersprache festgelegt und der Textrahmen aktiviert werden. Möchte man jedoch nur eine Textbox erstellen, so kann man dies auch mit dem `\fbox` Befehl tun.

1.9 Externe Dokumente

Mithilfe des `\input{file}` Befehls können weitere *.tex* Dateien in das Dokument so eingefügt werden, als würde der Inhalt der Datei im Originaldokument stehen. Die eingebundene Datei wird als normaler Teil des Quelltexts mitverarbeitet und sollte daher auch aus LaTeX Befehlen bestehen. Hierbei muss man allerdings beachten, dass die Präambel nur im Hauptdokument definiert werden darf (Pakete werden also nur im Hauptdokument geladen). Somit entfällt also auch die Kennzeichnung der Umgebung *document*. Die Pfadangaben können entweder *relativ* oder *absolut* sein. Den Unterdokumenten muss eine extra Zeile am Anfang eingefügt werden, die auf das Master Dokument hinweist. Alternativ kann im *TexMaker* das Master Dokument über das Menü Optionen eingestellt werden.

1.10 Längen

Mithilfe des `\setlength{variable}{länge}` Befehls können die in **TeX** auftretenden Längen verändert werden. **Alle Längen haben eine Einheit!** Wichtige Variablen sind zum Beispiel:

- `\parindent` - Definiert den Einzug eines Absatzes
- `\parskip` - Definiert den Abstand zwischen zwei Absätzen
- `\baselineskip` - Der Zeilenabstand

Anwendungsbeispiele:

1. `\setlength{\parindent}{10pt}` - Einzug auf 10pt setzen
2. `\setlength{\parskip}{0.5\baselineskip}` - Der Abstand zwischen zwei Absätzen wird auf die Höhe einer halben Zeile gesetzt.

Relative Längenangaben (siehe 2. Beispiel) beziehen sich auf einen Referenzwert. Dieser Bezug erfolgt mithilfe einer Multiplikation (in LaTeX wird dazu der Multiplikator einfach direkt an den Referenzwert geschrieben). Eine Addition geht nicht. Das wäre Stretching! *Absolute Längenangaben* werden mit Maßzahl und Einheit (**WICHTIG!**) direkt angegeben. Es ist sinnvoll **immer relative Längenangaben** zu nutzen, denn bei einer Änderung des Schriftgrads von 10pt auf bspw. 12pt ändert sich der Abstand zwischen den Absätzen genau richtig mit. Auch erweist sich dies bei der Breite von Bildern und bei der Spaltenweite von Tabellen als besonders hilfreich. Mithilfe des `\the` Befehls können Längen (Wert der Variablen) angezeigt werden. Es gibt viele verschiedene Längeneinheiten:

- Punkt - **pt** - entspricht: **1pt**
- Millimeter - **mm** - entspricht: **2.84pt**
- Zentimeter - **cm** - entspricht: **28.4pt**
- Inch - **in** - entspricht: **72.27pt**

Wichtiger Tipp:
Die wichtigsten Längen für die Gestaltung einer Seite können über den `\layout{}` Befehl angezeigt werden. Dazu muss aber vorher das Package *layout* mit `\usepackage{}` geladen werden. Interessante weitere Längen sind:

- `\columnsep`, `\columnwidth` - Bei der mehrspaltigen Gestaltung
- `\tabcolsep` - Für Tabellen
- `\itemsep` - Für Listen

Längen lassen außerdem über diverse Befehle verändern:

- `\setlength{\länge}{wert}` - Setzt eine Länge auf einen Wert
- `\addtolength{\länge}{wert}` - Addiert eine Länge zu einem Wert
- `\settowidth{\länge}{beispieltext}` - Breite des Beispieltexts auf `\länge` setzen
- `\settoheigth{\länge}{beispieltext}` - Höhe des Beispieltexts auf `\länge` setzen
- `\settodepth{\länge}{beispieltext}` - Tiefe des Beispieltexts auf `\länge` setzen

Bei letzeren Befehlen wird der Beispieltext jedoch **NICHT** angezeigt! Interessant ist, dass der Beispieltext selbst das Ergebnis eines Befehls sein kann.

Um ein schöneres Layout zu erreichen streckt LaTeX einige Größen. Hierzu zählen der Abstand zwischen Wörtern für Blocksätze und die Abstände zwischen den Absätzen um die Seite optimal auszunutzen. Wenn das nicht klappt, meckert LaTeX über eine *overfull* oder *underfull* `\vbox`. **Eigene Längen definieren:** Eigene Längen können mit dem Befehl `\newlength` definiert werden. Dies kann zum Beispiel bei den Spaltenbreiten von Tabellen im ganzen Dokument, oder für eine einheitliche Breite von Bildern (automatisch skaliert) nützlich sein. Die Verwendung geschieht dabei wie folgt:

```
\newlength{\imgWidth}
\setlength{\imgWidth}{0.8\textwidth}
```

Zuerst wird die Länge *imgWidth* mit `\newlength` erzeugt und dann mit den Längenänderungsbefehlen gesetzt. Alle Bilder im Text können mit dieser Variablen auf 80% der Textbreite skaliert werden. Um alle Bilder neu zu skalieren, muss nur die 0.8 geändert werden.

1.11 Befehle

1.11.1 Eigene Kommandos

Mit dem Befehl `\newcommand` lassen sich eigene Befehle definieren.

```
\newcommand{\name}{definition}
\newcommand{\zB}{z.\,B.\,\@}
\newcommand{\abk}{Abkuerzung}
```

In der hier gezeigten Form dienen sie als Abkürzung für häufig auftretende Befehle, die sonst umständlich einzutippen wären. Die fortgeschrittene Variante des `\newcommand` Befehls gebraucht **Argumente**. So wird der selbst-definierte Befehl zur Funktion.

```
\newcommand{\macheFett}[1]{\textbf{#1}}
```

In der Definition wird über #1 auf das erste Argument zugegriffen, über #2 auf das zweite Argument, etc. Dies ist für **maximal 9 Argumente** möglich! Möchte man bestehende Befehle verändern, so kann man dies mit dem `\renewcommand` Befehl tun. Die Syntax ist dabei die gleiche, wie bei `\newcommand`. Hierbei können auch **Argumente** übergeben werden!

```
\renewcommand{\name}[1]{definition}
\renewcommand{\em}{\bfseries}
```

Bei dem Beispiel mit dem `\em` geht allerdings durch die Erneuerung die Schalter-Logik verloren.

1.11.2 Zähler

Zähler sind wie **Längen** ein Variablen-Typ mit denen in LaTeX programmiert werden kann. Mit Zählern regelt LaTeX (fast) alle Verweise im Dokument. Wie bei **Längen** können eigene Zähler definiert werden. Diese können auch mit diversen Befehlen bearbeitet werden. Hierbei gibt es eine Liste vordefinierter Zähler:

- part
- chapter
- section
- subsection
- subsubsection
- paragraph
- subparagraph
- page
- equation
- figure
- table
- footnote
- mpfootnote

Für Listen gelten außerdem diese Zähler:

- enumi

- enumii
- enumiii
- enumiv

Beispiel:

```
\stepcounter{enumi}
\addtocounter{section}{zahl}
\setcounter{equation}{zahl}

\newcounter{numDoener}

\thenumDoener % Formatierter Text

\value{numDoener} % Wert unformatiert
```

Der Befehl `\the` zeigt die Zähler an (ohne \). Das Besondere an `\value` ist, ist dass der unformatierte Wert zum Gebrauch in Rechnungen genutzt werden kann. **Zählerdarstellung:** Mithilfe folgender Befehle kann die Darstellung von Zahlen mithilfe des `\renewcommand` Befehls verändert werden.

1. `\arabic` – 1. 2. 3. ...
2. `\roman` – i. ii. iii. ...
3. `\Roman` – I. II. III. ...
4. `\alph` – a. b. c. ...
5. `\Alph` – A. B. C. ...

Zur Änderung **aller** Aufzählungen des Dokuments wird `\theenumi` usw. in der Präambel neu definiert. Dies ist notwendig, da LaTeX anscheinend den Befehl indirekt zur Darstellung nutzt.

```
\documentclass{article}

\renewcommand{\theenumi}{\Roman{enumi}}

\begin{document}
    \begin{enumerate}
        ...
    \end{enumerate}
\end{document}
```

Zur Änderung einer einzigen Aufzählung wird `\theenumi` usw. direkt in der entsprechenden Aufzählung neu definiert.

```
\documentclass{article}

\begin{document}
    \begin{enumerate}

        \renewcommand{...}

        ...
    \end{enumerate}
\end{document}
```

1.12 Querverweise

Querverweise verweisen auf einen anderen Textteil. Diese Querverweise sind automatisiert und gebrauchen die Zähler. Dazu wird mit `\label` ein referenzierbarer Name erzeugt, der dann mit `\ref` gebraucht werden kann. Hierbei hat es sich im allgemeinen durchgesetzt, vor dem eigentlichen Labelnamen eine Bezeichnung einzufügen, die diese Verweisart beschreibt. Für Verweise auf Abschnitte schreibt man dazu einfach ein '**sec:**' vor den Labelnamen.

```
\section{Stand der Technik}
\label{sec:standTechnik}
.
.
.
Wie in Abschnitt ~\ref{sec:standTechnik}...
```

Das Label muss hierbei **direkt hinter dem referenzierten Objekt stehen!** Weitere Labelgruppen lauten wie folgt:

- sec:Abschnitt
- fig:Abbildung
- table:Tabelle
- eqn:Gleichung
- fn:Fußnote
- item:Aufzählungspunkt

Wenn an Stelle des Zählerwertes die Seite referenziert werden soll, auf der das Objekt steht, so gebraucht man den `\pageref` Befehl.

1.13 Gleitobjekte

LaTeX stellt zwei Umgebungen zur Verfügung, deren Position variabel ist:

1. figure
2. table

Diese Objekte gleiten durch den Text, d.h. die genaue Position wird erst beim Kompilieren festgelegt. Die Position hängt vom restlichem Layout ab und kann sich entsprechend mit einer Textänderung verschieben. Gleitobjekte sind also erst einmal nur Umgebungen mit einer variablen Postion, einer Beschriftung und einem Label. Der Inhalt ist völlig egal. Er kann Text, Formeln oder Bilder und Tabellen sein. Allerdings sind die Bildunterschriften an die Umgebung gebunden, d.h. eine figure-Umgebung wird mit 'Abb.' im Text gekennzeichnet. Die gewünschte Positionierung wird über ein optionales Argument übergeben. Die Positionen sind:

- **h**ere
- **t**op
- **b**ottom
- **p**age

Je nach Inhalt der Seite versucht LaTeX das Gleitobjekt in der angegebenen Reihenfolge der Positionen zu setzen (links nach rechts).

```
\begin{figure}[htbp]
    % Bild, Beschriftung, Referenz
\end{figure}
```

Bilder und Tabellen bekommen üblicherweise eine Bildunterschrift mit dem Befehl `\caption`. Diese Beschriftung geht in das Abbildungs- oder Tabellenverzeichnis ein (`\listoffigures` bzw. `\listoftables`). Optional kann `\caption` eine kürzere Beschriftung für die Verzeichnisse erhalten.

```
\begin{figure}[htbp]
    \fbox{Mein Bild in Text}
    \caption[Bild]{Textbild}
    \label{fig:tb1} % Referenz
\end{figure}
```

Wichtig: Bei den Gleitobjekten table und figure muss das Label hinter der `\caption` stehen! Mithilfe des `\centering` Befehls können sowohl die Abbildung als auch die Beschriftung mittig zentriert werden.

1.14 Grafiken

Für die Einbindung von Grafiken benutzen wir das Paket *graphicx*. Das Paket stellt den Befehl `\includegraphics` zur Verfügung. pdfLaTeX unterstützt dann 3 Grafik-Formate:

1. .pdf (Vektor)
2. .png (Raster)
3. .jpg (Raster)

Optimalerweise sollte eine Grafik so eingebunden werden:

```
\newlength{\imgWidth}
\setlength{\imgWidth}{0.9\textwidth}
...
\includegraphics[width=\imgWidth]{bild}
```

Der Befehl `\includegraphics` hat diverse Parameter:

```
width = xx
heigth = xx
keepaspectratio = (true/false)
scale = xx % als Faktor
angle = xx % in Grad
... und noch viele mehr.
```

Tipp: Möchte man den Kompiliervorgang und die Dateigröße verringern, so kann man in der `\documentclass` die Angabe *draft* hinzufügen.

1.15 Literatur

Mithilfe des Pakets *biblatex* und dem Tool *Zotero* oder *Citavi* kann ganz einfach eine Literaturverwaltung in LaTeX erfolgen. In den eckigen Klammern des `\usepackage` Befehls können dann Einstellungen vorgenommen werden, um z.B. den Zitationsstil zu ändern.

```
\usepackage[style = ieee, citestyle = ieee]{biblatex}
```

Anschließend kann in der *Präambel* mit dem `\addbibresource` Befehl eine *.bib* Datei aus *Zotero* eingebunden werden. Wichtig ist hierbei, dass bei dem Export die Zeichencodierung auf **UTF-8** eingestellt werden muss, da sonst Sonderzeichen/Umlaute in dem Literaturverzeichnis nicht angezeigt werden können. Nun kann an beliebiger Stelle auf Literatur verwiesen werden, indem man das Autorenkürzel (zu entnehmen aus *.bib* Datei) in den `\cite{Kürzel}` Befehl einfügt. Um das Literaturverzeichnis darzustellen, nutzt man den `\printbibliography` Befehl. Falls dann das Literaturverzeichnis nicht angezeigt wird, muss die Datei mehrmals kompiliert werden. Das Problem taucht auch auf, wenn noch kein Verweis im Text auf einen Eintrag in der *.bib* Datei getätigt worden ist. Es ist auch möglich, das Literaturverzeichnis in Unterkapitel oder Kategorien zu unterteilen. Beispielsweise kann so eine Kategorisierung nach Dokumententyp (Auszug aus Fachzeitschrift, Lehrbuch, etc.) erfolgen. Dazu fügt man dem `\printbibliography` Befehl etwaige Parameter zur Filterung der Literatur hinzu. **Sonstiges:** Fügt man dem *usepackage* Befehl den Parameter *backend = biber* hinzu, so wird der *biber* backend bibliography processor für *biblatex* geladen. *biber* behebt dabei einige Probleme von *bibtex* (richtige Sortierung mit Unicodeunterstützung, Speicherbedarf, Codierungen, etc.). Anschließend kann mithilfe der `\ExecuteBibliographyOptions{}` Anweisung zum Beispiel die Sortierung von Autor, Titel und Jahr verändert werden:

```
\ExecuteBibliographyOptions{
    sorting = nyt, % Sortierung Autor, Titel, Jahr
    bibwarn = true, % Probleme (Backend) anzeigen
    isbn = false, % Keine ISBN anzeigen
    url = false % Keine URL anzeigen
}
```

Zitate: Die Handhabung von Zitaten/Quotations gestaltet sich mithilfe des *csquotes* Pakets als einfach. Die Einbindung geschieht dabei wie folgt:

```
\usepackage[babel, german = quotes]{csquotes}
```

1.16 Farben

Farblicher Text kann besonders zur Hervorhebung eingesetzt werden. Um in LaTeX möglichst flexibel und leicht Textpassagen einfärben zu können, nutzt man das *xcolor* Paket. Dessen Befehl `\textcolor` kann unter Angabe der Farbe als ersten Parameter nun eine Textpassage einfärben. Die Farbe kann dabei eine der gängigen Farben (rot, gelb, grün, blau, usw.) sein. Möchte man eine größere Farbpalette nutzen, so kann man auch auf den RGB Farbraum zurückgreifen. Dazu definiert man sich einfach eine neue Farbe wie folgt:

```
\definecolor{farbe1}{RGB}{0, 255, 255}
```

Um die Seitenfarbe zu ändern nutzt man die Anweisung `\pagecolor`. Eine Farbe kann auch mit dem Befehl `\color` standardmäßig festgelegt werden (Default Schriftfarbe).

1.17 Tabellen

Für Tabellen wird das Gleitobjekt *table* verwendet. Das heißt, Tabellen können auch referenziert werden (mithilfe des *label*) und bekommen eine Abbildungsunterschrift. Dadurch sind sie dann auch in der Tabellenliste enthalten (`\listoftables`). Auch wird hier die Position über die Attribute *[htbp]* festgelegt. Die Tabellen-Umgebung **erzeugt selbst keine Tabelle!** Das Erzeugen einer Tabelle entsteht durch das Einfügen einer Umgebung in die *table*-Umgebung. Diese Umgebung heißt *tabular*. Die *tabular* Umgebung hat ein erforderliches Argument, welches das Aussehen der Tabelle festlegt. Jeder Buchstabe in diesem Argument steht für eine Spalte. Dadurch wird die Anzahl der Spalten festgelegt. Die Spalte kann links- oder rechtsbündig orientiert oder zentriert sein (l, r oder c). Mithilfe von zwei Zeichen kann so eine Tabelle aufgebaut werden. Das Symbol & zeigt eine neue **Zelle** an und \\ erzeugt eine neue Zeile. Das sieht dann so aus:

```
\begin{table}[htbp]
    \centering
    \begin{tabular}{lcr}
        Messung & Spannung & Strom \\
        1 & 2 & 3 \\
        4 & 5 & 6
    \end{tabular}
    \caption[Messreihe]{Induktionsspannung}
    \label{tab:messreihe}
\end{table}
```

Letzendlich sieht die Tabelle dann so aus:

Messung	Spannung	Strom
1	2	3
4	5	6

Tabelle 1: Induktionsspannung

1.17.1 Gestaltung der Tabelle

Standardmäßig setzt LaTeX die Tabelle so breit, wie der Inhalt ist. Dabei findet, wenn erforderlich, kein Umbruch statt! Jedoch ist es möglich, die Breite einer Spalte mit dem Argument `p{breite}` vorzugeben. Dann wird der Inhalt immer linksbündig umgebrochen. Am schönsten ist es hier natürlich, sich für die Spaltenbreite eine eigene Länge zu definieren. Die Zwischenräume zwischen den Zellen können auf zwei Weisen angepasst werden:

1. Mithilfe der Länge `\tabcolsep` (Verändert **alle** Spalten!)
2. Mithilfe von `@{zwischenraum}`

Der Befehl `@{zwischenraum}` wird im Argument von *tabular* verwendet. Damit kann für jede Spalte individuell der Zwischenraum angepasst werden.

```
...
\begin{tabular}{c@{\hspace{1cm}}cc}
...
\end{tabular}
...
```

Der Befehl `\hspace` erzeugt einfach einen horizontalen Freiraum. Um die Tabelle strukturiert aussehen zu lassen, werden häufig Gitterlinien eingesetzt. Diese sind in wissenschaftlichen Arbeiten **rein horizontal**. Möchte man trotzdessen vertikale Trennstriche in die Tabelle einfügen, so kann man dies mit dem Pipe-Symbol im Argument machen (z.B. |c|c|c). Hierfür wird das Paket *booktabs* benötigt. Das Paket stellt gutaussehende horizontale Linien zur Verfügung. Es werden drei Befehle mitgeliefert:

1. `\toprule` (Trennung oberhalb der Überschrift)
2. `\midrule` (Trennung zwischen Überschrift und Inhalt)
3. `\bottomrule` (Tabellenunterstrich)

1.17.2 Formatierungen

Das Paket *array* stellt weitere Befehle zur Formatierung der Spalten zur Verfügung. Diese Formatierungen werden auch direkt in das Argument von *tabular* eingesetzt. Zum einen bietet *array* Prä- und Suffixe innerhalb des *tabular*-Argumentes an. Damit kann jede Spalte einzeln formatiert werden. Es ist möglich fast alle Formatierungsbefehle (`\textbf`, `\textsc`, etc.) zu verwenden. Dadurch, dass jede der Formatierungen innerhalb der eigenen Zelle eingesperrt ist, können auch Formatierungen mit Schalter-Logik verwendet werden. Ein Beispiel mit **Präfixen** sieht dann so aus:

```
...
\begin{tabular}{>{\it}c >{\bf}c >{\sl}c}
...
\end{tabular}
...
```

Suffixe sind etwas komplizierter:

```
...
\begin{tabular}{c||<{\,MW} |<{\,MWh}}
...
\end{tabular}
...
```

Auffallend ist, dass das Positionsargument bei Präfixen rechts steht und bei Suffixen links. Weiter bietet *array* neben den standardmäßigen Spaltentypen l, c, r und `p{}` und den Prä- und Suffixen auch zwei neue Spaltentypen. `m{breite}` und `b{breite}` sind jeweils linksbündig und umbrechend. Jedoch ist `m` vertikal zentriert und `b` ist vertikal nach oben zentriert. Für die horizontale Orientierung müssen allerdings neue Befehle her. Neue Spaltentypen können mit dem Befehl `\newcolumntype` erzeugt werden. Dies kann dann so aussehen:

```
\newcolumntype{L}[1]{>{\raggedright\hspace{0pt}}p{#1}}
```

Hierbei wird ein neuer Spaltentyp **L** erzeugt, der 1 Argument entgegennehmen kann. Der eigentliche, maskierte Spaltentyp `p{}` wird um einen `\hspace` erweitert, der die Silbentrennung aktiviert. Gleichzeitig wird mit dem `\raggedright` Befehl die Bündigkeit auf links gesetzt (horizontale Orientierung). Der nun erzeugte Spaltentyp kann dann wie folgt eingesetzt werden:

```
\begin{table}[htbp]
    ...
    \begin{tabular}{cL{50pt}cL{50pt}cL{50pt}}
    ...
    \end{tabular}
    ...
\end{table}
```

Um die genannten Befehle innerhalb nur einer Zelle zu verwenden, können einfach neue Befehle definiert werden, die dann innerhalb dieser Zelle angewendet werden.

```
% Tabellenabschnitt linksbuendig
\newcommand{\ltab}{\raggedright\arraybackslash\hspace{0pt}}
% und fuer links und rechts
```

Hier ist eine sehr schöne Tabelle aus der Vorlesung: Manchmal möchte man zwei Spalten zu einer zusammenfassen. Dies ist mit dem `\multicolum` Befehl möglich. Der Befehl besitzt 3 Argumentenfelder:

1. Anzahl der Spalten

#	**N**ame	**L**eistung in MW	**S**peicher in MWh
1	Gemasolar	20 MW	300 MWh
2	Valle1	50 MW	375 MWh

Tabelle 2: Liste von Solarkraftwerken in Spanien

2. Ausrichtung (l, c, r)
3. Inhalt

Der Befehl steht dann anstelle der gleichen Anzahl an Stellen mitten in der Tabelle:

```
\begin{table}[htbp]
    \centering
    \begin{tabular}{clll}
        & & \multicolumn{2}{l}{Technische Daten} \\

        \# & \bf Name & \bf Leistung in & \bf Energie in \\
    ...
    \end{tabular}
    ... \end{table}
```

Natürlich gibt es zur Tabellenerstellung viele weitere Pakete und Möglichkeiten. Hier sind ein paar aufgelistet:

- longtable: Tabellen über mehr als eine Seite
- tabularx: Hilfestellung bei der Berechnung von Breiten
- ltxtable: Kombination aus den oberen Beiden
- rotating: gedrehte Tabellen
- multirow: mehrere Zeilen (ähnlich zu multicolumn)

Im Internet gibt es außerdem Tabellen-Rechner:

- Truben Table Editor
- Tables Generator

1.18 Formeln

Die Darstellung von mathematischen Formeln ist einer der Grundpfeiler für LaTeX. *amsmath* und *amssymb* sind die standard Pakete, die zur mathematischen Darstellung zusätzlich geladen werden. AMS steht dabei für *American Mathematical Society*.

1.18.1 Die math-Umgebung

Die *math-Umgebung* schaltet LaTeX in den Mathe-Modus. Innerhalb dieser Umgebung gelten ganz andere Formatierungsregeln als im Text. Zur Erzeugung gibt es zwei Möglichkeiten:

- Erstellung im Fließtext mit $...$
- Freistehende Gleichung mit diversen Umgebungen

1.18.2 Equation Umgebung

Eine einzelne Gleichung wird mit der Umgebung *equation* angegeben. Die Gleichung wird dann nicht im Fließtext sondern freistehend formatiert. In dieser Umgebung werden die Gleichungen automatisch durchnummeriert. Mit der equation* Umgebung kann dies vermieden werden.

$$n_1 \sin \alpha = n_2 \sin \beta \tag{1}$$

1.18.3 Eigenheiten

Ein Leerzeichen wird als Multiplikation interpretiert. Alle Buchstaben werden als Variablen interpretiert. Um einen Index zu erzeugen, schreibt man hinter die Variable einen Unterstrich mit dem entsprechenden Index. Folgende Zeichen können direkt in eine equation-Umgebung eingegeben werden: +-=!/()[]<>|':. Eine Multiplikation kann mit den folgenden Befehlen explizit hervorgerufen werden:

- `\cdot`
- `\times`

In der Umgebung werden griechische Buchstaben einfach ausgeschrieben. Hierbei ist zu beachten, dass große griechische Buchstaben groß geschrieben werden müssen. Auch für die gängigen Funktionen gibt es explizite Befehle. Diese müssen genutzt werden, da sonst die Buchstaben des Funktionsnamen als Variable interpretiert werden. Dadurch sind die Funktionsnamen dann auch nicht mehr kursiv! Um eine Hoch- oder eine Tiefstellung zu erzeugen, nutzt man die Befehle `_` (tiefgestellt) und `^` (hochgestellt). Das Argument muss hierbei mit geschweiften Klammern geklammert werden, da sonst z.B. nicht alle Variablen im Exponent landen würden. **Weiter auf Seite "Hoch- und Tiefgestellt, wo die 4 Formeln zu sehen sind!**

1.19 Sonstiges

Hier findest du sonstige Anweisungen und anderes nützliches Wissen:

- * Befehle lassen die Aufzählung verschwinden (z.B. `\section*{}`)
- `\tableofcontents` erzeugt ein Inhaltsverzeichnis
- `\noindent` Verhindert das automatische Einrücken der ersten Zeile
- `\,` erlaubt einen geringen Abstand (z.B. zu Einheiten)
- `\raggedleft` und `\raggedright` verändern die Bündigkeit
- Das hyperref Paket erlaubt die Nutzung von Links im Dokument
  - Mithilfe des `\href` Befehls kann eine Umleitung erfolgen
- `\addsec` ist ein **KOMA**-Befehl und erlaubt das Hinzufügen einer *section* ohne Nummerierung in das Inhaltsverzeichnis.