

# **Eine kleine Einführung in wissenschaftliches Arbeiten mit L<sup>A</sup>T<sub>E</sub>X**

Lukas Jans

Version 1.0  
12. Dezember 2021

# Inhaltsverzeichnis

<b>Einleitung</b>	<b>3</b>
<b>1 Setup</b>	<b>4</b>
1.1 Installation . . . . .	4
1.1.1 MiKTeX . . . . .	4
1.1.2 Visual Studio Code . . . . .	4
1.1.3 GIT . . . . .	4
1.2 Das Repository . . . . .	4
1.3 Kompilierung . . . . .	5
<b>2 L<sup>A</sup>T<sub>E</sub>X</b>	<b>7</b>
2.1 Textmodus . . . . .	7
2.2 Mathematikmodus . . . . .	8
<b>Schlusswort</b>	<b>10</b>
<b>Literaturverzeichnis</b>	<b>11</b>
<b>Abbildungsverzeichnis</b>	<b>12</b>

# Einleitung

Die geläufigsten Texteditoren (z. B. Word) basieren auf dem Prinzip „what you see is what you get“, kurz WYSIWYG. Dadurch ist deren Bedienung sehr intuitiv und leicht zu erlernen. Für anspruchsvollere Dokumente – wie etwa eine Abschlussarbeit – können diese Editoren jedoch an ihre Grenzen stoßen. Besonders im MINT-Bereich bietet sich stattdessen L<sup>A</sup>T<sub>E</sub>X für den Textsatz an. Dabei handelt es sich um eine Art Programmiersprache, in der Inhalte zunächst ohne Vorschau auf das eigentliche Dokument verfasst werden. Um Gestaltungselemente wie etwa Überschriften hervorzuheben, werden diese nicht etwa ausgewählt und per Klick auf ein Menüelement formatiert, sondern durch sogenannte *Makros* ausgezeichnet. Dabei handelt es sich im Wesentlichen um Funktionen, die beispielsweise eine Veränderung der Schriftgröße ihres Arguments bewirken, bei dem es sich um den zu formatierenden Text handelt. Die tatsächliche Gestalt des fertigen Dokuments ergibt sich erst durch *Kompilieren* des entsprechenden Quellcodes zu einer PDF-Datei.

In dieser Einleitung sollen die Grundzüge des wissenschaftlichen Arbeitens mit L<sup>A</sup>T<sub>E</sub>X nähergebracht sowie eine handliche Entwicklungsumgebung dafür bereitgestellt werden.

# 1 Setup

## 1.1 Installation

### 1.1.1 MiKTeX

Um den L<sup>A</sup>T<sub>E</sub>X-Quellcode kompilieren zu können, wird neben dem Texteditor ein weiteres Programm benötigt. In dieser Einführung kommt dafür „MiKTeX“ zum Einsatz, das unter <https://miktex.org/download> verfügbar ist.

### 1.1.2 Visual Studio Code

Als Editor für die L<sup>A</sup>T<sub>E</sub>X-Dateien wird hier Visual Studio Code (kurz VS Code) verwendet, der unter <https://code.visualstudio.com/> erhältlich ist. Nach der Installation können darüber die folgenden Erweiterungen (Extensions) bezogen werden:

**„LaTeX Workshop“ von James Yu.** Diese Erweiterung stattet VS Code mit umfangreichen Hilfsmitteln rund um L<sup>A</sup>T<sub>E</sub>X aus.

**„L<sup>A</sup>T<sub>E</sub>X – LanguageTool grammar/spell checking“ von Julian Valentin.** Dieses Tool bietet eine beeindruckend gute Rechtschreibprüfung für L<sup>A</sup>T<sub>E</sub>X-Dokumente.

### 1.1.3 GIT

Die flache Dateistruktur von L<sup>A</sup>T<sub>E</sub>X eignet sich gut für eine Versionsverwaltung, hier kommt dafür GIT zum Einsatz, das unter <https://git-scm.com/download/win> verfügbar ist. Bei der Installation werden zahlreiche Konfigurationsmöglichkeiten abgefragt, in den meisten Situationen kann allerdings der Standard verwendet werden.

## 1.2 Das Repository

Nun kann das Repository dieses Projekts von <https://github.com/ljans/thesis> geklont werden, am besten in ein Arbeitsverzeichnis wie etwa C:\Workspace\. Hat alles geklappt, dann kann das Projekt jetzt in VS Code geöffnet werden. Es enthält bereits einige Ordner und Dateien, deren Bedeutung im Folgenden erklärt wird.

**.vscode** Dieser Ordner enthält Konfigurationsdateien für VS Code, in diesem Fall die sogenannten „Workspace-Settings“. Speziell für dieses Projekt werden darin zahlreiche Einstellungen für die Erweiterung „LaTeX Workshop“ getroffen, um ein reibungsloses Arbeiten mit L<sup>A</sup>T<sub>E</sub>X zu ermöglichen.

**Grafiken** In diesem Ordner werden Screenshots (oder andere Grafiken) abgelegt, die in diesem Dokument eingebunden werden.

**Kapitel** Hier sind die Inhalte der jeweiligen Kapitel dieses Dokuments in eigenen Dateien ausgelagert. Dadurch lässt sich die Größe der einzelnen Dateien reduzieren und somit die Übersichtlichkeit verbessern.

**.gitattributes und .gitignore** Dies sind Konfigurationsdateien für die Versionsverwaltung GIT und können unberührt gelassen werden.

**Literatur.bib** In diese Datei werden die (exemplarischen) Quellen der Arbeit in einer speziellen Syntax angelegt. Im eigentlichen Dokument wird diese Literaturliste dann eingebunden, um die automatische Erstellung von Quellenangaben in L<sup>A</sup>T<sub>E</sub>X nutzen zu können. Der Name „Literatur“ kann dabei angepasst werden und auch mehrere Dateien sind möglich.

**Thesis.pdf** Dies ist die Ausgabedatei, die von MiKTeX aus dem L<sup>A</sup>T<sub>E</sub>X-Quellcode generiert wird.

**Thesis.tex** Hierbei handelt es sich um die „Hauptdatei“ des Projekts. In ihr befindet sich die sogenannte *Präambel* mit den Metadaten des Dokuments; darin werden zum Beispiel Funktionspakete geladen, Einstellungen vorgenommen und Makros definiert. Neben der Steuerung von Abbildungs-, Inhalts- und Literaturverzeichnis werden in dieser Datei dann auch die ausgelagerten Kapitel eingebunden.

## 1.3 Kompilierung

Ist eine L<sup>A</sup>T<sub>E</sub>X-Datei in VS Code geöffnet, kann über die Erweiterung „LaTeX Workshop“ im linken oberen Teil des Fensters ein manueller Kompilervorgang gestartet werden (siehe Abbildung 1.1). Bei der Option „Full typesetting“ wird der Vorgang mehrmals durchlaufen, was gerade beim ersten Mal notwendig ist, damit alle Querverweise und Referenzen richtig zugeordnet werden. Das Projekt ist so konfiguriert, dass beim Speichern einer L<sup>A</sup>T<sub>E</sub>X-Datei automatisch ein „Partial typesetting“ ausgeführt wird, das zwar schneller geht, aber nur für Änderungen am Text verwendet werden sollte.

Beim ersten Kompilieren wird MiKTeX erkennen, dass die eingebundenen Pakete zum Teil noch nicht installiert sind und entsprechend dazu auffordern. Dieser Vorgang kann

## 1 Setup

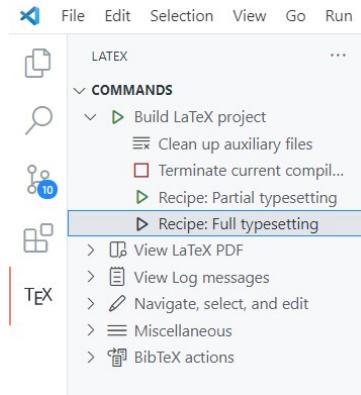


Abbildung 1.1: Manuelles Kompilieren einer L<sup>A</sup>T<sub>E</sub>X-Datei.

einen Moment dauern und sich mehrmals wiederholen. Anschließend startet die eigentliche Kompilierung. Dabei werden einige Hilfsdokumente im Projekt angelegt, die ein erneutes Kompilieren beschleunigen sollen.

Nun kann das fertige PDF-Dokument angezeigt werden; dafür nutzt man die Schaltfläche im oberen rechten Teil des Fensters, wie in Abbildung 1.2 zu sehen. Wird die



Abbildung 1.2: Kompilierte PDF-Datei anzeigen.

Datei auf diesem Weg geöffnet, aktualisiert sich das PDF-Dokument bei jedem erneuteten Kompilieren automatisch. Darüber hinaus steht so *SyncTeX* zur Verfügung: Durch Linksklick auf eine Stelle im PDF-Dokument bei gehaltener **Strg**-Taste, springt VS Code automatisch an die jeweilige Stelle im Quellcode. Andersherum wird die aktuelle Cursorposition in VS Code durch die Tastenkombination **Strg + Alt + J** im PDF-Dokument hervorgehoben.

## 2 L<sup>A</sup>T<sub>E</sub>X

### 2.1 Textmodus

Dieses Kapitel ist hauptsächlich dafür gedacht, die Ausgabe mit dem Quellcode zu vergleichen, um etwas Erfahrung im Textsatz mit L<sup>A</sup>T<sub>E</sub>X zu gewinnen. Makros – die zentralen Steuerelemente – bestehen aus einem \ gefolgt von einem Namen. Die einfachste Gruppe von Makros ist für das Einfügen von Sonderzeichen gedacht, beispielsweise für Symbole wie ✓ aber auch den L<sup>A</sup>T<sub>E</sub>X-Schriftzug.

Andere Makros benötigen ein Argument, das nach dem Namen in geschweiften Klammern angegeben wird. Dazu gehören beispielsweise die Makros `chapter` oder `section` zum Erstellen von Überschriften. Auch Makros zur Formatierung von *kursivem* oder unterstrichenem Text funktionieren auf diese Weise.

**Achtung:** Besteht das Argument aus nur einem Zeichen, kann es ohne Klammern angegeben werden, zum Beispiel `so`. Das hat allerdings zur Folge, dass auch Leerzeichen nach einem Makro als Argument interpretiert werden. So wird zwischen den Ausdrücken L<sup>A</sup>T<sub>E</sub>TEXtest kein Leerzeichen gesetzt, obwohl es im Quellcode vorkommt. Auch mehrere Leerzeichen in Folge helfen nicht, da diese automatisch reduziert werden. Zur Abhilfe kann hier mit einem Backslash gefolgt von einem Leerzeichen aber bewusst ein Leerzeichen    gesetzt werden. Anspruchsvollere Makros können auch mehrere Argumente erfordern, die dann nacheinander jeweils in geschweiften Klammern dem Namen angehängt werden.

Als Nächstes betrachtet man eine *Umgebung*, die vom Makro `begin` geöffnet und von `end` geschlossen wird. Darunter fällt zum Beispiel die Aufzählungs-Umgebung *itemize*:

- Erster Aufzählungspunkt
- Zweiter Aufzählungspunkt
- ...

Wurde im Präambel eine Literaturliste geladen, dann können im Text leicht Quellenangaben gemacht werden (Tonomura et al., 1989). Nach Ohanian und Markert (2007) stehen hierfür mehrere Makros zur Verfügung. Diese lassen einen optionalen Parameter zu, der in eckigen Klammern angegeben wird beispielsweise eine Seitenzahl angibt. (vgl. Tonomura et al., 1989, S. 7)

## 2.2 Mathematikmodus

Standardmäßig befindet sich L<sup>A</sup>T<sub>E</sub>X im *Textmodus*, zum Setzen mathematischer Ausdrücke muss in den *Mathematikmodus* umgeschaltet werden. Dies ist zum Beispiel durch das Dollarzeichen \$ möglich. In einem Fließtext lässt sich damit eine kleine Formel wie etwa  $a = b + c$  einbinden.

Für größere Ausdrücke gibt es eigene Mathematik-Umgebungen, zum Beispiel `align`:

$$a = b + c. \quad (2.1)$$

Hierbei wird automatisch eine Formelnummer generiert, die über (2.1) referenziert werden kann, falls ihr ein entsprechendes `label` zugewiesen wurde. Die *gesternete* Umgebung `align*` unterdrückt das Erzeugen einer Formelnummer.

Im Mathematikmodus gibt es nun eine Fülle weiterer Makros für den Aufbau der gewünschten Ausdrücke. Das folgende Beispiel zeigt eine kleine Auswahl. Es ist zu beachten, dass Makros allgemein auch verschachtelt genutzt werden können. Das doppelte Backslash bewirkt einen Zeilenumbruch.

$$\begin{aligned} f(x) &= \alpha x + \beta \\ \frac{1}{q} + \frac{1}{p} &= n! \\ \forall n \in \mathbb{N} : \sqrt{5\pi^n} &\geq 7 \\ \mathcal{F}\vec{x} &=: \lim_{b \rightarrow \infty} \int_a^b \langle \psi | \phi \rangle \delta_k(x) dx. \end{aligned}$$

Sollen Formeln an einem bestimmten Zeichen (beispielsweise dem Gleichheitszeichen) ausgerichtet werden, kann das &-Zeichen davorgesetzt werden. Häufig verwendete Formelbausteine können über `newcommand` als eigene Makros definiert werden (siehe Präambel), um den Code lesbarer zu gestalten:

$$\begin{aligned} \sum_{n=1}^{\infty} |x_n + z_n|^2 &\leq \sum_{n=1}^{\infty} (|x_n + z_n|^2 + |x_n - z_n|^2) \\ &= \sum_{n=1}^{\infty} ((x_n^* + z_n^*)(x_n + z_n) + (x_n^* - z_n^*)(x_n - z_n)) \\ &= \sum_{n=1}^{\infty} (2|x_n|^2 + 2|z_n|^2) \\ &= 2 \sum_{n=1}^{\infty} |x_n|^2 + 2 \sum_{n=1}^{\infty} |z_n|^2 < \infty. \end{aligned}$$

Kleinere Mathematik-Ausdrücke ohne Ausrichtung können etwas kompakter auch über die Umgebung mit den Begrenzern `\[` und `\]` gesetzt werden.

Für mathematische Sätze, Beweise, Definitionen, usw. werden ebenfalls entsprechende Umgebungen eingesetzt. Auch diese lassen einen optionalen Parameter zu, der als Titel fungiert.

### Definition 1 (Hilbertraum)

Ein Innenproduktraum, der bezüglich der vom Skalarprodukt induzierten Norm vollständig ist, heißt auch *Hilbertraum*.

### Satz 2 (Cauchy-Schwarz'sche Ungleichung)

Ist  $X$  ein Skalarproduktraum, dann gilt für alle  $x, z \in X$

$$|\langle x|z\rangle|^2 \leq \langle x|x\rangle\langle z|z\rangle.$$

BEWEIS. Seien  $x, z \in X$  beliebig. Da die Behauptung für  $z = 0$  trivialerweise erfüllt ist, nimmt man im Folgenden jedoch  $z \neq 0$  an. Für alle  $\alpha \in \mathbb{C}$  gilt

$$0 \leq \langle x + \alpha z | x + \alpha z \rangle = \langle x|x\rangle + \alpha\langle x|z\rangle + \alpha^*\langle z|x\rangle + |\alpha|^2\langle z|z\rangle.$$

Einsetzen von

$$\alpha = -\frac{\langle z|x\rangle}{\langle z|z\rangle} = -\frac{\langle x|z\rangle^*}{\langle z|z\rangle}$$

liefert

$$0 \leq \langle x|x\rangle - \frac{|\langle x|z\rangle|^2}{\langle z|z\rangle} - \frac{|\langle z|x\rangle|^2}{\langle z|z\rangle} + \frac{|\langle z|x\rangle|^2}{\langle z|z\rangle} = \langle x|x\rangle - \frac{|\langle x|z\rangle|^2}{\langle z|z\rangle}$$

und nach Multiplikation mit  $\langle z|z\rangle$

$$0 \leq \langle x|x\rangle\langle z|z\rangle - |\langle x|z\rangle|^2,$$

woraus die Behauptung folgt. □

# Schlusswort

Diese Einführung konnte hoffentlich einen kleinen Einblick in die Möglichkeiten von L<sup>A</sup>T<sub>E</sub>X liefern. Allen Lesern wird empfohlen, den Quellcode des Dokuments zu untersuchen, darin zu experimentieren und auszuprobieren – nur so kann ein sicherer und produktiver Umgang erlernt werden. Ausführliche Dokumentationen zu sämtlichen Makros und Prinzipien sind leicht im Internet zu finden. Es steht außer Frage, dass sich der Einsatz beim Schreiben einer Abschlussarbeit auszahlt wird.

In Zukunft soll dieses Dokument weiter ausgebaut und mit mehr Screenshots ergänzt werden. Wer dabei mitwirken möchte, ist herzlich eingeladen, ein Pull Request auf GitHub zu erstellen.

# Literaturverzeichnis

- Ohanian, H. C. & Markert, J. T. (2007). *Physics for Engineers and Scientists: Volume 3* (5. Aufl.). W.W. Norton & Company, New York, London.
- Tonomura, A., Endo, J., Matsuda, T., Kawasaki, T. & Ezawa, H. (1989). Demonstration of single-electron buildup of an interference pattern. *Physics World*, 57.

# Abbildungsverzeichnis

1.1	Manuelles Kompilieren einer L <sup>A</sup> T <sub>E</sub> X-Datei. . . . .	6
1.2	Kompilierte PDF-Datei anzeigen. . . . .	6