Eine Einführung

Mathematik mit LaTeX

Jana Peters

Zusammenfassung

In dieser Einführung halte ich mich hauptsächlich an die Dokumentation der $\mathcal{AMS}$-Pakete. Der Leser sollte LaTeX installiert haben (und es sollte funktionieren). Außerdem sollte er bereits in der Lage sein einfache Dokumente mit LaTeX zu erstellen. Im letzten Kapitel werde ich einige Dinge über dieses Dokument selbst sagen, um Anderen zu erleichtern ebenfalls etwas Ähnliches zu erstellen.
Die ersten drei Kapitel sind eine grobe Einführung in den Mathematikmodus. Kapitel 2 **Gleichungssysteme und Matrizen** ist eigentlich nur der Vollständigkeit halber aufgeführt, weil man sowohl Gleichungssysteme als auch Matrizen mit dem $\mathcal{AMS}$-Paket besser machen kann. In Kapitel 4 steht einiges zu den Optionen der $\mathcal{AMS}$-Pakete. Dieses Kapitel kann man eigentlich auch überspringen.

Inhaltsverzeichnis

1 Der Mathematikmodus

Es gibt zwei grundsätzliche Möglichkeiten Formeln in einem Dokument darzustellen:

Beispiel Eine Folge (a_n) heißt konvergent, wenn es eine Zahl $a \in \mathbb{C}$ mit folgender Eigenschaft gibt: Zu jedem $\varepsilon > 0$ existiert ein $N \in \mathbb{R}$ so, daß

$$|a_n - a| < \varepsilon \quad \text{für alle } n > N$$

Eingebettete Formeln Eine Formel heißt eingebettet, wenn sie von Text umflossen wird. Dabei wird zum Umschalten zwischen Textmodus und Mathematikmodus das `$...$` benutzt:

```
Eine Folge $(a_n)$ heißt konvergent, wenn es eine Zahl $a \in \mathbb{C}$...
```

Abgesetzte Fomeln Abgesetzte Formeln setzen sich, wie der Name schon sagt, vom Text ab. Hier gibt es mehrere Möglichkeiten zwischen Text- und Mathematikmodus umzuschalten:

```
...so, daß \[ |a_n-a| < \varepsilon \quad \text{für alle } n > N \]
...so, daß $$ |a_n-a| < \varepsilon \quad \text{für alle } n > N $$
```

außer `\[...\]` und `$$...$$` gibt es noch eine weitere Möglichkeit (mehr sind mir nicht bekannt aber wer weiß): `\begin{displaymath}... \end{displaymath}` Allerdings finde ich die syntaxtechnisch zu aufwendig.
Der Befehl `$$...$$` ist veraltet und sollte nicht mehr benutzt werden, da es Schwierigketen mit Abständen geben kann.

Nummerierte Formeln Will man eine abgesetzte Formel mit Formelnummer versehen benutzt man `\begin{equation}...\end{equation}`:

$$|a_n - a| < \varepsilon \quad \text{für alle } n > N \tag{1}$$

also

```
\begin{equation}
|a_n-a|< \varepsilon \quad \text{für alle } n > N
\end{equation}
```

1.1 Text im Mathematikmodus

Manchmal ist es notwendig im Mathematikmodus zum Textmodus zurückzuschalten. Das ist bei eingebetteten Formeln kein Problem, aber bei abgesetzten:

```
\[a+b = c hier kommt Text in Formeln\]
```

ergibt

$$a + b = chierkommtTextinFormeln$$

Buchstaben werden im Mathematikmodus als Variablen betrachtet und *kursiv* gedruckt. Außerdem macht LaTeX im Mathematikmodus keine Abstände zwischen Buchstaben bzw. Variablen weil es mehrere Buchstabenkombinationen quasi als eine Variable betrachtet. Wenn man also normalen Text im Mathematikmodus schreiben will, muß man wieder in den Textmodus umschalten. Das geht mit dem Befehl `\text{...}`:

```
\[a+b = c \text{ hier kommt Text in Formeln}\]
```

ergibt

$$a + b = c \text{ hier kommt Text in Formeln}$$

2 Gleichungssysteme und Matrizen

2.1 Gleichungssysteme

Um Gleichungssysteme darzustellen, wobei alle Gleichungen an einem Zeichen (z.B. dem Gleichheitszeichen) ausgerichtet werden, gibt es die `eqnarray` - Umgebung. Man sollte `eqnarray` nicht mehr verwenden (warum später), aber der Vollständigkeit halber:

$$\begin{aligned} f_1(x) &= x_1 + x_2 + x_3 && (2)\\ f_2(x) &= a_2x^2 + a_1x + a_0 && (3)\\ u &= v + w && (4) \end{aligned}$$

```
\begin{eqnarray}
f_1(x) & = & x_1 + x_2 +x_3  \\
f_2(x) & = & a_2 x^2 + a_1 x + a_0 \\
u & = & v + w
\end{eqnarray}
```

2.2 Matrizen

Matrizen und Ähnliches kann man mit der `array` - Umgebung darstellen. Matrizen sollten jedoch lieber mit den `matrix`-Befehlen aus dem $\mathcal{AMS}$-Paketen dargestellt werden.

$$A = \left(\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array} \right)$$

```
A=\left(
\begin{array}{ccc}
a_{11} & a_{12}  & a_{13}\\
a_{21} & a_{22}  & a_{23}\\
a_{31} & a_{32}  & a_{33}
\end{array}\right)
```

mit den `\left, \right` -Befehlen kann man das Array mit verschiedenen Klammersymbolen umklammern. Will man die Klammer nur auf einer Seite, z.B. auf der rechten, ersetzt man `\left(` durch `\left.`. Ein anderes Beispiel: wegen den vertikalen Strichen kann man sowas nicht mit den `matrix`-Befehlen lösen:

	c_1	c_2	
d_1	a_{11}	a_{12}	h_1
d_2	a_{21}	a_{22}	h_2
	$T(g_1)$	$T(g_2)$	

```
$\begin{array}{c|c c| c}
 & c_1 & c_2 & \\
\hline
d_1 & a_{11} & a_{12} & h_1 \\
d_2 & a_{21} & a_{22} & h_2 \\
\hline
 & T(g_1) & T(g_2) & \\
\end{array}$
```

3 Exponenten, Indizes, ...

Hoch- oder tiefstellen kann man mit ^ und _, wobei dabei immer nur das direkt folgende Zeichen hoch- oder tiefgestellt wird. Mehrere Zeichen hoch- bzw. tiefstellen oder einen Exponenten indizieren kann man mit verschieden Klammerungen erreichen:

`$x^3$` x^3	`$x^2y$` x^2y	`$x^{2y}$` x^{2y}	`$x^{a^b}$` x^{a^b}
`$x_i$` x_i	`$x_{3+5}$` x_{3+5}	`$x_{i_3}$` x_{i_3}	
`$x^2_j$` x^2_j	`$x_j^2$` x^2_j	`$x_{i^2}$` x_{i^2}	`$x^{t_0}$` x^{t_0}

Vor allem braucht man das Hoch- bzw. Tiefstellen bei Integralen, Summen, Produkten, ...

$$\int_a^b f(x)dx \qquad \sum_{i=1}^{\infty} a_i \qquad \prod_{j=3}^{6} b_j$$

```
\int_a^b f(x)dx      \sum_{i=1}^\infty a_i      \prod_{j=3}^6 b_j
```

4 Das $\mathcal{AMS}$-Paket und seine Optionen

Die folgenden Pakete sollte man einbinden

```
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{amsthm}
```

Das `amssymb`-Paket stellt zum Beispiel Mengenbezeichnungen wie $\mathbb{R}$ zur Verfügung. Die mathematischen Schriftformen:

$\mathbb{R}$	`\mathbb{R}`	$\mathbf{R}$	`\mathbf{R}`	$\mathcal{R}$	`\mathcal{R}`	$\mathfrak{R}$	`\mathfrak{R}`
R	`\mathrm{R}`	R	`\mathsf{R}`	$\mathtt{R}$	`\mathtt{R}`	R	`\mathit{R}`

4.1 Die Optionen

Optionen hinter denen ein default steht sind standardmäßig eingestellt. Angegeben werden die Optionen folgendermaßen: `\usepackage[intlimits]{amsmath}`

- **sumlimits**(default) Setzt die obere und untere Grenze bei Summationszeichen und Ähnlichem ($\sum, \prod, \coprod, \bigotimes, \bigoplus, \dots$). Exklusive Integrale!
- **nosumlimits** Das Gegenteil von sumlimits.
- **intlimits** Wie sumlimits nur bei Integralen.
- **nointlimits**(default) Das Gegenteil von intlimits.
- **namelimits**(default) Wie sumlimits für sonstige Operatoren (z.B.: $\lim, \max, \dots$)
- **nonamelimits** Das Gegenteil von namelimits.

4.1.1 Beispiele zu den Optionen

Die kursiven Optionen sind jeweils default. Wie man sieht, braucht man eigentlich nichts zu ändern.

***sumlimits*:**

$$\sum_{n=0}^{\infty} \frac{1}{2^n} = 2 \qquad \prod_{i=0}^{m-1} n - i$$

nosumlimits:

$$\sum\nolimits_{n=0}^{\infty} \frac{1}{2^n} = 2 \qquad \prod\nolimits_{i=0}^{m-1} n - i$$

intlimits:

$$\int_0^a \sqrt{a^2 - x^2}dx \qquad \int_0^{\frac{\pi}{2}} \cos^2 t dt$$

***nointlimits*:**

$$\int\nolimits_0^a \sqrt{a^2 - x^2}dx \qquad \int\nolimits_0^{\frac{\pi}{2}} \cos^2 t dt$$

***namelimits*:**

$$\lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n = e$$

nonamelimits:

$$\lim\nolimits_{n\to\infty} \left(1 + \frac{1}{n}\right)^n = e$$

Diese Einstellungen der oberen und unteren Grenzen gilt nur in abgesetzten Formeln. In eingebetteten Formeln werden aus Platzgründen obere und untere Grenzen immer neben das Symbol gesetzt:

Beispiel Es sei (a_n) eine monoton fallende Nullfolge.
Dann gilt: die Reihe $\sum_{n=0}^{\infty}(-1)^n a_n$ konvergiert.

```
Es sei $(a_n)$ eine monoton fallende Nullfolge.
Dann gilt: die Reihe $\sum_{n=0}^\infty (-1)^n a_n$ konvergiert.
```

Natürlich kann man das händig ändern indem man den Befehl `\limits` bennutzt:
Es sei (a_n) eine monoton fallende Nullfolge. Dann gilt: die Reihe $\sum\limits_{n=0}^{\infty}(-1)^n a_n$ konvergiert.

```
Es sei $(a_n)$ eine monoton fallende Nullfolge.
Dann gilt: die Reihe $\sum\limits_{n=0}^\infty (-1)^n a_n$ konvergiert.
```

Sieht nicht schön aus, ist aber manchmal ganz nützlich.

Genauso funktioniert das mit allen anderen Operatoren auch. Die folgenden Optionen werden hinter `\documentclass` angegeben also z.B.: `\documentclass[leqno]{article}`

- **leqno** Die Formelnummern erscheinen links
- **reqno**(default) Die Formelnummern erscheinen rechts
- **fleqn** Die Formeln werden nicht zentriert sondern erscheinen am Rand ausgerichtet

Diese Optionen habe ich noch nie benutzt, weil die Voreinstellung in LaTeX eigentlich schon super aussieht. Es gibt noch einige weitere Optionen, die aber an anderer Stelle behandelt werden.

5 Formelsatz mit dem $\mathcal{AMS}$-Paket

5.1 Einführung

Das $\mathcal{AMS}$-Paket stellt eine Reihe von Umgebungen zur Verfügung:

`equation`	`equation*`	`align`	`align*`	
`gather`	`gather*`	`flalign`	`flalign*`	
`multline`	`multline*`	`alignat`	`alignat*`	`split`

Obwohl die `eqnarray`-Umgebung noch verfügbar ist, sollte man sie besser nicht benutzen, da LaTeX Schwierigkeiten mit der Formelnummer hat wenn die Formel sehr lang ist und weil die Abstände zwischen Formel und Gleichheitszeichen (bzw. anderen Zeichen an denen man ausrichtet) zu groß sind. Beispiele warum man `eqnarray` nicht mehr verwenden sollte später.

Außer zu `split` gibt es zu jeder Umgebung eine `*`-Version, die keine Formelnummern setzt. Bis auf `split` sind alle Umgebungen eigene mathematische Umgebungen, die `$...$` oder `\[...\]` nicht erfordern. `split` kann nur innerhalb anderer mathematischer Umgebungen außer `multline` verwendet werden.

Anders als `eqnarray` wird im Folgenden an Zeichen ausgerichtet (hier am Gleichheitszeichen) indem man das `&` nur vor das Gleichheitszeichen setzt. Weitere `&` haben andere Auswirkungen.

5.2 Einfache Formeln

Einzelne abgesetzte Formeln kann man mit `\begin{equation}...\end{equation}` oder mit `\begin{equation*}...\end{equation*}` setzen.
Dabei liefert `\begin{equation}...\end{equation}` eine Formel mit automatisch generierter Formelnummer (siehe auch auf Seite 1).
Da `\begin{equation*}...\end{equation*}` eine abgesetzte Formel ohne Formelnummer liefert, kann man eigentlich auch `\[...\]` benutzen.

5.3 Mehrzeilige Formeln

5.3.1 gather-Umgebung

Die `gather` - Umgebung zentriert die Formeln einfach nur, ohne sie auszurichten. Man braucht also auch kein `&`:

$$a_1 = b_1 + c_1 \tag{5}$$
$$a_2 = b_2 + c_2 + d_2 + e_2 \tag{6}$$

```
\begin{gather}
a_1 = b_1 + c_1 \\
a_2 = b_2 + c_2 + d_2 + e_2
\end{gather}
```

5.3.2 align-Umgebung

Die `align` - Umgebung richtet die Formeln nach einem beliebigen Zeichen (meistens das Gleichheitszeichen) aus. Hier mal eine `*` Variante:

$$a_1 = b_1 + c_1$$
$$a_2 = b_2 + c_2 + d_2 + e_2$$

```
\begin{align*}
a_1 & = b_1 + c_1 \\
a_2 & = b_2 + c_2 + d_2 + e_2
\end{align*}
```

Im Gegensatz zu `eqnarray` kann man auch mehrere Formelgruppen gestalten. Um einzelne Formelgruppen zu trennen benutzt man ein weiteres `&`:

$$a_1 = b_1 \qquad a_2 = b_2 \tag{7}$$
$$a_3 = b_3 \qquad a_4 = b_4 \tag{8}$$

```
\begin{align}
a_1 &= b_1 & a_2 & = b_2 \\
a_3 & = b_3 & a_4 & = b_4
\end{align}
```

Warum aber `eqnarray` durch `align` ersetzen wenn man nicht mehrere Formelgruppen gestalten möchte? Es gibt mindestens zwei Gründe: Einmal hat `eqnarray` bei langen Formeln Schwierigkeiten mit der Formelnummer:

$$a = b+v+c+e+f+g+r+s+w+r+cf+v+g+r+s+x+c+f+r+e+r+t+r+e+s \tag{9}$$

$$a = b+v+c+e+f+g+r+s+w+r+cf+v+g+r+s+x+c+f+r+e+r+t+r+e+s \tag{10}$$

```
\begin{eqnarray}
a&=&b+v+c+e+f+g+r+s+w+r+cf+v+g+r+s+x+c+f+r+e+r+t+r+e+s
\end{eqnarray}
\begin{align}
a&=b+v+c+e+f+g+r+s+w+r+cf+v+g+r+s+x+c+f+r+e +r+t+r+e+s
\end{align}
```

Und außerdem sieht `align` besser aus weil der Abstand zwischen Gleichheitszeichen und Formel bei `eqnarray` zu groß ist:

$$a = b$$
$$c = d$$

```
\begin{align*}
a &= b  \\
c & = d
\end{align*}
```

$$a \;=\; b$$
$$c \;=\; d$$

```
\begin{eqnarray*}
a &=& b  \\
c & =& d
\end{eqnarray*}
```

5.3.3 flalign-Umgebung

Wie `align`, nur werden die Formelgruppen am Rand ausgerichtet. (Hier wieder ohne Formelnummer weils schicker ist)

$a_1 = b_1$ $a_2 = b_2$
$a_3 = b_3$ $a_4 = b_4$

```
\begin{flalign*}
a_1 &= b_1 & a_2 & = b_2 \\
a_3 & = b_3 & a_4 & = b_4
\end{flalign*}
```

5.3.4 alignat-Umgebung

Die `alignat` - Umgebung lässt keinen Zwischenraum zwischen den einzelnen Formelgruppen bzw. gestattet es diesen Zwischenraum sebst festzulegen. `alignat` hat als Argument die Anzahl der Spalten:

$$a_1 = b_1 \qquad a_2 = b_2$$
$$a_3 = b_3 \qquad a_4 = b_4$$

```
\begin{alignat*}{2}
a_1 &= b_1  \hspace{2cm}& a_2 & = b_2 \\
a_3 & = b_3 & a_4 & = b_4
\end{alignat*}
```

Läßt man das `\hspace{2cm}` weg sieht es so aus:

$$a_1 = b_1 a_2 = b_2$$
$$a_3 = b_3 a_4 = b_4$$

5.3.5 multline und split

`Multline` und `split` gestatten es, eine Formel zu setzen die länger ist als eine Zeile und diese mit einer Formelnummer zu versehen.

multline

$$\begin{aligned} a+b+c+d+e+f+g+h+i+ & \\ +j+l+m+n+o+p+1+2+3+4+5+6+7 & \\ +q+r+s+t+u+10+1+2+3+4+5 & \\ +v+w+x+y+z & \end{aligned} \tag{11}$$

```
\begin{multline}
a+b+c+d+e+f+g+h+i+ \\
+j+l+m+n+o+p+1+2+3+4+5+6+7\\
+q+r+s+t+u+10+1+2+3+4+5\\
+v+w+x+y+z
\end{multline}
```

Die `multline`-Umgebung liefert nur eine Formelnummer die, wenn keine andere Option angegeben ist, rechts unten steht. Wenn man `leqno` als Option angegeben hat, steht die Formelnummer links oben.
Es ist in der `multline`-Umgebung also nicht nötig Formelnummern mit `\notag` oder Ähnlichem zu unterdrücken. Die erste Zeile in der `multline`-Umgebung ist am linken Rand ausgerichtet und die letzte zeile am rechten. Alle Zeilen dazwischen sind zentriert.

split

$$\begin{aligned} a &= b + c + d + e \\ &\quad - e - f \\ &= b + c + d - f \\ &= g + h + i \end{aligned} \tag{12}$$

```
\begin{equation}
\begin{split}
a & = b + c+d+e\\
& \quad - e - f\\
& = b+c+d-f\\
& = g +h + i
\end{split}
\end{equation}
```

Auch `split` ist dazu gedacht Formeln, die zu lang für eine Zeile sind, umzubrechen. Anders als `multline` gestattet es `split` die Formelteile aneinander auszurichten. Das geschieht wie bei `align` durch das `&`. `split` benötigt eine zusätzliche mathematische Umgebung (hier `equation`). `split` selbst liefert keine Formelnummerierung. Diese kommt von der zusätzlichen mathematischen Umgebung. Es ist also nicht sinnvoll `split` in `\[...\]` oder `align*` zu nutzen da man da auch nur `align*` nutzen kann. Der Vorteil von `split` liegt in der Formelnummer, die in der Mitte angeordnet wird (in unserem Beispiel zwischen der zweiten und dritten Zeile).

5.3.6 gathered, aligned und alignedat

`gather`, `align` und `alignat` nutzen die gesammte Seitenbreite so, daß es nicht möglich ist Bemerkungen neben die Formeln zu schreiben. Dafür gibt es `gathered`, `aligned` und `alignedat`. Ihre Gesamtbreite ist die Breite der Formel. Ein Beispiel:

$$\left.\begin{aligned} \frac{dB}{dt} &= -\nabla \times E \\ \frac{dE}{dt} &= \nabla \times B - 4\pi j \end{aligned}\right\} \quad \text{Maxwellsche Gleichungen}$$

```
\begin{equation*}
\left. \begin{aligned}
\frac{dB}{dt} & = -\nabla \times E \\
\frac{dE}{dt} & = \nabla \times B - 4 \pi j
\end{aligned}
\right\}\qquad \text{Maxwellsche Gleichungen}
\end{equation*}
```

`\left. ... \right\}` erzeugt die große geschweifte Klammer wobei `\left.` nur ein Dummy ist. Dazu mehr unter Sonstiges.

5.4 Sonstiges

5.4.1 Fallunterscheidung

Für Fallunterscheidungen stellt das `amsmath`-Paket die `cases`-Umgebung zur Verfügung:

$$f(x) = \begin{cases} \frac{1}{5} & \text{für } x \geq b \\ 0 & \text{für } x < b \end{cases}$$

```
\[f(x) =
\begin{cases}
\frac{1}{5} & \text{für } x \geq b\\
0 & \text{für } x < b
\end{cases}\]
```

Durch `&` werden die 'für' aneinander ausgerichtet

5.4.2 Umbrüche

Die mathematischen Umgebungen für mehrzeilige Formeln (z.B. `align`) mögen keine Leerzeilen. Will man aber zusätzliche Lehrzeilen einfügen geht das mit `\\[Anzahl]`:

$$a = b$$

$$c = d$$

```
\begin{align*}
a &= b  \\\\
c & = d
\end{align*}
```

Normalerweise ist es nicht erlaubt einen Seitenumbruch innerhalb einer mathematischen Umgebung zu machen. LaTeX macht das nicht von sich aus, so daß manchmal eine Formel komplett auf die nächste Seite gerückt wird oder die Seite sehr gequetscht aussieht. Möchte man aber, daß die Seite innerhalb einer Formel umgebrochen wird kann man das manuell mit dem Befehl `\displaybreak` vor dem entsprechenden `\\` erreichen. `\displaybreak` hat ein optionales Argument zwischen 0 und 4. `\displaybreak[0]` bedeutet: Es ist verboten hier umzubrechen (die Seite), `\displaybreak[4]` ist das gleiche wie `\displaybreak` und erzwingt einen Seitenumbrucht. Alles zwischen 0 und 4 sind abgeschwächte Wichtigkeiten. Möchte man ganz allgemein festlegen mit welcher Wichtigkeit LaTeX Formeln am Ende einer Seite umbrechen kann, kann man das global (vor `\begin{document}`) mit `\allowdisplaybreaks[1]` einstellen. Hier läuft das optionale Argument von 1-4. Dabei bedeutet `[1]`: erlaube Seitenumbrüche aber vermeide sie so gut es geht und 2,3,4 bedeuten jeweils erhöhte Verbotsstufen.
Wenn man mit `\allowdisplaybreaks` das Verhalten von LaTeX global eingestellt hat, kann man immernoch mit dem Befehl `\\*` einen Seitenumbruch manuell verhindern.
`\displaybreak` und `\allowdisplaybreaks` haben keinen Einfluss auf die Umgebungen: `split, gathered, aligned` und `alignedat`.

5.4.3 Unterbrochene Formeln

Manchmal möchte man eine Formelgruppe kurz unterbrechen um einen Umformungsschritt näher zu erklären oder etwas in der Art. Allerdings möchte man, daß die nachfolgenden Gleichungen (oder Ähnliches) weiterhin an den vorhergehenden ausgerichtet bleiben. Man kann also nicht einfach die Umgebung beenden, den Text schreiben und eine neue aufmachen.
Für solche Fälle gibt es den Befehl `\intertext{...}`

$$A_0 = b_0 \qquad A_1 = a_1 + b_0 b_1$$
$$B_0 = 1 \qquad B_1 = b_1$$

rekursiv weiter

$$A_k = b_k A_{k-1} + a_k A_{k-2}$$
$$B_k = b_k B_{k-1} + a_k B_{k-2}$$

```
\begin{align*}
A_0 & = b_0 & A_1 &=a_1 + b_0b_1\\
B_0 &=1 & B_1 &=b_1\\
\intertext{rekursiv weiter}
A_k &= b_k A_{k-1}+a_kA_{k-2} &&\\
B_k &= b_k B_{k-1}+a_kB_{k-2} &&
\end{align*}
```

5.4.4 Bezüge

Eine der größten Stärken von LaTeX (meiner Meinung nach) ist die Fähigkeit sich auf Dinge zu beziehen. Damit meine ich Dinge wie „siehe Formel (3.1) auf Seite 5“. Denn das Schöne ist, daß LaTeX ganz automatisch auch diese Bezüge ändert, wenn sich die Formelnummer oder die Seite ändert.
Um so einen Bezug herzustellen braucht man zwei Dinge: ein Label und eine Referenz. Das Label setzt man mit `\label{Name}` und beziehen kann man sich mit `\ref{Name}`. `\pageref{Name}` liefert die Seite auf der das Label gesetzt wurde und wenn man sich auf Formelnummern beziehen möchte gibt es da noch `\eqref`. Das liefert nicht nur die Nummer sondern auch gleich noch die Klammer mit.

Beispiel

$$a = 1 + 2 + 3 + 4 + 5 \tag{13}$$

Ich möchte mich jetzt auf Formel (13) auf Seite 10 beziehen.

```
\begin{align}\label{xy}
a&=1+2+3+4+5
\end{align}
Ich möchte mich jetzt auf Formel \eqref{xy} auf
Seite \pageref{xy} beziehen.
```

Zum Schluß noch eine Sache: Wenn man das Paket `\usepackage{hyperref}` und `\hyperbaseurl{.}` benutzt und dann pdfLaTeX nimmt hat man die Links im Acrobat auch zum anklicken. Warum das `\hyperbaseurl{.}` nötig ist weiß ich nicht aber ohne gehts nicht.
Sobald man Bezüge nutzt, muß man LaTeX mehrmals durchlaufen lassen.

5.4.5 Formelnummern

Zu diesem Thema kann man, glaube ich, beliebig viel schreiben aber da ich nicht sehr viel mit Formelnummern mache, wenn ich LaTeXe, halte ich das hier etwas kürzer. Es gibt einen Zähler der für die Formelnummern zuständig ist, der nennt sich `\theequation`. Den kann man mit `\renewcommand{\theequation}{irgendwas}` ändern.
Zum Beispiel möchte man die Formelnummern am Kapitel bzw. der section orientiern also $(1.1), (1.2), \ldots$ in Kapitel 1 und $(2.1), (2.2), \ldots$ in Kapitel 2, ...:
`\renewcommand{\theequation}{\thesection.\arabic{equation}}`

6 Matrizen, Punkte, Sonstiges

6.1 Matrizen

Siehe auch Seite 2. Dort wurde der `array`-Befehl vorgestellt, mit dem man auch Matrizen basteln kann. Besser und einfacher geht es aber mit xmatrix-Befehlen des $\mathcal{AMS}$-Paketes, weil `array` Probleme mit Abständen produzieren kann. Dabei ist x zu ersetzen durch: gar nichts, p, b, B, v oder V:

$$\begin{matrix} r & s & t \\ u & v & w \\ x & y & z \end{matrix}$$ matrix

$$\begin{pmatrix} r & s & t \\ u & v & w \\ x & y & z \end{pmatrix}$$ pmatrix

$$\begin{bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{bmatrix}$$ bmatrix

$$\begin{Bmatrix} r & s & t \\ u & v & w \\ x & y & z \end{Bmatrix}$$ Bmatrix

$$\begin{vmatrix} r & s & t \\ u & v & w \\ x & y & z \end{vmatrix}$$ vmatrix

$$\begin{Vmatrix} r & s & t \\ u & v & w \\ x & y & z \end{Vmatrix}$$ Vmatrix

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

```
\begin{pmatrix}
a_{11} & a_{12} & \ldots & a_{1n}\\
a_{21} & a_{22} & \ldots & a_{2n}\\
\vdots & \vdots & \ddots & \vdots\\
a_{m1} & a_{m2} & \ldots & a_{mn}
\end{pmatrix}
```

Im Gegensatz zu `array` muß man hier auch keine Spaltenangaben machen. Man hat bis zu 10 Spalten.
Besser gesagt: Die maximale Anzahl der Spalten wird durch den Zähler `MaxMatrixCols` bestimmt, den man auch ändern kann mit: `\setcounter{MaxMatrixCols}{12}`.

Außerdem gibt es noch den Befehl `\begin{smallmatrix} ... \end{smallmatrix}`. Dann bekommt man eine kleine Matrix, $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ die sich gut im Fließtext macht. Allerdings gibts dazu keine Klammern. Die muß man sich mit z.B. `\left( ... \right)` selbst drum machen.

Erwähnenswert in diesem Kapitel ist noch der Befehl `\hdotsfor{Anzahl der Spalten}`. Dieser produziert Punkte über die angegebene Anzahl von Spalten. Optional kann man noch den Raum zwischen den Punkten ändern: `\hdotsfor[Faktor]{Anzahl der Spalten}` Standard ist 1.0.

6.2 Punkte

$\mathcal{AMS}$ hat einen Standad festgelegt, was Punkte angeht. Das hat einmal den Vorteil, daß sich die Punkte an der richtigen Stelle befinden und außerdem ist aus dem Quelltext heraus ersichtlich. welche Punkte gemeint sind:

- `\dotsc` Punkte zwischen Kommas: $a, \dots, z$
  `$a, \dotsc ,z$`
- `\dotsb` Punkte zwischen binären Operatoren: $A_1 + \dots + A_n$
  `$A_1 + \dotsb + A_n$`
- `\dotsm` Punkte bei Faktoren: $A_1 A_2 \cdots$
  `$A_1A_2 \dotsm$`
- `\dotsi` Punkte bei Integralen

$$\int_{A_1}\int_{A_2}\cdots \qquad \texttt{\int\_\{A\_1\} \int\_\{A\_2\} \dotsi}$$

- `\dotso` sonstige Punkte

Sollte irgendjemand mal in die Verlegenheit kommen einen Text für $\mathcal{AMS}$ zu schreiben ist es auch vorteilhaft sich schonmal an diese Notation gewöhnt zu haben.

6.3 Umbruch verhindern

Manchmal möchte man nicht, daß LaTeX nach einem – die Zeile Umbricht. Z. B. bei ε–Umgebung. Um das Umbrechen nach dem Bindestrich zu verhindern gibt es den Befehl `\nobreakdash`: `$\varepsilon$\nobreakdash--Umgebung`

6.4 Sonstiges

$\acute{a}$	$\bar{a}$	$\breve{a}$	$\check{a}$
`\acute{a}`	`\bar{a}`	`\breve{a}`	`\check{a}`
$\grave{a}$	$\hat{a}$	$\hat{\hat{a}}$	$\tilde{a}$
`\grave{a}`	`\hat{a}`	`\hat{\hat{a}}`	`\tilde{a}`

$\sqrt[a]{b}$ `\sqrt[a]{b}`, $\dot{t}$ `\dot{t}`, $\ddot{t}$ `\ddot{t}`, $\dddot{t}$ `\dddot{t}`, $\ddddot{t}$ `\ddddot{t}`

$$\iiiint \quad \texttt{\iiiint} \qquad \int\cdots\int \quad \texttt{\idotsint} \qquad \oint \quad \texttt{\oint}$$

6.5 Umrandete Formeln

Es gibt den befehl `\boxed` um einen Rahmen um das Argument zu erstellen. Das funktioniert wie `\fbox` nur, daß der Inhalt im Mathemodus steht:

$$\boxed{e^{i\pi}+1=0}$$

```
\[\boxed{e^{i \pi}+1=0}\]
```

6.6 Pfeile

LaTeX stellt die Befehle `\overleftarrow` und `\overrightarrow` zur Verfügung um Pfeile über irgendwas zu LaTeXen:

$$\overrightarrow{abcdefg} \qquad \texttt{\overrightarrow\{abcdefg\}}$$

Außerdem gibt es noch den Befehl `\vec{}` der einen Vektorpfeil über das Argument schreibt. Der Unterschied zu `\overrightarrow`: $\vec{abcdefg}$ `\vec{abcdefg}`
$\mathcal{AMS}$ stellt zusätzlich noch die (selbsterklärenden) Befehle
`\overleftrightarrow, \underleftarrow,\underrightarrow, \underleftrightarrow`
zur Verfügung.

Für wachsende Pfeile gibt es folgende Befehle:
`\xleftarrow[unten]{oben}, \xrightarrow[unten]{oben}`:

$$A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n \pm i-1} C$$

```
\[A \xleftarrow{n+\mu-1}B\xrightarrow[T]{n \pm i-1}C \]
```

6.7 Brüche und Ähnliches

6.7.1 Brüche

Der Befehl um einen Bruch zu LaTeXen ist `\frac{Zähler}{Nenner}`. Bei eingebetteten Formeln wird der Bruch klein: $\frac{1}{2}$, bei abgesetzten entsprechend größer:

$$\frac{1}{2}$$

dargestellt. $\mathcal{AMS}$ ermöglicht es nun mit den beiden zusätzlichen Befehlen
`\tfrac{}{}, \dfrac{}{}` selbst zu bestimmen, ob ein Bruch in displaystyle oder in textstyle (also groß oder klein) dargestellt werden soll.

6.7.2 Binomialkoeffizienten

Auch für Ausdrücke wie $\binom{n}{k}$ hat $\mathcal{AMS}$ Befehle: `\binom{}{}, \dbinom{}{}, \tbinom{}{}`:

$$(a+b)^n = a^n + \binom{n}{1}a^{n-1}b + \cdots + \binom{n}{n-1}ab^{n-1} + b^n$$

```
\[(a+b)^n=a^n+\binom{n}{1}a^{n-1}b+\dotsb+\binom{n}{n-1}ab^{n-1}+b^n\]
```

6.7.3 Kettenbrüche

Um Kettenbrüche darzustellen ist es nicht schön mehrmals den Befehl `\frac{}{}` zu verwenden. Stattdessen gibt es den Befehl `\cfrac{}{}`. Dieser Befehl kann auch mit dem optionalen Argument `\cfrac[l]{}{}, \cfrac[r]{}{}` benutzt werden, das zur Folge hat, daß der Zähler rechts oder links ausgerichtet wird:

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}}}}$$

```
\[\cfrac[l]{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+ \cfrac[r]{1}{\sqrt{2}}}}\]
```

6.8 Operatoren

Mathematische Funktionen wie sin usw. werden in normaler Schrift gesetzt, um sie von Variablen zu unterscheiden. Die üblichsten Funktionen wie sin, cos, lim sind bereits vordefiniert. Häufig treten jedoch Funktionen auf, die noch nicht definiert sind. Mit dem Befehl `\DeclareMathOperator{\xxx}{xxx}` kann man sich selbst neue Operatornamen definieren. Diese werden dann in normaler Schrift gedruckt und es wird, wenn nötig, entsprechender Abstand hinzugefügt, so daß man mit `A\xxx B` Dann A xxx B bekommt und nicht AxxxB.

Der Befehl `\DeclareMathOperator*{\xxx}{xxx}` sorgt dafür, daß Indizes und Exponenten in abgesetzten Formeln unter bzw. über der Operatornamen gedruckt werden statt nur schräg oben bzw. unten wie es ja `_, ^` normalerweise tun:

$$\underset{123}{\mathrm{xxx}} \quad \texttt{\xxx\_\{123\}} \quad \text{im Gegensatz zu} \quad \mathrm{test}_{abc} \quad \texttt{\test\_\{abc\}}$$

Wobei die Operatoren test und xxx definiert wurden mit:
`\DeclareMathOperator*{\xxx}{xxx}, \DeclareMathOperator{\test}{test}`
Diese Befehle werden übrigens im Dokumentenkopf, also vor `\begin{document}` verwendet. Wenn man sich die Funktion nicht definieren möchte (weil man sie vielleicht nur einmal braucht oder so) dann gibt es noch den Befehl `\operatorname{abc}`, den man direkt im Mathemodus verwenden kann. Auch hier gibt es eine `*`-Variante für Grenzen.

6.8.1 mod und seine Verwandten

Es gibt verschiedene Möglichkeiten Modulo zu schreiben. Dafür gibt es auch verschiedene Befehle: `\mod, \bmod, \pmod, \pod`
`\mod` und `\pod` sind Varianten von `\pmod` wobei `\mod` die Klammern und `pod` das mod weglässt. `\bmod` und `\pmod` underscheiden sich im Abstand zwischen den Ausdrücken:

$\gcd(n, m \bmod n)$ `\gcd(n,m \bmod n)`, $x \equiv y \pmod b$ `x \equiv y \pmod b`

$x \equiv y \mod c$ `x \equiv y \mod c`, $x \equiv y \pod d$ `x \equiv y \pod d`

6.9 Multiple sub- und superscripts und sideset

Ganz nützlich sind `\substack` und `\subarray`:

$$\sum_{\substack{0 \leq i \leq m \\ 0 < j < n}} P(i,j) \qquad \texttt{\sum\_\{\substack\{0 \leq i \leq m \\ 0 < j < n\}\}P(i,j)}$$

`\substack` zentriert die Indizes, während `\subarray` sie links- bzw. rechtsbündig ausrichtet:

$$\sum_{\substack{i\in\Lambda \\ 0<j<n}} P(i,j) \quad \texttt{\sum\_\{\begin\{subarray\}\{l\}i\in\Lambda\\0<j<n \end\{subarray\}\}P(i,j)}$$

Superscripts funktionieren natürlich genauso.

6.10 Eigene mathematische Umgebungen

Das ist auch ein Thema, zu dem man beliebig viel schreiben kann. Es ist sicher sinnvoll, sich eigene Umgebungen schaffen zu können wie zum Beispiel

SATZ 6.1 Dies ist ein Satz

Beweis. Und dies der Beweis q.e.d.

```
\begin{satz}
Dies ist ein Satz
\end{satz}
\begin{proof}
Und dies der Beweis
\end{proof}
```

Die `proof`-Umbegung ist dabei schon vordefiniert. Ich habe hier lediglich den `\proofname` und das `\qedsymbol` neu definiert mittels:

```
\renewcommand{\qedsymbol}{q.e.d.}
\renewcommand{\proofname}{\textup{\textbf{Beweis:}}}
```

Zur Sache: Für die Definition eigener Umgebungen benötigt man das Paket `amsthm`. Eigene Umgebungen werden folgendermaßen definiert:

```
\newtheorem{satz}{\begin{large}SATZ\end{large}}[section]
\newtheorem*{bsp}{Beispiel}
```

Die `*`-Variante liefert eine Umgebung ohne Nummerierung. das `[section]` bedeutet, daß die Nummerierung der Sätze sich an der Kapitelnummer orientiert. Ohne dieses würde oben z.B. **SATZ 1** stehen.
Es gibt noch verschiedene Theoremstyles, die man sich auch selbst definieren kann mit:

```
\newtheoremstyle{citing}% name
  {20pt}%      Space above, empty = ‘usual value’
  {}%      Space below
  {}% Body font
  {}%         Indent amount (empty = no indent, \parindent = para indent)
  {\bfseries}% Thm head font
  {}%        Punctuation after thm head
  {.9em}%     Space after thm head: " " = normal interword space;
        %       \newline = linebreak
  {}% Thm head spec
```

Das muß man natürlich alles im Dokumentenkopf und vor der Definition der Umgebungen machen. den Style gibt man dann an mit `\theoremstyle{citing}`

Welche Theoremstyles es da so gibt und mehr Infos zu `amsthm` gibts in der Dokumentation (amsthdoc.pdf).

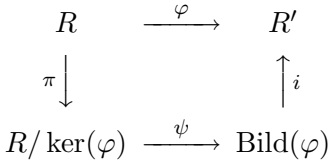
Noch eine kleine Anmerkung: Es gibt noch mehr Pakete die es erlauben, sich eigene Umgebungen zu definieren. Diese vertragen sich allerdings nicht miteinander. Also: Immer nur eins verwenden..

6.11 Kommutative Diagramme

Für kommutative Diagramme gibt es zwei empfehlenswerte Pakete.
Einmal `\usepackage{pictexwd,dcpic}` und `\usepackage{amscd}` . Mit dem $\mathcal{AMS}$-Paket `amscd` kann man einfachere kommutative Diagramme ohne Diagonalpfeile machen. Für kompliziertere Dinge muß man andere Pakete verwenden. Es gibt da mehrere, ich hab hier mal `dcpic`.

amscd

Hier erst mal ein Beispiel:



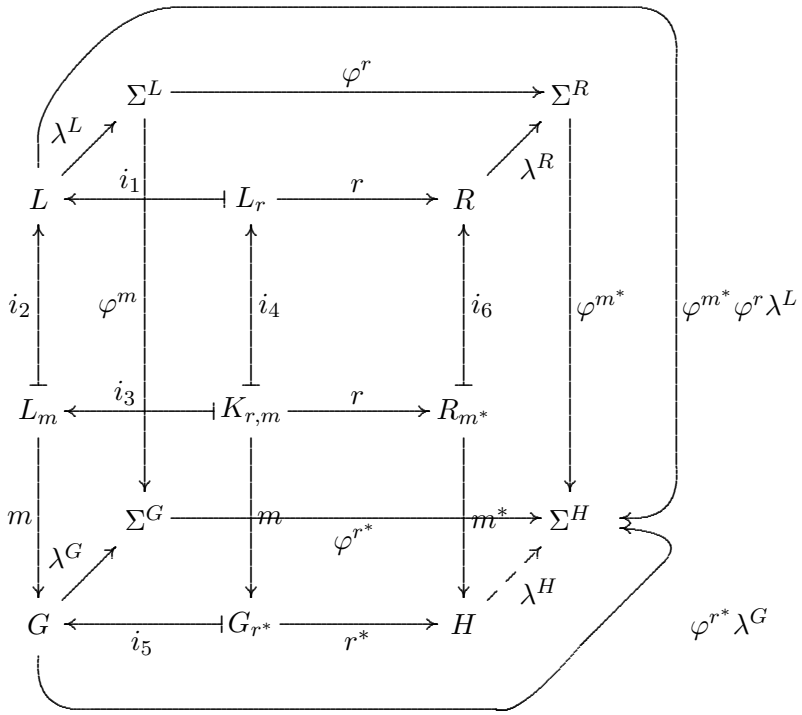

```
\begin{CD}
R @>{\varphi}>> R'\\
@V{\pi}VV @AAiA\\
R/\ker(\varphi) @>{\psi}>> \Bild(\varphi)
\end{CD}
```

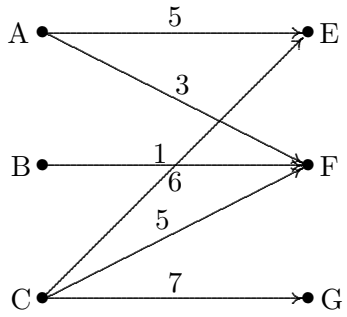
Die `CD`-Umgebung liefert die Befehle `@>>>, @<<<, @VVV, @AAA` für rechts-, links-, down- und up-Pfeile. `@=` und `@|` liefern horizontale oder vertikale Doppellinien. Einen Nullpfeil bekommt man mit `@.`.

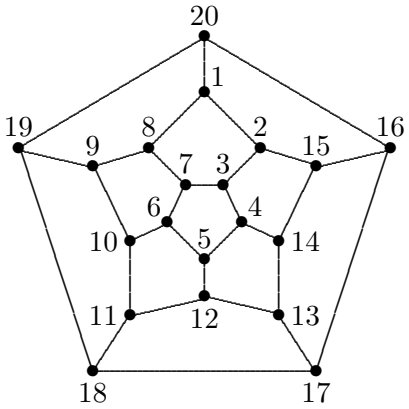
- Um etwas über einen Pfeil zu schreiben setzt man es zwischen das erste und zweite `>` bzw. `<`
- Um etwas unter einen Pfeil zu schreiben setzt man es zwischen das zweite und dritte `>` bzw. `<`
- genauso mit rechts und links neben die Pfeile.

dcpic

Das Paket ist mir zu kompliziert um es hier eingehend zu beschreiben. Die Dokumentation findet man auf:
`ftp://ftp.dante.de/pub/tex/macros/generic/diagrams/dcpic/`
Und hier ein paar Beispiele was man so machen kann:







Unter der URL findet man auch eine Beispieldatei, der ich diese Diagramme entnommen habe.

7 Dokumentendokumentation

Ich habe hier mal den Dokumentenkopf dieser Einleitung:

```
\documentclass[twoside, abstracton, titlepage]{scrartcl}

\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{amsfonts}
\usepackage{amsthm}
\usepackage{pictexwd,dcpic}
\usepackage{amscd}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\usepackage{fancyheadings}
\usepackage{hyperref}
\hyperbaseurl{.}

\setlength{\parindent}{0mm}
\setlength{\topmargin}{-2mm}
\setlength{\headheight}{3mm}
\setlength{\headsep}{5mm}
\setlength{\textheight}{232mm}
\setlength{\textwidth}{150mm}
\setlength{\oddsidemargin}{-1mm}
\setlength{\evensidemargin}{-1mm}

\newtheoremstyle{citing}% name
  {20pt}%      Space above, empty = 'usual value'
  {}%      Space below
  {}% Body font
  {}%         Indent amount (empty = no indent, \parindent = para indent)
  {\bfseries}% Thm head font
  {}%        Punctuation after thm head
  {.9em}%     Space after thm head: " " = normal interword space;
        %       \newline = linebreak
  {}% Thm head spec

\theoremstyle{citing}
\newtheorem{satz}{\begin{large}SATZ\end{large}}[section]
\newtheorem*{bsp}{Beispiel}
\newtheorem*{definition}{\underline{\underline{Definition}}}
\newtheorem*{bem}{\underline{Bemerkung}}

\renewcommand{\parallel}{\|}
\renewcommand{\sectionmark}[1]{\markboth {\thesection #1}}
\renewcommand{\qedsymbol}{q.e.d.}
\renewcommand{\proofname}{\textup{\textbf{Beweis:}}}

\DeclareMathOperator{\Bild}{Bild}
\DeclareMathOperator*{\xxx}{xxx}
\DeclareMathOperator{\test}{test}
\allowdisplaybreaks[1]

\pagestyle{fancy}

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

`\documentclass[twoside, abstracton, titlepage]{scrartcl}` Die Optionen `abstracton` und `titlepage` liefern folgendes: `abstracton` schreibt über den Abstract vor dem Inhaltsverzeichnis 'Zusammenfassung'. Kann man auch weglassen. `titlepage` macht es mir möglich die Titelseite zu erstellen:

```
\subject{Eine Einführung}
\title{Mathematik mit \LaTeX{}}
\date{ }
\author{Jana Peters}
\maketitle[-3]
\thispagestyle{empty}
\begin{abstract}
...
\end{abstract}
\tableofcontents
```

Das `{scrartcl}` ist meine Dokumentenklasse. Diese entspricht im Allgemeinen der normalen Klasse `article` allerdings benutze ich das KOMA-Skript. Näher Informationen zu KOMA-Script: `http://www.komascript.de/` KOMA-Script stellt Klassen zur Verfügung, die den Klassen `book, article, report, letter` ähnlich sind. Die KOMA-Script Klassen leiten sich aus den normalen Klassen dadurch ab, daß man vor den normalen Namen ein `scr` (steht für Script) setzt und, wenn der daraus resultierende Name länger als 8 Buchstaben ist, von hinten die Vokale rauslöscht. Für deutsche Skripte und Dokumentationen ist KOMA-Script besser geeignet als die üblichen LaTeX Klassen. Für mehr informationen über KOMA-Script empfehle ich einmal die Website (die istallerdings sehr konfus) oder aber die sehr ausführliche Dokumentation.

Die optionale Angabe `[-3]` bei `\maketitle[-3]` heißt, daß der interne Seitenzähler um 3 vermindert wird, damit das erste Kapitel mit Seite 1 beginnt.

Informationen zu den eingebundenen Paketen

`\usepackage{amsmath}`	allgemeines $\mathcal{AMS}$ Paket
`\usepackage{amssymb}`	u.A. für Dinge wie $\mathbb{R}$
`\usepackage{amsfonts}`	kann man glaub ich auch weglassen
`\usepackage{amsthm}`	für selbstdefinierte mathematische Umgebungen
`\usepackage{amscd}`	für 4-eckige kommutative Diagramme
`\usepackage{pictexwd,dcpic}`	für komplizierte kommutative Diagramme
`\usepackage[ngerman]{babel}`	neue deutsche Rechtschreibung, deutsches Inhaltsverzeichnis
`\usepackage[latin1]{inputenc}`	für ü,ä,ö
`\usepackage{hyperref}`	für die tollen Links im Inhaltsverzeichnis
`\usepackage{fancyheadings}`	für die schönen Kopf-Fußzeilen

die `\setlength...` Sachen definieren die Ränder und Abstände neu.
Zum Darstellen von Quellcode kann man die verbatim-Umgebung benutzen
(also `\begin{verbatim}...\end{verbatim}`) oder kürzer: `\verb+...+` dabei wird der

Text, der innerhalb der verbatim-Umgebung oder zwischen den +-Zeichen steht, so geschrieben wie er ist. Befehle werden nicht ausgeführt. Dabei kann statt des +-Zeichens auch jedes beliebige andere Zeichen verwendet werden, welches innerhalb der Umgebung nicht vorkommt (da ja sonst die Umgebung beim ersten Auftreten des Zeichens für beendet erklärt wird).