Fachhochschule Nordwestschweiz
Hochschule für Technik

Der Einstieg in LaTeX

Vorlage und Beispiele

Windisch, 13.08.2018

Hochschule	Hochschule für Technik - FHNW
Studiengang	Elektro- und Informationstechnik
Autor	Patrick Studer und Hanspeter Schmid
Betreuer	Anita Gertiser
Auftraggeber	Sebastian Gaulocher
Version	1.2

Zusammenfassung

Das Abstract ist eine Art Zusammenfassung des ganzen Dokuments. Es gibt einen Einblick in die Aufgabenstellung, wie diese umgesetzt wurde und welches Ergebnis erreicht wurde. Aus diesem Grund wird das Abstract immer ganz am Schluss der Arbeit verfasst. Es besteht aus einem zusammengehörenden Absatz und umfasst ungefähr 10 bis 20 Zeilen. Formeln, Referenzen oder andere Unterbrechungen haben im Text nichts zu suchen. Direkt unter dem Abstract folgt eine Liste von drei bis vier Stichworten/Keywords. Diese werden in alphabetischer Reihenfolge aufgelistet und beschreiben das Themengebiet der Arbeit.

Keywords: Anleitung, LaTeX, Thesis, Vorlage

Bitte schicken Sie jeglichen Feedback auch an `hanspeter.schmid@fhnw.ch`
Er wird dieses Template langfristig unterhalten.

Inhaltsverzeichnis

1 Grundlagen

Diese Vorlage soll den Studenten im Studiengang Elektro- und Informationstechnik den Einstieg in LaTeX vereinfachen und anhand von Beispielen **(siehe direkt im tex-File)** einige Tipps und Tricks auf den Weg geben. Zudem wird auch auf Vorgaben eingegangen, welche für eine technische Dokumentation wichtig sind und sicherstellen, dass alle Dokumentationen des Studiengangs gleich designed werden. Für LaTeX-Neulinge wird das Dokument *LaTeX2e-Kurzbeschreibung* (siehe **l2kurz**) empfohlen. Die elektronische Datei ist im Ordner */bibliography/* zu finden. Daraus stammt auch das folgende Zitat:

> Typographisches Design ist ein Handwerk, das erlernt werden muss. Ungeübte Autoren machen dabei oft gravierende Fehler. Fälschlicherweise glauben viele Laien, dass Textdesign vor allem eine Frage der Ästhetik ist – wenn das Schriftstück vom künstlerischen Standpunkt aus „schön“ aussieht, dann ist es schon gut „designed“. Da Schriftstücke jedoch gelesen und nicht in einem Museum aufgehängt werden, sind die leichtere Lesbarkeit und bessere Verständlichkeit wichtiger als das schöne Aussehen.
> **l2kurz**

1.1 Teamfähige Installation

Bei der Arbeit mit LaTeX empfiehlt es sich sehr, dass alle im Team dieselbe Version installiert haben. Am einfachsten ist das mit einer Installation von `https://tug.org/texlive/`. Dies ist eine nahezu vollständige Installation von allem was es braucht, so aufbereitet, dass die Dateiversionen zusammenpassen, und lauffähig auf Linux, MacOS und Windows.

1.2 Grundgerüst

Um ein LaTeX-Dokument zu erstellen braucht es eigentlich nicht viel, nur eine Textdatei mit der Dateiendung `.tex`, welche mit dem Befehl `\documentclass[]{}` beginnt und die Umgebung `\begin{document}` ... `\end{document}` enthält. Darin kann nun die ganze Dokumentation erstellt und kompiliert werden, jedoch wird das schnell unübersichtlich und eignet sich sehr schlecht für das gemeinsame Arbeiten am Dokument.

Durch das „Outsourcen“ von zusammengehörenden Blöcken (z. B. ganze Kapitel) in externe Dateien kann eine **übersichtliche und teamfähige Struktur** geschaffen werden. Damit sie bei der Kompilation des Dokuments beachtet werden, müssen sie in der Masterdatei eingebunden werden. Einerseits wird der Befehl `\input{}` dazu verwendet, Texte **1:1** aus einer Datei einzubinden. Dies kann beispielsweise dazu dienen, die Dokumenteinstellungen und Packages aus einer externen Datei (hier `header.tex`) zu laden.[1] Andererseits können ganze Kapitel mit `\include{}` eingebunden werden. Dieser Befehl fügt ein zusätzliches `\clearpage` **vor** und **nach** dem geladenen Text hinzu. Dadurch wird sichergestellt, dass ein Seitenumbruch am Anfang und Schluss gesetzt und die Gleitobjekt-Warteschlange geleert wird.[2] Zu beachten ist, dass der `\include{}`-Befehl nur in der Masterdatei benutzt wird, da er nicht verschachtelt werden kann.

In Abbildung 1.1 ist die Struktur des aktuellen LaTeX-Projekts aufgezeigt.
Diese Struktur ist als Minimalstruktur zu verstehen und ist für Berichte des Studiengangs EIT vorgeschrieben!

[1] Beim Kompiliervorgang wird das aktuelle `.aux`-File weitergeführt.
[2] Für das eingefügte Kapitel wird ein separates `.aux`-File erzeugt.

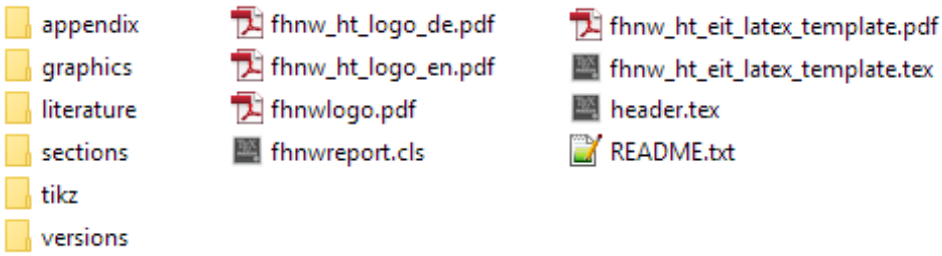


Abbildung 1.1: Minimalstruktur des LaTeX-Projekts.

Die Ordner sind für folgende Dateien vorgesehen:

`appendix/`	im Anhang eingebundene Dokumente oder Bilder
`graphics/`	Grafiken (vorzugsweise als Vektorgrafik – pdf)
`literature/`	zitierte Dokumente, Paper, Webseiten und die Bibliographie-Datenbank
`sections/`	mit `\include{section.tex}` eingefügte Kapitel
`tikz/`	tikz-Grafiken (Standalone-Modus) zum Einbinden mit `\input{tikz.tex}`
`versions/`	Ablage für verschiedene Dokumentversionen (z. B. bei Abgabe)

1.3 Packages

Packages können dem Benutzer helfen, gewisse Dinge einfacher zu Programmieren, indem vorgefertigte Funktionen geladen werden. Dies ist zwar nützlich, sollte jedoch mit Vorsicht genossen werden. **Unterschiedliche Packages können einander gegenseitig beeinflussen und dadurch unerklärliche Fehler verursachen. Aus diesem Grund empfiehlt sich, nur wirklich verwendete Packages dem Header hinzufügen!** Auch bei Beispielen aus dem Internet ist immer etwas Vorsicht geboten. Viele Examples binden unnötige Packages ein. Am besten also alle ungebrauchten Packages im Header auskommentieren und erst wieder einkommentieren, wenn der Kompiler meckert, dass er ein Befehl nicht kennt.

1.4 Gliederung

Die `fhnwreport`-Klasse basiert auf der LaTeX-Klasse `article`. Für die Gliederung können drei nummerierte und eine nicht nummerierte Ebenen verwendet werden. Da unsere Berichte nicht mehrere Hundert Seiten beinhalten, sind Kapitelseiten (`\chapter{}`) nicht vorgesehen. Somit wird mittels `\section{}` die oberste Gliederungsebene angesprochen. Mittels `\subsection{}` resp. `\subsubsection{}` werden die nächsten zwei Ebenen definiert.

Das ist ein Paragraph
Ein Paragraph kann mit dem Befehl `\paragraph{}` erstellt werden und wird nicht im Inhaltsverzeichnis aufgelistet. Es setzt nur einen fetten Titel und beginnt danach auf einer neuen Zeile.

1.4.1 Tiefe des Inhaltsverzeichnis beschränken

Das Inhaltsverzeichnis ist durch die Klasse auf eine Tiefe von drei Ebenen eingestellt. Nun kann es vorkommen, dass man in einem Kapitel die unterste Ebene ausblenden möchte, jedoch nicht auf die nummerierten Überschriften verzichten möchte. Dies kann einfach gemacht werden, indem man mit dem Befehl `\addtocontents{toc}{\protect\setcounter{tocdepth}{2}}` die Tiefe auf Ebene 2 setzt. Die darauf folgenden Überschriften der 3. Ebene sind dann nicht mehr im Inhaltsverzeichnis aufgelistet.

1.4.2 Dieser Titel ist nicht im Inhaltsverzeichnis

Am Schluss muss jedoch die Tiefe wieder zurück auf die Standardeinstellung gesetzt werden (`\addtocontents{toc}{\protect\setcounter{tocdepth}{3}}`), damit im nächsten Kapitel wieder alle Gliederungsebenen angezeigt werden.

1.5 Kommentare

Möchte man Kommentare setzen kann dies einfach mit dem Package `todonotes` realisiert werden. Die Kommentare können mit dem Befehl `\todo{}` am Seitenrand oder für längere Kommentare mit `\todo[inline]{}` direkt im Text angeordnet werden.

Todo am Seitenrand

Todo im Text

Das Wichtigste jedoch ist, vor Abgabe alle Todo's abgearbeitet zu haben. Dabei hilft ein weiteres schönes Feature – die `\listoftodos`. Sie bietet eine Übersicht über alle definierten Todo's. Ähnlich wie beim Inhaltsverzeichnis braucht diese Liste zwei Kompilierdurchgänge. Aktuell wird sie nur erstellt, solange das Dokument im `mode=draft` ist (siehe am Schluss des Hauptdokuments).

1.6 Verweise

Verweise helfen dem Leser, Zusammenhänge zwischen verschiedenen Themen besser zu erkennen. Die Verweise auf andere Kapitel wie z. B. (siehe Kapitel 1.5) können nach eigenem Ermessen platziert werden. **Verweise auf Abbildungen und Tabellen sind jedoch Pflicht!** Das Dokument darf keine Bilder oder Tabellen enthalten, ohne dass im dazugehörigen Abschnitt darauf referenziert wird.

1.6.1 Label setzen

Um auf etwas verweisen zu können, muss zuerst ein Label gesetzt werden. Dies wird mit dem Befehl `\label{}` gemacht. Den Befehl setzt man direkt hinter Überschriften oder hinter die Beschreibung (`\caption{}`) von Abbildungen und Tabellen. Die Funktion von `\label{}` ist ganz einfach: der Befehl speichert die letzte automatisch erzeugte Nummer ab, was immer das auch war. Deshalb ist im obigen Text das Wort **nach** ganz besonders wichtig.

Grundsätzlich können die Labels beliebig benannt werden. Jedoch ist es für die Teamarbeit oder auch für eure Nachfolger extrem hilfreich, wenn ein einheitliches Namenssystem eingehalten wird. **Deshalb gilt für Dokumentationen des Studiengangs EIT folgende Richtlinie für die Vergabe von Labelnamen:**

`\label{sec:name}`	für Kapitel (sections)
`\label{fig:name}`	für Bilder (figures)
`\label{tab:name}`	für Tabellen (tables)
`\label{equ:name}`	für Formeln (equations)
`\label{app:name}`	für Objekte im Anhang (appendix)

1.6.2 Auf Labels referenzieren

Wurde erstmal ein Label gesetzt, kann von einer beliebigen Stelle im Dokument darauf referenziert werden. Dazu kann man zwei Befehle nutzen: `\ref{}` und `\pageref{}`. Zu beachten ist, dass diese Befehle nur eine Zahl (Index oder Seite) des Objektes zurückgeben. Damit die Formelreferenz automatisch in runde Klammern gesetzt wird, bietet sich der Befehl `\eqref{}` an. Referenzen werden erst nach **zwei Kompilierdurchgängen** richtig angezeigt.

1.7 Abstände, Bindestriche und Silbentrennung

Die wichtigsten Befehle für horizontale Abstände sind:

`~`	Leerschlag mit unterdrücktem Zeilenumbruch
`\,`	Kleiner Abstand
`\;`	Mittelgrosser Abstand
`\!`	Den Abstand verkürzen (kann auch mehrmals angewendet werden: zum Beispiel)
`\quad`	Abstand so breit wie ein Buchstabe hoch ist
`\qquad`	Doppelter `\quad`
`\enspace`	So breit wie eine Ziffer (123 56 8)
`\phantom{}`	So breit wie der übergebene Text (Das ist ein -Beispiel)
`\hspace{}`	So breit wie man will (Das sind 2 cm Abstand)
`\hfill`	Abstand, bis die Zeile voll ist
`\dotfill`	Punkte, bis die Zeile voll ist
`\hrulefill`	Linie, bis die Zeile voll ist

Die wichtigsten Befehle für vertikale Abstände sind:

`\smallskip`	Kleiner Abstand
`\medskip`	Mittlerer Abstand
`\bigskip`	Grosser Abstand
`\parskip`	Definierter Absatzabstand
`\vphantom`	So hoch wie der übergebene Text (hilfreich in Formeln)
`\vspace{}`	So hoch wie man will
`\vfill`	Abstand, bis die Seite voll ist

Für Minuszeichen, Bindestriche und Gedankenstriche wird in Latex das selbe Symbol verwendet:

`$-$`	Mathematisches Minus: $1 - 2 = -1$
`-`	Bindestrich: O-Beine, AD-Wandler
`"~`	Bindestrich der nicht unterbrochen werden darf
`--`	Gedankenstriche: Ja – oder doch nein?
	Bis/Nach: 11–18 Uhr, Zürich–Paris
	Versus: Basel – Zürich
`---`	Langer Gedankenstrich ohne Abstand (nur im Englischen): yes—or no?

Möchte man die automatische Silbentrennung von LaTeX beeinflussen helfen folgende Befehle:

`\-`	Dieses Wort darf nur an dieser Stelle
	oder nach einem Bindestrich getrennt werden
`"-`	Diesem Wort eine zusätzliche Trennstelle hinzufügen
`\mbox{}`	Dieser Satz wird weder umgebrochen, noch werden die Wörter getrennt
`\hyphenation{}`	Die aufgelisteten Wörter dürfen im gesamten Dokument
	nur an den mit `-` markierten Stellen getrennt werden

1.8 Zahlen und Einheiten

Um die Abstände zwischen Zahlen und deren Einheitsangabe richtig darzustellen bietet sich das Package `siunitx` an. Es hilft bei Angaben von Einheiten, Zahlenbereichen oder Zehnerpotenzen.

Die Zahlen `1234` und `1.234e3` können mit `\num{}` angezeigt werden. Für Winkel wie 45° oder 60°20′10″ verwenden wir `\ang{}`. Für alleinstehende Einheiten wie `km/h` oder `m/s^2` gibt es den `\si{}`-Befehl, und schlussendlich bildet die Kombination aus `\num{}` und `\si{}` den Befehl `\SI{}{}`.

Beispiele: 1234 ist gleichviel wie 1.234×10^3. $2\,\pi$ sind 360°0′0″. 1, 2, 3, 4 und 5 sind Zahlen von 1 bis 5. $\mathrm{kg\,m/s^2}$ sind $\mathrm{kg\,s^{-2}}$. $112\,\mathrm{km/h}$ oder $112\,\mathrm{km\,h^{-1}}$.

1.9 Sprache umstellen

Man kann an einer beliebigen Stelle im Dokument die Sprache wechseln. Dies bewirkt, dass automatisch generierte Texte in der jeweiligen Sprache eingefügt werden. Der Befehl dazu lautet `\selectlanguage{ngerman}` für Deutsch und `\selectlanguage{english}` für Englisch.

2 Gleitobjekte (floats)

Im Normalfall werden Objekte nicht genau dort im Text platziert, wo sie inhaltlich auch hingehören. LaTeX versucht viele typographische Regeln zu erfüllen, um ein „korrektes“ Schriftbild zu erreichen. Diese Regeln schaffen etwas Freiraum und erhöhen die Lesefreundlichkeit. Deshalb werden Tabellen und Abbildungen automatisch an die nächstmögliche geeignete Stelle „gleiten“ und nachfolgende Objekte vor sich her schieben (quasi eine Warteschlange), bis das Dokument zu ende ist oder der Befehl `\clearpage` eingegeben wird.

Um so eine Gleitumgebung zu schaffen stellt LaTeX die beiden Umgebungen `figure` und `table` zur Verfügung. Bei Blocksatz ist es also so, dass Bilder und Tabellen am oberen oder unteren Blattrand, oder auf einer eigenständigen Seite (ohne Text) platziert werden. Dies, weil der Float-Specifier ohne Angabe auf `[tbp]` gesetzt ist (top, bottom, page). **Dies sollte immer die erste Wahl sein.**

Hat man jedoch sehr viele Bilder oder ist mit den automatisch gesetzten Bildpositionen nicht einverstanden, so ist eine Nachbearbeitung ganz am Schluss sinnvoll. Durch Setzen von `[t]` oder `[b]` kann nach Belieben eingegriffen werden. Sollte das immer noch nicht reichen können durch ein zusätzliches `!` (bang) die LaTeX-Regeln für den Mindestanteil an normalem Text pro Seite umgangen werden.

2.1 Abbildungen (figure)

Eine Abbildung beginnt mit `\begin{figure}` und endet mit `\end{figure}`. Dazwischen stehen Anweisungen für das Hinzufügen der Grafik, der Bildunterschrift und des Labels für die Referenzierung. Oft wird der Parameter `size=faktor` verwendet. Dieser hat jedoch den Nachteil, dass er abhängig von der Originalgrösse der Grafik ist. Besser ist die Verwendung des Parameters `width`. Dazu folgendes Beispiel, welches der Abbildung 2.1 entspricht:

```
\begin{figure}[b]
  \centering
  \includegraphics[width=0.9\linewidth]{beispiel.pdf}
  \caption{Dies ist ein Beispiel für eine Abbildung.}
  \label{fig:Figure}
\end{figure}
```

2.1.1 Subfigures

Hat man mehrere kleine Bilder (die irgendwie zusammengehören), kann man diese platzsparend in einer Subfigure anordnen. In Abbildung 2.2 auf Seite 7 sieht man ein kleines Beispiel von Subfigures. Für diese wird das Package `subfig` verwendet.

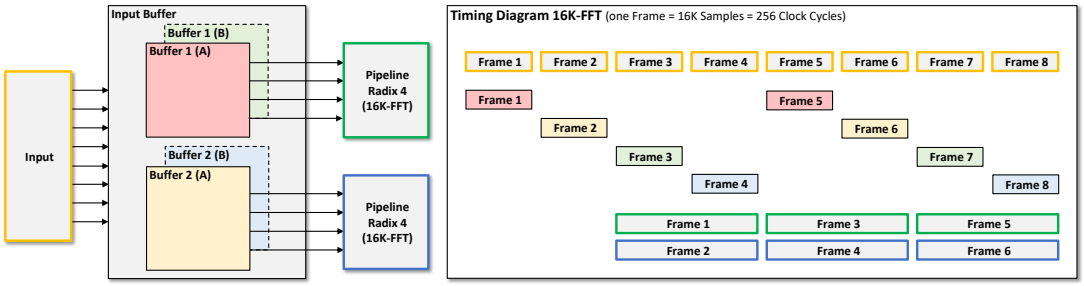


Abbildung 2.1: Dies ist ein Beispiel für eine Abbildung.

(a) Bild 1 **(b)** Bild 2 **(c)** Bild 3

Abbildung 2.2: Ein einfaches Beispiel für eine Abbildung mit mehreren Bildern.

2.2 Tabellen (table)

Eine Tabellen beginnt mit `\begin{table}` und endet mit `\end{table}`. Dazwischen stehen Anweisungen für das Erstellen der eigentlichen Tabellenanordnung, die Tabellenbeschreibung und ein Label für die Referenzierung. Dazu ein Beispiel, welches der Tabelle 2.1 entspricht:

```
\begin{table}[b]
  \centering
  \begin{tabular}{C{1cm} C{2.5cm} C{2cm}|C{2cm} C{2.5cm} C{1cm}}
  \multicolumn{3}{c|}{\textbf{Normally Ordered Input}}
    & \multicolumn{3}{c}{\textbf{Digit-Reversed Output}} \\
  {Index}& {Base 2} & {Base 4}& {Base 4} & {Base 2} & {Index}\\ \hline\hline
    60   & 11 11 00 &   330   &   033    & 00 11 11 &   15   \\
    61   & 11 11 01 &   331   &   133    & 01 11 11 &   31   \\
    62   & 11 11 10 &   332   &   233    & 10 11 11 &   47   \\
    63   & 11 11 11 &   333   &   333    & 11 11 11 &   63   \\
  \end{tabular}
  \caption{Das erste Beispiel für eine Tabelle}\label{tab:DigitReverse}
\end{table}
```

Bei Tabellen gilt grundsätzlich: „Weniger ist manchmal mehr“. Somit sollte man auf möglichst viele durchgezogene Linien verzichten. Vor allem die äussere Umrandung ist oft überflüssig. Nur wenn Linien dazu gedacht sind, Gruppen zu trennen oder zusammengehörige Blöcke zu kennzeichnen, dann sollen sie eingefügt werden.

Oft hat man im ersten Bericht keine Zeit für die saubere Gestaltung mit LaTeX. Da bietet sich die Möglichkeit, Tabellen im Excel zu erstellen und diese als Bild mit `\includegraphics{}` zwischen die `table`-Umgebung zu setzen.

Normally Ordered Input			**Digit-Reversed Output**		
Index	Base 2	Base 4	Base 4	Base 2	Index
60	11 11 00	330	033	00 11 11	15
61	11 11 01	331	133	01 11 11	31
62	11 11 10	332	233	10 11 11	47
63	11 11 11	333	333	11 11 11	63

Tabelle 2.1: Das erste Beispiel für eine Tabelle.

Weiter gilt auch, dass sehr grosse Tabellen (z. B. Messprotokolle, Portmaps, ...) im Anhang abgelegt werden können. Falls notwendig kann ein kleiner, aussagekräftiger Ausschnitt davon im Dokument eingefügt werden.

Tabellen, welche eine komplizierte Formatierung aufweisen (z. B. Projektpläne), können im Standalone-Modus in eine eigenständige Datei geschrieben werden und dann eingefügt werden oder man nimmt die vorhandene Excel-Tabelle und wandelt sie in ein PDF und fügt sie ebenfalls als ganze Seite dem Anhang hinzu.

Weitere Tabellen-Beispiele sind in Tab. 2.2 gezeigt.

Parameter	**Structure** (row vector)	**Description**
`W`	`isSigned`	1 = signed, 0 = unsigned
	`WordLength`	Total word length of output (incl. sign bit)
	`FracLen`	Number of fractional bits
`options`	`Scaling`	Number of decimal places to shift. For more information read the manual.
	`RoundingMethod`	0 = Nearest, 1 = Ceiling, 2 = Convergent
	`OverflowAction`	0 = Saturate, 1 = Wrap
`reporting`	`ShowOuputString`	Displays information to each conversation
	`CheckOverflow`	Enable the overflow check

Measures	Task	Method 1				Method 2				Method 3				p-value
		1	2	3	4	1	2	3	4	1	2	3	4	
Quality	A													
	B													
	C													
Time	A													
	B													
	C													
Cost	A													
	B													
	C													

Tabelle 2.2: Beispiele mit komplizierter Zellenstruktur und automatischer Schrifteinstellung in zwei Kolonnen.

3 Mathematische Formeln

Der Mathematikmodus ist sehr mächtig und kann nicht in wenigen Sätzen erklärt werden. Aus diesem Grund wird nochmals auf die LaTeX2e-Kurzbeschreibung verwiesen, welche alles wichtige erklärt. Möchte sich jemand noch tiefer in die Materie einlesen, hilft die Dokumentation **doc_mathmode**.

3.1 Mathematische Umgebungen

Selbst komplizierte Formeln können mit LaTeX sehr schnell umgesetzt werden. Zur Verfügung stehen verschidene Modi:

`$ ... $`	Einfacher Mathe-Modus direkt im Text

`\[`
`...`
`\]`
	Abgesetzter Mathe-Modus ohne Nummerierung

`\begin{equation}`
`...`
`\end{equation}`
	Abgesetzter Mathe-Modus mit Nummerierung

3.2 Bekannte Fehler

Dieses Kapitel soll auf die häufigsten Fehler aufmerksam machen, welche öfters falsch gemacht werden. Die Liste ist selbstverständlich nicht abgeschlossen, doch für den Anfang sollten diese Tipps schon reichen.

Falsch	**Richtig**	**Beschreibung**
V_{IN}	V_{IN}	Variablen sind kursiv dargestellt. Im linken Fall würde sich der tiefgestellte Index aus den Variablen $I \cdot N$ berechnen. Bezeichnungen/Namen werden jedoch mit aufrechter Schrift dargestellt. Dazu benutzt man `\mathrm{}`.
$e^{j \cdot \omega \cdot t}$	$\mathrm{e}^{j\omega t}$	Zwischen einzelnen Variablen werden Multiplikationen impliziert und daher weggelassen. Es kann jedoch sinnvoll sein, für die optische Hervorhebung von wichtigen Thermen ein Punkt (`\cdot`) zu setzen.
$sin(\alpha)$	$\sin(\alpha)$	Funktionen sind keine Variablen und stehen deshalb nicht kursiv.
$\exp(\frac{A}{B})$	$\exp\left(\frac{A}{B}\right)$	Klammern müssen mit `\left` und `\right` skaliert werden
$\frac{A}{B} = \frac{\frac{C}{D}}{B}$	$\frac{A}{B} = \frac{C/D}{B}$	Nicht unterschiedlich skalierte Brüche verwenden. Lieber mal einen normalen Schrägstrich setzen.

4 Tikz-Grafiken

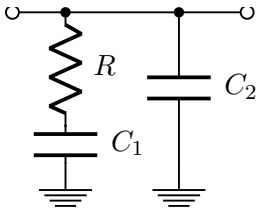
Tikz ist ebenfalls sehr mächtig und auf den ersten Blick auch sehr kompliziert. Schon alleine die Dokumentation des Grundpackages erstreckt sich über 1000 Seiten. Tikz lohnt sich vor allem, wenn die erstellte Grafik (oder nur Teile davon) wiederverwendet werden können. Es folgen drei Beispiele für Tikz-Grafiken.

Man beachte, dass diese in einem eigenen „LaTeX-Projekt“ erstellt wurden. Dieses hat die `\documentclass{standalone}` und kann deswegen eigenständig kompiliert werden. Dabei werden automatisch Unterstüzungslinien/Grid eingeblendet (wurde programmiert), welche die Gestaltung der Grafik extrem erleichtern. Schaut euch doch die Tikz-Dateien an und kompiliert sie separat, es lohnt sich!

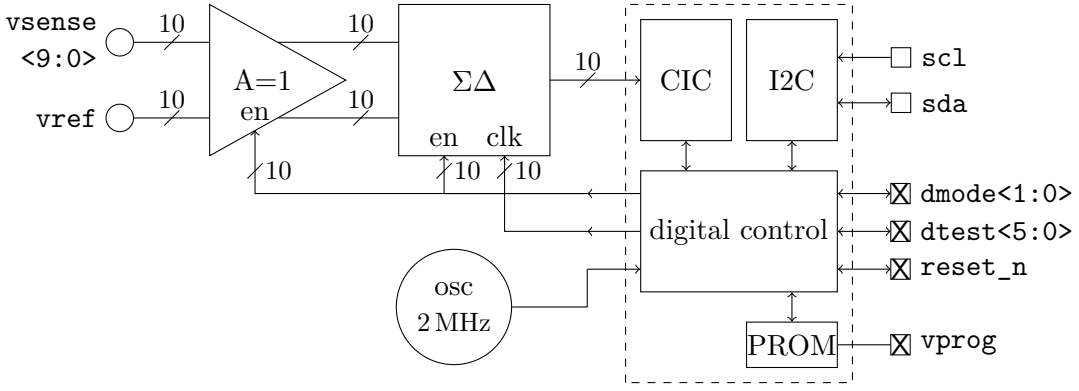
Mittels Befehl `\includestandalone{}` werden dann diese in jedes andere Projekte eingebunden, und zwar nicht als PDF sondern direkt erstellt beim Kompilieren.

Somit können wir nun einfache elektrische Schaltungen wie in Figur 4.1a oder auch komplizierte Blockschaltbilder wie in Figur 4.1b programmieren.

Im Dokumenteordner */tikz/* findet ihr noch zwei weitere Beispiele. Eines zeigt ein **animiertes Tikz** und das andere interagiert mit **gnuplot**, um Plots zur Laufzeit zu erstellen. Um gnuplot nutzen zu können sind ein paar zusätzliche Installationen notwendig. Weiter muss der Kompilierbefehl für `pdflatex` mit `-shell-escape` erweitert werden. Das Internet bietet gute Unterstützung bei der Integration von gnuplot. Viele weitere coole Beispiele findet ihr auf http://www.texample.net/tikz/examples/.



(a) Einfach



(b) Kompliziertes Blockschaltbild

Abbildung 4.1: Zwei tikz-Beispiele: (a) einfach, (b) kompliziert.

5 Bibliographien

Dieses Template arbeitet mit bibLaTeX und biber; einige Informationen dazu findet man in **biblatex_biber**. Die Anwendung `biber.exe` ist standardmässig installiert, muss jedoch anstelle von `bibtex.exe` aufgerufen werden. **Dazu muss im verwendeten Editor der Bib(la)tex Befehl durch biber ersetzt werden.**

5.1 Literaturdatenbank

Um zu Referenzieren braucht man nun nur die Datei `literature/bibliography.bib` auszufüllen (BibLaTeX-Mode), zum Beispiel mit Hilfe des Quellenverwaltungsprogramm JabRef **jabref**. Danach muss das Dokument mehrfach zu kompilieren: einmal mit pdfLaTeX, damit die Literaturverweise erkannt und festgehalten werden, dann einmal mit biber, welches die Daten aus `literature/bibliography.bib` herausliest und in das richtige Format bringt, und dann zweimal mit pdfLaTeX, damit das Literaturverzeichnis korrekt wird und alle Nummern im Text stimmen.

5.2 Referenzieren

Man kann nun mit verschiedenen Versionen des Befehles `\cite` nun einzelne Publikationen **Mason1953**, mehrere miteinander **Mason1953**, **Mason1956**, oder Abschnitte aus einer Publikation **Schmid2018** zitieren. Für die genaue Positionierung der Referenzen bitte den Leitfaden verwenden.

5.3 Literaturverzeichnis

Der Befehl `\printbibliography` erstellt ein Literaturverzeichnis. Wie auf Seite 13 zu sehen ist, passt sich das Literaturverzeichnis so automatisch der gewählten Sprache an.

5.4 Was bedeutet eigentlich zitieren und referenzieren?

Woher habe ich meine Information?

Meine Ansichten darüber, wie Wissenschaft funktioniert, decken sich weitgehend mit **Schmid2003**.

Woher genau?

Die genaue Zusammenstellung der drei Kriterien für empirische Wissenschaftlichkeit ist zu finden in **Schmid2003**

Was steht denn dort?

Praphrasieren: In **Schmid2003** beschreibt Schmid, dass es für empirische Wissenschaften nicht nur wesentlich ist, sich auf die Wirklichkeit zu beziehen und sich darüber Gedanken zu machen, sonder auch diese Gedanken mit anderen Wissenschaftlern zu teilen und zu besprechen.

Was steht denn dort *genau*?

In **Schmid2003** steht:

> All that is empirical science has three things in common: a practical injunction (if you want to know this, you have to do this); an apprehension, illumination, or experience (if you do this, you see this), and communal checking (did others who did this also see the same?).

Das habe ich oben gemeint mit „[. . .] dass es für empirische Wissenschaften nicht nur wesentlich ist, sich auf die Wirklichkeit zu beziehen und sich darüber Gedanken zu machen, sonder auch diese Gedanken mit anderen Wissenschaftlern zu teilen und zu besprechen.“

Und wenn jemand einen Fehler gemacht hat?
Tellegen publizierte das 1954 schon in seinem Paper *La recherche pour una [sic!] série complète d'éléments de circuit ideaux non-linéaires* **Tellegen1954**.

EIT-Projekt P2 - Aufgabenstellung vom Auftraggeber (FS_2018)

Array-Antennen (Gruppenstrahler): Eigenschaften, Beispiele und Visualisierung der Richtcharakteristik

1. Einleitung

In der modernen Gesellschaft spielt die weltweite und schnelle Kommunikation eine wichtige und zentrale Rolle. Dank drahtloser Übertragung (wireless) kann z.B. mit dem Smartphone oder Tablet PC fast verzögerungsfrei auf beliebig viele Daten zugegriffen, oder Informationen gesucht werden. Die Geschichte der drahtlosen Übertragung ist relativ jung und begann mit den "Maxwellschen Gleichungen", wo 1873 der Physiker Maxwell in genialer Weise die Gleichungen so darstellte, dass daraus die (theoretische) Existenz von elektromagnetischen Wellen ersichtlich war. Erst Heinrich Hertz konnte in Jahren 1885 bis 1889 mit genialen Experimenten die Existenz von elektromagnetischen Wellen nachweisen. Die erste drahtlose Übertragung demonstrierte Marconi im Jahre 1898. Heute werden Marconi (1874-1934) und auch Popow (1859-1905) als Erfinder der Antenne genannt.

Da man die elektromagnetischen Wellen weder sieht noch hört (uns fehlt das entspr. Sinnesorgan) bleibt für die meisten Anwender die drahtlose Übertragung immer noch geheimnisvoll und rätselhaft. Die zugehörige Theorie ist relativ abstrakt, aber äusserst erfolgreich, denn sie beschreibt alle Phänomene korrekt und hat uns die moderne drahtlose Kommunkation ermöglicht. Elektromagnetische Wellen können von ganz tiefen Frequenzen (Wellenlängen im km-Bereich) bis zu sehr hohen Frequenzen (Wellenlängen im mm-Bereich) abgestrahlt und empfangen werden. Die "Umwandlung" von "drahtgebundener Wellen-ausbreitung" zu "freier Wellenausbreitung" (Sendefall) und umgekehrt (Empfangsfall) geschieht mittels **Antennen**, welche für die drahtlose Technologie eine zentrale Rolle spielen.

Antennen kommen in unterschiedlichsten Bauformen und Ausprägungen vor. Das Prinzip und die wichtigen Begriffe werden meistens am "Hertzschen Dipol" erläutert (einfachster Fall) und dann auf weitere Bauformen erweitert ($\lambda/2$ - Dipol, Faltdipol).

Eine wichtige und sehr verbreitete Bauform sind sog. "Gruppenantennen" oder "Arrays", welche in allen möglichen Ausprägungen und Technologien vorkommen. Dabei werden "Strahler" einzeln angesteuert und bilden dann zusammen die Antenne (Array). Diese Art von Antennen sind sehr flexibel und durch die individuelle Ansteuerung der Einzelstrahler (Amplitude, Phase, oder beides) lassen sich "Antennen-Eigenschaften" realisieren, welche äusserst attraktive Anwendungen möglich machen:

- Elektronische Strahlschwenkung
- Adaptives Diagrammsynthese
- Störer ausblenden (Nullstelle im Diagramm)
- Konfigurierbare Diagramm
- Gleichzeitig in mehrere Richtungen schauen
- Bodenechos unterdrücken (Radar / AWACS)

Diese Möglichkeiten sind für Hightech Anwendungen (militärischen Bereich, Raumfahrt) sehr interessant und es ist nicht überraschend, dass dort der grösste Einsatzbereich dieser Array-Technologie liegt. Aber auch im Amateurfunk werden solche Arrays als aktive Empfangsantennen für bestimmte Frequenzbänder verwendet (z.B. 8 Einzelstrahler kreisförmig angeordnet mit wählbarer Richtung in Winkelschritten von 45°).

Das Array-Prinzip kommt auch in Anwendungen in der Akustik, Medizin und Optik vor (Lautsprecher-Arrays, Ultraschallköpfe, Beugung am Gitter, usw.). Überall wo Wellen und die zugehörigen Phänomene (konstruktive und destruktive Interferenz) auftreten, lässt sich das Array-Prinzip anwenden.

Konkret geht es in diesem Projekt um die Entwicklung und Realisierung eines Tools, mit welchem der Benutzer das Array-Prinzip "erleben und verstehen" kann. Dazu sollen mit illustrativen Beispielen wichtige Erkenntnisse gewonnen werden. Die Berechnung und Darstellung der Strahlungscharakteristik (Strahlungsdiagramm) ist ein zentrales Element des Tools.

2. Aufgaben/Anforderungen an Tool

Entwerfen und realisieren Sie ein benutzerfreundliches Tool/Programm/GUI/usw. mit welchem ebene Array-Antennen berechnet werden können und Antennendiagramme in geeigneter Art und Weise visualisiert werde können. Folgende wichtige "Spezialfälle" und Anwendungen sollen dabei auch berücksichtigt werden:

- Diagrammsynthese mit unterdrücken Nebenkeulen
- Diagrammsynthese mit Nullstelle an gewünschter Stelle
- Strahlschwenkung (phased array)
- Anordnung der Strahler in typischen Geometrien (Zeile, Quadrat, Kreis,…)
- Diskrete Amplituden (Einfluss Quantisierung)
- …

Ergeben sich im Laufe des Projekts neue Erkenntnisse, so können obige Anforderungen auch angepasst und/oder erweitert werden.

3. Bemerkungen

Die Software und das GUI sind in enger Absprache mit dem Auftraggeber zu entwickeln. Der Auftraggeber steht als Testbenutzer zu Verfügung und soll bei der Evaluation des GUI eingebunden werden. Alle verwendeten Formeln, Algorithmen und Berechnungen sind zu verifizieren, eine vorgängige oder parallele Programmierung in Matlab ist zu empfehlen. Zur Fachthematik des Projektes werden Inputs durchgeführt.

21.02.2018
Peter Niklaus

Semester Planning: Autumn Semester 2017 (P8a)

Week	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3	4	5	6	7	8 - 37	MSE		Days off		Project		IME	
	Sep.		October				November				December					January				February												
	18.09 - 24.09	25.09 - 01.10	02.10 - 08.10	09.10 - 15.10	16.10 - 22.10	23.10 - 29.10	30.10 - 05.11	06.11 - 12.11	13.11 - 19.11	20.11 - 26.11	27.11 - 03.12	04.12 - 10.12	11.12 - 17.12	18.12 - 24.12	25.12 - 31.12	01.01 - 07.01	08.01 - 14.01	15.01 - 21.01	22.01 - 28.01	29.01 - 04.02	05.02 - 11.02	12.02 - 18.02	19.02 - 16.09	Total Hours	Total Days	Total Hours	Total Days	Total Hours	Total Days	Total Hours	Total Days
MSE (Lesson, EV, Exams, ...)	17	17	27	27	27	17	17	17	17	17	17	17	17	17				20	30	40	32		210	600	71.4						
Holiday, Public Holiday															42	42						42	210			336	40.0				
Project SS-OCT	6	6	13	10		14	19	23	23	23	23	23	18	19			40	20	10				770					1060	126.2		
IME, CoachING, Wilda, ...	19	19	2	5	15	11	6	2	2	2	2	2	7	6			2	2	2	2	10		70							188	22.4
Total Hours per Week	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	1260	600.0		336.0		1060.0		188.0	
Differenz zu 42 h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					P8a		290	
																												P8b		770	
																												Total		2184	

Project Planning: FFT for OCT - Matlab Model

Week		38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3	4	5	6	7	8 - 37	Zeit		
		Sep.		October				November				December					January				February						
		18.09 - 24.09	25.09 - 01.10	02.10 - 08.10	09.10 - 15.10	16.10 - 22.10	23.10 - 29.10	30.10 - 05.11	06.11 - 12.11	13.11 - 19.11	20.11 - 26.11	27.11 - 03.12	04.12 - 10.12	11.12 - 17.12	18.12 - 24.12	25.12 - 31.12	01.01 - 07.01	08.01 - 14.01	15.01 - 21.01	22.01 - 28.01	29.01 - 04.02	05.02 - 11.02	12.02 - 18.02	19.02 - 16.09	Total in WP	Total Hours	Total Days
WP3.0	**Introductory phase**																										
i	Transfer	6	6																						69	12	1
ii	Understand Structure			13																						13	2
iii	Prepare existing Matlab-Files						14																			14	2
iii	Introduction to VHDL/Quartus																							30		30	4
WP3.1	**Analysis of FFT structure**																										
i	Analysis Bitwidth																	10							10	10	1
ii	Comparison Generateable Code																									0	0
WP3.2	**Development of a model**																										
i	Implementation 16K FFT-Core							19	23	23	10														164	75	9
ii	Expansion to 32K FFT											23	23	18												64	8
iii	Verification										10				15◆											25	3
WP3.3	**Implementation on FPGA**																										
i	Implementation 16K/32K FFT-Core																							350	500	350	42
ii	Verification																							150		150	18
WP3.4	**Administratives**																										
i	Project Clarification P8a				10◆																				57	10	1
ii	Presentation P8a																			10◆						10	1
iii	Documentation P8a										3				4			20	10	◆						37	4
iv	Administratives P8b																							115	115	115	14
x	Reserve/Buffer																	10	10					125		145	17
Total Hours per Week		6	6	13	10	0	14	19	23	23	23	23	23	18	19	0	0	40	20	10	0	0	0	770		1060	126.2

P. Studer, 16.10.2017

	Date:	Milestone:
◆	22.12.17	Golden Model of FFT Core
◆	13.10.17	Delivery Project Clarification
◆	25.01.18	Presentation
◆	26.01.18	Delivery Tech. Report

C MATLAB-Code Snippets

```
% Random Code Example:

for c = 1:ncols
    for r = 1:nrows
        if r == c
            A(r,c) = 2;
        elseif abs(r-c) == 1
            A(r,c) = -1;
        else
            A(r,c) = 0;
        end
        if isempty(line), break, end
    end
end

if any(A > limit)
    disp('There is at least one value above the limit.')
else
    disp('All values are below the limit.')
end

x = 10;
minVal = 2;
maxVal = 6;

if (x >= minVal) && (x <= maxVal)
    disp('Value within specified range.')
elseif (x > maxVal)
    disp('Value exceeds maximum value.')
else
    disp('Value is below minimum value.')
end

switch(grade)
   case 'A'
      fprintf('Excellent!\n' );
   case 'B'
      fprintf('Well done\n' );
   case 'C'
      fprintf('Well done\n' );
   case 'D'
      fprintf('You passed\n' );
   case 'F'
      fprintf('Better try again\n' );
   otherwise
      fprintf('Invalid grade\n' );
   end
```