Wiskunde en Computers: LaTeX

B. Souvignier
(op basis van documenten van B. Polman)
((met aanpassingen door W. Bosma))

versie najaar 2005

Hoofdstuk 1

Inleiding

1.1 Tekstverwerking

Iedereen heeft vast wel eens een *tekstverwerkingssysteem* gebruikt om gedachten netjes op papier te zetten en vervolgens een mooie uitdraai te krijgen. Een van de standaard systemen is het Word programma onder het Windows besturingssysteem. Om goed te begrijpen wat de taken van een tekstverwerkingssyteem zijn is het instructief om even naar een heel ouderwetse methode te kijken:

Bij het traditionele boekdrukken levert de *auteur* een (soms handgeschreven) manuscript aan. Vervolgens bepaalt de grafisch ontwerper van de uitgeverij de layout, d.w.z. bladspiegel, regelafstand, lettertype (font), lettergrootte enz. Tenslotte zet de zetter het boek in lood en wordt het gedrukt. Het is dus duidelijk dat de ontwerper een groot aandeel in de opmaak van het boek heeft.

Als we het algemeen over het produceren van geschreven documenten (van korte notities over letters tot boeken) hebben, wordt duidelijk dat het wenselijk is dat de auteur naast de inhoud ook de opmaak kan bepalen. Dit wordt door het gebruik van tekstverwerkingssystemen mogelijk.

Het systeem dat we hier gaan bespreken (en gebruiken) heet LaTeX. Eigenlijk bestaat het uit twee delen, namelijk LaTeX, dat de rol van de ontwerper over neemt, en TeX dat door LaTeX als zetter gebruikt wordt. Maar in het algemeen krijgen we TeX helemaal niet te zien en hoeven we ons alleen met LaTeX te bemoeien.

1.2 Historie van LaTeX

TeX is door Donald E. Knuth geschreven uit frustratie over de slechte opmaak van zijn artikelen. Hiermee is hij begonnen in 1977 en de min of meer definitieve versie zoals we die nu nog gebruiken was klaar in 1982. TeX zit erg dicht tegen de zetmachine aan, veel van de commando's zijn tamelijk primitief en corresponderen één op één met echte zetconstructies in lood. Vanaf 1980 begon Leslie Lamport aan de ontwikkeling van een macropakket bovenop TeX om meer afstand te creëren tussen de auteur en de typografische details. Dit resulteerde in LaTeX, nu bekend als latex209. Vanaf '89 wordt er gewerkt aan het LaTeX3 project en het resultaat hiervan was LaTeX2e, een sterk verbeterde versie van LaTeX.

1.3 Nieuwe rol van de auteur

Omdat er geen ontwerper meer tussen auteur een publicatie zit, moet de auteur meer informatie in het document opnemen om structuur en opmaak aan te geven. Voorbeelden hiervan zijn:

- hier begint een hoofdstuk
- dit is een lijst
- dit is een stelling, formule, opmerking etc.

Dit gebeurt door extra commando's in de tekst op te nemen die de structuur bepalen. Op basis van een stijlkeuze bepaalt LaTeX vervolgens de layout. De invoer is daarmee vrijwel onafhankelijk van de uitvoer op papier. Zonder enige kennis van typografisch ontwerpen ben je dus in staat met LaTeX direct professioneel ogende uitvoer te verkrijgen. Dit in tegenstelling met de meeste tekstverwerkers waarbij de layout interactief bepaald wordt.

1.4 Voor- en Nadelen

1.4.1 Voordelen

- professionele layouts beschikbaar
- uitgeverijen hebben vaak eigen voorgedefinieerde layout
- mooie uitvoer van wiskundige formules en tabellen
- complexe structuren, voetnoten, literatuuropgave, verwijzingen, index, eenvoudig te maken
- veel uitbreidingen voor speciale zetproblemen (denk aan electrische schakelingen, chemische formules, muziek, schaakpartijen)
- redelijk makkelijk te leren
- nauwelijks bemoeienis met druktechnische details
- beschikbaar voor vrijwel alle computersystemen en meestal gratis (Linux)

1.4.2 Nadelen

- grote veranderingen aan voorgedefinieerde layouts is moeilijk
- invoer is ingewikkelder dan aanbrengen van layout in moderne tekstverwerker (met menubesturing)
- makkelijker om fouten te maken en foutmeldingen zijn vaak slecht te begrijpen

1.5 Werkcyclus

Omdat je in LaTeX tegelijkertijd tekst en commando's in een bestand hebt moet je zo'n bestand verwerken om er de bedoelde publicatie van te maken. Het programma dat hiervoor zorgt heet (geen verrassing) `latex` en produceert van een `.tex`-bestand een `.dvi`-bestand. Hierbij staat `dvi` voor *device independent* omdat het resulterende formaat op verschillende manieren (onafhankelijk van het systeem) verwerkt kan worden. Een van de mogelijkheden is, het `.dvi`-bestand met een *previewer* op het scherm te bekijken, het standaard programma hiervoor is `xdvi`. Met behulp van het `dvips` programma is het ook mogelijk, een `.dvi`-bestand in een *postscript*-bestand om te zetten. Hierbij kun je van de verschillende opties van `dvips` gebruik maken, bijvoorbeeld maar enkele pagina's produceren, de oriëntatie veranderen enz. Je kunt het `.dvi`-bestand natuurlijk ook meteen naar de printer sturen (als deze de goede driver heeft), maar dan moet je er wel zeker van zijn dat het al zo uitziet als je dat wilt. Een typische werkcyclus met LaTeX is dus als volgt:

- schrijven: `nedit` of `vi` *bestand.tex*
- verwerken: `latex` *bestand* of *bestand.tex*
- bekijken: `xdvi` *bestand* of *bestand.dvi*
- postscript maken: `dvips` *bestand* of *bestand.dvi*
- printen: `lpr` *bestand.ps* of *bestand.dvi*

Hoofdstuk 2

Eenvoudige teksten

2.1 De structuur van een LaTeX document

Een typisch LaTeXdocument ziet er als volgt uit:

> `\documentclass[`*options*`]{`*class*`}`
> *preambule*
> `\begin{document}`
> *de eigenlijke tekst*
> `\end{document}`

Hierbij kan de eigenlijke tekst natuurlijk ook commando's bevatten zo als we later zullen zien. De preamble is het stuk waarin de je allerlei instellingen kunt veranderen (bijvoorbeeld hoogte en breedte van de tekst), eigen commando's kunt definiëren en extra opties kunt laden.

Met de argumenten van `documentclass` wordt aangegeven wat voor een soort document je gaat produceren. Merk op dat er twee types van argumenten zijn: Argumenten in accolades (zoals *class*) zijn verplicht, argumenten in rechthoek hakjes (zoals *options*) zijn optioneel.

2.1.1 Klassen van documenten

Mogelijke waarden van het *class* argument bij `documentclass` zijn:

- `article`: voor gewone verslagen
- `report`: voor uitgebreide verslagen, rapporten, scripties (met hoofdstukken)
- `book`: voor de publicatie van boeken
- `letter`: voor brieven (met voorzieningen voor adres, signatuur enz.)
- `slides`: voor presentaties

2.1.2 Document class opties

Mogelijke waarden van het *options* argument bij `documentclass` zijn:

- `11pt, 12pt`: Voor een 11 punts letter of 12 punts letter. Zonder een van deze twee opties gebruikt LaTeX een 10 punts letter, wat in boekdruk de 'normale' grootte is. Omdat 10 punt de default is, mag je die niet als argument mee geven.
- `fleqn`: Voor links uitgelijnde in plaats van gecentreerde vergelijkingen.
- `leqno`: Om vergelijkings nummers links in plaats van rechts van de vergelijking te krijgen.

- `titlepage`: Om een 'losse' titel pagina te verkrijgen (in documentclass `article`).
- `notitlepage`: Om geen 'losse' titel pagina te verkrijgen (in documentclass `report` en `book`).
- `twocolumn`: Om een tweekoloms output te verkrijgen.
- `twoside`: Om het document dubbelzijdig te zetten (de linker en rechter pagina zijn verschillend, kijk eens naar de linker en rechter marge). In document class `book` is dit standaard. Daar kan de `oneside` optie gebruikt worden om dit niet te doen.
- `landscape`: Om het papier in de breedte te gebruiken, dus met de lange kant horizontaal. In het algemeen zal dan ook een speciale printopdracht gegeven moeten worden.
- `a4paper, a4, a4wide`: Om alles netjes op een A4-tje te krijgen (de amerikaanse papierformaten zijn anders).

2.2 Platte tekst

In principe wordt platte tekst net zo als op een schrijfmachine in getikt. Het eerste opmerkelijke verschil is dat LaTeX een nieuwe regel als een spatie beschouwt en dat meerdere spaties zo als één spatie worden behandeld. Om een nieuwe regel te krijgen, voeg je een `\\` of `\newline` in. Met `\\[1cm]` kun je zelfs te afstand tot te volgende regel bepalen.

Een lege regel betekent een nieuwe alinea, deze springt afhankelijk van de parameter `parindent` in. Dit is een voorbeeld van een aanpassing die je in de *preambule* maakt: Door hier de regel
`\setlength{\parindent}{0pt}`
in te voegen springen nieuwe alinea's in het hele document helemaal niet meer in. Lengtes mag je overigens met `1in` (inch), `2cm`, `3mm`, `4pc` of `5pt` aangeven (een inch is ongeveer 2.54cm, 1pc (pica) is 12pt en 72.27pt zijn 1 inch, dus is 1pt ongeveer 0.35mm).

Om een afstand tussen alinea's te creëren, gebruik je `smallskip`, `medskip` of `bigskip` maar je kunt ook met `\vspace{5cm}` een grote afstand maken.

Een paar verdere dingen om op te letten:

- Er zijn enkele karakters die een speciale betekenis voor LaTeX hebben en dus niet zo maar in de tekst gebruikt mogen worden, bijvoorbeeld: `$ & % # _ { }`. Deze worden verkregen door er een `\` voor te zetten. Verdere speciale karakters zijn: `~ ^ " \ | < >`.
- Aanhalingstekens worden met twee linkse en rechtse accenten gemaakt: ` ``hoi'' ` print als "hoi".
- Door streepjes achter elkaar te zetten worden streepjes van verschillende lengtes verkregen:
  - verbindingsstreepje: `e-mail` print als e-mail
  - reeks: `8--10` print als 8–10
  - gedachtestreepje: `ja---of nee?` print als ja—of nee?
  - in wiskundige formules is een min teken langer dan een verbindingsstreepje: -2 tegenover -2.
- Drie puntjes voor enz. worden niet door ... maar door `\ldots` verkregen: $a, b, c, \ldots, z$ ziet er beter uit dan $a, b, c, ..., z$.
- LaTeX denkt dat een punt altijd een zin afsluit en maakt dan een grotere afstand dan bij een gewone spatie tussen twee woorden. Dit is door een `~` te voorkomen, bijvoorbeeld print `D.~Knuth` als D. Knuth. Als je `\frenchspacing` in de preambule zet, maak je de afstand tussen zinnen gelijk aan de afstand tussen woorden.

2.3 Commando's

We hebben al een paar voorbeelden van commando's gezien, bijvoorbeeld `\documentclass`, `\ldots` of `\setlength`. Gemeenschappelijk aan alle commando's is de vorm `\` gevolgd door alleen letters. De commandonaam eindigt door de eerstvolgende spatie, cijfer of bijzonder teken. Kennis te verwerven van de commando's is eigenlijk de hoofdklus bij het leren van LaTeX, want hiermee controleer je de opmaak van je document. Een van de voordelen van LaTeX is dat je ook eigen commando's kunt definiëren. In zekere zin is LaTeX dus een programmeertaal en we zullen later zien hoe we eigen commando's kunnen maken.

Voorbeelden voor het gebruik van commando's zijn:

- het aangeven van structuur, bijvoorbeeld begint een nieuw hoofdstuk met `\chapter`
- het aangeven van symbolen, bijvoorbeeld maak je de Griekse letter α door `\alpha`
- als afkorting, bijvoorbeeld print `\LaTeX` als LaTeX
- het wijzigen van layout, bijvoorbeeld maakt `\large` een groter lettertype

Zoals we bij het `\documentclass` commando al hebben gezien mogen commando's argumenten hebben, deze zijn verplicht als ze in accolades staan en ze zijn optioneel als ze in rechthoekige haakjes staan.

Merk op dat een spatie na een commando door LaTeX'opgegeten' wordt, als je dus na een commando een spatie in de uitvoer wilt hebben, dan bereik je dit door `\LaTeX{}` of `\LaTeX\`.

2.4 Lettergroottes en -types

Je kunt in LaTeX natuurlijk ook het type ('font') en de grootte ('fontsize') van letters veranderen. Voor verschillende groottes van letters zijn er de volgende commando's:

`\tiny`	vreselijk klein schrift
`\scriptsize`	heel klein schrift (indices)
`\footnotesize`	klein schrift (voetnoten)
`\small`	klein schrift
`\normalsize`	normaal schrift
`\large`	groot schrift
`\Large`	groter schrift
`\LARGE`	heel groot schrift
`\huge`	reuzegroot
`\Huge`	gigantisch groot

De fontsizes blijven van kracht binnen de kleinst omhullende groep, die gewoonlijk door accolades wordt aangegeven. Om bijvoorbeeld S.O.S te krijgen, schrijf je `{\Large S O S}`. Voor grote stukken tekst kan je ook de `\begin{large} ... \end{large}` constructie gebruiken, hierdoor wordt de structuur van het document duidelijker.

Ook het type van fonts laat zich aanpassen. De fonts zijn georganiseerd in *families*, *series* en *shapes*, je kunt natuurlijk een familie met een shape combineren.

Commando	switch	kort	
`\textrm`	`\rmfamily`	`\rm`	normaal schrift (roman)
`\textsf`	`\sffamily`	`\sf`	'sans serif' schrift
`\texttt`	`\ttfamily`	`\tt`	`schrijfmachineschrift`
`\textmd`	`\mdseries`		medium schrift
`\textbf`	`\bfseries`	`\bf`	**vet schrift (boldface)**
`\textup`	`\upshape`		rechtopstaand schrift
`\textit`	`\itshape`	`\it`	*cursief schrift (italic)*
`\textsl`	`\slshape`	`\sl`	*schuin schrift (slanted)*
`\textsc`	`\scshape`	`\sc`	'SMALL CAPS' SCHRIFT
`\textnormal`	`\normalfont`		standaard schrift voor document

Het *emphasize* commando `\emph` schakelt automatisch tussen normaal en cursief schrift heen en weer; je krijgt dus met `\textit{een heel {\em mooi} resultaat}` het stukje *een heel* mooi *resultaat*.

Hoofdstuk 3

Structuur in LaTeX

3.1 Introductie

Dit document laat kort zien hoe je LaTeX commando's kunt gebruiken om een duidelijke structuur in je document aan te brengen. De keuze van document *class* en eventuele opties is bepalend voor welke structuurcommando's gebruikt kunnen worden en hoe de uitvoer er uit zal zien.

3.2 Sectie commando's

Voor het opdelen van je tekst beschikt LaTeX over de volgende commando's:

`\part` (Alleen te gebruiken in *book* of *report*.)

`\chapter` (Idem.)

`\section` Het meest gebruikte sectie commando.

`\subsection` Subsectie.

`\subsubsection` Subsubsectie.

`\paragraph` Paragraaf, meestal niet genummerd, komt niet in de inhoudsopgave en de titel wordt in de lopende tekst gezet.

`\subparagraph` Subparagraaf, idem als paragraaf.

Van al deze commando's bestaat ook een 'ster vorm', bijvoorbeeld
`chapter*{Index}`. Dit zorgt ervoor dat dit hoofdstuk niet genummerd wordt en niet in de inhoudsopgave wordt opgenomen.

3.3 Omgevingen

In tekst mode, worden de enumerate, itemize en description omgevingen waarschijnlijk het meest gebruikt. Deze kunnen allemaal genest gebruikt worden. In elk van deze omgevingen wordt het commando `\item` gebruikt om een nieuw item te beginnen. Daarbij kun je een optioneel argument achter `\item` zetten, tussen rechte haken, als je een ander label wilt dan wat standaard door de omgeving wordt gebruikt.

Een voorbeeld van *description* is al in sectie 4.1 gegeven.

3.3.1 Stellingen

Zie sectie 6.3 voor voorbeelden van het gebruik van de *theorem* omgeving, deze is natuurlijk voor wiskunde-teksten onmisbaar.

3.4 Wiskunde omgevingen

Voor wiskunde heeft LaTeX aparte omgevingen, er wordt daarbij onderscheid gemaakt tussen wiskunde tekst in de lopende tekst zoals bijvoorbeeld: $1 + 1 = 2$, en wiskunde die als een aparte formule wordt gezet in zogeheten 'display'-stijl,

$$i^2 = -1.$$

In plaats van de lange commandonamen bestaan hiervoor ook standaard afkortingen in LaTeX, `\(...\)` en `\[...\]` voor de 'math' en 'displaymath' omgevingen. Er bestaat zelfs nog een derde manier om deze twee omgevingen te openen en te sluiten, nml. $ voor 'math' mode en $$ voor 'displaymath' mode. Dit is een gevolg van het feit dat LaTeX een uitbreiding is op TeX, maar om verwarring te voorkomen (de $ om een omgeving te openen is niet te onderscheiden van de $ om de omgeving af te sluiten) kun je deze maar beter vermijden.

In wiskunde mode (LaTeX bevindt zich altijd of in tekst mode of in wiskunde mode) zijn er nog een groot aantal andere omgevingen die je kunt gebruiken voor allerlei speciale constructies, zie sectie 5.1 voor een aantal voorbeelden (en natuurlijk de handleiding).

3.5 Tot slot een goede raad

Zorg dat je de tekst overzichtelijk houdt, zodat het duidelijk is waar een omgeving begint en weer ophoudt. Daarbij is de hoofdregel om altijd alle wiskunde objecten in wiskunde mode te typen en de rest in tekst mode.

Beginners met LaTeX typen vaak dingen als

```
f(x)=sin($\sigma$(x) $x^2$) waar $\sigma$(x)=0
voor x in $\{x$: $x$ is even $\}$.
```

met als resultaat

f(x)=sin(σ(x) x^2) waar σ(x)=0 voor x in $\{x$: x is even $\}$.

(Zoals boven al opgemerkt, de $ is een andere manier om te switchen tussen tekst en wiskunde mode, hier zie je meteen een voorbeeld waarom dat niet verstandig is.) Dit is een erg onoverzichtelijk stukje en de kans op fouten wordt dan vanzelf groot. Het probleem is hoofdzakelijk dat de structuur niet helder is, wat zit in wat ? Het is net zo gemakkelijk om tekst binnen wiskunde (met het `\mbox` commando) te schrijven als andersom, wiskunde binnen tekst en het is helemaal niet ongebruikelijk om tekst-binnen-wiskunde-binnen-tekst te hebben of zelfs wiskunde-binnen-tekst-binnen-wiskunde-binnen-tekst. Bijvoorbeeld

```
\begin{math}
f(x)= \sin(\sigma(x)x^2)
     \mbox{ waar }
     \sigma(x)=0 \mbox{ voor \(x\) in }
                 \{x: \mbox{\(x\) is even} \}
\end{math}.
```

met als resultaat

$f(x) = \sin(\sigma(x)x^2)$ waar $\sigma(x) = 0$ voor x in $\{x : x \text{ is even}\}$.

Er is een uitzondering op de regel om een strikte scheiding tussen tekst en wiskunde te maken, het betreft interpunctie aan het eind van een 'mathdisplay', de punt of puntkomma is geen wiskunde maar moet toch in de wiskunde omgeving geplaatst worden. Als voorbeeld:

$$1 + 1 = 2.$$

Terwijl dit het resultaat is als je de punt buiten de ‘displaymath’ omgeving zet (wat op zich wel logischer zou zijn)

$$1 + 1 = 2$$

. Merk op dat dit geen probleem is voor de ‘math’ omgeving in $1 + 1 = 2$.

Hoofdstuk 4

Complexere structuren

4.1 Lijsten

Er zijn in LaTeX verschillende omgevingen om lijsten te maken: ongenummerd met `itemize`, genummerd met `enumerate` en met `description` voor eigen labels. De verschillende types van lijsten kunnen natuurlijk ook genest zijn, hierbij mag je 4 levels diep gaan.

Ongenummerde en genummerde lijsten hebben voor elk 'level' verschillende standaard labels, bijvoorbeeld bullets en streepjes voor de eerste twee levels bij ongenummerde lijsten.

Een typische lijst maak je dus zo:

```
\begin{itemize}
\item itemize
  \begin{itemize}
  \item het tweede level
  \end{itemize}
\item enumerate
\item description
\end{itemize}
```

print als

- itemize
  - de tweede level
- enumerate
- description

Je kunt elk label ook apart door een argument bij het `\item` commando aangeven; bij de `description` omgeving is dit zelfs verplicht.

```
\begin{itemize}
\item[$\Leftrightarrow$:] d.e.s.d.a.
\item[$\cong$:] is isomorf met
\end{itemize}
```

print als

$\Leftrightarrow$: d.e.s.d.a.

$\cong$: is isomorf met

Het systematisch veranderen van de labels in genummerde lijsten gebeurt typisch in de preambule. Hiervoor worden de parameters `\labelenumi, \labelenumii` enz. gewijzigd. Grote romeinse cijfers op het buitenste niveau en gewone cijfers in haakjes op de tweede level krijg je zo:

```
\renewcommand{\labelenumi}{\Roman{enumi}}
\renewcommand{\labelenumii}{(\arabic{enumii})}
\begin{enumerate}
\item het eerste punt
  \begin{enumerate}
  \item een onderpunt
  \end{enumerate}
\item het tweede punt
\end{enumerate}
```

I het eerste punt

(1) een onderpunt

II het tweede punt

Als je de wijze van nummering binnen je document verandert, geldt het nieuwe type binnen de kleinste omhullende groep, in de meeste gevallen dus tot het eind van het document. Om alleen maar één lijst anders te nummeren moet je de `\renewcommand` commando's een de lijst in accolades insluiten.

4.2 Tabellen

Een tabel wordt met behulp van de `table` omgeving gemaakt, bijvoorbeeld zo:

```
\begin{tabular}{c|l}       % bepaalt aantal kolommen en uitlijnen per kolom
                           % de | produceert verticale lijn
Student & Cijfer \\
\hline                     % produceert een horizontale lijn
002345 & 8.5 \\
002347 & 6.5 \\
002348 & 9 \\
\end{tabular}
```

Dit geeft de volgende tabel:

Student	Cijfer
002345	8.5
002347	6.5
002348	9

Achter de `\begin{tabular}` geef je in een verplicht argument voor elke kolom aan, of ze links uitgelijnd, rechts uitgelijnd of gecentreerd is (met `l`, `r` of `c`) en kun je door `|` aangeven dat er een verticale lijn tussen de kolommen (of aan de rand) is. Op een regel van de tabel zijn de elementen door `&` gescheiden, (voor lege velden schrijf je niets of een spatie tussen de `&`'s). Er moet dus één `&` minder zijn dan het aantal kolommen. Elke regel wordt door een `\\` afgesloten, tussen regels kan je met `\hline` een of meerdere horizontale lijnen produceren.

Vaak is het handig een tabel in een `table` omgeving in te sluiten. Dit heeft meerdere voordelen:

- Een `table` is een *floating* environment, LaTeX bepaalt zelf de geschikte plaats waar de tabel terecht komt. Je kunt met een optioneel argument een voorkeur aangeven, `h` voor *here*, `t` voor *top* en `b` voor *bottom*.
- Je kunt een label aan de tabel toevoegen om er later op te kunnen verwijzen.
- Je kunt met `\caption` een titel boven of onder de tabel krijgen.
- De tabel wordt automatisch genummerd en je kunt met `listoftables` een lijst van tabellen in je document krijgen.

Het volgende voorbeeld levert een veel gecompliceerdere tabel, waarbij ook het samenvatten van kolommen m.b.v. `\multicolumn` en het `\cline` commando gedemonstreerd worden.

```
\begin{table}[htb] \label{koffietabel}
\begin{center}
\begin{tabular}{|c|c|c|r|r|r|}
\hline
\multicolumn{6}{|c|}{{\bf Koffie consumptie van een wiskundige}}\\
                               % met multicolumn kun je een
```

```
                                % aantal kolommen samen nemen
\hline \hline                   % produceert twee horizontale lijnen
Jaargetijde & Sterkte & Prijs &
\multicolumn{3}{|c|}{Aantal koppen per}\\ \cline{4-6}
                                % cline maakt een horizontale lijn
                                % over de aangegeven kolommen
& Koffie & per kop & uur & dag & week \\
\hline
Lente  & 1.  & 0.50 & 0.23 & 1.84 & 9.201 \\
Zomer  & 0.8 & 0.50 & 0.2  & 1.59 & 8.01 \\
Herfst & 0.9 & 0.50 & 0.69 & 5.52 & 28.3 \\
Winter & 1.2 & 0.50 & 1.2  & 9.6  & 48.0 \\
\hline
\end{tabular}
\end{center}
\caption{Een typisch geval?}
\end{table}
```

Koffie consumptie van een wiskundige					
Jaargetijde	Sterkte	Prijs	Aantal koppen per		
	Koffie	per kop	uur	dag	week
Lente	1.	0.50	0.23	1.84	9.201
Zomer	0.8	0.50	0.2	1.59	8.01
Herfst	0.9	0.50	0.69	5.52	28.3
Winter	1.2	0.50	1.2	9.6	48.0

Tabel 4.1: Een typisch geval?

Aan deze tabel kunnen we nu met `\ref{koffietabel}` refereren, bijvoorbeeld om te vermelden dat Tabel 4.1 helemaal niet representatief is.

Soms is het ook handig om in een tabel sommige rijen te kunnen samenvoegen, daarvoor bestaat het pakket *multirow*, min of meer het rij equivalent van `multicolumn`.

Voor heel erg lange tabellen (meer dan een pagina) bestaat er ook het pakket *supertabular*.

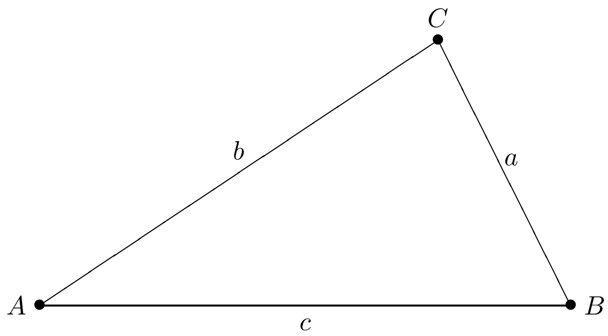
4.3 Een plaatje in de tekst

Het komt nog al eens voor dat je een plaatje in de tekst wilt opnemen. Dat kan op meerdere manieren, je kunt het plaatje maken met behulp van een van de tekenpakketten die als uitbreiding op LaTeX beschikbaar zijn (meest gebruikte zijn `epic`, `pstricks`, `xypic` en `texdraw`), of je maakt het plaatje met een extern programma, bijvoorbeeld `maple`, `xfig`, `gnuplot`.

Met het `epic` pakket maak je vooral eenvoudige plaatjes, die uit lijnen, symbolen en tekst bestaan. Het driehoek hieronder krijgen we bijvoorbeeld zo:

```
\begin{center}
\begin{picture}(200, 110)(0, 0)
\put(0,0){\makebox(0,0)[c]{$\bullet$}}
\put(-5,0){\makebox(0,0)[r]{$A$}}
\put(200,0){\makebox(0,0)[c]{$\bullet$}}
\put(205,0){\makebox(0,0)[l]{$B$}}
\put(150,100){\makebox(0,0)[c]{$\bullet$}}
\put(150,105){\makebox(0,0)[b]{$C$}}
```

```
\drawline(0,0)(200,0)
\put(100,-5){\makebox(0,0)[t]{$c$}}
\drawline(0,0)(150,100)
\put(75,55){\makebox(0,0)[b]{$b$}}
\drawline(200,0)(150,100)
\put(175,55){\makebox(0,0)[l]{$a$}}
\end{picture}
\end{center}
```

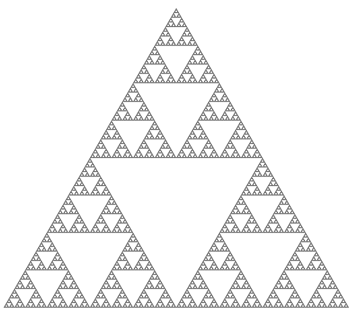


Omdat de meeste plaatjes te ingewikkeld zijn om ze zelf te maken, zullen we vervolgens alleen kijken naar hoe je een extern plaatje kunt gebruiken. Het gaat daarbij vrijwel altijd om `postscript`-plaatjes. Daarvoor is een buitengewoon krachtig macropakket beschikbaar: `graphicx`. Daarmee kun je een plaatje schalen, draaien, spiegelen, deels afdekken etc.

Het basis commando is `\includegraphics`, zie de documentatie in
`/vol/texlive/texmf/doc/latex/graphics/grfguide.dvi`
voor de volledige lijst van opties die je kunt gebruiken. De meest gebruikte vorm is als volgt

```
\begin{figure}[htbp]
\begin{center}
\includegraphics[0.3]{sierpiet}
\caption{Een fractal}
\end{center}
\end{figure}
```

De `\figure` omgeving zorgt net als de `table` omgeving ervoor dat LaTeX het plaatje mag verschuiven om een mooiere paginaverdeling te krijgen. Naast `[htb]` voor h(ere), t(op) of b(ottom) kun je ook `p` aangeven voor een aparte pagina met andere floating elementen, bijvoorbeeld andere plaatjes of tabellen. Als je wilt dat het plaatje persee op deze positie in de tekst moet komen gebruik je `[h!]`. Het resultaat van de boven aangegeven commando's is:



Figuur 4.1: Een fractal

4.4 Bestanden en conversies

Er is nog iets om op te letten als je externe plaatjes in je document hebt opgenomen: om het bestand te printen kun je nu niet uitgaan van het `dvi`-bestand. Dat komt omdat de postscript-plaatjes niet in het `dvi`-bestand zijn opgenomen maar alleen de verwijzing naar de plaatjes. Als je nu het `dvi`-bestand naar de printer stuurt kan deze de plaatjes vervolgens niet vinden. Om dit te ondervangen moet je zelf al de conversie naar postscript uitvoeren, in `xtem` is dat eenvoudig omdat de standaard instelling voor de printer dit al automatisch regelt. Als je zelf `lpr` gebruikt moet je eerst het commando `dvips` gebruiken.

```
dvips bestand.dvi -o nieuwbestand.ps
lpr -Pmath bestand.ps
```

De `-o` optie van `dvips` bepaalt de naam voor de uitvoer, zonder deze optie wordt het altijd de basisnaam met de `.ps` uitgang. Het resulterende postscript bestand kun je op het scherm bekijken met `gv`.

Tot slot meteen maar wat over de mogelijkheden om `pdf`-bestanden te maken. PDF staat voor Portable Document Format en is met name in de pc-wereld een steeds meer gebruikt bestandsformaat. Een mogelijkheid is, een `pdf`-bestand uit een `ps`-bestand te maken door het commando `ps2pdf bestand.ps` (het programma `ps2pdf` is een script dat is gebaseerd op `ghostscript`). Dit levert dan het bestand `bestand.pdf` op, maar de kwaliteit hiervan kan iets minder zijn.

Beter (en ook eenvoudig) is het om uitgaande van je LaTeX bestand direct een `pdf`-bestand te maken. In plaats van `latex bestand.tex` gebruik je daartoe `pdflatex bestand.tex`. Dit levert automatisch `bestand.pdf` op. Waarom ik dat hier vertel heeft te maken met het feit dat als er plaatjes in je bestand voorkomen je iets meer moet doen om een goed pdf-bestand te krijgen inclusief de plaatjes. Ten eerste moet je van al je postscript plaatjes `pdf`-versies maken met behulp van het commando `epstopdf plaatje.ps`. Dit produceert het bestand `plaatje.pdf`.

Vervolgens moet je nog de volgende twee regels in de preamble opnemen

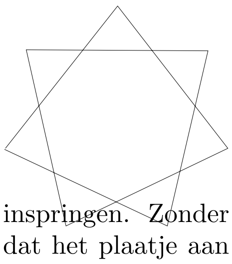
```
\usepackage[pdftex]{graphicx}
\DeclareGraphicsExtensions{.jpg,.pdf,.mps,.png}
```

in plaats van de regel

```
\usepackage[dvips]{graphicx}
```

Het resulterende `pdf`-bestand kun je op het scherm bekijken met `acroread` of `gv`. Je kunt het met behulp van `lpr` rechtstreeks naar de `math` printer sturen. Merk op dat het handig is om in het `\includegraphics` commando bij de naam van het plaatje geen uitgang te gebruiken. Op die manier hoef je alleen maar in de preamble een paar regels te veranderen om een `pdf`- dan wel een `dvi`-versie van het document te maken.

Er zijn nog heel wat pakketten die zich met plaatjes bemoeien, een mooi voorbeeld is het *wrapfigure* pakket, waarmee je de tekst om de plaatjes heen kunt laten vloeien.



Dit wordt heel eenvoudig door de volgende drie regels bereikt:

```
\begin{wrapfigure}[3]{l}{4cm}
\includegraphics[width=3cm]{ster}
\end{wrapfigure}
```

waarbij `ster.eps` een postscript-bestand met het plaatje is. Het eerste (optionele) argument geeft aan hoeveel regels er minstens moeten inspringen. Zonder dit argument gaat het door tot het eind van de pagina. Met `l` geef je aan dat het plaatje aan de linkerkant terecht komt, en de `4cm` geven aan aan hoe ver de tekst moet inspringen.

Hoofdstuk 5

Wiskunde

5.1 De *math*-mode

Wiskundige symbolen en formules worden gezet in *math*-mode:

- in lopende tekst binnen een alinea op een van de volgende manieren:
  - `zij $f(x)=x^2$, bereken $f'(x)$`
  - `als \( f''(x)>0 \) noemen we \( f \) convex`
  - `\begin{math} a^2+b^2=c^2 \end{math}`
- op een aparte regel op een van de volgende manieren:
  - `$$\sin^2(x) + \cos^2(x) = 1$$`
  - `De vergelijking \[ a x^2 + b x + c \] heeft als oplossing`
  - ```
    \begin{displaymath}
    x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}
    \end{displaymath}
    ```

De resultaten hiervan zijn

- zij $f(x) = x^2$, bereken $f'(x)$
- als $f''(x) > 0$ noemen we f convex
- $a^2 + b^2 = c^2$

en

- $$\sin^2(x) + \cos^2(x) = 1$$
- De vergelijking
$$ax^2 + bx + c$$
heeft als oplossing
- $$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Merk op dat spaties en regeleinden in math-mode geen betekenis hebben. Losse letters zijn variabelen, gebruik `\mathrm{tekst}` of `\text{ bla }` om gewone tekst te zetten in een formule. Lege regels zijn niet toegestaan, een display moet in één alinea.

In de voorbeelden zie je al commando's voor wiskundige formules. Sommige van deze commando's zijn redelijk voor de hand liggend, maar het is handig om ergens een tabel met de commando's voor de symbolen bij de hand te hebben.
    ```

5.1.1 Wiskunde formules

Met de `\begin{equation} ... \end{equation}` omgeving kunnen formules automatisch genummerd worden:

```
\begin{equation} \label{pyth}
c = \sqrt{a^2+b^2}
\end{equation}
Zie (\ref{pyth}) voor een andere formulering.
```

geeft als resultaat:

$$c = \sqrt{a^2 + b^2} \tag{5.1}$$

Zie (5.1) voor een andere formulering.

In dit voorbeeld is de nummering van de vorm $(n.i)$ waarbij n het nummer van het hoofdstuk en i een lopende index in dit hoofdstuk is. Deze stijl kan in de preambule veranderd worden. Let ook op de verwijzing.

Met de `eqnarray` omgeving laten zich ook meerdere formules groeperen:

```
\begin{eqnarray}
f(x) &=&\cos x \\
f'(x) &=& -\sin x\\
\sin x &=& x - \frac{x^3}{3!}
        + \frac{x^5}{5!} - \nonumber\\
    & & -\frac{x^7}{7!} + \cdots
\end{eqnarray}
```

geeft als resultaat:

$$\begin{array}{rcll}
f(x) & = & \cos x & (5.2) \\
f'(x) & = & -\sin x & (5.3) \\
\sin x & = & x - \frac{x^3}{3!} + \frac{x^5}{5!} - & \\
 & & -\frac{x^7}{7!} + \cdots & (5.4)
\end{array}$$

Als je wel de groepering van formules maar niet de nummering wilt gebruiken, kun je dit met `eqnarray*` bereiken.

5.2 Het gebruik van `amsmath`

$$\tau = \begin{cases} x^2 & \text{if } r-j \text{ is odd,} \\ x^4 & \text{if } r-j \text{ is even.} \end{cases}, \qquad \tau = \begin{cases} x^3 & x > 0 \text{if } r-j \text{ is odd,} \\ x^5 & x < 0 \text{if } r-j \text{ is even.} \end{cases}.$$

Een van de moeilijkste onderdelen van het prepareren van een wiskunde-tekst is het afbreken van lange formules over meerdere regels en het netjes onder elkaar zetten, uitlijnen, van een groep formules. Dit kan voor een groot deel niet geautomatiseerd worden, het is aan de auteur om ervoor te zorgen dat de formules er zo leesbaar mogelijk uitzien.

Met behulp van LaTeX's `eqnarray` and `eqnarray*` omgevingen heb je wel enige controle maar de mogelijkheden zijn beperkt en als je aan ziet komen dat je document veel en lange formules zal gaan bevatten dan is het aan te bevelen om de AMS pakketten te gebruiken.

5.2.1 Hoe gebruik je het AMS pakket

Wanneer je één van de AMS documentclasses gebruikt wordt het `amsmath` pakket automatisch geladen; anders moet je in de *preamble* het commando `\usepackage{amsmath}` opnemen. Dit voegt een groot aantal nuttige commando's toe waaronder de *alignment*-omgevingen.

1. `equation` — min of meer als normaal
2. `multline` — voor formules over meerdere regels
3. `split` — als multline, maar met extra controle over het uitlijnen
4. `gather` — voor het groeperen van meerdere vergelijkingen
5. `align` — idem maar met uitlijnmogelijkheden
6. `alignat` — vergelijkbaar met align maar met nog meer uitlijnmogelijkheden

Van de meeste bestaan er versies met en zonder ster—met ster vindt er geen automatische nummering plaats. Er zijn nog een paar verwante commando's, zoals: het `\intertext` commando voor het invoegen van tekst zonder het uitlijnen te verknoeien; de `subequations` omgeving, deze nummert een groep van vergelijkingen als *5a*, *5b*, *5c*, etc.; de `cases` omgeving voor definities met gevalsonderscheiding. We beschrijven hier niet in detail hoe deze commando's werken, zie daarvoor de documentatie van AMS, in plaats daarvan geven we in de volgende sectie een groot aantal voorbeelden van het gebruik. Je kunt snel van start door de uitvoer te vergelijken met de LaTeX source.

5.2.2 Voorbeelden

Veel van deze voorbeelden zijn direct ontleend aan de documentatie van AMSLaTeX, mocht je meer van de details willen weten dan is dat de aanbevolen tekst, deze kun je vinden in `/vol/texlive/texmf/doc/latex/amslatex`.

Gebruik van `equation*`:

$$a = b$$

Gebruik van `equation`:

$$a = b \tag{5.5}$$

Gebruik van `split` en `equation`:

$$\begin{split} a &= b + c - d \\ &\quad + e - f \\ &= g + h \\ &= i \end{split} \tag{5.6}$$

Gebruik van `multline`:

$$\begin{multline} a + b + c + d + e + f + b + c + d + e + f + b + c + d + e + f \\ + b + c + d + e + f + b + c + d + e + f + i + j + k + l + m + n \end{multline} \tag{5.7}$$

Gebruik van `gather`:

$$a_1 = b_1 + c_1 \tag{5.8}$$

$$a_2 = b_2 + c_2 - d_2 + e_2 \tag{5.9}$$

Gebruik van `align`:

$$a_1 = b_1 + c_1 \tag{5.10}$$

$$a_2 = b_2 + c_2 - d_2 + e_2 \tag{5.11}$$

Andere voorbeelden van `align`:

$$a_{11} = b_{11} \qquad a_{12} = b_{12} \tag{5.12}$$

$$a_{21} = b_{21} \qquad a_{22} = b_{22} + c_{22} \tag{5.13}$$

Gebruik van `flalign*`:

$$a_{11} = b_{11} \qquad a_{12} = b_{12}$$

$$a_{21} = b_{21} \qquad a_{22} = b_{22} + c_{22}$$

Gebruik van `\equation` en `\split`:

$$\begin{split} H_c = \frac{1}{2n} \sum_{l=0}^{n} (-1)^l (n-l)^{p-2} \sum_{l_1+\dots+l_p=l} \prod_{i=1}^{p} \binom{n_i}{l_i} \\ \cdot [(n-l)-(n_i-l_i)]^{n_i-l_i} \cdot \Big[(n-l)^2 - \sum_{j=1}^{p} (n_i-l_i)^2\Big]. \end{split} \tag{5.14}$$

Gebruik van `\align` voor het uitlijnen van textuele annotaties:

$$\begin{aligned} x &= y_1 - y_2 + y_3 - y_5 + y_8 - \dots && \text{by (5.2)} \qquad (5.15) \\ &= y' \circ y^* && \text{by (5.9)} \qquad (5.16) \\ &= y(0) y' && \text{by Axiom 1.} \qquad (5.17) \end{aligned}$$

Gebruik van `\aligned` voor intern uitlijnen

$$\begin{aligned} \alpha &= \alpha\alpha \\ \beta &= \beta\beta\beta\beta\beta \\ \gamma &= \gamma \end{aligned} \qquad \text{versus} \qquad \begin{aligned} \delta &= \delta\delta \\ \eta &= \eta\eta\eta\eta\eta\eta \\ \varphi &= \varphi \end{aligned}$$

"Cases" constructions:

$$P_{r-j} = \begin{cases} 0 & \text{if } r-j \text{ is odd,} \\ r!\,(-1)^{(r-j)/2} & \text{if } r-j \text{ is even.} \end{cases} \tag{5.18}$$

Gebruik van `\smash` en `\vphantom` om verticale afmeting te regelen:

$$\begin{aligned} \Big\langle u \,\Big|\, \sum_{i=1}^{n} F(e_i, v) e_i \Big\rangle &= \sum_{i=1}^{n} F(e_i, v) \langle u | e_i \rangle \\ &= \sum_{i=1}^{n} \langle u | e_i \rangle F(e_i, v) \\ &= \sum_{i=1}^{n} \overline{\langle e_i | u \rangle} F(e_i, v) \\ &= F\Bigg(\sum_{i=1}^{n} \langle e_i | u \rangle e_i, v \Bigg) = F(u, v), \end{aligned}$$

en zonder smash en vphantom

$$\left\langle u \,\middle|\, \sum_{i=1}^{n} F(e_i, v) e_i \right\rangle$$

(merk op dat zonder `\phantom` en `\smash` de haakjes te groot zijn vanwege de onder- boven-grens voor de som)

Gebruik van `\intertext`: *Bewijs:* (a) Given $\mathbf{v}$, we have

$$\begin{aligned} 1\,\mathbf{v} &= (-1)(-1)\mathbf{v} \\ &= (-1)(-1)\mathbf{v} + \mathbf{0} \\ &= (-1)(-1)\mathbf{v} + (\mathbf{v} + (-1)\mathbf{v}) \\ &= (-1)(-1)\mathbf{v} + ((-1)\mathbf{v} + \mathbf{v}) \\ &= ((-1)(-1)\mathbf{v} + (-1)\mathbf{v}) + \mathbf{v} \\ &= \mathbf{0} + \mathbf{v} \\ &= \mathbf{v}. \end{aligned}$$

(b) Again,

$$\begin{aligned} 0\,\mathbf{v} &= (1 + (-1))\mathbf{v} \\ &= 1\mathbf{v} + (-1)\mathbf{v} \\ &= \mathbf{v} + (-1)\mathbf{v} \\ &= \mathbf{0}. \end{aligned}$$

For (c),

$$\begin{aligned} \lambda\mathbf{0} &= \lambda(\mathbf{0} + (-1)\mathbf{0}) \\ &= \lambda\mathbf{0} + \lambda((-1)\mathbf{0}) \\ &= \lambda\mathbf{0} + (-1)(\lambda\mathbf{0}) \\ &= \mathbf{0} \end{aligned}$$

as required. $\square$

Hoofdstuk 6

Omgevingen

6.1 Theorem omgevingen

Omdat allerlei soorten stellingen, lemma's, gevolgen, definities etc. een veel voorkomend element zijn in wiskunde teksten is er een speciaal commando gemaakt waarmee je makkelijk dit soort omgevingen kunt maken. Het `\newtheorem` commando, wat je alleen in de *preamble* mag gebruiken, biedt de mogelijkheid om de naam te kiezen, de nummering te regelen en verschillende fonts te gebruiken voor de kop en de inhoud van de omgeving. Voor een volledige beschrijving van alle mogelijkheden zie `/vol/texlive/texmf/doc/latex/tools/theorem.dvi`.

```
\newtheorem{naam}[anderetheoremnaam]{tekst}[deel]
```

Hierin is `naam` de naam van de te definiëren omgeving, `anderetheoremnaam` optioneel een andere theorem omgeving waar de nummering in meegaat, `tekst` de tekst die voor het nummer komt, `deel` een optioneel documentdeel waarbinnen nummering opnieuw begint (bijvoorbeeld `chapter` of `section` – let op dat er geen \ voor `chapter` en dergelijke komt).

Bij het gebruik heeft de omgeving zelf ook nog een optioneel argument wat dan tussen haakjes achter het nummer wordt afgedrukt (zie 'Peter Principle' hieronder):

```
\newtheorem{stelling}{Stelling}[section]
{\theorembodyfont{\upshape}
\newtheorem{gevolg}[stelling]{Gevolg}}

\begin{stelling}[Peter Principle]
Iedereen stijgt tot zijn niveau van onbekwaamheid
\end{stelling}

\begin{gevolg}
Na verloop van tijd wordt iedere post bezet door
een werknemer die daar niet geschikt voor is
\end{gevolg}
```

6.1.0.1 Stelling (Peter Principle) *Iedereen stijgt tot zijn niveau van onbekwaamheid*

6.1.0.2 Gevolg Na verloop van tijd wordt iedere post bezet
door een werknemer die daar niet geschikt voor is

6.2 Eigen commando's (macro's)

Macro's zonder argumenten worden meestal gebruikt als afkorting, zoals

```
\newcommand{\Gi}{$\Gamma_i$}
```

dan werkt vervolgens `\Gi` om Γ_i te verkrijgen; maar beter is

```
\newcommand{\Gi}{\ensuremath{\Gamma_i}}
```

dat voorkomt problemen als je `\Gi` zowel in tekst- als in math-mode wilt gebruiken. De $ in de definitie sluit anders math mode af.

Je kunt ook macro's met (maximaal 9) argumenten maken:

```
\newcommand{\vect}[3]{\ensuremath{#1_{#2},
                \ldots,#1_{#3}}}
```

dan levert `\vect{a}{1}{n}` als resultaat op $a_1, \ldots, a_n$, wat zich weer laat afkorten door

```
\newcommand{\veca}{\vect{a}{1}{n}}
```

Namen van macro's mogen alleen letters bevatten.

Voordeel van macro's is interne consistentie in de tekst en de mogelijkheid om achteraf met alleen een herdefinitie van de macro objecten (bijvoorbeeld vectoren) anders weer te geven (met een pijltje erboven of vet afgedrukt).

Reeds bestaande commando's kunnen veranderd worden middels `\renewcommand`, maar wees daar voorzichtig mee!

6.3 Eigen omgevingen

Hier is een zelf-gedefinieerde omgeving `beschrijving`:

```
\newenvironment{beschrijving}[1]%
{\begin{quote}\emph{#1}:}          % begin-stuk
{\end{quote}}                      % eind-stuk
```

die dan als volgt gebruikt kan worden:

```
\begin{beschrijving}{Voorbeeld}
Dit is de binnenkant van de omgeving
niet een argument
\end{beschrijving}
```

Het resultaat zal dan zijn:

> *Voorbeeld*: Dit is de binnenkant van de omgeving, niet een argument

Dit lijkt dus veel op `newcommand`, maar belangrijk verschil is dat er een actie aan het eind uitgevoerd kan worden (om de omgeving weer af te sluiten). Meestal worden environments in termen van al bestaande environments gedefinieerd.

Argumenten mogen alleen in het beginstuk worden gebruikt !

6.4 Meer Omgevingen

Uitvullen:

- center:
  `\begin{center}In het midden\end{center}`

  In het midden

- flushleft:

```
\begin{flushleft}
Deze tekst is naar links\\
uitgelijnd
\end{flushleft}
```

Deze tekst is naar links
uitgelijnd

- flushright

```
\begin{flushright}
Terwijl\\
deze juist naar rechts is uitgelijnd
\end{flushright}
```

Terwijl
deze juist naar rechts is uitgelijnd

- verbatim

```
\begin{verbatim}
Voor het letterlijk weergeven van de inhoud in \tt.
Met name erg handig voor het weergeven van programma code:
   for(sum=0,i=1; i<10; i++) {
      sum += i^2;
   }
\end{verbatim}
```

```
Voor het letterlijk weergeven van de inhoud in \tt.
Met name erg handig voor het weergeven van programma code:
   for(sum=0,i=1; i<10; i++) {
      sum += i^2;
   }
```

Behalve als omgeving bestaat er ook een commando versie, `\verb`, die erg handig is voor een kort stukje, bijv. `\verb|\verb|`. In plaats van `|` mag je ook een ander teken gebruiken, bijv. een +.

6.5 Verwijzingen

`\label{naam}`	definiëert referentiepunt
`\ref{naam}`	gebruikt referentie
`\pageref{naam}`	geeft pagina nr waar referentie is gedefiniëerd.

Labels kunnen in iedere omgeving die door LaTeX genummerd wordt gebruikt worden, de labels worden weggeschreven in het .aux file en worden pas bij de volgende latex run gebruikt, overal waar dan een `\ref{naam}` voorkomt wordt het betreffende nummer gesubstitueerd. Als een naam niet gedefiniëerd is, krijg je een waarschuwing en in de uitvoer verschijnt een vraagteken.

De volgende LaTeX code geeft als resultaat de uitvoer eronder:

```
\setcounter{chapter}{1}
\setcounter{section}{4}
\section{Algoritmes}
Het bewijs daarvoor is in hoofdstuk~\ref{sec:bew} gevonden.

\subsection{Iteratie}\label{subsec:Iter}
Zie vergelijking~\ref{pythagoras} op pagina~\pageref{pythagoras} voor de
details.

\section{Bewijzen} \label{sec:bew}

\begin{equation}
  a^2+b^2=c^2 \label{pythagoras}
\end{equation}

Het bewijs daarvoor is in subsectie~\ref{subsec:Iter} gevonden.
```

1.5 Algoritmes

Het bewijs daarvoor is in hoofdstuk 1.6 gevonden.

1.5.1 Iteratie

Zie vergelijking 1.1 op pagina 23 voor de details.

1.6 Bewijzen

$$a^2 + b^2 = c^2 \tag{1.1}$$

Het bewijs daarvoor is in subsectie 1.5.1 gevonden.