Einführung in LaTeX

Burkhard Bunk

13.3.2013

1 Struktur von LaTeX

Ein LaTeX–File wird mit einem Text–Editor geschrieben, es enthält Text und Formeln und ist von Befehlen durchsetzt. Das Format des Files ist weitgehend frei, denn Zeilenumbruch, Seitenumbruch, Formeln usw. werden von TeX nach den Regeln des Schriftsatzes gestaltet.
Ein % und alles weitere bis zum Zeilenende wird ignoriert, so kann man Kommentare schreiben und Zeilen 'auskommentieren'.
Befehle sind an einem vorangestellten \ zu erkennen, ihre Argumente werden in `[]` und `{}` eingeschlossen. In LaTeX findet man oft 'Blöcke' der Form

```
\begin{irgendwas}
...
\end{irgendwas}
```

Damit werden 'Umgebungen' definiert, in denen spezielle Regeln gelten.
Ein LaTeX–File sieht häufig so aus wie das hier verwendete:

```
\documentclass[12pt,a4paper]{article}
\usepackage{graphicx,ngerman}

\begin{document}

\title{Einf"uhrung in LaTeX}
\author{Burkhard Bunk}
\date{29.2.2012}
\maketitle

\section{Struktur von LaTeX}

...Text...
```

```
\section{Text}

...Text...

\end{document}
```

`12pt` bezeichnet die Schriftgröße, `a4paper` das Papierformat A4.
Die Gliederung durch `\title`, `\section`, `\subsection` ist typisch für den Stil `article`. Andere Formen sind `letter, book, report`.
In `\usepackage` sind einige Makros benannt, die zusätzlich geladen werden sollen: `graphicx` bezeichnet das unten besprochene Paket zum Einbinden von Bilddateien, `ngerman` aktiviert die Regeln für ein deutsches Dokument: u.a. gelten die deutschen Trennungsregeln (nach der neuen Rechtschreibung) und man kann Umlaute und ß eingeben als `"a "o "u "A "O "U "s`.

2 Text

Text kann man einfach eingeben, die Gestaltung erledigt TeX. Einen Zeilenumbruch erzwingt man mit `\\`, eine Leerzeile trennt Absätze.

3 Formelsatz

Zum Gestalten von Formeln gibt es den 'mathematischen Modus'. Innerhalb des Textes wird er durch `$ ... $` aktiviert, für abgesetzte (und nummerierte) Formeln benutzt man am besten die Umgebung

```
\begin{eqnarray}
...
\end{eqnarray}
```

Sie erlaubt mehrzeilige Formeln (Zeilenumbruch mit `\\`) und eine Art Tabulator (`&..&`) zum Ausrichten der Zeilen untereinander. Ein Beispiel:

```
\begin{eqnarray}
      (a + b)^2 &=& a^2 + 2ab + b^2 \\
      (a - b)^2 &=& a^2 - 2ab + b^2 \\
                &=& (a + b)^2 - 4ab
\end{eqnarray}
```

erzeugt

$$
\begin{aligned}
(a+b)^2 &= a^2 + 2ab + b^2 && (1)\\
(a-b)^2 &= a^2 - 2ab + b^2 && (2)\\
&= (a+b)^2 - 4ab && (3)
\end{aligned}
$$

Die *eqnarray*–Umgebung verträgt übrigens keine Leerzeilen, warum auch immer.
Die Formeln werden automatisch nummeriert. Für eine einzelne Zeile kann man das mit `\nonumber` unterdrücken, die Umgebung `eqnarray*` lässt die Nummern ganz weg.
Einige häufig gebrauchte Befehle im Formelsatz sind
`\frac{..}{..}` für einen Bruch,
`\alpha` usw. für griechische Buchstaben,
`\sin` usw. für die Standard–Funktionen,
`\sum` für ein Summenzeichen,
`\int` für ein Integralzeichen.
Zum Hoch– und Tiefstellen dienen `^` und `_` (auch bei Summations– und Integrationsgrenzen). Wenn mehr als ein Zeichen betroffen ist, muß man den Ausdruck in `{}` einschließen.
Runde und eckige Klammern kann man einfach eingeben, geschweifte sind durch `\{ ... \}` zu schützen, damit sie nicht als TeX–Klammern interpretiert werden.
Für die zahlreichen Sonderzeichen und Tricks sei auf die Literatur verwiesen, Kopka[1] gibt erschöpfend Auskunft, Wonneberger[2] hat das wichtigste zum Nachschlagen zusammengestellt.

4 Tabellen

Eine Datentabelle baut man im mathematischen Modus mit Hilfe der Umgebung `array` auf. Hier ein Beispiel:

```
\begin{eqnarray}
   \begin{array}{|c|c|cc|c|}
            \hline
      d & \lambda_{max}(H_{ff}) & \lambda_{min}(A_{ff})
        & \lambda_{max}(A_{ff}) & \mbox{cond}(A_{ff}) \\
            \hline
      2 & 2.8284 & 0.2929 & 1.7071 &  5.83 \\
      3 & 5.2915 & 0.1181 & 1.8819 & 15.94 \\
      4 & 7.5696 & 0.0538 & 1.9462 & 36.17 \\
```

```
            \hline
   \end{array} \nonumber
\end{eqnarray}
```

erzeugt

d	$\lambda_{max}(H_{ff})$	$\lambda_{min}(A_{ff})$	$\lambda_{max}(A_{ff})$	$\mathrm{cond}(A_{ff})$
2	2.8284	0.2929	1.7071	5.83
3	5.2915	0.1181	1.8819	15.94
4	7.5696	0.0538	1.9462	36.17

Im Argument `|c|c|cc|c|` steht jedes `c` für eine Spalte mit zentriertem Eintrag und jeder `|` für eine senkrechte Trennlinie. Horizontale Linien sind durch `\hline` erzeugt. In der Tabelle selbst sind Spalten durch `&` getrennt und Zeilenenden durch `\\` markiert.
Die ganze Box kann noch in eine Umgebung

```
\begin{table}
...
\end{table}
```

eingekapselt werden, die eine Tabellennummer erzeugt und eine `caption` ermöglicht, genauso wie unten bei der Abbildung.
Zahlentabellen, die als Dateien vorliegen (z.B. als Ausgabe einer Tabellenkalkulation), kann man mit dem Befehl `csv2latex` (auf der Unix–Kommandozeile) mit den Latex–Formatierungen versehen, so dass sie mit geringem Aufwand in ein Latex–Dokument eingebaut werden können.
Matrizen kann man auch mit der *array*–Umgebung schreiben, siehe Anhang B.5.
Im Textmodus kann man eine Tabelle ganz ähnlich mit der Umgebung `tabular` (statt `array`) konstruieren, man hätte dann aber für jeden mathematischen Ausdruck im Tabellenfeld ein `$ ... $` gebraucht.
Auch hier muß man Einzelheiten in der Literatur[1] nachschlagen.

5 Abbildungen

Die Abbildung 1 ist mit Hilfe des Pakets `graphicx` eingebaut:

```
\begin{figure}
   \begin{center}
      \includegraphics[width=80mm]{sinus}
      \caption{Ein Sinus, was sonst.}
   \end{center}
\end{figure}
```

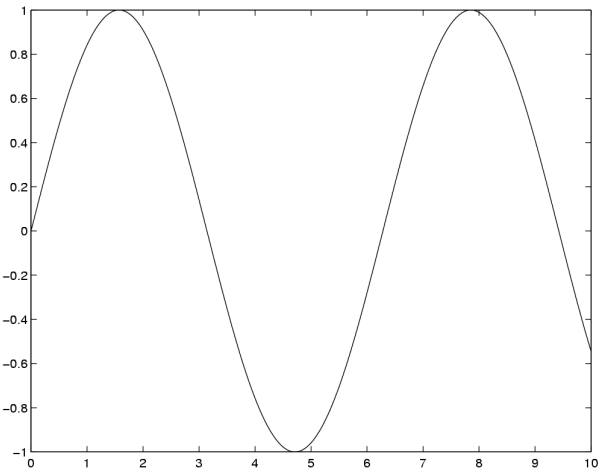

Abbildung 1: Ein Sinus, was sonst.

Hier wird die PostScript–Grafik eingebunden, die im File `sinus.ps` (im aktuellen Verzeichnis) gespeichert ist. Sie wird zentriert und auf 80mm horizontale Größe skaliert. Bilder im Querformat werden manchmal in gedrehter Form (*landscape*) gespeichert, das kann man durch eine Option wie `angle=-90` bei `\includegraphics[..]` kompensieren.
Beim Aufruf von `latex` wird, wenn keine Erweiterung angegeben ist, erst `sinus.eps` gesucht und dann `sinus.ps`. Andere Bildformate kann LaTeX nicht einbinden.
Hier kommt man mit `pdflatex` weiter: mit demselben Paket `graphicx` kann es PDF, JPEG, PNG und viele andere Formate einbinden (aber kein PS/EPS!) und erzeugt dann direkt eine PDF–Datei. Es wird, wenn nichts explizit angegeben wurde, erst nach `sinus.png`, dann nach `sinus.pdf`, schließlich nach `sinus.jpg` gesucht.
Für größere Texte empfiehlt sich folgende Strategie: bei `includegraphics` gibt man die Bilddatei immer *ohne* Erweiterung an. Für `latex` stellt man etwa `sinus.ps` bereit. Sollte man sich zum Umstieg auf `pdflatex` entschließen, dann erzeugt man zusätzlich z.B. `sinus.pdf` und braucht den Quelltext nicht zu ändern. Schon verhandene Bilder produziert man nicht noch einmal, sondern konvertiert sie, z.B. auf der Kommandozeile mit

```
prompt>epstopdf sinus.ps            => sinus.pdf
prompt>convert -density 150 sinus.ps sinus.jpg
prompt>convert -density 150 sinus.ps sinus.png
```

Der erste Befehl erzeugt eine PDF–Datei, in der das Seitenformat genau die Bildgröße widerspiegelt. Die Grafik bleibt dabei skalierbar. Im Gegensatz dazu erzeugen die beiden anderen Konvertierungen gerasterte Bilder, hier in einer Auflösung von 150dpi.
Die *figure*–Umgebung nummeriert die Abbildung und erlaubt eine `caption`. Ohne weitere Angaben setzt LaTeX die Abbildung bevorzugt oben auf die laufende oder folgende Seite. Wenn das nicht gefällt, kann man versuchen, die Position zu beeinflussen durch eine Option bei `\begin{figure}[X]`, wobei für X eine Angabe wie `h` (*here*) oder `b` (*bottom*) in Frage kommt an Stelle von `t` (*top*). Mit der Positionsangabe `[!h]` kann man eine Abbildung an Ort und Stelle nahezu erzwingen. Bei der Tabellenumgebung *table* geht das übrigens genauso.
Mehr zu diesem heiklen Thema der Gleitumgebungen findet man in [3].

6 Literaturverzeichnis

Das Literaturverzeichnis wird durch einen Block der Form

```
\begin{thebibliography}{9}
\bibitem{Kop1}    H. Kopka, LaTeX, Bd.1, Addison--Wesley, 1994.
                        Addison--Wesley, 1988.
...
\end{thebibliography}
```

angelegt. Die Einträge werden automatisch durchnummeriert. Jede Referenz ist durch ein Kürzel wie `Kop1` markiert, man zitiert sie als `\cite{Kop1}`, LaTeX ersetzt dann die Marke durch die Nummer. So kann man Referenzen nachträglich einfügen, und die Nummerierung paßt sich an.

7 LaTeX aufrufen

Nachdem man ein LaTeX–File, z.B. `aufgabe1.tex`, mit einem Texteditor geschrieben (und abgespeichert!) hat, will man sich das Kunstwerk natürlich ansehen. Dazu 'übersetzt' man es in eine DVI–Datei

```
prompt>latex aufgabe1          =>aufgabe1.dvi
```

In der Regel bekommt man eine Reihe (schwer verständlicher) Fehlermeldungen. Man kann versuchen, den Durchlauf durch wiederholtes `<ENTER>` zum Ende zu bringen, oder man bricht ihn ab (z.B. mit `<Q>` oder `<CTRL-C>`). Nachdem alle Fehler behoben sind, entsteht eine vom Ausgabegerät unabhängige

Beschreibung (DVI = *device independent*) des gestalteten Textes. Womöglich muss man den Befehl zweimal geben, damit Literaturliste, Verweise auf Abbildungen und Tabellen usw. auf dem neuesten Stand sind. Das Ergebnis kann man in einem Bildschirmfenster betrachten:

```
prompt>xdvi aufgabe1 &
```

das sich automatisch aktualisiert, wenn man neu übersetzt.
Schliesslich verwandelt man DVI in Postscript

```
prompt>dvips aufgabe1          =>aufgabe1.ps
```

das sich dann ausdrucken lässt:

```
prompt>lpr aufgabe1.ps
```

In der Praxis zirkuliert man lange zwischen Editieren/Speichern, Übersetzen und Ansehen, bis ein Text fertig ist, dabei sollte man die benötigten Fenster neben– oder übereinander stehen lassen.

8 PDFLaTeX aufrufen

Unter PDFLaTeX sieht der Ablauf so aus:

```
prompt>pdflatex aufgabe1       =>aufgabe1.pdf
```

Für PDF stehen viele Viewer zur Verfügung, vor allem `xpdf, gv, acroread`. Wichtig ist die Aktualisierung der Anzeige nach jedem Aufruf von `pdflatex`. In `acroread` gibt es dafür keinen einfachen Weg, `gv` braucht zweimal [Reload] (und gibt erstmal viele Fehlermeldungen), bei `xpdf` hilft Blättern oder die Taste [R]. Am besten ist also noch

```
prompt>xpdf aufgabe1.pdf &
```

Unter KDE kann auch `kpdf` versuchen, das aktualisiert sich durch Anklicken. `xdvi` und `dvips` braucht man hier nicht mehr.

A Eigene Befehle

Für einige längliche Befehle, die man sehr oft braucht, möchte man sich Abkürzungen definieren. Der Übersicht halber macht man das oben im Dateikopf (oberhalb von `\begin{document}`), dort könnte z.B. stehen:

```
\newcommand{\BE}{\begin{eqnarray}}
\newcommand{\EE}{\end{eqnarray}}
\newcommand{\NN}{\nonumber\\}
\newcommand{\N}{\nonumber}
```

Statt `\begin{eqnarray}` schreibt man dann nur noch `\BE` usw.
Die Abkürzung darf nur (große oder kleine) Buchstaben enthalten, selbst Ziffern sind nicht zulässig.
Wenn sich `latex` beklagt, dass ein Befehl, den man definieren will, schon existiert, dann muss man ihn duch `\renewcommand{..}{..}` überschreiben. Man kann auch Befehle mit Argumenten definieren...

B Einige Tricks beim Formelsatz

B.1 Horizontaler Zwischenraum

Die Kürzel

```
\!  \,  \:  \;  \quad  \qquad
```

erzeugen Zwischenraum von zunehmender Größe (beginnend mit *negativem* `\!`). Ein Beispiel:

`\int a x dx`	erzeugt	$\int axdx$
`\int a x \, dx`	erzeugt	$\int ax\,dx$

B.2 Text in Formelzeilen

Mit `\mbox{..}` schaltet man vorübergehend vom mathematischen in den Textmodus zurück und kann Text eingeben, der dann nicht den Regeln des Formelsatzes unterworfen ist. Oft wird man mit `\quad\mbox{..}\quad` etwas Platz um den Text herum schaffen, damit er nicht an den Formelteilen "klebt":

`f(x) = 0 \quad\mbox{f"ur}\quad x < 0` erzeugt

$$f(x) = 0 \quad \text{für} \quad x < 0$$

B.3 Lange Formeln

Um lange Formelausdrücke zu gestalten, ist `eqnarray` schon die richtige Umgebung, trotzdem ist es oft nicht einfach, eine befriedigende Platzaufteilung

zu erreichen. Ein (nicht sehr extremes) Beispiel:

$$\int dx\,(x+2)^2 \ln x \tag{4}$$
$$= \frac{1}{3}\left[(x+2)^3 \ln x - \int dx\,(x+2)^3/x\right] \tag{5}$$
$$= \left(\frac{1}{3}x^3 + 2x^2 + 4x\right)\ln x \tag{6}$$
$$-\frac{1}{9}x^3 - x^2 - 4x \tag{7}$$

Der Tabulator `& &` wird meist in der Form `&=&` benutzt, um die Gleichheitszeichen untereinander zu bringen, wie in Glg.(5), (6). Man kann aber auch ein leeres Paar `&&` zur Ausrichtung benutzen wie in Glg.(7). Die Gleichungszeile (4) läuft, weil sie als linke Seite zu lang schien, ganz "außer der Reihe", dazu wurde sie in `\lefteqn{..}` eingeschlossen (und ohne Tabulator geschrieben).

B.4 Klammergrößen anpassen

`[A + \frac{B}{C}]`	erzeugt	$[A + \frac{B}{C}]$
`\left[ A + \frac{B}{C} \right]`	erzeugt	$\left[A + \frac{B}{C}\right]$

Das Paar `\left .. \right` funktioniert allerdings nur bei Klammern, die in derselben Formelzeile stehen. Bei Formeln mit Zeilenumbruch (z.B. innerhalb von `eqnarray`) muss man "von Hand" anpassen durch Voranstellen von

```
\big  \Big  \bigg  \Bigg
```

B.5 Matrizen und Determinanten

Eine Matrix kann man schreiben als eine Tabelle in angepassten Klammern:

```
\begin{eqnarray*}
      \left( \begin{array}{cc}
            a & b \\
            c & d
      \end{array} \right)
\end{eqnarray*}
```

ergibt

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right)$$

Für ein Determinante klammert man die Tabelle mit `\left| ... \right|`, oder man erzeugt die senkrechten Linien mit der Spaltenbeschreibung, z.B. `\begin{array}{|cc|}`.

C Umflossene Abbildungen und Tabellen

Die Umgebungen `figure` und `table` platzieren ihren Inhalt so, dass der Text auf ganzer Breite unterbrochen wird. Manchmal will man aber die Abbildung/Tabelle an den Rand des Satzspiegels setzen, und der Text soll daran vorbeifließen. Dazu dient das Paket `wrapfig`, das die Umgebungen `wrapfigure` und `wraptable` bereitstellt. Schematisch sieht die Anwendung so aus:

```
\usepackage{wrapfig}
...

\begin{wrapfigure}{pos}{width}
      ...
      \caption{...}
\end{wrapfigure}
...

\begin{wraptable}{pos}{width}
      ...
      \caption{...}
\end{wraptable}

% pos = R r   right
%       L l   left
% R,L => floating vertical placement
% r,l => "here"
```

Literatur

[1] H. Kopka, LaTeX, Bd.1, Addison–Wesley, 1994.

[2] R. Wonneberger, Kompaktführer LaTeX, Addison–Wesley, 1988.

[3] `http://people.ee.ethz.ch/~dominikb/l2picfaq/l2picfaq.pdf`