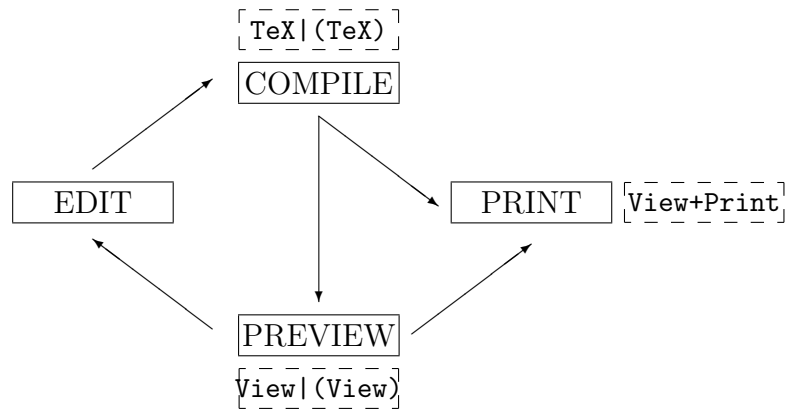


Imperial College of Science Technology & Medicine



# An Introduction to L<sup>A</sup>T<sub>E</sub>X

Phillip Kent

Version 5 (March 2001) for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> and BaKoMa TeX

# Contents

<b>1</b>	<b>A first <math>\LaTeX</math> document</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.2	When, Why & Where $\LaTeX$ ? . . . . .	4
1.3	About $\TeX$ . . . . .	5
1.4	Doing $\LaTeX$ . . . . .	6
1.5	EXERCISE 1: A first $\LaTeX$ document . . . . .	8
1.6	EXERCISE 1A . . . . .	10
1.7	The $\LaTeX$ philosophy . . . . .	12
<b>2</b>	<b>Math mode and more</b>	<b>13</b>
2.1	Math mode . . . . .	13
2.2	$\LaTeX$ environments . . . . .	17
2.3	EXERCISE 2—typing maths . . . . .	18
2.4	Handling $\LaTeX$ errors . . . . .	18
<b>3</b>	<b>Complex documents</b>	<b>20</b>
3.1	Cross-referencing . . . . .	20
3.2	Tables of contents . . . . .	20
3.3	Bibliographies . . . . .	21
3.4	The auxiliary ( <code>.aux</code> ) file . . . . .	21
3.5	Putting a document in pieces . . . . .	22
3.6	EXERCISE 3 . . . . .	23
<b>4</b>	<b>Pictures</b>	<b>24</b>
4.1	The <code>figure</code> environment . . . . .	24
4.2	The scissors-and-paste method . . . . .	25
4.3	$\LaTeX$ 's <code>picture</code> environment . . . . .	25
4.4	Using the <code>graphics</code> package . . . . .	26
<b>A</b>	<b><math>\LaTeX</math> on the web</b>	<b>28</b>
<b>B</b>	<b>Glossary</b>	<b>29</b>

# Chapter 1

## A first L<sup>A</sup>T<sub>E</sub>X document

### 1.1 Introduction

Welcome to this introductory course in L<sup>A</sup>T<sub>E</sub>X. It is very short, and is a sort of crash, or head-on collision, into L<sup>A</sup>T<sub>E</sub>X. It should get you started—after that, experience is crucial: you learn by doing, studying a certain aspect of L<sup>A</sup>T<sub>E</sub>X when you need to use it in your documents. There are numerous free and published guides to L<sup>A</sup>T<sub>E</sub>X, and you will certainly have to consult these during the experience phase. Also, a good tip is to keep a notebook (electronic and/or paper) for recording any useful pieces of L<sup>A</sup>T<sub>E</sub>X code that you discover.

All the documents for this course, including these notes and the exercises, are on a website:

<http://metric2.ma.ic.ac.uk/latex/>

This course describes the current standard version of L<sup>A</sup>T<sub>E</sub>X called “2<sub>ε</sub>” (pronounced, “two E”) which appeared in 1994. The previous version, which first appeared in 1985 and is usually known as “2.09” (since the last version released was 2.09) works slightly differently, and it’s important to bear this in mind when you are looking up information in books or on the web.

The standard reference book is the *L<sup>A</sup>T<sub>E</sub>X User Guide & Reference Manual* by Leslie Lamport (Addison-Wesley). The second edition of 1994 describes L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. The first edition of 1986 describes L<sup>A</sup>T<sub>E</sub>X 2.09. I don’t think Lamport is the best book for learning, but you will need access to a copy sometimes. A much better book, especially for “advanced” L<sup>A</sup>T<sub>E</sub>X features, is *A Guide to L<sup>A</sup>T<sub>E</sub>X* by Kopka & Daly (Third edition, Addison-Wesley, 1999).

Any unfamiliar technical terms used in these notes should be in the Glossary on page 29. If not, tell me.

I think the most important thing to know about  $\LaTeX$  at the start is that it is NOT helpful to think of it as a word processor. Well, it is a “word processor” in the sense that you can use it to make printed documents, but how you go about this is very different from using the type of software that is called a word processor, such as Microsoft Word. I would say that using  $\LaTeX$  requires having the attitude of a computer programmer, and this is quite a problematic issue in today’s “user friendly” computer culture where most computer users are discouraged from thinking about programming.

By the way, the “X” in the name  $\LaTeX$  is not pronounced as an X, but rather as a Greek *chi* ( $\chi$ ), so  $\TeX$  is “tech” (as in “technical”) and  $\LaTeX$  is “lay tech”.

## 1.2 When, Why & Where $\LaTeX$ ?

### When?

To learn  $\LaTeX$  requires an investment of time somewhat greater than for word processing software. It just isn’t the same kind of slick, idiot-proof package, and you need to know a little bit about computers to use it. So, when is it worthwhile to make the effort?

- Typing mathematics. Easy typing of maths was central to the creation of  $\LaTeX$ .
- Typing a complex document, such as a scientific paper, report or thesis, where you need a bibliography, a table of contents, an index, labelling of equations and cross-referencing, and so on.

An *apparent* disadvantage of  $\LaTeX$  is that it is non-WYSIWYG (‘What-You-See-Is-What-You-Get’), that is, as you’re typing you can’t see what the output will look like. This is *not* a disadvantage, if you can adapt to the idea, because you should find yourself caring more about the content of a document and less about its format on the printed page, which is 95% of the time handled automatically. This is, I believe, how word processing should be. A WYSIWYG program like Word is actually quite good at automatic formatting, *if* you know how to use it properly (but, in my experience, this aspect of Word is not easy to use). However, there is nothing to compete with  $\LaTeX$  for typing mathematics.

There are circumstances where  $\LaTeX$  is not the best choice of software. For documents with little text and a lot of images (eg. photos), WYSIWYG is probably preferable. Or, if you use the “object linking” feature of Microsoft

Office to include spreadsheet and database objects into a Word document;  $\LaTeX$  has nothing comparable to this.

## Why?

- $\LaTeX$  output is beautiful, virtually of professional-typeset quality.
- The basic  $\LaTeX$  system is FREE.
- $\LaTeX$  makes typing “easy” in the sense that almost all formatting is automatic, leaving the user to concentrate on content.
- $\LaTeX$  is written as plain text. Therefore it is compact, portable across machines and operating systems, and transferable across the internet via World-Wide-Web and email.
- All major academic publishers accept papers and articles in  $\LaTeX$  form. This speeds up the publishing process and reduces the chance of printing errors.

## Where?

Free versions of  $\LaTeX$  exist for Intel-based PCs (Microsoft Windows, Linux), Unix and Apple Macintosh. Commercial (non-free) versions offer extra features (roughly varying with the price charged), and in this course we will be using a cheap, shareware package called *BaKoMa TeX*<sup>1</sup>, which offers a pleasanter working environment than the basic.

There is a commercial WYSIWYG package, *Scientific Word*<sup>2</sup>, which creates documents in  $\LaTeX$  format, without the user needing to know any  $\LaTeX$ . If you find yourself strongly attached to WYSIWYG, but needing to produce  $\LaTeX$  for other people, this may be a compromise solution.

## 1.3 About $\TeX$

$\LaTeX$  is a super-set (macro package) of the typesetting language  $\TeX$ , created by Donald Knuth. Knuth was obsessive in making a program that could format text like a human typesetter, and was very successful: most academic publishers in the world use  $\TeX$  in some part of their operations.

---

<sup>1</sup>See: [www.tex.ac.uk/tex-archive/nonfree/systems/win32/bakoma/](http://www.tex.ac.uk/tex-archive/nonfree/systems/win32/bakoma/)

<sup>2</sup>See: [www.mackichan.com](http://www.mackichan.com)

But  $\text{T}_{\text{E}}\text{X}$  by itself, *plain T<sub>E</sub>X* as Knuth calls it, is not very easy to use without programming skills because Knuth deliberately left out “high-level” structure like standard formats (*e.g.* for reports and letters), leaving that for other people to do.  $\text{\LaTeX}$  (originally created by Leslie Lamport) adds those formats and other things to make using  $\text{T}_{\text{E}}\text{X}$  easier. For a while, there were other  $\text{T}_{\text{E}}\text{X}$  macro packages around, notably  $\mathcal{A}\mathcal{M}\mathcal{S}\text{T}_{\text{E}}\text{X}$  and  $\mathcal{A}\mathcal{M}\mathcal{S}\text{\LaTeX}$  used and promoted by the American Mathematical Society. Nowadays, these have been incorporated into the standard  $\text{\LaTeX}$  system.

$\text{\LaTeX}$  and  $\text{T}_{\text{E}}\text{X}$  are not two different languages so most  $\text{T}_{\text{E}}\text{X}$  commands can be used in  $\text{\LaTeX}$ . I’ll point out some useful commands as we go along. But there is sometimes a conflict where  $\text{\LaTeX}$  has re-defined a  $\text{T}_{\text{E}}\text{X}$  command.

The *T<sub>E</sub>Xbook* by Donald Knuth is the reference for  $\text{T}_{\text{E}}\text{X}$ . You may need to refer to it for doing things which are not in  $\text{\LaTeX}$  or to change the  $\text{\LaTeX}$  standard for your own purposes.

## 1.4 Doing $\text{\LaTeX}$

The process of doing  $\text{\LaTeX}$  is essentially like writing computer programs in, say, Fortran or C, and especially (you should be warned) in terms of its frustrations. Programmers will recognise this in my description below. This makes  $\text{\LaTeX}$  very different from any WYSIWYG program, where everything is mouse and menu controlled.

For this course, we will use *BaKoma TeX*, which enhances the basic  $\text{\LaTeX}$  system with a “typing environment” that provides a number of useful shortcuts.

The  $\text{\LaTeX}$  *source file* contains the plain (ASCII) text of a document combined with formatting commands. Commands are usually preceded by a backslash “\”. This and nine other characters are *reserved* for special purposes:

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

You’ll see later how they are all used. So if you actually want a “%”, for example, you type: \%. The source file name MUST end in “.tex”.

A NOTE ON CASE: In different computer operating systems, command and file names may or may not be sensitive to upper or lower-case letters. In Unix (and Linux) they are; in Microsoft Windows they are not. I’ll always use lower case in these notes.

Figure 1.2 shows the file `skeleton.tex`, illustrating the essential components of a  $\text{\LaTeX}$  source file.

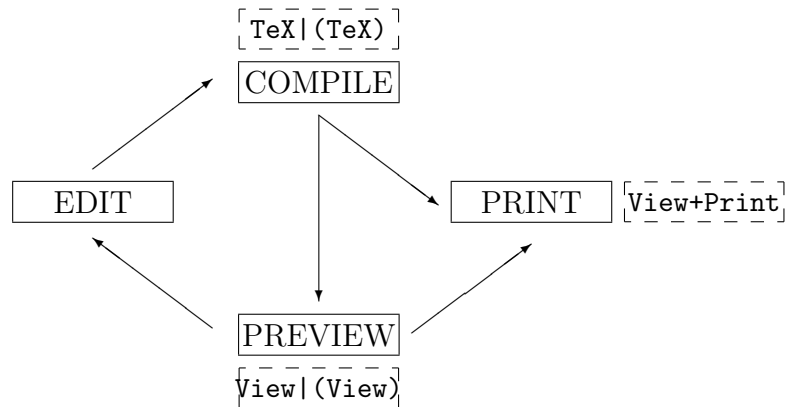


Figure 1.1: The steps of doing  $\LaTeX$ .

---

```

% skeleton.tex:
% the essential components of a LaTeX file
%
% (N.B. % is the comment character, everything to
% the right of it on a line is IGNORED.)

\documentclass{article}

% **** PREAMBLE ****
% contains:
% title/author/date information
% definitions, short-hands, macros etc.
% BUT NO text

\begin{document}

% **** BODY OF DOCUMENT ****
% ...the text itself
%
% N.B. the RESERVED CHARACTERS:
% \ % $ ^ _ & # ~ { }
%

\end{document}

```

---

Figure 1.2: The essential components of a  $\LaTeX$  file.

Given a source file, there is a program called `latex` which “compiles” it to a typeset document, known as the *DVI file* with its name ending in “.dvi”. DVI means DeVice Independent: this file may be used to show the document on any viewing or printing device. Typically you will do one of two things: *preview* it on the computer screen (using a “DVI viewer” program), or *print* it. The final program needed to use L<sup>A</sup>T<sub>E</sub>X is a *text editor* to create and edit source files.

Working in the most basic way, you will have three separate programs for doing L<sup>A</sup>T<sub>E</sub>X: a text editor, `latex` and a DVI viewer. With *BaKoma TeX*, these are all combined and used through one program interface.

The steps in the “L<sup>A</sup>T<sub>E</sub>X cycle” are shown in Figure 1.1 (this is an example of a diagram created with L<sup>A</sup>T<sub>E</sub>X `picture` commands—see Chapter 4). Most time is spent in the EDIT/COMPILE/PREVIEW loop, with PRINTing at the end of a typing session or when the document is finally complete. I think you can now see why I compare doing L<sup>A</sup>T<sub>E</sub>X to programming, and you’ll discover another similarity: *debugging!* Your source file may contain “syntax errors” in the L<sup>A</sup>T<sub>E</sub>X commands, *e.g.* a mis-spelt command name or a missing bracket. These you can only discover by running L<sup>A</sup>T<sub>E</sub>X, which will provide error messages. Correcting errors can be tricky, since like any other sequential compiler program the effect of an error may only be trapped by the program a long way after the error occurs in the source. More on this later.

## 1.5 EXERCISE 1: A first L<sup>A</sup>T<sub>E</sub>X document

For this exercise, either work on producing some text of your own in L<sup>A</sup>T<sub>E</sub>X, or work with the text contained in the file `exer1.tex`, which may be downloaded from the WWW pages.

In the latter case, you should either edit that file, or type everything from scratch, to make a document that looks like the one shown in the printed answer.

### Using BaKoma TeX

To use the text editor that BaKoma provides, launch the program which is called CENTAUR. To create a new L<sup>A</sup>T<sub>E</sub>X source file, go to **Open/New...** in the **File** menu, choose the location for the file and its name (you do not need to type the `.tex` extension); you can then choose the “document template” for the file—ignore this for now and just press the OK button.

To “compile” a source file (with the `latex` program), click on the button marked **TeX** at the top of the program window. A sub-window will appear showing the progress of the compilation, followed immediately by (if there are no errors!) the preview window containing your typeset document.

To do a preview only, without the compilation step, press the **View** button.

## Basics of L<sup>A</sup>T<sub>E</sub>X

Consult the sample file `skeleton.tex` shown in Figure 1.2 to see the basic framework of a L<sup>A</sup>T<sub>E</sub>X file. You should add this first to `exer1.tex`. Note the following:

- All L<sup>A</sup>T<sub>E</sub>X commands are case-sensitive, `\Begin` is different to `\begin`.
- L<sup>A</sup>T<sub>E</sub>X automatically formats paragraphs and pages, so text in the source file can be broken-up across lines in whatever way is convenient. Extra spaces are usually ignored, but L<sup>A</sup>T<sub>E</sub>X will not put spaces after punctuation marks; insert these manually.
- A “\” command which ends in a letter must be followed by a SPACE to tell L<sup>A</sup>T<sub>E</sub>X that the command has finished.
- Paragraphs are separated by one or more BLANK LINES. The first line of a paragraph is usually indented; to prevent this, put `\noindent` at the start. (The amount of indentation can also be set globally for a whole document.)
- Fonts: To emphasise text, use `\em` which gives italic within a block of roman (default) text, and roman within italic; for example

```
here is some {\em emphasised} text
```

will appear as

here is some *emphasised* text

Note that a font-switching command affects all the text following, therefore always enclose the text to be changed by a pair of braces (curly brackets: `{}`).

To access italic and bold font styles, the approved (but not the only) way for L<sup>A</sup>T<sub>E</sub>X<sub>2<sub>ε</sub></sub> is to use the commands `\textit` and `\textbf`, for example:

here is some `\textbf{bold text}`, some `\textit{italic text}`  
and some `\textbf{\textit{bold italic}}` text

here is some **bold text**, some *italic text* and some ***bold italic***  
text

Font selection in  $\text{\LaTeX}$  is rather sophisticated; consult one of the guide documents for the details (eg. *A Guide to  $\text{\LaTeX}$* , section 4.1).

- For this exercise, print the mathematical symbols as italic text, with `\textit`. Maths typing is introduced in Chapter 2.
- Quotation marks: single using ‘ and ’ (left and right quotes, you may have to search a little for the left quote key on your keyboard). Double using ‘‘ and ’’. The normal double quote character " is converted to ’’.
- Dots “...” are got with `\dots`.
- All the standard accents for languages that use the latin alphabet are available. For example, `\’e` makes é and `\"o` makes ö. NOTE: If you wish to use  $\text{\LaTeX}$  for typing in a language other than English, there are macro packages around to make this easier, including for languages such as Russian and Arabic which do not use the latin alphabet; see Appendix D of *A Guide to  $\text{\LaTeX}$*  for information.

## 1.6 EXERCISE 1A

Having completed EXERCISE 1 successfully, try the following modifications to `exer1.tex` and see the effects on the printed document.

### Making a title

Insert into the preamble the following lines:

```
\title{Poincar\’e and Fish}  
\author{Max ‘‘6 by 2’’ Plank}
```

and then put the command `\maketitle` directly after `\begin{document}`.

Thus you should have a main title and author in a bigger font, and a date automatically added. If you want no date, add to the preamble `\date{}`, or for an explicit date (or any other text): `\date{1 January 0000}`. There is a useful command `\today` which will generate the current date.

## Changing `\documentclass`

In `exer1.tex` I specified a “document class” called `article`. With this argument, the `\documentclass` command loads a standard format file which defines the page layout, sizes and styles of fonts for headings and other things appropriate for a technical or scientific “article”. Other document classes include `report` and `letter`, which we’ll come to later. Try changing `article` to `report`. What happens?

By default, documents are formatted for a physical paper size known as “US legal”, the North American equivalent of A4 (it’s shorter and wider but still fits on A4 paper), and a basic font size of “10 point”, where 72 points = 1 inch. To change these settings you use a second argument to the `\documentclass` command, with square brackets:

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

this specifies A4 paper size and 12-point type. Some people prefer 11-point, `11pt`.

*BaKoMa TeX* allows you to choose the initial document class and class options (such as `12pt`) when you create a new file with **Open/New...**, which saves a little bit of typing, though of course you can change the class and options at any time later on.

## Sectioning

Sectioning commands in  $\text{\LaTeX}$  include `\chapter`, `\section`, `\subsection` and `\subsubsection`. Numbering is automatic. Modify `exer1.tex` to:

```
\section{Extract from a recent press article}
```

You must leave a blank line following any sectioning command, and by default the first line of the first paragraph is not indented. Sometimes you don’t want numbering, and for this there are corresponding commands `\section*` *etc.* Try it.

## Sizes of fonts

Fonts may have different typefaces (bold-face, italic, *etc.*) and sizes. The sizes are, for example, `\small`, `\normalsize`, `\large`, `\Large`, `\huge`. Like the typeface commands, the size commands affect all text following, so enclose the region of text within braces, `{...}`. Experiment with some font sizes.

## Typing maths

Maths is introduced in Chapter 2 but you might want to start here. Mathematical type is enclosed in  $\$$  signs (“because math is expensive,” Knuth says). For example,  $\$ x_1^2 + 2x_1 + 2 = 0 \$$ . Have a play and see what comes out. Like ordinary text, maths text is formatted by  $\LaTeX$ , so ordinary spaces are ignored. What happens if you put two  $\$$  signs,  $\$\$ y = 3x+1 \$\$$  ?

## 1.7 The $\LaTeX$ philosophy

Hopefully after the first exercises you’ve got a feel for what creating a  $\LaTeX$  document involves. Now it’s worthwhile to emphasise the fundamental philosophy of  $\LaTeX$ .

The aim of  $\LaTeX$  is to make the formatting of a document as automatic as possible. Of course, there will be times when the standard format is not what you want, or there is not a formatting command that you need. When this happens you should strenuously avoid making *ad hoc* manual changes to your text. You should instead create your own new commands, or modifications to existing ones, in the preamble of the source file. We’ll do some examples of this later. If the changes involve extensive modification to the “global” appearance of a file you might even want to create your own document format file, basing it on a standard one. It is also a good idea to insert *comments* in a source file (a line beginning with  $\%$ ) when anything non-obvious is being done.

The advantages of this approach are: (1) the  $\LaTeX$  source will be more comprehensible both to you and other people. (2) The source will be portable between different document formats without modification, so for example the text of a research paper can (almost) instantly become a part of a thesis, or *vice versa* (and often does). (3) You’ll save yourself a lot of work overall by making some effort to organise things properly.

# Chapter 2

## Math mode and more

### 2.1 Math mode

So far we have been working in text mode. All mathematics is typed in *math mode*, which is quite distinct from text (it even uses a different italic font to the text one). Math mode is very sophisticated and knows all the standard typesetting rules for mathematical equations. For example, variables appear in italic and functions in roman. Ordinary spaces in math mode are ignored by  $\LaTeX$ , there are special commands for inserting space when necessary.

Math can be typed either inside a block of text, or in its own *display* separate from text. Here is some math in text:

Let  $G(x)$  be defined for  $a < x < b$ ,  $a, b \in \mathbb{R}$ . Therefore  $\dots$

which looks like:

Let  $G(x)$  be defined for  $a < x < b$ ,  $a, b \in \mathbb{R}$ . Therefore  $\dots$

Notice that the spacing between variables and operators ( $<$ ,  $\in$ ) is inserted automatically. The spacing and layout of math that you use in the source file is purely for legibility. Now some display math:

```
Let
\[
G\colon \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3
\]
be defined for the semi-intervals
\[
0 < \mathbf{x} < \infty, \quad 0 < t < \infty.
\]
```

which looks like:

Let

$$G: R^3 \times R \rightarrow R^3$$

be defined for the semi-intervals

$$0 < \mathbf{x} < \infty, \quad 0 < t < \infty.$$

Notice the use of bold-face: font-switching commands in math mode should be placed within braces as in text mode. `\quad` inserts extra space. A “quad” is an old printer’s unit equal to the width of an “M”; `\qqquad` is double this, and should be used for equations with a side condition:

$$\l[ F_n = F_{n-1} + F_{n-2}, \qqquad n \geq 2. \r]$$

It is good style to punctuate equations correctly, especially putting a full stop or comma on the end as appropriate. Sometimes it looks better to insert a little more space before the punctuation: `\,` inserts a half space.

The commands `$` (maths in text) and `$$` (maths display) belong to `TEX`, and `LATEX` has `\(...\)` and `\[...\]` instead. You should definitely avoid using the `$$` form, but also it is good practice to use `LATEX`’s bracket forms because it is a common mistake to miss out a closing `\)` or `$` and most good text editors can check for matching brackets, but can’t check for `$`’s.

## Sub- and super-scripts

The operators `_` for subscripts and `^` for superscripts work only on the next character, so `$x_{12}$` is  $x_{12}$ . Use braces if more than one character is to be subscripted, `$x_{12}$` ( $x_{12}$ ). Sub- and super-scripts may be combined straightforwardly: `$X^{2_{a+1}}$` is the same as `$X_{a+1}^2$` ( $X_{a+1}^2$ ). You can put sub-sub- and super-super-scripts, or indeed to any level, by nesting: `$X^{y^{2+x_1}}$` ( $X^{y^{2+x_1}}$ ). Note, however, that there are only three font sizes: normal, subscript and sub-subscript.

## Fractions

Fractions may be indicated just with a `/`, as in `x/(x^2+1)`, or using `\frac`:

$$\frac{x}{x^2+1}$$

$$\frac{x}{x^2+1}$$

It is considered bad style to use `\frac` for maths in text, because this uses smaller fonts than display maths, and fractions come out small and hard to read.

## Functions

Functions should be typeset in roman font, and L<sup>A</sup>T<sub>E</sub>X has a fair number of function names built-in, for example  $\mathbf{z} = \sin^2 x \cos^3 y$  ( $z = \sin^3 x \cos^2 y$ ). Suppose the one you want isn't there. You might do the following:

```
y = \mathrm{spoon}\,x
```

gives  $y = \text{spoon } x$ , where `\,` inserts a bit of space. This is OK if you only need to refer to that function once or twice, what about many times? We can *define a new command* in the preamble:

```
\newcommand{\spoon}{\mathrm{spoon}\,}
```

so that the new command `\spoon` does the job.

You can also define new commands to create shorthands for frequently-used complicated constructions. For example:

```
\newcommand{\Nb}[1]{\parallel #1\parallel^{\flat_\rho}}
```

Notice the “[1]”, this specifies that the command has 1 argument, so that  $\mathbf{\Nb\{X\}}$  gives  $\| X \|^{\flat_\rho}$ . You can put any expression you like into the argument, and it will be inserted at the place marked by #1.

A big advantage of defining commands like this is that it becomes easy to change things. For example, you decide that the function “spoon” should really be “Spoon”. It's a trivial matter to change the definition, and all the `\spoon`'s are untouched.

## Delimiters (brackets)

L<sup>A</sup>T<sub>E</sub>X can make brackets of any size. The easiest way is to use the `\left` and `\right` commands, which will create a bracket big enough for whatever is inside:

```
\[ \left( \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \right) \].
```

$$\left( \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \right).$$

Here we used parentheses, but it works for any delimiter (`[...]`, `{...}`, *etc.*).

## Arrays and matrices

L<sup>A</sup>T<sub>E</sub>X's `\array` command builds arrays of expressions, and in the case of matrices we can put brackets around by the method just described. For example:

```
\[ \left(
\begin{array}{ccc}
1-\lambda & -4 & x \\
3 & -2-\lambda & y \\
-1 & \alpha & z-\lambda
\end{array}
\right). \]
```

$$\left( \begin{array}{ccc} 1-\lambda & -4 & x \\ 3 & -2-\lambda & y \\ -1 & \alpha & z-\lambda \end{array} \right).$$

The `{ccc}` specifies the position of the entries in each column: flush left (`l`), centred (`c`) or flush right (`r`).

A possible shortcut for matrices (surrounded by parentheses) is T<sub>E</sub>X's `\pmatrix` command. This gives the same output with less typing:

```
\pmatrix{
1-\lambda & -4 & x \\
3 & -2-\lambda & y \\
-1 & \alpha & z-\lambda
}
```

## Numbered equations and multi-line equations

The following creates a numbered equation. `\[` and `\]` are not required, here math mode is switched on and off by the `\begin` and `\end` commands:

```
\begin{equation}
x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, .
\end{equation}
```

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \quad (2.1)$$

Sometimes a single equation is too big for one line, or you want a set of equations grouped together. For example, again no `\[` and `\]` required,

```

\begin{eqnarray}
x^2-3x+2 & = & 0 ,\nonumber \\
\Rightarrow x & = & \frac{3\pm\sqrt{9-4\cdot 2\cdot 1}}{2} \,, \nonumber \\
& = & 1, \quad 2.
\end{eqnarray}

```

$$\begin{aligned}
x^2 - 3x + 2 &= 0, \\
\Rightarrow x &= \frac{3 \pm \sqrt{9 - 4 \cdot 2 \cdot 1}}{2}, \\
&= 1, \quad 2.
\end{aligned} \tag{2.2}$$

`eqnarray` automatically numbers every line unless there is a `\nonumber`. The `&`'s define the columns of the array. As written above, the “=” signs are vertically aligned, which is the usual situation. Breaking long equations over several lines is one of those few occasions where  $\text{\LaTeX}$ 's formatting ability fails. For advice on doing it, see *A Guide to  $\text{\LaTeX}$*  section 5.4.7.

## 2.2 $\text{\LaTeX}$ environments

We've now seen a number of commands that have a `\begin{...}` statement and an `\end{...}` statement: `document`, `array`, `equation` and `eqnarray`. These are called *environments* and are a fundamental part of  $\text{\LaTeX}$ . We can easily define some of our own in the preamble. This defines a `proof` environment:

```

\newenvironment{proof}{\scshape Proof. }\itshape}
{\hfill$\spadesuit$\par}

```

How it works: the second set of braces defines what to do at the `\begin{proof}` command, and the final set what to do at the `\end{proof}` command. An example of its use:

```

\begin{proof}
The proof is obvious to {\em any animal} with an ounce of
brain tissue between its ears!  $E=mc^2$ , right?
\end{proof}

```

PROOF. *The proof is obvious to any animal with an ounce of brain tissue between its ears!  $E = mc^2$ , right?* ♠

## Theorems and such like

Talking of proofs leads naturally to theorems, and there is a special command for defining theorem-like environments. Examples:

```
\newtheorem{theorem}{Theorem}[section]
\newtheorem{conjecture}[theorem]{Conjecture}
```

These define a `theorem` environment which will number Theorem's according to section (1.1, 1.2, ..., 2.1, ...3.1), and a `conjecture` environment which numbers Conjectures in the *same* sequence as Theorem's. An example of use:

```
\begin{theorem}[Gackworth]
 $\text{spoon } f \geq 79/42$ .
\end{theorem}
\begin{conjecture}[Whackabath]
If  $f$  is dithyrambic and iconoclastic, then
 $\text{spoon } f > 107/42$ .
\end{conjecture}
```

**Theorem 2.1 (Gackworth)**  $\text{spoon } f \geq 79/42$ .

**Conjecture 2.2 (Whackabath)** *If  $f$  is dithyrambic and iconoclastic, then  $\text{spoon } f > 107/42$ .*

The arguments in square brackets add the attributions.

## 2.3 EXERCISE 2—typing maths

---

Some exercises on math mode are given separately. See the  
WWW pages.

---

## 2.4 Handling L<sup>A</sup>T<sub>E</sub>X errors

### When latex finds an error...

...it stops crunching the file, prints out some form of error message, the line number in the source file where the error occurred (useful), and some

often meaningless expression containing lots of low-level  $\TeX$ . There isn't much you can do to correct the problem at this point, although a very crude correction system is there.

Your responses include the following: ignore the error message and press `<Return>` at the “?” prompt—the problem may clear itself. If it doesn't and a whole pile more messages appear, you can do “q” (for quiet) and `latex` will crunch through to the end, putting error messages just in a `.log` file. Alternatively, type “x” to stop `latex`.

Now you are faced with some potentially nasty debugging! Check the source code at the line numbers given by `latex`, there may be some trivial syntax error at the line specified. The problem is that if you miss a closing `\]` or `$` this may not cause any syntax error until much later on, so you don't know where the error actually is. Avoid such problems by running `latex` pretty frequently as you're typing, that way the error will get pinned down to the text you typed since the last `latex`.

A useful feature of *BaKoMa TeX* is that you can typeset a “block” (selected part) of the source file by highlighting it and then pressing the button marked (**TeX**).

$\LaTeX$  errors can be nasty and obscure. This is why you really need a *LaTeX Book* or *Guide* close by. Also be ready to consult “local experts”.

## Over- and under-full boxes

`latex` frequently generates warnings about overfull and underfull hboxes. An “overfull hbox” means that a line is too long and the text has encroached into the right margin: check the amount given in the warning message. This is quoted in points, where 72 points = 1 inch, so a few points will be OK. The problem is that `latex` could not find a sensible place to break a line according to its rules. If the amount is big you will have to re-arrange the paragraph (or math display). An “underfull hbox” occurs when a line is too short and `latex` has had to insert more blank space than it thinks proper to fill up a line. You can ignore these or re-arrange the offending paragraph.

If you use the `draft` option with `\documentclass` then a black rectangle will appear in the typeset document wherever an overfull hbox occurs.

`hbox` problems get especially common when only a few words can fit on a line. This may be when you're using an extra big font size, or extra narrow margins. It may be a good idea in these cases to use `\raggedright` (with braces to localise its effect), so that lines are not justified to the right margin.

# Chapter 3

## Complex documents

This chapter covers some topics related to the preparation of large and/or complex documents with cross-references, bibliographies and related things.

### 3.1 Cross-referencing

Cross-referencing is very simple. You put a `\label` with a reference key on any numbered object (chapter, section, equation, theorem, figure, *etc.*) and then use `\ref` to get that number, or `\pageref` to get the page number. Example: I did the following just now,

```
\section{Cross-referencing} \label{sec.cross}
```

so if I type

```
Welcome to Section \ref{sec.cross} on page \pageref{sec.cross}!
```

I get

Welcome to Section 3.1 on page 20!

The key can be anything, but it's a good idea to include tags like “sec”, “eq” or “thm” for clarity.

### 3.2 Tables of contents

Tables of contents are also very simple. Insert the command

```
\tableofcontents
```

at the place you want the table to appear, and the information will be read from the `.aux` file (see below). There are similar commands for figures and tables: `\listoffigures` and `\listoftables`. More auxiliary files will appear when you use these commands, ending in `.toc`, `.lof` or `.lot`.

### 3.3 Bibliographies

The posh way to do bibliographies is to use a separate program called `BIBTEX`, which allows you to create a single master file of references which you can use in any document. It also allows a lot of control over the style of a bibliography. I'm not talking about that here: see the *TEX Book* Appendix B, and *A Guide to TEX* Appendix B.

Here's a more basic way of doing it. Usually (but not necessarily) at the end of a document, you put the bibliography information like this:

```
\begin{thebibliography}{99}

\bibitem{wilkins}
B. J. M. Wilkins, ‘‘Topological Dynamics and the Haddock
Fishery’’, Unpublished, 1987.

\bibitem{strainer-wilkins}
T. I. Strainer \& B. J. M. Wilkins 1993,
A new result on Drivle’s Theorem, {\em Proc. Iceland Cod Fish
Soc. Lond. Ser. D}, {\bf 134} (8678--8679).

\end{thebibliography}
```

The “99” means a maximum of 99 items (this is to get a neat alignment for the item numbers). The `\begin` command has the effect of starting a new, un-numbered section called “References”. Each item begins with `\bibitem` and a citation key, so that in the document you cite an item with its key, `\cite{wilkins}`, which `LATEX` replaces by a (sequential) number in square brackets.

`BIBTEX` is a lot more powerful than the method just described, and well worth a look.

### 3.4 The auxiliary (.aux) file

Every time you run `latex` a “.aux” file is made or refreshed. This file contains things like the names of sections with their page numbers, cross-reference `\label` and bibliography `\bibitem` information. When you run `latex` a *second* time this information is read to create a table of contents, and insert the right numbers into the `\ref`’s and `\cite`’s. So if any of these numbers change in editing a document, `LATEX` must be run *twice* for the changes to appear in the output.

## 3.5 Putting a document in pieces

When you are working on a large document, it can become inconvenient to keep the whole thing in a single, large source file. Typically, you're only working on a small part of the document at any one time, so recompiling the whole document each time you run the `latex` program becomes tedious. Then the thing to do is to split the source file into separate files, perhaps each containing one chapter, and use a small master file to collect everything together. The command:

```
\input{stuff}
```

reads the contents of the file `stuff.tex` and inserts it into the document at the point of the `\input`. This is useful if, say, you have some big source code for a diagram or picture which you may want to leave out until the final printing—just comment out the `\input` with a `%`.

For split documents, though, you really want to load the `.aux` information (with its cross-reference, bibliographic etc. information) but not the `.tex`, and the `\include` command does exactly this. Here's an example of a master file `master.tex` for a split document:

```
\includeonly{chap1,chap3}%files to be included when latex runs

\documentclass[a4paper,12pt]{report}

%% preamble

\begin{document}

\include{titlepage}
\include{chap1}
\include{chap2}
\include{chap3}

\end{document}
```

Together with this master file there will be four files: `titlepage.tex`, `chap1.tex`, `chap2.tex` and `chap3.tex`. You use the `\includeonly` command to specify which files to load, but the `.aux` files for all the `\included` files will be read. You compile the document just by compiling `master.tex`.

## **3.6 EXERCISE 3**

---

An exercise on the Chapter 3 topics is given separately. See the  
WWW pages.

---

# Chapter 4

## Pictures

Putting pictures into  $\LaTeX$  documents can be quite tricky. This is one area where WYSIWYG programs have a certain advantage, because you can create pictures, diagrams and graphs using other windows software and then paste them directly into your document. Typically when writing a technical document you have a lot of mathematics, which is why you choose  $\LaTeX$ , and then want to include the odd diagram or graph of results. This chapter covers some of the options for doing this.

### 4.1 The figure environment

As far as  $\LaTeX$  is concerned, any picture is just a rectangle of information which has to be inserted somehow into your document. In  $\TeX$  language this is called a “box”. It is best not to manually format the position of figure boxes, but to let the `figure` environment position it for you. (The reason is that manual formatting will usually be fragile, it will work for one arrangement of the document, but if you change the order of the document or insert new material then the manual layout will no longer work). You can specify the box to go “here”, which is OK for small boxes of height up to about 5cm, but anything larger may give  $\LaTeX$  indigestion. In that case, you must give  $\LaTeX$  the right to choose its own page, where you specify the top or bottom of a text page, or on a page dedicated to figures. If you’re unlucky this might be 3 or 4 pages away from where you refer to it, but usually it will be sensibly placed (and you can always use the `\pageref` command to get its page number). Insert a figure like this:

```
\begin{figure}[h]
```

```
% the figure itself---see below
```

```

\caption{Some types of fish} %% optional
\label{fig.fish} %% optional
\end{figure}

```

where the [h] means here. Use **t** for top-of-page, **b** for bottom-of-page or **p**, which puts the figure on a page devoted to figures: if space allows several figures may appear on the same page. `\caption` is optional, the text inside will be copied by the `\listoffigures` command.

## 4.2 The scissors-and-paste method

The computationally least sophisticated method to include figures is just to define an adequate space within the figure box, then glue the pictures in by hand. The command for making space is:

```
\vspace*{5cm}
```

and both “cm” and “in” are understood. The `*` here has a special meaning:  $\LaTeX$  treats space as both compressible and expandable for purposes of formatting pages nicely, and usually this is OK, but for figure boxes it is crucial that they have the exact dimension specified, so we use a `*` to impose fixed, unchangable space.

This is of course a very crude method, but if you’re in a hurry and don’t have all the diagrams you need in electronic form it can be the simplest and fastest.

## 4.3 $\LaTeX$ ’s picture environment

The `picture` environment allows for the creation of diagrams with text, boxes, circles, lines and vectors (lines with arrows). It is convenient because it is internal to  $\LaTeX$ , but the limitations are that lines may only take certain slopes and that everything must be specified by explicit coordinates, which can get very painful. The diagram on page 7 was drawn using it. Here is the source code to give you an idea:

```

\begin{figure}[p]

% define the 'plotting space'

\begin{picture}(210,140)(30,0)

```

```

% boxes

\put(5,70){\framebox(60,15){EDIT}}
\put(90,115){\framebox(60,15){COMPILE}}
\put(90,20){\framebox(60,15){PREVIEW}}
\put(170,70){\framebox(60,15){PRINT}}

% dashed boxes

\put(90,135){\dashbox{5}(60,15){\tt\small TeX|(TeX)}}
\put(90,0){\dashbox{5}(60,15){\tt\small View|(View)}}
\put(235,70){\dashbox{5}(60,15){\tt\small View+Print}}

% lines with arrows

\put(120,110){\vector(0,-1){70}}
\put(35,90){\vector(4,3){45}}
\put(80,30){\vector(-4,3){45}}
\put(120,110){\vector(4,-3){45}}
\put(155,28){\vector(4,3){45}}

\end{picture}

\caption{The steps of doing \LaTeX.}
\label{fig.steps}
\end{figure}

```

You don't need to specify any space for the figure, this is worked out automatically.

`picture` environment is certainly worth considering: see the *LaTeX Book* §5.5, and *A Guide to LaTeX* Chapter 6. The thing to bear in mind is that line slopes may only take restricted values; therefore, plan your diagram fully beforehand and do the lines first.

## 4.4 Using the graphics package

LaTeX 2<sub>ε</sub> (but not 2.09) includes several standard packages for importing graphic images produced by other software.

A good image format to use is *encapsulated Postscript* (EPS), since this can retain image quality when resized. Many programs will output EPS

format directly, and if they don't you can use a general graphics manipulation program to convert an image from other formats (jpg, gif, bmp, etc.) into EPS.

On the web site, you'll find a small sample file illustrating the basic use of the `graphics` package. For the details on this package and others, see *A Guide to L<sup>A</sup>T<sub>E</sub>X* Chapter 6.

# Appendix A

## L<sup>A</sup>T<sub>E</sub>X on the web

There is a lot of material for L<sup>A</sup>T<sub>E</sub>X available on the web. The basic system is freely distributed on many web sites, and you will find versions of the software for all the common types of computers, operating systems and printers.

Also, since (as you've seen) L<sup>A</sup>T<sub>E</sub>X is a programmable, extensible system, there are many thousands of packages that extend L<sup>A</sup>T<sub>E</sub>X in different ways. For example, packages are available for using L<sup>A</sup>T<sub>E</sub>X to write in all the European languages that use Latin script. They provide special keyboard mappings to make accented letters easier to type. More importantly, they provide hyphenation rules to replace L<sup>A</sup>T<sub>E</sub>X's rules for English—these are essential for the correct breaking of lines and formatting paragraphs correctly. Beyond Latin script, you may also use L<sup>A</sup>T<sub>E</sub>X to write in Greek, Russian, Arabic, Farsi, Japanese, Chinese and others.

The main United Kingdom T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X archive is:

`www.tex.ac.uk`

This is part of the international Comprehensive T<sub>E</sub>X Archive Network:

`www.ctan.org`

CTAN is very comprehensive but not so easy to browse around for basic things. The following site provides a very good beginners overview, including links to free online and printable manuals:

`http://www.cs.technion.ac.il/~yogi/latex.html`

A search engine (such as `www.google.com`) is often the quickest way to track down specific information, but make sure whether the information relates to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> or 2.09.

# Appendix B

## Glossary

**ASCII:** stands for *American Standard Code for Information Interchange*. A standard set of 128 characters (upper case, lower case, numerals, symbols) which is used almost universally in computer systems. The term “ASCII text” is fairly synonymous with “plain text”.

**Case-sensitive:** used to describe a computer program that considers capital (upper case) and small (lower case) letters to be different.  $\LaTeX$ , UNIX and Linux are; Windows is not.

**DVI file:** the output file that results from running  $\LaTeX$  on a source (`.tex`) file is called the DVI file. DVI stands for *DeVice Independent*, meaning that the file contains the typeset document in a form which may be displayed or printed on any viewing/printing device.

**Knuth (Donald):** inventor of the  $\TeX$  typesetting language, and Professor at Stanford University (<http://www-cs-faculty.stanford.edu/~knuth/>).

**$\LaTeX$  Book:** the *User’s Guide and Reference Manual*, by Leslie Lamport (pub. Addison-Wesley, 1994, ISBN 0-201-52983-1). Also look at *A Guide to  $\LaTeX$*  by Helmut Kopka and Patrick Daly (Third edition, Addison-Wesley, 1999, ISBN 0-201-39825-7).

**Macro:** a sequence of  $\LaTeX$  commands grouped into a *definition* using `\newcommand`. Existing macros can be changed with `\renewcommand`. Macros with arguments can become simple functions and “subroutines”. A powerful programming feature of  $\LaTeX$ .

**Preamble:** the part of a  $\LaTeX$  source file between `\documentstyle` and `\begin{document}`. Used for title/author details, (re-)defining commands

and macros.

**T<sub>E</sub>X**: is a *typesetting language*, designed for the creation of technical documents.

**T<sub>E</sub>Xbook**: by Knuth (Addison-Wesley, ISBN 0-201-13448-9).

**Windows**: with a capital “W” refers to the Microsoft Windows operating system. “windows” with a small “w” refers to windows-type programs in general.

**WYSIWYG**: stands for *What-You-See-Is-What-You-Get*. A term applied to word processing software when the display on the screen is exactly equal (in principle!) to what will appear on the paper output. What L<sup>A</sup>T<sub>E</sub>X is not.