LaTeX and WinEdt for MA4053

Stephen J Wills

Abstract

These notes provide a bare bones approach to the LaTeX typesetting system, and in particular using the WinEdt program to edit the final project report for the module MA4053.

Contents

1 What is LaTeX/TeX?

LaTeXis a powerful typesetting programme developed in the 1980s by Leslie Lamport (and others), which in essence is actually just a large number of macros for use by the lower level TeX programme written by Donald Knuth. It is *the* package for producing documents/papers/articles in the (pure) mathematical community.

Some of the advantages it has over other programmes include

- High quality professional looking output.
- It is (relatively) easily to create complicated mathematical formulae.
- Cross-referencing and bibliographies can be achieved painlessly.
- The programs are *free* and available on a wide variety of platforms. Moreover there is continual development by a large number of enthusiasts around the world, resulting in the production of many add-on packages that increase flexibility.
- The language emphasises the logical structure of the document under preparation, rather than getting too concerned with fancy fripperies...

This last point enables us to draw an analogy with HTML, the language in which (basic) web pages are written. There the layout is left to the particular browser the page is being displayed upon, with the author of the page describing only what fundamental features should be present. A well written web page should be viewable (and readable!) on any browser.

2 Processing documents

When creating a document you will end up dealing with a number of different files. Most important is the input file which will be named `MyFile.tex`. This is a plain ASCII text file and can be generated by any text editor. In our case we shall be using WinEdt, an editor specially developed for preparing LaTeX documents, and which consequently has a few additional useful features. Some other widely used editors have similar capabilities (e.g. emacs, which is available on a large number of platforms), but in reality one could use Notepad or Word if one was so inclined.

Having produced the `.tex` file, one then has two routes depending on the type of output required, as summarised in figure 1.

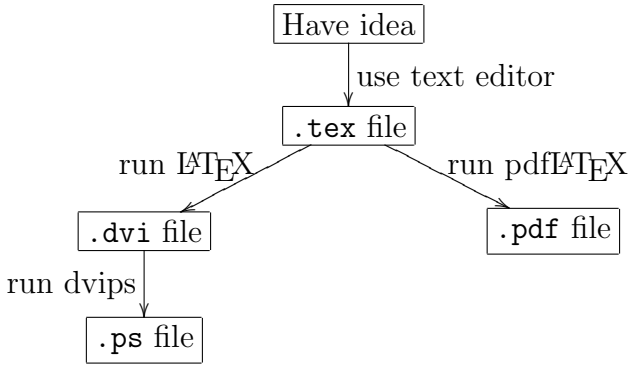


Figure 1: Steps in the production of a document

Route 1: Before the introduction of pdf files there was only POSTSCRIPT, as understood by printers. By running LaTeX on the `.tex` file, a `.dvi` file is produced, which can be viewed, enabling any editing that is necessary before it is converted to postscript using dvips, and then sent to the printer. Both the compilation to `.dvi` and conversion to `.ps` can be accomplished by clicking on the relevant buttons in WinEdt.

In fact this last step of conversion is largely superfluous in day-to-day use since it is possible to print from YAP, the default application for viewing `.dvi` files.

Route 2: Nowadays one can produce a pdf file direct from the `.tex` input by using the programme pdfLaTeX. Again this is done by a single click in WinEdt.

With either route you will end up with a number other files produced as LaTeX deals with your document, and for the most part you will not need to do anything with these. Two that will definitely turn up are `MyFile.aux` and `MyFile.log`. The first contains a lot of information about where equations and sections in your document are to be found — this is discussed further in Section 5. The second contains a record of all the things LaTeX had to say about the processing, in particular it contains any error messages that turn up. These helpfully list the line number in your `.tex` file that created the error which helps you to track down what is wrong — with the aid of the somewhat cryptic error message. Most often it is a missing `$` or `}`, or one environment is ended by another, through incorrect nesting or just plain forgetting to finish the previous environment. These comments should become more clear as we progress...

There is an alternative editor to be found on the School's computers, namely Scientific Workplace (or Word, the beefed up version). This is the

result of an attempt to produce a WYSIWYG version of LaTeX that feels a bit more like Microsoft Word. You are still editing LaTeX code, but it is hidden from you — you see something closer to the final product as you type. This has some advantages in that you do not have to learn so much about LaTeX at the outset, but in my opinion any such advantage is out-weighed by various disadvantages (it produces clunky code, moreover if you want to send the code to a native LaTeX user you need to do some fiddling; it is harder to make changes in overall style/control of the document; it is expensive...)

3 Text-only documents

3.1 Basic file structure

The `.tex` file is (essentially) always of the form

```
\documentclass[options]{article}
preamble
\begin{document}
text of document
\end{document}
```

Here the first line marks out the fact that we are dealing with a LaTeX 2_ε document, and that we are writing an article. The part `options` will contain a number of possibilities, e.g. whether we are using A4 paper, want the equation number to appear on the left or right, etc. Next comes the preamble to the file, rather than the article. That is, we next supply any further instructions that will be in force throughout the document such as user defined commands, or requesting the loading of packages. The template I have supplied to you contains a lot of stuff set up here already. Finally, the actually text is enclosed by the `\begin{document}`...`\end{document}` pair.

3.2 Spaces and special characters

The amount of space between words in a `.tex` document is largely irrelevant and ignored. For instance

```
Nicely typed input
```

and

```
Nicely          typed          input
```

both produce the same input. What is important is that a blank line indicates the end of the current paragraph. So while the two inputs above both produce

Nicely typed input

if we typed

```
Nicely


typed input
```

instead we would get

Nicely
typed input

Because of their use for commands etc., the following will not print as you would expect

```
\ & % { } ^ _ # $ ~
```

If you want to use any of the first nine of these symbols in some text or mathematics you must instead enter

```
\backslash \& \% \{ \} \^ \_ \# \$
```

respectively.

3.3 Breaking

LaTeX is programmed to work very hard to find the optimal place to break lines, pages etc., and so you are advised to leave it to get on with this task. However, when preparing the *final version* of you document, if you are not happy with a particular break, or if LaTeX cannot find a good place to break a line and needs help, then the following commands should be used:

```
\\   \newline        \newpage   \pagebreak[n]
```

The first two break the current line, and start the next without indentation. The final two break the page, with the first one leaving all text on the page where the break takes place together, the final one trying to spread the text out to fill the space. The `n` should be replaced with a number 0...4 depending on how much you want the break to take place (4 being the strongest request).

If you use long and/or unusual words then LaTeX may not know hoe to hyphenate it, and can be taught by input of the form `sjam\-bok`, the symbols `\-` showing where a break is permissible. If the unusual word will be used repeatedly then put `\hyphenation{sjam-bok}` in the preamble.

One place where typesetting conventions dictate that linebreaks should *not* take place is when a particular numbered Theorem or similar is being referred to. For instance there should not be a break between the 1 and the word Theorem when you type Theorem 1. To prevent this insert `~` between them, i.e. `Theorem~1`.

3.4 Quotes, dashes, dots and accents

Shift+2 should *not* be used to produce quotation marks. Instead for a single opening quotation mark use the symbol ‘ (found in the top-left of the keyboard) and a closing quotation mark is produced with ’. For double quotation marks use two of each! The difference between this and an erroneous use of shift+2 is shown by:

”Bad quote” and “Good quote”

Other important symbols are hyphens and dashes. There are three separate types of dash: an intraword dash, a numerical range, and a dash indicating a pause in the sentence. These are given by 1, 2 and 3 copies of - respectively. For example

bye-law 1–10 and That is true — in this case.

are produced by

```
bye-law     1--10    That is true --- in this case.
```

An ellipsis (the symbol . . .) is produced with `\ldots`, since typing . three times produces ... which does not have the correct spacing.

Accented letters such as ã, é and ô can be produced by the inputs `\~{a}`, `\'{e}` and `\^{o}`.

3.5 Fonts and sizing

By default LaTeX documents are set in Computer Modern fonts, which look fine and I would not encourage you to go down the route of trying to change this. Times Roman, Courier, etc. are available, as are a myriad of foreign scripts (Russian, Korean,. . .). More important at the moment is the ability to emphasis words and change size. The follow commands achieve a change in the shape:

`\textit`	*Italics*	`\textbf`	**Bold face**
`\textup`	Upright	`\textrm`	Roman
`\textsf`	Sans serif	`\texttt`	`Typewriter style`

They can be combined in obvious ways. For example **bold sans serif** is achieved by typing `\textbf{\textsf{bold sans serif}}`. Most of the time the default is for roman upright text so these two commands are not immediately useful, except in the statements of theorems and the like, when the default is italics. There is another command, `\emph`, that can be used to emphasis text. In general this achieves the same effect as using `\textit`, but

the philosophical point is that you should be able to make a global change at a latter stage if you decided for instance that emphasis would be better generated by bold sans serif or underlined typewriter style. See the subsection on defining commands later on for more information.

To change the size of text there are the following commands, in order of decreasing size:

```
\tiny        \scriptsize  \footnotesize  \small
\normalsize  \large       \Large         \LARGE
\huge        \Huge
```

For example the input

```
{\Large Starting big} and {\small getting} {\tiny smaller}
```

produces

{\Large Starting big} and {\small getting} {\tiny smaller}

3.6 Environments

To insert lists, theorems, tables etc. requires the use of the appropriate *environment*. The text for such an object is enclosed in a `\begin{environment}` `\end{environment}` pair. For example there are three default styles of list which are `itemize`, `enumerate` and `description`. The first produces bullet points or equivalent for each item in the list, the second numbers the entries, and the third permits user defined labelling. For example the list

1. Item 1
2. Item 2
3. Item 3

is produced by

```
\begin{enumerate}
\item Item 1
\item Item 2
\item Item 3
\end{enumerate}
```

These lists can be successively nested, giving rise to new numbering or bulleting styles in the different levels. For example two layers of `enumerate` gives

1. Item 1
    (a) Subitem 1 of first item
    (b) Subitem 2 of first item
2. Item 2

The standard labelling in `itemize` and `enumerate` can be overridden by input of the form `\item[New label]`, and this is how the user defined labels are given in the `description` environment. Two further list types are provided by the template which are both variants on the `enumerate` environment. One is `alist` which labels the items (a), (b) etc., and the other is `ilist` which labels the items (i), (ii) etc. These save having to type `\item[(a)]` etc.

The `center` environment and `quote` environment are useful for displaying tables and quotes in text documents. To produce tables use the `tabular` environment. This begins with a command like `\begin{tabular}[cr|lr|]`, where the letters `l`, `c` and `r` indicate the number of columns and whether the entries should be left justified, centred or right justified. An upright line indicates that there should be a vertical line separating the columns. To insert horizontal lines use `\hline` at the end of the preceding line of entries. The entries themselves are separated into the relevant columns by `&` in the input, and the end of the line is indicated by `\\`. For example the input

```
\begin{center}
\begin{tabular}{|c|lr} \hline
{\large Centred} & {\large left} & {\large right} \\
{\small centred} & {\small left} & {\small right} \\ \hline
\end{tabular}
\end{center}
```

produces

Centred	left	right
centred	left	right

Not enclosing the `tabular` environment in the `center` environment will cause the table to be set in the middle of the current paragraph.

The most important environments that you will need for the project write up are the so-called *theorem-like* environments. Since it is impossible to know from the outset what is required in a given document in terms of theorems, propositions etc., there is a means by which the user specifies the types required in the preamble of the document. I have done this in the given template, creating the following environments:

```
thm  propn  lemma  cor  defn  note  rem  rems
```

for producing a Theorem, Proposition, Lemma, Corollary, Definition, Note, Remark or Remarks respectively. The first five will also produce numbering, and are numbered consecutively together. It is possible to have Theorems numbered on one scale, Propositions under another etc. The first four italicise the text inside them, and the final four leave it upright.

So the inputs

```
\begin{thm}          \begin{rem}
$E = mc^2$           General relativity is complicated
\end{thm}            \end{rem}
```

produce

Theorem 3.1. $E = mc^2$

and

Remark. General relativity is complicated

respectively. The differing possibilities for italic text, bold or italicised labeling etc., are implemented by AMS-LaTeX, which also provides as standard the `proof` environment. This writes *Proof.* at the start of your proof and the □ symbol at the end.

3.7 Large scale structure

Any document tends to have a large scale structure starting with front matter (such as a title, author and so on) before splitting the document into chapters, sections, etc., possibly with appendices, and then ending with a bibliography and maybe an index.

The front matter for your project can be achieved by filling in the blanks in

```
\title{}
\author{}
\maketitle

\begin{abstract}

\end{abstract}
```

which appears just after the `\begin{document}` command.

The appropriate divisions within a document are produced by commands such as

```
\chapter   \section   \subsection   \subsubsection
```

— this section was started by the input `\section{Text-only documents}`. In fact the commands available depend on the declaration of document type contained in the `\documentclass` line at the start of your `.tex` file. I have set up the template as an `article`, so the highest level section is in fact `section`, rather than `chapter`. LaTeX also generates the numbers here automatically, although it is possible to control just how far down the parts are labelled through use of the `\secnumdepth` command. Bibliographies are dealt with in section 5.

3.8 Defining commands etc.

When producing a document it is common to find that certain constructs are used repeatedly, and it would involve a lot of repetitive typing each time the construct appears. Time can be saved by creating user defined commands in the preamble. For example in one set of lecture notes that I prepared I decided that any term that was being defined should be set in bold sans serif, which normally would involve typing `\textbf{\textsf{`*term*`}}` each time. Instead I included the line

```
\newcommand{\dn}[1]{\textbf{\textsf{#1}}
```

in the preamble. The `\dn` in the curly brackets indicates that I am defining a command called `dn`. The 1 in the square brackets indicates that there is to be one input into the command. Finally, the actual command appears enclosed in the final set of brackets, with the `#1` indicated the point where the (first) input should be used. So now `\dn{topological space}` produces **topological space**. A philosophical viewpoint on the use of such constructs is that if at a later stage I decide to change how a defined term is displayed then I need only change the definition of the command `\dn`.

If you try to create a command with a name that is already in use then LaTeX will complain. If you really want to use that name then there is the `\renewcommand` but this is a somewhat dangerous path to go down. It is not necessary to have inputs in the command. For example rather than type `\begin{thm}` and `\end{thm}` at the beginning and end of each of your theorems you could put

```
\newcommand{\bt}{\begin{thm}}
\newcommand{\et}{\end{thm}}
```

in the preamble, and then need only type `\bt` and `\et`. You can also create new environments in a similar way.

4 Inputting mathematics

4.1 Text and math modes; spacing

There are two main modes in which LaTeX works. So far we have only made use of text mode, but more importantly there is also math mode used for inputting mathematics — the reason we are dealing with the program in the first place! There are two ways of including mathematics in a document. It can either be placed *in-line*, that is in the middle of the paragraph, or can be *displayed*. The first is used for relatively small portions of mathematics, in particular if they are not being emphasised. For large equations or for emphasis one should use displayed equations.

Mathematical text within a paragraph is obtained by enclosing the required commands between `$` and `$`, or `\(` and `\)`, or between `\begin{math}` and `\end{math}`. So for instance

```
And Pythagoras said $x^2+y^2=z^2$, and all was right-angled.
```

produces

> And Pythagoras said $x^2 + y^2 = z^2$, and all was right-angled.

On the other hand, if we wanted to emphasis the equation then the mathematics should appear between `\begin{equation}` and `\begin{equation}` if the equation is to be numbered, or between `\[` and `\]`, or `$$` and `$$`, or `\begin{displaymath}` and `\end{displaymath}`, or `\begin{equation*}` and `\end{equation*}` if the equation is not to be numbered. So our example above could be entered as

```
And Pythagoras said
\[
x^2+y^2=z^2,
\]
and all was right-angled.
```

to produce

> And Pythagoras said
> $$x^2 + y^2 = z^2,$$
> and all was right-angled.

Letters appearing in math mode are processed by LaTeX as if they are variables, and are usually typeset in italics. However the spacing between them differs from that in text mode, and so a `$...$` should not be used as a

shorthand for `\textit`. For example `\textit{different}` and `$different$` produce the outputs *different* and $different$ respectively.

It is common to have a few words in a given formula. Indeed, certain journals are keen that the symbol $\forall$ should not appear in definitions and displayed equations, but should be replaced by 'for all'. When dealing with in-line mathematics you could come out of math mode and re-enter it after the text part is dealt with. This is not possible in displayed equations, nor in some circumstances with in-line mathematics. A more robust way of doing things is to enclose the text part inside `\text{...}`. For example

```
\begin{equation}
\exists x,y,z \in \mathbb{Z} \text{such that}
x^{101}+y^{101}=z^{101}
\end{equation}
```

produces

$$\exists x, y, z \in \mathbb{Z} \text{such that} x^{101} + y^{101} = z^{101} \tag{4.1}$$

This example illustrates another problem/feature of entering mathematics. As with text mode, spaces between parts of the input are ignored. Indeed, in math mode they do not even produce a space. One remedy in the above would be to change `\text{such that}` to `\text{ such that }`, since the spaces inside the `\text` command are *processed in text mode* and so are not ignored. There can be a need to alter/adjust the spacing between mathematics and commands for doing this include, in increasing order

`\!`	thin negative space	`\,`	thin space
`\:`	medium space	`\;`	thick space
`\␣`	interword space	`\quad`	space width of M
`\qquad`	biggest space		

4.2 Basic constructs

The following are a non-exhaustive list of common constructs in mathematics, and how to obtain them in LaTeX.

Super- and subscripts

These are produced by `^` and `_` respectively. For example the inputs `$x^2$` and `$a_{13}$` produce x^2 and a_{13}. Note that the curly brackets are required around the `13` to ensure that both numbers are used as the subscript. Without them we get $a_1 3$.

Roots

`$\sqrt{x}$` produces $\sqrt{x}$. To take nth roots use `$\sqrt[n]{x+iy}$` to get $\sqrt[n]{x+iy}$.

Lines, braces, accents

To get a line over the top of some mathematics, enclosed the required code inside `\overline{...}`. For example `$\overline{x+iy} = x-iy$` produces $\overline{x+iy} = x - iy$. The command `\underline` works similarly. There are also the commands `\overbrace` and `\underbrace` which can be combined usefully with super- and subscripts. For example

$$(a^m)^n = \underbrace{(\overbrace{a \cdots a}^{m}) \cdots (\overbrace{a \cdots a}^{m})}_{n}$$

is produced by

```
\[
(a^m)^n = \underbrace{(\overbrace{a \cdots a}^m) \cdots
(\overbrace{a \cdots a}^m)}_n
\]
```

A number of different accents are possible, for example $\widehat{ab}$, $\widetilde{T}$ and so, as shown in Table 2. The `\widehat` and `\widetilde` are reasonably stretchy, but do have their limits.

Greek and other symbols

The Greek alphabet begins α, β, γ,..., and is produced by `\alpha` etc. For uppercase Greek letters change the first letter in the command to uppercase. For example `$\Gamma$` gives Γ, noting that an uppercase alpha is just A, given by `$A$`. Note also that some letters have variants. See Tables 5 and 6.

Infinity is given by `\infty`. There are further variants on the ellipsis `\ldots` discussed earlier, which are `\cdots` ($\cdots$), `\cdot` ($\cdot$), `\vdots` ($\vdots$) and `\ddots` ($\ddots$). Unlike `\ldots` these variants only work in math mode. The final two are useful in particular when typesetting large matrices.

Integrals, sums and products

These are produced by `\int`, `\sum` and `\prod`. The ranges are given by the appropriate sub- and superscripts. For instance $\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$ is produced by `$\sum_{i=1}^n i = \frac{1}{2} n(n+1)$`. These symbols

have different sizes and place the limits in different places depending on whether they are used in-line or in a display. For instance when the previous example is displayed we get

$$\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$$

where the sum is now given by a bigger symbol and the limits are placed on top and bottom, rather than to the side. This can be altered by hand — see the section below.

Multiple integrals can either be achieved by repeating `\int` the required number of times (and using `\!` to get better spacing), or, if there is only one subscript giving the range of integration, then `\iint`, `\iiint`, `\iiiint` and `\idotsint` give $\iint$, $\iiint$, $\iiiint$ and $\int\cdots\int$, where the spacing between the integral signs is done automatically.

Set and vector space operations

Unions, intersections and set differences are given by the commands `\cup`, `\cap` and `\setminus` when dealing with a given (finite) numbers of sets. For unions and intersections of families of sets use `\bigcup` and `\bigcap`. For example $A \cup B$ and $\bigcap_{i=1}^{\infty} A_i$ are given respectively by `$A \cup B$` and `$\bigcap_{i=1}^\infty A_i$`.

Direct sums ($\oplus$) are given by `\oplus`, and tensor products ($\otimes$) are given by `\otimes`, and again both of these have the larger variants `\bigoplus` and `\bigotimes` for use on families of vector spaces, rings etc.

As with integrals, the big versions of all of these symbols behave differently when in displays, compared to in-line use.

Text and display styles

Sometimes the change in behaviour concerning placing of limits and/or the size of the symbol for commands such as `\int` and `\bigcup` are not what you want to happen. This behaviour can be over-ruled by use of the `\textstyle` and `\displaystyle` commands which change it to the named variety. For example normally `$\bigcup_{i=1}^n A_i$` produces $\bigcup_{i=1}^n A_i$, but when enclosed in a `\displaystyle` command gives $\displaystyle\bigcup_{i=1}^{n} A_i$. The relevant symbols that change are listed in Table 7.

Fractions

`$\frac{\pi}{2}$` produces $\frac{\pi}{2}$, although when displayed will give the larger result $\displaystyle\frac{\pi}{2}$. To produce textstyle fractions in displayed material either use the `\textstyle` command, or type `$\tfrac{\pi}{2}$`. The `\binom` variant provides binomial coefficients: $\binom{3}{2}$ is produced by `$\binom{3}{2}$`.

Log-like functions

Certain function names are usually set with upright letters, with special spacing on either side. Rather than have to fiddle around with `\mathrm` and spacing commands, a number of such functions have been pre-programmed with command names like `\log`, `\sin` and so on — see Table 3. To create new functions you should use the `\DeclareMathOperator` command, rather than `\newcommand`, since this sorts out the font type and size issues automatically. It is used in exactly the same way, for example

```
\DeclareMathOperator{\Range}{Ran}
```

produces $\operatorname{Ran} V$ when you type `$\Range V$`.

Modulo arithmetic

The AMS-LaTeX package provides the commands `\mod`, `\bmod`, `\pmod` and `\pod` for modulo arithmetic. They produce, respectively:

$$13 = 4 \mod 9; \quad 13 = 4 \bmod 9; \quad 13 = 4 \pmod 9; \quad 13 = 4 \pod 9 \tag{9}$$

Bracketing

Standard parentheses (and) are given by `(` and `)`. Similarly for square brackets [and]. For curly brackets you need to type `\{` and `\}`, since `{` and `}` are used throughout in giving the arguments of commands. Further examples of delimiters, used to give inner products or norms for example, can be found in Table 4.

The size of brackets can be adjusted to accommodate large contents. For example consider

$$(\int_0^t f(t)^2\,dt)^{1/2} \quad \text{and} \quad \left(\int_0^t f(t)^2\,dt\right)^{1/2}$$

The brackets in the latter are generated by `\left(` and `\right)`. It is vital that there is a matching pair of `\left` and `\right` commands — but the

brackets need not match each other. The commands `\left.` and `\right.` produce the 'empty' bracket.

Sometimes the sizing given by `\left` and `\right` can be a bit excessive. These can be adjusted by hand by using the commands (in increasing order of size) `\bigl`, `\Bigl`, `\biggl` and `\Biggl` in front of the left hand bracket, and `\bigr` etc. in front of the right hand bracket. Moreover, these no longer need to match, which can be helpful if the brackets appear on different lines of a (broken) displayed equation.

Arrays and matrices; cases

The standard math mode version of the `tabular` environment discussed previously is the `array` environment, and works in a very similar way. In combination with brackets given by commands such as `\left(` and `\right)` this gives one method of producing matrices. More efficient, both to type and in terms of the horizontal space used, are the environments `matrix`, `pmatrix`, `bmatrix` and `vmatrix` from AMS-LaTeX. With these you are not required to specify in advance the number columns in your matrix, nor the relative positioning of the entries since they are all assumed to be centred. So

```
\left( \begin{array}{ccc} 1 & 2 & 3 \\ e^1 & \log 3 & \cos 7
  \end{array} \right)
```

and

```
\begin{pmatrix} 1 & 2 & 3 \\ e^1 & \log 3 & \cos 7
  \end{pmatrix}
```

produce

$$\left(\begin{array}{ccc} 1 & 2 & 3 \\ e^1 & \log 3 & \cos 7 \end{array} \right) \quad \text{and} \quad \begin{pmatrix} 1 & 2 & 3 \\ e^1 & \log 3 & \cos 7 \end{pmatrix}$$

respectively.

To define a function involving many cases either use the `array` environment inside brackets given by `\left\{` and `\right.`, or use the `cases` environment. For example

```
\cos (n\pi) = \begin{cases} -1 & \text{if } n \text{ is odd} \\
  +1 & \text{if } n \text{ is even} \end{cases}
```

gives

$$\cos(n\pi) = \begin{cases} -1 & \text{if } n \text{ is odd} \\ +1 & \text{if } n \text{ is even} \end{cases}$$

Fonts and sizes

The commands for changing between type faces in mathematics are similar to those in for use with text. Also, there are calligraphic and blackboard bold versions of the uppercase letters, and fraktur versions of both upper- and lowercase letters.

`\mathit`	*Italics*	`\mathrm`	Roman
`\mathbf`	**Boldface**	`\mathsf`	Sansserif
`\mathtt`	`Typewriter`	`\mathcal`	$\mathcal{CALLIGRAPHIC}$
`\mathbb`	$\mathbb{BLACKboardbold}$	`\mathfrak`	$\mathfrak{Fraktur}$

There are fewer sizing commands, and these reflect the number of levels of super- and subscripts, and whether or not one is in-line or using a display. The commands that change the current style are `\displaystyle`, `\textstyle`, `\scriptstyle` and `\scriptscriptstyle`. For example ordinarily `$e^{y(i)}$` produces $e^{y(i)}$, whereas `$e^{\textstyle y(i)}$` produces $e^{y(i)}$.

4.3 Aligning material

It often happens that even though you are trying to display material that would look wrong in the text, your equations are too long to fit onto the one line. Alternatively you are trying to give an account of a number of steps in a calculation. The following environments give a number of options for dealing with these and other circumstances. In what follows the term equation could equally well mean inequality, or another mathematical expression involving a binary operation.

If the equation you have is just too long, but there is only one equation then there are two choices: `multline` and `split`. The first is an environment in itself. Each time you want to break the line you should insert `\\`; the first line is placed at the left of the page, the final one shifted to the right, and any intervening lines are centred. The split environment, on the other hand, has to be used inside the `equation` or `equation*` environments. Moreover it allows for alignment between the lines. Again `\\` is used to denote the point to break, and `&` is used to indicate the points to align. Examples include

```
\begin{multline}
H_c = \frac{1}{2n} \sum^n_{l=0} (-1)^{l} (n-{l})^{p-2}
\sum_{l_1 +\cdots + l_p=l} \prod^p_{i=1} \binom{n_i}{l _i} \\
\cdot [(n-l )-(n_i-l _i)]^{n_i-l _i} \cdot
\Bigl[(n-l )^2 - \sum^p_{j=1} (n_i-l _i)^2 \Bigr]
\end{multline}
```

which produces

$$H_c = \frac{1}{2n}\sum_{l=0}^{n}(-1)^l(n-l)^{p-2}\sum_{l_1+\cdots+l_p=l}\prod_{i=1}^{p}\binom{n_i}{l_i} \cdot [(n-l)-(n_i-l_i)]^{n_i-l_i}\cdot\left[(n-l)^2-\sum_{j=1}^{p}(n_i-l_i)^2\right] \quad (4.2)$$

Here you might want the items on the second line to line up with the = sign on the line above, so could use a combination of `equation` and `split` by typing

```
\begin{equation}
\begin{split}
H_c &= \frac{1}{2n} \sum^n_{l=0} (-1)^{l} (n-{l})^{p-2}
\sum_{l_1 +\cdots + l_p=l} \prod^p_{i=1} \binom{n_i}{l _i} \\
&\quad \cdot [(n-l )-(n_i-l _i)]^{n_i-l _i} \cdot
\Bigl[(n-l )^2 - \sum^p_{j=1} (n_i-l _i)^2 \Bigr]
\end{split}
\end{equation}
```

which produces

$$\begin{aligned} H_c &= \frac{1}{2n}\sum_{l=0}^{n}(-1)^l(n-l)^{p-2}\sum_{l_1+\cdots+l_p=l}\prod_{i=1}^{p}\binom{n_i}{l_i} \\ &\quad \cdot [(n-l)-(n_i-l_i)]^{n_i-l_i}\cdot\left[(n-l)^2-\sum_{j=1}^{p}(n_i-l_i)^2\right] \end{aligned} \quad (4.3)$$

Note that the `&` should come before the binary relation, so the = in the first line above. If there is no binary relation in any of the subsequent lines then you should have `&\quad` to obtain the appropriate spacing.

There is an unnumbered version of `multline`, namely `multline*`, but no such thing for `split`. Indeed, because a `split` environment is meant to go inside another equation-like environment, it is the outer environment that should be unnumbered. So in our example above we should use the `equation*` environment.

For a sequence of aligned equations there is the `align` environment, together with the unnumbered version `align*`. Again the symbol `&` is used to denote the alignment points. For example

```
\begin{align}
a_1 &= b_1+c_1\\
```

```
a_2 &= b_2+c_2-d_2+e_2
\end{align}
```

produces

$$a_1 = b_1 + c_1 \tag{4.4}$$
$$a_2 = b_2 + c_2 - d_2 + e_2 \tag{4.5}$$

In fact you can have (almost) any odd number of `&` symbols in an `align` environment. The odd numbered ones indicate alignment points, and the even ones separate the columns. So

```
\begin{align*}
a_{11} &= b_{11}  &  a_{12} &= b_{12}\\
a_{21} &= b_{21}  &  a_{22} &= b_{22}+c_{22}
\end{align*}
```

produces

$$a_{11} = b_{11} \qquad a_{12} = b_{12}$$
$$a_{21} = b_{21} \qquad a_{22} = b_{22} + c_{22}$$

If you have a number of equations or expressions to display over more than one line, but do not need to align them then there are the `gather` and `gather*` environments. Once more `\\` give the line breaks, but should be no `&`s present. For example

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

produces

$$\begin{aligned} a_1 &= b_1 + c_1 \\ a_2 &= b_2 + c_2 - d_2 + e_2 \end{aligned} \tag{4.6}$$

All of these environments produce output that stretches across the width of the page. If you need to create a block that is as wide as the text it produces in order to insert it within further displayed material then there are the environments `aligned` and `gathered`. For example

$$\cos n\pi = \left\{ \begin{array}{l} -1 \text{ if } n \text{ is odd} \\ +1 \text{ if } n \text{ is even} \end{array} \right\} = (-1)^n$$

is produced by

```
\[
\cos n\pi = \left\{
\begin{aligned} -1 & \text{ if } n \text{ is odd} \\
+1 & \text{ if } n \text{ is even} \end{aligned}
\right\} = (-1)^n
\]
```

If you have a run of aligned equations and wish to include a one or two word interjection before the end, whilst retaining the alignment, then there is the command `\intertext`. For example

$$x^2 = 0$$

and hence

$$x^2 \not< -10000 \times 10000$$

is produced by

```
\begin{align*}
x^2 &= 0 \\
\intertext{and hence}
x^2-200 &\not< -10000 \times 10000
\end{align*}
```

Suppressing and changing numbering

If you are using one of the numbered equation environments but do not want every line to be numbered then insert `\notag` on those lines where a number is not required. On the other hand if you wish to override the current number, or supply one if you are using a starred environment then there are the commands `\tag{n}` and `\tag*{..}`. The first produces the number (n) for the equation in question. The `\tag*` produces a similar effect, but without the bracketing which is useful if you wish to add adornment to that part of the labelling.

4.4 Available symbols

The following tables list a number of the available symbols. Some require the loading of extra packages such as `latexsym` and `amssymb`, but this has been done for you in the template so I have not noted where they are needed. On the other hand the euro symbol in Table 1 needs the package `eurosym` which has not been loaded on your behalf. There are many more symbols available — the document "The Not So Short Introduction to LaTeX 2ε" contains a more comprehensive list from which the following was derived/filched.

Table 1: Non-Mathematical Symbols.

These symbols can also be used in text mode.

†	`\dag`	§	`\S`	©	`\copyright`	€	`\euro`
‡	`\ddag`	¶	`\P`	£	`\pounds`		

Table 2: Math Mode Accents.

$\hat{a}$	`\hat{a}`	$\check{a}$	`\check{a}`	$\tilde{a}$	`\tilde{a}`	$\acute{a}$	`\acute{a}`
$\grave{a}$	`\grave{a}`	$\dot{a}$	`\dot{a}`	$\ddot{a}$	`\ddot{a}`	$\breve{a}$	`\breve{a}`
$\bar{a}$	`\bar{a}`	$\vec{a}$	`\vec{a}`	$\widehat{A}$	`\widehat{A}`	$\widetilde{A}$	`\widetilde{A}`

Table 3: Log-like functions.

arccos	`\arccos`	deg	`\deg`	lg	`\lg`	proj lim	`\projlim`
arcsin	`\arcsin`	det	`\det`	lim	`\lim`	sec	`\sec`
arctan	`\arctan`	dim	`\dim`	lim inf	`\liminf`	sin	`\sin`
arg	`\arg`	exp	`\exp`	lim sup	`\limsup`	sinh	`\sinh`
cos	`\cos`	gcd	`\gcd`	ln	`\ln`	sup	`\sup`
cosh	`\cosh`	hom	`\hom`	log	`\log`	tan	`\tan`
cot	`\cot`	inf	`\inf`	max	`\max`	tanh	`\tanh`
coth	`\coth`	inj lim	`\injlim`	min	`\min`	$\varliminf$	`\varliminf`
csc	`\csc`	ker	`\ker`	Pr	`\Pr`	$\varlimsup$	`\varlimsup`

Table 4: Delimiters.

(	`(`	)	`)`	↑	`\uparrow`	⇑	`\Uparrow`
[	`[` or `\lbrack`	]	`]` or `\rbrack`	↓	`\downarrow`	⇓	`\Downarrow`
{	`\{` or `\lbrace`	}	`\}` or `\rbrace`	↕	`\updownarrow`	⇕	`\Updownarrow`
⟨	`\langle`	⟩	`\rangle`	|	`|` or `\vert`	‖	`|` or `\Vert`
⌊	`\lfloor`	⌋	`\rfloor`	⌈	`\lceil`	⌉	`\rceil`
/	`/`	\	`\backslash`		. (dual. empty)		

Table 5: Lowercase Greek Letters.

α	\alpha	θ	\theta	o	o	υ	\upsilon
β	\beta	ϑ	\vartheta	π	\pi	ϕ	\phi
γ	\gamma	ι	\iota	ϖ	\varpi	φ	\varphi
δ	\delta	κ	\kappa	ρ	\rho	χ	\chi
ϵ	\epsilon	λ	\lambda	ϱ	\varrho	ψ	\psi
ε	\varepsilon	μ	\mu	σ	\sigma	ω	\omega
ζ	\zeta	ν	\nu	ς	\varsigma		
η	\eta	ξ	\xi	τ	\tau		

Table 6: Uppercase Greek Letters.

Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

Table 7: BIG Operators.

$\sum$	\sum	$\bigcup$	\bigcup	$\bigvee$	\bigvee	$\bigoplus$	\bigoplus
$\prod$	\prod	$\bigcap$	\bigcap	$\bigwedge$	\bigwedge	$\bigotimes$	\bigotimes
$\coprod$	\coprod	$\bigsqcup$	\bigsqcup	$\bigodot$	\bigodot		
$\int$	\int	$\oint$	\oint	$\biguplus$	\biguplus		

Table 8: Arrows.

$\leftarrow$	\leftarrow or \gets	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\rightarrow$	\rightarrow or \to	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow
$\rightleftharpoons$	\rightleftharpoons	$\Longleftrightarrow$	\iff (bigger spaces)	$\rightsquigarrow$	\leadsto

Table 9: Binary Relations.

You can produce corresponding negations by adding a `\not` command as prefix to the following symbols, e.g. `\not\sim` gives $\not\sim$

$<$	`<`	$>$	`>`	$=$	`=`
$\leq$	`\leq` or `\le`	$\geq$	`\geq` or `\ge`	$\equiv$	`\equiv`
$\ll$	`\ll`	$\gg$	`\gg`	$\doteq$	`\doteq`
$\prec$	`\prec`	$\succ$	`\succ`	$\sim$	`\sim`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\simeq$	`\simeq`
$\subset$	`\subset`	$\supset$	`\supset`	$\approx$	`\approx`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\cong$	`\cong`
$\sqsubset$	`\sqsubset`	$\sqsupset$	`\sqsupset`	$\bowtie$	`\Join`
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\bowtie$	`\bowtie`
$\in$	`\in`	$\ni$	`\ni` , `\owns`	$\propto$	`\propto`
$\vdash$	`\vdash`	$\dashv$	`\dashv`	$\models$	`\models`
$\mid$	`\mid`	$\parallel$	`\parallel`	$\perp$	`\perp`
$\smile$	`\smile`	$\frown$	`\frown`	$\asymp$	`\asymp`
$:$	`:`	$\notin$	`\notin`	$\neq$	`\neq` or `\ne`

Table 10: Binary Operators.

$+$	`+`	$-$	`-`		
$\pm$	`\pm`	$\mp$	`\mp`	$\triangleleft$	`\triangleleft`
$\cdot$	`\cdot`	$\div$	`\div`	$\triangleright$	`\triangleright`
$\times$	`\times`	$\setminus$	`\setminus`	$\star$	`\star`
$\cup$	`\cup`	$\cap$	`\cap`	$*$	`\ast`
$\sqcup$	`\sqcup`	$\sqcap$	`\sqcap`	$\circ$	`\circ`
$\vee$	`\vee` , `\lor`	$\wedge$	`\wedge` , `\land`	$\bullet$	`\bullet`
$\oplus$	`\oplus`	$\ominus$	`\ominus`	$\diamond$	`\diamond`
$\odot$	`\odot`	$\oslash$	`\oslash`	$\uplus$	`\uplus`
$\otimes$	`\otimes`	$\bigcirc$	`\bigcirc`	$\amalg$	`\amalg`
$\bigtriangleup$	`\bigtriangleup`	$\bigtriangledown$	`\bigtriangledown`	$\dagger$	`\dagger`
$\lhd$	`\lhd`	$\rhd$	`\rhd`	$\ddagger$	`\ddagger`
$\unlhd$	`\unlhd`	$\unrhd$	`\unrhd`	$\wr$	`\wr`

Table 11: Miscellaneous Symbols.

$\dots$	\dots	$\cdots$	\cdots	$\vdots$	\vdots	$\ddots$	\ddots
$\hbar$	\hbar	$\imath$	\imath	$\jmath$	\jmath	ℓ	\ell
$\Re$	\Re	$\Im$	\Im	$\aleph$	\aleph	$\wp$	\wp
$\forall$	\forall	$\exists$	\exists	$\mho$	\mho	∂	\partial
$'$	'	$\prime$	\prime	$\emptyset$	\emptyset	∞	\infty
∇	\nabla	$\triangle$	\triangle	$\Box$	\Box	$\Diamond$	\Diamond
$\bot$	\bot	$\top$	\top	$\angle$	\angle	$\surd$	\surd
$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit	$\clubsuit$	\clubsuit	$\spadesuit$	\spadesuit
$\neg$	\neg or \lnot	$\flat$	\flat	$\natural$	\natural	$\sharp$	\sharp

Table 12: AMS Miscellaneous.

$\hbar$	\hbar	$\hslash$	\hslash	$\Bbbk$	\Bbbk
$\square$	\square	$\blacksquare$	\blacksquare	$\circledS$	\circledS
$\vartriangle$	\vartriangle	$\blacktriangle$	\blacktriangle	$\complement$	\complement
$\triangledown$	\triangledown	$\blacktriangledown$	\blacktriangledown	$\Game$	\Game
$\lozenge$	\lozenge	$\blacklozenge$	\blacklozenge	$\bigstar$	\bigstar
$\angle$	\angle	$\measuredangle$	\measuredangle	$\sphericalangle$	\sphericalangle
$\diagup$	\diagup	$\diagdown$	\diagdown	$\backprime$	\backprime
$\nexists$	\nexists	$\Finv$	\Finv	$\varnothing$	\varnothing
$\eth$	\eth	$\mho$	\mho		

You will find that within a short time of using LaTeX/WinEdt that you begin to remember the names of the symbols that you use most commonly. Moreover, rather than have to keep looking at this document or some other source for less frequently used symbols you can make use of tool bars that list a lot of these. If these bars are not visible then click on the Options folders, then Appearance, and check the Show GUI Page Control.

5 Cross-referencing; bibliographies

All scientific documents place themselves in context by citing related work and source materials, and also aid their readers by the use of (internal) cross-referencing. Both of these tasks are easily accomplished using LaTeX, in a manner that is also painless when it comes to editing.

5.1 Internal cross-references

When you run LaTeX on your file `MyFile.tex` another file is produced called `MyFile.aux`. This contains a lot of information about the location of various parts and structures of your document. Some information is placed into the file automatically, other parts you have to tell LaTeX to write there, but all is then available the next time you compile your document.

Information that is automatically added to the file includes the page number where each section, subsection etc. begins, along with the number of that section. For most other items you need to place a marker at the point you wish to recall. This is done by the command `\label{`*marker*`}`. To make use of this information you can then use the commands `\ref{`*marker*`}`, `\eqref{`*marker*`}` and `\pageref{`*marker*`}`. The first produces the number of the section (or theorem, or equation...) that is being referred to, the second is really only for referring to equations, and produces the equation number in brackets, and in an upright font if you are in the middle of an italicised portion of text, and the third gives the page number of the location of the marker. For instance the code for the beginning of this particular section is

```
\section{Cross-referencing; bibliographies}
\label{referencing}
```

and a paragraph on page 9 ends with

```
Bibliographies are dealt with in
section~\ref{referencing}.\label{referred to}
```

The insertion of `\label{referencing}` allowed me to refer on page 9 to the fact that we are now in Section 5. More importantly if I were to change this document at a later stage, perhaps by introducing a new section before this one, then the number of this section will change. But this change is recorded in the `.aux` file, and the correct section number will always be given. Note also that I included the code `\label{referred to}` on page 9, so that in this section I would be referring back to the correct page that referred to this section...

This machinery can also be used to refer to theorems, propositions, etc., as well as particular equations. For example if we were to write

```
\begin{thm}  \label{Fermat}
$x^n + y^n \neq z^n$ for all $x,y,z \in \mathbb{Z}, \ n \geq 3$
\end{thm}
```

then any use of the code `Theorem~\ref{Fermat}` would replace the part `\ref{Fermat}` by the appropriate number for the theorem. The same works

with equations, although you should take a little care when using `align` and `gather` that you put the label in the correct place. For example

```
\begin{align}
a_1& =b_1+c_1 \label{first eqn}\\
a_2& =b_2+c_2-d_2+e_2 \label{second eqn}
\end{align}
```

produces

$$a_1 = b_1 + c_1 \tag{5.1}$$
$$a_2 = b_2 + c_2 - d_2 + e_2 \tag{5.2}$$

and then `equation~\eqref{second eqn}` produces 'equation (5.2)'.

5.2 Bibliographies

A bibliography is produced using the environment `thebibliography`, which will be the last environment in your `.tex` file. The beginnings of one would look like

```
\begin{thebibliography}{99}
\bibitem{Slava1} V P Belavkin, Quantum stochastic calculus and
quantum nonlinear filtering, \emph{J Multivariate Anal}
\textbf{42} (1992), 171--201.

\bibitem{EvansM} M P Evans, Existence of quantum diffusions,
\emph{Probab Theory Related Fields} \textbf{81} (1989),
473--483.
```

Each item you wish to refer to is begun with the `\bibitem` command (cf. the `\item` commands used in lists from subsection 3.6), and the text in brackets after the `\bibitem` gives the marker with which you will refer to the work. This produces an ordered list of the works that you are referencing, numbered 1., 2. etc. To cite one of these works you then type something of the form `\cite{Slava1}` to refer to the first work above, or `\cite[p.480]{EvansM}` if you wished to refer to page 480 of the second work. You can cite multiple works by putting a comma separated list of markers inside the curly brackets.

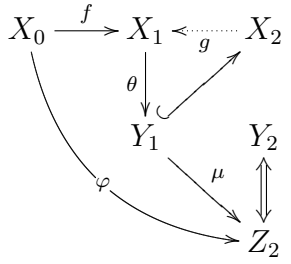
If you prefer you can change the labelling of your bibliography — for the above you might use something like [Bel] and [Eva], or [B 92] and [E 89] to indicate the name and/or year of publication. This is achieved by changing the `\bibitem` commands above to `\bibitem[Bel]{Slava1}` etc. Also, the `{99}` that appears in the first line is an indication of what your largest label

in the bibliography will be. Using 99 for standard numbered labels normally suffices (unless you've been reading very widely...), but something like `[MMM]` would be better for three letter labelling.

6 Further possibilities

This document barely scratches the surface of what is possible within LaTeX, but should cover most of what you need for your project write-up. Most obviously I have left out all discussion about changing the layout, type size, font etc. as this would take you further from the goal of producing a report in a short space of time that conveys your researches of the last few months. References [1, 3, 4] can be consulted for more information if you are interested (at a later date...)

Many of the further extensions are achieved by loading additional package. We are already using some packages produced by the American Mathematical Society, and many more are available already on the computers in the lab. It is possible to find the relevant documentation there, or, possibly more easily, by googling for it. For example commutative diagrams such as



can be produced by the XY-pic package. Searching for `xypic` produces a whole host of resources.

Similarly using either of the `graphics` or `graphicx` packages allows you to include graphics in your work. If you want to produce postscript output then the graphics to be included should be in encapsulated postscript format (i.e. as an `.eps` file); for pdf output they should be put in as a `.pdf` file. It is possible to save the plots from programs such as Maple as `.eps` files, and conversion from that to `.pdf` is possible with the program `epstopdf`, which you can run at the command prompt.

Once you have the line `\usepackage{graphicx}` in your preamble, the command `\includegraphics[options]{filename}` at the required point inserts your file. Some useful options that can be given include:

- `scale=1.5` — increases all linear dimensions by 1.5. The factor can be varied...

- `angle=90` — rotate through 90 degrees, measured anti-clockwise.
- `height=100mm` — if you want to fix the height. Acceptable units include `mm`, `cm`, `in` and `pt`. The `width` option works similarly.

You may want to adjust the horizontal and vertical positioning of you figure, which can be done simply with the `\hspace` and `\vspace` commands respectively. If you want to have side-by-side graphics with captions etc., then you may have to learn about the `figure` and `minipage` environments, and similar constructs — google for `graphicx` for more information.

References

[1] The Not So Short Introduction to LaTeX 2_ε, Tobias Oetiker *et. al.*, 2002.

[2] User's Guide for the `amsmath` Package (Version 2.0), American Mathematical Society, 1999.

[*These first two are included on the disc with the template.*]

[3] LaTeX: A document preparation system, Leslie Lamport, 2nd revised ed., Addison-Wesley, Reading, MA, 1994. (686.2 LAMP)

[*Note: the first edition is for LaTeX 2.09, rather than the current version of LaTeX 2_ε — but is still partly relevant.*]

[4] The LaTeX companion, Michel Goossens, Frank Mittelbach, and Alexander Samarin, Addison-Wesley, Reading, MA, 1994. (686.2 GOOS)

[*Note: The first edition (1994) is not a totally reliable guide for the* `amsmath` *package. I have the recently published second edition.*]