LaTeX Seminar Series

by Dr. John P. Costella

School of Physics, The University of Melbourne

Contents

Seminar 1: Introduction to LaTeX

1.1 Objectives

The objectives of this series of seminars are:

1. To understand why LaTeX is the most suitable and efficient document processing system for scientific documents.

2. To be able to create documents using LaTeX, and modify or correct documents created by others.

3. To gain familiarity with the macro capabilities of LaTeX, to further streamline the writing of, and clarify the logical structure of, your documents.

4. To be made aware of common errors of grammar, punctuation, notation, style, nomenclature, spelling and formatting, that reduce the professionalism of self-typeset documents.

1.2 Textbooks

The defining reference for standard LaTeX is the first edition of Leslie Lamport's "LaTeX: A Document Preparation System" (Addison-Wesley, 1986).

Unfortunately, this text is now out of print. In the past few months a *second* edition has become available, which covers a new, slightly incompatible version of LaTeX that I will discuss in Sec. 1.3.2. If you cannot get hold of a copy of the first edition, then the second edition will do, as 95% of it is identical. It costs about $60.00.

As an alternative source of information, *these seminar notes* contain most of the important information contained in the LaTeX manual—and, in any case, *all* the information you need to write a paper, preprint, book, thesis, *etc.*, of arbitrary length and complexity. There is also available a companion document to this one, which contains all of the LaTeX source code that I have written to produce these notes. By comparing the source code and the resulting output, you can get an immediate "tutorial" on the use of LaTeX. I will also be making available a "skeleton thesis", suitable for use by Honours, Masters or Ph.D. students.

Note that *all* the material that I am making available can be accessed through the World Wide Web, from my home page:

`http://www.ph.unimelb.edu.au/~jpc/homepage.htm`

There are also available a considerable number of books on LaTeX from technical bookshops. Again, unless you want to delve into the intricacies of LaTeX, you should not have need to buy these books.

In all cases, you should be careful to *never* buy a textbook on "TeX", rather than "LaTeX", unless you want to delve into plain TeX (which I strongly recommend against).

1.3 Why use LaTeX?

LaTeX is a complete *document processing system*, built on the TeX typesetting system.

You tell LaTeX, in plain text, the *logical structure* of your document; LaTeX takes care of the visual layout. If at any time you decide to use a different visual layout, you need simply change a single command at the beginning of your document.

LaTeX automatically numbers cross-references for you: equations, chapters, sections, tables, figures, and entries in your bibliography. Instead of using explicit numbers, you only need to invent "names" for the objects you wish to refer to (*e.g.*, `Gauss` for an equation that lists one of Gauss's laws); you then cross-reference the name, and LaTeX inserts the appropriate number automatically.

You can also make use of LaTeX's powerful macro capabilities, to encapsulate repetitive or complicated descriptions frequently appearing in your documents into a small set of short, efficient, easily understood and easily modified commands.

A properly-written LaTeX document is about as efficient a description of the document—in terms of time, memory and disk space—as is possible.

LaTeX is the standard format for the interchange of scientific documents throughout the world.

1.3.1 The names TeX and LaTeX

There are commands in LaTeX for producing the logos TeX and LaTeX.

If you are writing about TeX or LaTeX in a plain text document (*e.g.*, in e-mail, or with Microsoft Word), then you should spell them TeX and LaTeX respectively.

The name TeX is actually a capitalisation of the greek root $\tau\epsilon\chi$-, which is used to form English words such as "technical" and "technology"; TeX is a *technical typesetting system*; the offset E in the logo TeX symbolises advanced typesetting capabilities. The "X" at the end of TeX is thus a capital Greek chi, *not* an English ex. Thus, TeX is pronounced "teck"—like the word "technical", without the "-nical".

LaTeX is pronounced like the plastic latex, but with a Greek chi as the last letter: "lay-teck". The "La" in LaTeX come from the surname of Leslie Lamport, who created LaTeX.

1.3.2 I hear there is a new version of LaTeX?

The LaTeX reference manual describes version 2.09 as "the final version of LaTeX". It has remained so for almost a decade. If you send a paper to a colleague overseas, a preprint bulletin board, or a journal, this is what they will understand as "LaTeX".

Unfortunately, a "new and improved" version of LaTeX hit the streets in the middle of last year (June 1994), which the creators called "LaTeX 2_ε". It is sufficiently incompatible (about 5%) with standard LaTeX to be a hassle; but it is not sufficiently more advanced to be useful.

My advice: if you want to communicate with scientists and publishers around the world, then *do not touch the new version of* LaTeX *at the present time*.

If the new version of LaTeX does in fact become the standard over the next decade, it will be trivial for you to "upgrade" your documents. However, this is an unlikely scenario, as the people who have released "LaTeX 2_ε" are working on a *third* incompatible version of LaTeX, which they have tentatively dubbed "LaTeX 3".

If it ain't broke, don't fix it.

1.4 CostelLaTeX: My favourite macros

During my Ph.D., I wrote many "macros" that simplified or streamlined the task of writing a large LaTeX document. By the time I had completed the thesis, this collection was quite comprehensive.

Since then, I have rewritten the macros extensively, and added many more simplifying features. I have called this package of macros "CostelLaTeX" (pronounced "Costella-TeX"), to remind any users that they are my personal additions to LaTeX: they are *not* included with LaTeX itself (but you can obtain them from me free of charge).

I faced a dilemma when presenting these seminars for the first time, in October 1994: I could teach

1. Bare LaTeX only.

2. LaTeX first, and then the extra CostelLaTeX features in later seminars.

3. A seamless combination of LaTeX and CostelLaTeX features, as a single document processing package.

For that first presentation, I chose to go via route 2, for the simple reason that my audience consisted of both novices *and* experts in LaTeX, the latter of which would want a clear delineation of what new features my macros added to the language.

However, the price was high for the novices: I have found that many of them were confused by the presentation of many of the same subjects *twice* (firstly for bare LaTeX, and then again via the simplified CostelLaTeX commands), and have been in this way led to using a random (and confusing) mix of bare LaTeX and CostelLaTeX commands.

Since I only received a single complaint that my seminars presupposed the acceptance of the CostelLaTeX macros (everyone else seemed very grateful for them!), and since most of the "bare LaTeX experts" have now been through the seminars, *I have decided to follow choice 3 above for these revised seminars:* CostelLaTeX commands will be presented where they are simplest; bare LaTeX commands will only be listed where my macros have not needed to improve them.

If your requirements for LaTeX are such that you definitely *do not* want to use my macros, then, unfortunately, this seminar series is not for you. You should instead refer to a copy of either edition of Lamport's book.

You should also refer to Lamport for a further categorisation of bare LaTeX commands into "plain TeX" commands, and those that Lamport himself added in creating LaTeX. In these seminar notes, all commands not defined by CostelLaTeX are explicitly allowed as LaTeX commands in Lamport's reference, and I refer to them as "LaTeX commands", regardless of whether or not they were present in plain TeX.

You should also note that CostelLaTeX does *not* modify any of LaTeX's built-in commands in any way. In most situations, you can fruitfully graft an existing piece of bare LaTeX code into a CostelLaTeX document, and have it work without need for change.

(The only exception to this rule is the command `\include{}`. If you have existing files that use `\include` commands, you need to change them to `\input` or `\Include`. As a consequence, CostelLaTeX does not support LaTeX's `\includeonly` facility.)

1.4.1 UnMacro: A sneaky program

One problem with using the CostelLaTeX package is that, if you want to send a paper or report over the Internet, it is not usually practicable to send with it a copy of the CostelLaTeX macros (over twenty individual files).

To overcome this problem, I have written a new program, called UnMacro, which reads in an arbitrary LaTeX file—and in turn any files included by that file,—expanding out all macros that are found. The result is written to a *single* LaTeX file, that contains *no macros whatsoever*. You can then send this "unmacroed" file over the Internet, and any recipients will be able to process it simply using LaTeX, without needing to have even heard of "CostelLaTeX".

The UnMacro program is in the final stages of debugging, and will be released shortly. Again, check my World Wide Web home page for the latest information about both CostelLaTeX and UnMacro.

1.5 Running LaTeX and CostelLaTeX

LaTeX is a public domain program that can run on many different types of computer. There are no licensing requirements: you can install it on as many machines as you like.

However, some fancier "front-ends" for LaTeX (*e.g.*, Textures for the Macintosh) are proprietary software, and must be purchased and licensed.

The best way to find out how to run LaTeX on your machine is to ask someone successfully running it on a similar machine. If you are using a computer maintained by one of the departments of the University, ask the system operators for general advice as to the best procedures to follow.

Different people run LaTeX using different programs, but all implementations of LaTeX break the task into three fundamental components:

1. You prepare your LaTeX *source document*. This is a plain text document, containing special commands that tell LaTeX about the logical structure of your document. You can use any text editor to create this document. It should have extension `.tex`; *e.g.*, you might create a document called `myfile.tex`.

2. You run the LaTeX program on this source document. On many computer systems, you do this by typing a command like `latex myfile`; on others (*e.g.*, Textures for the Mac) you simply choose a menu item. LaTeX produces a "DVI" (device-independent) output file, which is a binary file that describes the typeset document, independent of what device (screen, printer, of whatever brand) you will eventually use to view it. This file has extension `.dvi` (*e.g.*, for the example above it would be `myfile.dvi`).

3. You then use a program that either shows you this DVI output on your screen, or else converts it into a form that your printer will understand.

This structure is shown diagrammatically in Fig. 1.1, as well as suitable choices of programs for IBM-PC, Macintosh and Unix X-Windows computers.

Note that Textures for the Mac incorporates these three tasks into one easy-to-use application; other similar "front-ends" are available for other machines. In these cases, instead of three separate programs, there are three simultaneously viewable *windows*: one showing the source text file, one showing LaTeX "talking" to you as it processes your file, and one showing the output.

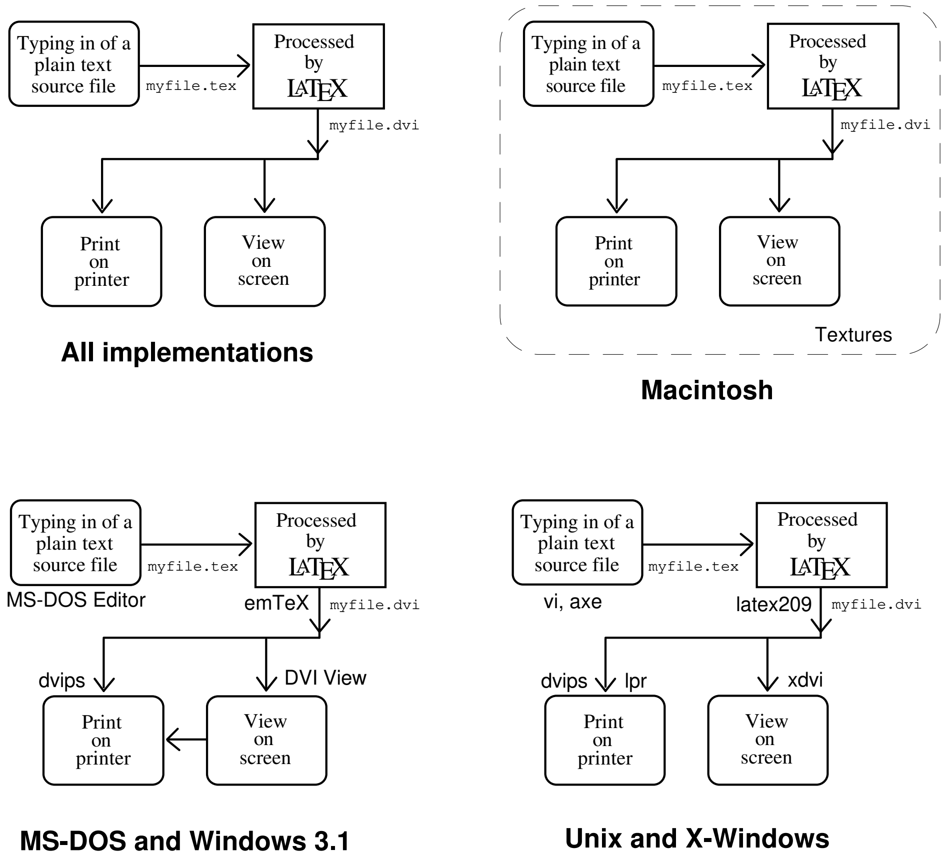


Figure 1.1: The three component parts of all implementations of LaTeX: text input, processing by LaTeX, and output viewing or printing.

1.5.1 A simple "Hello, world!" document

Half of the work in learning how to use LaTeX is in finding out *how* to perform the above three tasks on your computer. If you can create a document that simply says, "Hello, world!", successfully LaTeX it, and view it or print it, then you are 50% there already.

Start by typing the following into your text editor:

```
\documentstyle{article}
\begin{document}
Hello, world!
\end{document}
```

When you enter this text, take note of the following:

- At the end of each line, you press the ENTER or RETURN key on your keyboard.
- Do not leave any spaces in the text (except the space between `Hello,` and `world!`, of course).
- Note that the `\` character appearing before `documentstyle`, `begin` and `end` is a *backslash*: it is *not* the forward slash `/` that is usually underneath the question mark key on your keyboard. (The backslash key is in different places on different keyboards, unfortunately.)
- The characters `{` and `}` are *braces*, which are usually (but not always) above the square bracket keys `[` and `]` on your keyboard.

Once you have entered the above text, save it with the name `myfile.tex`, and exit from the text editor. (Although not forced to, Macintosh users of Textures *should* use `.tex` as the end of the name of a LaTeX source file—it's good practice, especially if you may want to transfer the file to colleagues with other types of computer. Users of Textures also need not quit the text editor: it remains a separate window, in the background. IBM-PC users can achieve the same effect by using the Windows Notepad as their text editor.)

You are now ready to let LaTeX process your source file. On most computers, you will type a command like `latex myfile` to do this. You will then see something like the following appear on your screen:

```
This is emTeX, Version 3.0 [3a] (no format preloaded)
**&lplain myfile
(myfile.tex
LaTeX Version 2.09 <25 March 1992>
(c:\emtex\texinput\article.sty
Standard Document Style ‘article’ <14 Jan 92>.
(c:\emtex\texinput\art10.sty))
No file myfile.aux.
[1] (myfile.aux) )
Output written on myfile.dvi (1 page, 232 bytes).
Transcript written on myfile.log.
```

The exact text that appears depends somewhat on your computer system: I myself use emTeX on an IBM-PC-compatible 486, installed in the directory `c:\emtex`. Regardless of what your implementation of LaTeX calls itself, the only important thing is the line

```
LaTeX Version 2.09 <revision date>
```

As long as you see this line appear, you can be sure that you are using standard LaTeX.

If everything has gone well so far, and you have received a "transcript" similar to that above, then you will have noticed a line of the form

```
Output written on myfile.dvi (1 page, 232 bytes).
```

This represents the end of LaTeX's responsibilities: it has processed your document, and written its output into a DVI file called `myfile.dvi`.

You now use another program to view or print the DVI file. If all goes well, you should see the text

> Hello, world!

somewhere near the top-left corner of the page, as well as the page number "1" centred near the bottom of the page. You have created your first LaTeX document!

1.5.2 How do I install CostelLaTeX?

The CostelLaTeX "package" simply consists of a number of plain text LaTeX source files. There are a number of ways that you may get the files:

- I can give them to you on a floppy disk.
- I can send them to you by e-mail.
- They may already be on your computer, if you are logging onto a machine maintained by one of the departments of the University.

If the files are not already on your machine, you must simply copy them to the directory where "TeX input files" are stored on your computer. If you don't know where this is, then follow the procedures outlined in the previous section, and take careful note of the lines in the transcript that look like

```
(c:\emtex\texinput\article.sty
(c:\emtex\texinput\art10.sty))
```

These lines will tell you the directory where you should put the CostelLaTeX files. For example, on my IBM-compatible 486, the above lines tell me that the "TeX input files" are stored in the directory

```
c:\emtex\texinput\
```

On your machine it could be anything—but the name of the directory will generally end in something like "`texinput`" or "`TeX inputs`". You can ask me for help if you get stuck on this.

1.5.3 CostelLaTeX's "Hello, world!" document

The CostelLaTeX equivalent of the "Hello, world!" document of Sec. 1.5.1 is simply

```
\include{costella}
\BeginDocument
Hello, world!
\EndDocument
```

If you write this into a file `hiagain.tex` using a text editor (note carefully the capitalisation and lack of braces of the `\BeginDocument` and `\EndDocument` commands!), and LaTeX it, you will get a transcript something like the following:

```
This is emTeX, Version 3.0 [3a] (no format preloaded)
**&lplain hiagain
(hiagain.tex
LaTeX Version 2.09 <25 March 1992>
\@input{costella.aux}
(c:\emtex\texinput\costella.tex
CostelLaTeX Version 1.12 <1 March 1995>
Copyright (C) 1992-1995 John P. Costella
)
\@input{macros.aux}
(macros.tex) (c:\emtex\texinput\default.tex
Default document conventions <1 March 1995>
) (c:\emtex\texinput\citlatex.tex
Standard LaTeX citation style <1 March 1995>
) (c:\emtex\texinput\british.tex
British spelling conventions <1 March 1995>
) (c:\emtex\texinput\report.sty
Standard Document Style `report' <14 Jan 92>.
(c:\emtex\texinput\rep12.sty)
(c:\emtex\texinput\titlepag.sty
Document Style Option `titlepage', Released 16 June 1991.
)) (c:\emtex\texinput\epsf.tex)
No file hiagain.aux.
[1] (hiagain.aux) )
Output written on hiagain.dvi (1 page, 232 bytes).
Transcript written on hiagain.log.
```

(The exact text depends on which implementation of LaTeX you have, and how it is set up on your computer.) You have created your first CostelLaTeX document!

If you cannot get the above procedure to work, then your copy of LaTeX probably does not have my macros installed correctly. Please contact me to sort out your problem.

1.5.4 Using New LaTeX

If you really want to use the new version of LaTeX (version 2_ε), then you simply need to insert the command `\NewLaTeX` before the `\BeginDocument` command, and CostelLaTeX will automatically adjust itself to use the modified LaTeX commands required by the new version.

If you have been given a set of files that automatically assumes the new version of LaTeX, you can revert to the standard version by inserting the command `\StandardLaTeX` before `\BeginDocument`.

In CostelLaTeX, these are the *only* changes you need make to switch between the two versions of LaTeX.

1.5.5 Handling errors

Since LaTeX documents contain a description of their logical structure, one must accept a slight amount of inconvenience on the occasions that you happen to type in a command containing an error in syntax or logic.

If you made a mistake in entering the sample "Hello, world!" documents above, then you will have already been confronted with LaTeX's way of reporting errors to you. If not, then let us create an error. Change the line

```
\BeginDocument
```

to

```
\BeginFocument
```

and save this modified file as `hierror.tex`. If you now LaTeX this modified file, you will receive a transcript something like the following:

```
This is emTeX, Version 3.0 [3a] (no format preloaded)
**&lplain hierror
(hierror.tex
LaTeX Version 2.09 <25 March 1992>
\@input{costella.aux}
(c:\emtex\texinput\costella.tex
CostelLaTeX Version 1.12 <1 March 1995>
Copyright (C) 1992-1995 John P. Costella
)
```

```
! Undefined control sequence.
l.2 \BeginFocument

?
```

Although it looks quite daunting at first, you will quickly get used to LaTeX's way of reporting errors to you. The line

```
! Undefined control sequence.
```

explains clearly that LaTeX doesn't understand a command in your file; in this case, it doesn't know how on earth to "`\BeginFocument`". Sometimes, however, your error might be quite subtle, and LaTeX will not be able to be quite as specific as to what *your* mistake was—it will only tell you the point at which *it* detects an inconsistency in logic. Sometimes, the only way to figure out what you did wrong is to ask someone who is more experienced with LaTeX; you will eventually build up a knowledge of the most common mistakes, and how LaTeX reacts to them.

The next piece of information in the above transcript is the most important: the *line number* in your source file that LaTeX foundered on:

```
l.2 \BeginFocument
```

The `l.2` means "line number 2"; the rest of the line shows the text of this line in your source file.

LaTeX then displays a question-mark prompt, and waits for you to type something. There are a number of options you have. If you simply hit the Enter or Return key (I will refer to this key as Enter from now on, regardless of what it is called on your keyboard), LaTeX will attempt to continue on, ignoring your error. In this case, the error is a little too fundamental (the `\BeginDocument` command is quite important!) to continue on very fruitfully; but in general you *should* try and continue on as if possible, to pick up any other errors before reprocessing the entire document. On the other hand, *ignoring* your first error may cause LaTeX to get a little confused later on, and it may report that *other* lines in your source file are incorrect, when in fact they are fine. You will learn to balance these two competing considerations as you gain familiarity with LaTeX.

Let us therefore assume that we wish to immediately address the error found. We have several choices:

- Press `x` and hit ENTER. This will exit from LaTeX. We can then open the source file with our text editor and go to the line containing the error.

- Press `e` and hit ENTER. This also exits from LaTeX; but, furthermore, on many systems, this will start your text editor, and *automatically* jump to the line that LaTeX reported as containing the error (line 2 in the above). Ask a local guru whether this works with your setup of LaTeX.

- Press `h` and hit ENTER, to receive some help from LaTeX. For our example above, doing so gives the help message

  ```
  The control sequence at the end of the top line
  of your error message was never \def'ed. If you have
  misspelled it (e.g., `\hobx'), type `I' and the
  correct spelling (e.g., `I\hbox'). Otherwise just
  continue, and I'll forget about whatever was
  undefined.
  ```

  In this case, you can follow the advice given by typing

  ```
  I\BeginDocument
  ```

  and LaTeX will then happily process the document. Note, however, that this *does not* change the error in the file itself: it is only a way to "help" LaTeX in its time of need.

- Type `I\stop` and hit ENTER. This is a special example of the suggestion `I<command>` given when we asked for help above. In this case, we are inserting the TeX command `\stop` which will attempt to quit LaTeX immediately. (Sometimes it doesn't work, however, depending on what sort of error has occurred; but it can never hurt to try it.) The reason that `I\stop` is often better than simply pressing `x` to exit is that using `x` *will not output the page on which the error occurred*, whereas using `I\stop` (if it is successful) *will* output the final page. The advantage is that you can then look directly at the DVI output, to see precisely where LaTeX stopped; the error is then usually obvious.

The most cautious (but not necessarily the most productive) way to work is to type `x` or `e` after any error that you encounter, and immediately repair the mistake. This can be wasteful of time if you start to work on lengthy documents, since if you have an error on the twentieth page of the document, you need to let LaTeX reprocess the first nineteen pages again before it gets past the repaired error. On the other hand, we will see in a later seminar that there are ways to split large documents into smaller component documents in a very efficient way, and in such cases it *may* still be quite efficient to repair practically every error as it appears.

1.6 Typesetting of plain text

Even though LaTeX's great attraction for a scientist is its powerful ability to typeset mathematical material to a professional level, you should not overlook the fact that LaTeX is a powerful document processing system for *plain text* documents too.

1.6.1 Human versus LaTeX typesetting

Fig. 1.2 shows a page scanned from an old printing of Charles Dickens' book *Dombey and Son*, which was typeset in the traditional (*i.e.*, human) way. Fig. 1.3 shows the same text produced by LaTeX, using a suitable document style.

You will see that the differences between them are minor; the same conventions for punctuation and spacing apply.

1.6.2 Generating a preprint

If Dickens *had* used LaTeX to write his books, he could have easily issued a *preprint* of the above section of *Dombey and Son*, as illustrated in Fig. 1.4, by simply changing a few commands at the start of his document.

This illustrates the power of LaTeX's *logical design*: you do not have to spend hours "fiddling" with the visual format of a document to get it just right; LaTeX's style files (either those supplied with LaTeX, or ones that colleagues or professional visual designers have created) take care of the visual layout for you.

DOMBEY AND SON

"We met Mr. Carker the junior in the street, queerer than ever. He bade me good-bye, but came behind us here —there's an odd thing!—for when we reached the shop door, I looked round, and saw him going quietly away, like a servant who had seen me home, or a faithful dog. How does she look now, Uncle?"

"Pretty much the same as before, Wally," replied Uncle Sol.

"That's right. Now I *am* off!"

And this time he really was: and Solomon Gills, with no appetite for dinner, sat on the opposite side of the fire, watching Florence in her slumber, building a great many airy castles of the most fantastic architecture; and looking, in the dim shade, and in the close vicinity of all the instruments, like a magician disguised in a Welsh wig and a suit of coffee colour, who held the child in an enchanted sleep.

In the meantime, Walter proceeded towards Mr. Dombey's house at a pace seldom achieved by a hack horse from the stand; and yet with his head out of window every two or three minutes, in impatient remonstrance with the driver. Arriving at his journey's end, he leaped out, and breathlessly announcing his errand to the servant, followed him straight into the library, where there was a great confusion of tongues, and where Mr. Dombey, his sister, and Miss Tox, Richards, and Nipper, were all congregated together.

"Oh! I beg your pardon, Sir," said Walter, rushing up to him, "but I'm happy to say it's all right, Sir. Miss Dombey's found!"

The boy with his open face, and flowing hair, and sparkling eyes, panting with pleasure and excitement, was wonderfully opposed to Mr. Dombey, as he sat confronting him in his library chair.

"I told you, Louisa, that she would certainly be found," said Mr. Dombey, looking slightly over his shoulder at that lady, who wept in company with Miss Tox. "Let the servants know that no further steps are necessary. This boy who brings the information, is young Gay, from the office.

98

Figure 1.2: A page from Dickens' book *Dombey and Son*.

DOMBEY AND SON

"We met Mr. Carker the junior in the street, queerer than ever. He bade me good-bye, but came behind us here—there's an odd thing!—for when we reached the shop door, I looked round, and saw him going quietly away, like a servant who had seen me home, or a faithful dog. How does she look now, Uncle?"

"Pretty much the same as before, Wally," replied Uncle Sol.

"That's right. Now I *am* off!"

And this time he really was: and Solomon Gills, with no appetite for dinner, sat on the opposite side of the fire, watching Florence in her slumber, building a great many airy castles of the most fantastic architecture; and looking, in the dim shade, and in the close vicinity of all the instruments, like a magician disguised in a Welsh rig and a suit of coffee colour, who held the child in an enchanted sleep.

In the meantime, Walter proceeded towards Mr. Dombey's house at a pace seldom achieved by a hack horse from the stand; and yet with his head out of the window every two or three minutes, in impatient remonstrance with the driver. Arriving at his journey's end, he leaped out, and breathlessly announcing his errand to the servant, followed him straight into the library, where there was a great confusion of tongues, and where Mr. Dombey, his sister, and Miss Tox, Richards, and Nipper, were all congregated together.

"Oh! I beg your pardon, Sir," said Walter, rushing up to him, "but I'm happy to say it's all right, Sir. Miss Dombey's found!"

The boy with his open face, and flowing hair, and sparkling eyes, panting with pleasure and excitement, was wonderfully opposed to Mr. Dombey, as he sat confronting him in his library chair.

"I told you, Louisa, that she would certainly be found," said Mr. Dombey, looking slightly over his shoulder at that lady, who wept in company with Miss Tox. "Let the servants know that no further steps are necessary. This boy who brings the information, is young Gay, from the office.

98

Figure 1.3: The same text as Fig. 1.2, typeset using LaTeX.

DICKENS–P–1846/01

DOMBEY AND SON

Mr. C. Dickens

No fixed address, London, England

(25 July 1846)

"We met Mr. Carker the junior in the street, queerer than ever. He bade me good-bye, but came behind us here—there's an odd thing!—for when we reached the shop door, I looked round, and saw him going quietly away, like a servant who had seen me home, or a faithful dog. How does she look now, Uncle?"

"Pretty much the same as before, Wally," replied Uncle Sol.

"That's right. Now I *am* off!"

And this time he really was: and Solomon Gills, with no appetite for dinner, sat on the opposite side of the fire, watching Florence in her slumber, building a great many airy castles of the most fantastic architecture; and looking, in the dim shade, and in the close vicinity of all the instruments, like a magician disguised in a Welsh rig and a suit of coffee colour, who held the child in an enchanted sleep.

In the meantime, Walter proceeded towards Mr. Dombey's house at a pace seldom achieved by a hack horse from the stand; and yet with his head out of the window every two or three minutes, in impatient remonstrance with the driver. Arriving at his journey's end, he leaped out, and breathlessly announcing his errand to the servant, followed him straight into the library, where there was a great confusion of tongues, and where Mr. Dombey, his sister, and Miss Tox, Richards, and Nipper, were all congregated together.

Figure 1.4: A hypothetical Dickens preprint version of Fig. 1.3.

1.6.3 A sample piece of real LaTeX text

In Fig. 1.5, I show an extract of text taken directly from the first two pages of my Ph.D. thesis. The LaTeX source code for this text is shown in Fig. 1.6 (excluding the commands to begin and end the document, which we are taking as given).

You will note that the LaTeX source code is (perhaps surprisingly) quite *readable*. It is, in large part, plain text.

It will be useful to refer to Figs. 1.5 and 1.6 when considering the instructions in the following sections.

1.6.4 Characters you use to type plain text

As you can see in Fig. 1.6, the plain English words of your document are produced in the same way as with a normal typewriter or word processor: you simply type them in, pressing the space bar between words. This is how I will write the plain alphabetic keys on your keyboard:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Note that the uppercase i (`I`) is different from the lowercase L (`l`); you should *never* type an `l` instead of an `I`, as some people used to do on typewriters:

```
l went down the street.  % WRONG!
I went down the street.  % correct.
```

You also type in numbers as you would with a typewriter: you use the ten digit keys along the top row of the keyboard:

```
1234567890
```

Note that the digit one (`1`) is different from a capital i (`I`) and a lowercase L (`l`); you should *never* use `l` or `I` instead of `1`, as used to be necessary on some typewriters:

```
The year is l994.  % WRONG!
The year is I994.  % WRONG!
The year is 1994.  % correct.
```

Classical electrodynamics is not "trendy". It is, in the minds of many, a dead subject—a vein that was mined out long ago. Its reputation is not helped by the disturbing tendency for "new" results to be so revolutionary as to be incredible—literally. Many seasoned veterans of the field can relate to Feynman's description [86] of another area of classical physics, since improved:

> I am getting nothing out of the meeting. I am learning nothing. Because there are no experiments this field is not an active one, so few of the best men are doing work in it. The result is that there are hosts of dopes here and it is not good for my blood pressure: such inane things are said and seriously discussed that I get into arguments outside the formal sessions (say, at lunch) whenever anyone asks me a question or starts to tell me about his "work". The "work" is always: (1) completely ununderstandable, (2) vague and indefinite, (3) something correct that is obvious and self-evident, but worked out by a long and difficult analysis, and presented as an important discovery, or (4) a claim based on the stupidity of the author that some obvious and correct fact, accepted and checked for years, is, in fact, false (these are the worst: no argument will convince the idiot), (5) an attempt to do something probably impossible, but certainly of no utility, which, it is finally revealed at the end, fails, or (6) just plain wrong. There is a great deal of "activity in the field" these days, but this "activity" is mainly in showing that the previous "activity" of somebody else resulted in an error or in nothing useful or in something promising. It is like a lot of worms trying to get out of a bottle by crawling all over each other. It is not that the subject is hard; it is that the good men are occupied elsewhere.

Figure 1.5: Text taken from the first two pages of my Ph.D. thesis.

```
Classical electrodynamics is not ``trendy''.
It is, in the minds of many, a dead subject---a vein that
was mined out long ago.
Its reputation is not helped by the disturbing tendency for
``new'' results to be so revolutionary as to be
incredible---literally.
Many seasoned veterans of the field can relate to
Feynman's description \cite{Feynman88} of another
area of classical physics, since improved:
%
\begin{quote}
I am getting nothing out of the meeting.
I am learning nothing.
Because there are no experiments this field
is not an active one,
so few of the best men are doing work in it.
The result is that there are hosts of dopes here and it is not
good for my blood pressure: such inane things are said
and seriously discussed that I get into arguments outside the
formal sessions (say, at lunch) whenever anyone asks me a
question or starts to tell me about his ``work''.
The ``work'' is always:
(1)~completely un-understandable,
(2)~vague and indefinite,
(3)~something correct that is obvious and self-evident, but
worked out by a long and difficult analysis, and presented
as an important discovery, or
(4)~a claim based on the stupidity of the author that some
obvious and correct fact, accepted and checked for years,
is, in fact, false (these are the worst: no argument will
convince the idiot),
(5)~an attempt to do something probably impossible,
but certainly of no utility, which, it is finally revealed at
the end, fails, or
(6)~just plain wrong.
There is a great deal of ``activity in the field'' these days,
but this ``activity'' is mainly in showing that the
previous ``activity'' of somebody else resulted in an error
or in nothing useful or in something promising.
It is like a lot of worms trying to get out of a bottle by
crawling all over each other.
It is not that the subject is hard; it is that the good men are
occupied elsewhere.
\end{quote}
```

Figure 1.6: The LaTeX source code for the output shown in Fig. 1.5.

Likewise, the digit zero (`0`) is different from the uppercase letter o (`O`), and they should never be exchanged:

```
Wait for 30 minutes.  % WRONG!
Wait for 30 minutes.  % correct.

Oh no!  % WRONG!
Oh no!  % correct.
```

You also type in the following punctuation symbols as you would on a normal typewriter:

```
. : ; , ? ! ( ) [ ] / * @ + =
```

which produce the output

. : ; , ? ! ( ) [ ] / * @ + =

respectively. The symbols `(` and `)` are sometimes called "brackets", but should properly be referred to as *parentheses*. It is the symbols `[` and `]` that are in fact "brackets"—sometimes redundantly referred to as "square brackets" to ensure no confusion on the part of the listener.

1.6.5 Punctuation symbols you can't type straight in

We now come to seven punctuation symbols that you can use, but *not* by typing them straight in:

```
# $ % & _ { }
```

The reason is that LaTeX reserves these symbols for special purposes. It also reserves the characters

```
\ ~ ^ | < > "
```

for special uses, so you should not type them into your plain text either.

The way to get the seven extra punctuation symbols promised above is to precede them by a *backslash* (`\`). In other words, you type

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

to obtain one of the symbols

$ % & _ { }

in your text. For example, if you type in the source code

```
The price in the Black \& Decker catalogue
is \$60.00, which is a 25\% reduction.
It is listed as part \#46524 in the
computer file all\_data.dbs,
in the \{prices.data\} section of the file.
```

you will get the output

> The price in the Black & Decker catalogue is $60.00, which is a 25% reduction. It is listed as part #46524 in the computer file all_data.dbs, in the {prices.data} section of the file.

You can already see what LaTeX reserves the character \ for: it denotes that a *special command* is following. We used it earlier to begin the special commands

```
\documentstyle{article}
\begin{document}
\end{document}
\include{costella}
\BeginDocument
\EndDocument
```

and will shortly see many more examples of such commands.

1.6.6 Quotation marks

If you examine Figs. 1.2 and 1.3, you will note that left and right double quotation marks are typographically distinct: they are what primary school teachers refer to as “66’s” and “99’s” respectively. Similarly, there are distinct left and right versions of the single quotation mark; the latter is also used for apostrophes, such as in

> Arriving at his journey’s end ...

Unfortunately, the normal typewriter keyboard only contains “straight up-and-down” versions of single and double quotes, namely, `'` and `"`. Thus, we need to have some way of uniquely specifying whether we want the left or the right versions of these symbols. The solution used in LaTeX is as follows:

1. The `"` key is discarded altogether, at least as far as quotation marks are concerned.

2. The “normal” single-quote key, which probably looks something like ´ on your keyboard, represents a *right* quotation mark. I will denote this key by `’` in these notes.

3. There is another key that can be found on all computer keyboards (but not many typewriters), that probably looks something like ` on your keyboard. This key represents a *left* quotation mark. I will denote this key by `‘` in these notes.

4. To produce double quotes, you simply type two single quotes—of the left or right variety as required.

You can see this at work in the first sentence of Figs. 1.5 and 1.6: the output

> Classical electrodynamics is not “trendy”.

is produced by typing the code

```
Classical electrodynamics is not ‘‘trendy’’.
```

where you type the following keys to produce the last word:

`‘` `‘` `t` `r` `e` `n` `d` `y` `’` `’` `.`

Another good example, adapted from one in the LaTeX manual, both illustrates the usage of all of the quotes—including the right single quote’s use as an apostrophe,—while also making an important point about changing typographical trends:

```
‘Historical convention’ dictates that punctuation go
inside quotes, like ‘‘this,’’ but modern writers think
it’s more logical to do ‘‘this’’.
```

produces

> 'Historical convention' dictates that punctuation go inside quotes, like "this," but modern writers think it's more logical to do "this".

Incidentally, I recommend you use the latter, more logical punctuation convention described here (except in dialogue, if you are writing a novel), since it is now the preferred convention by most publishers of scientific material.

Finally, if you need to place a single quote next to a double quote, then you need to use the `\,` command, which indicates that a small amount of space should be inserted in the output, to separate symbols that would otherwise be ambiguous. The LaTeX manual gives the example

```
``\,`Fi' or `fum?'\,'' he asked.
```

which produces the output

> " 'Fi' or 'fum?' " he asked.

Note carefully that the `\,` command is a *logical* command, and should *not* be used to arbitrarily put extra space in your output; it should *only* be used in the situations I describe in these notes.

1.6.7 Hyphens and dashes

If you look at Figs. 1.2, 1.3 and 1.5, you will see that there are at least two different types of "dashes" that are used in typeset material. One is the "long dash", such as in the line

> It is, in the minds of many, a dead subject—a vein that was mined out long ago.

Another is the *hyphen*, as in

> ... (1) completely un-understandable, ...

In fact, if you look carefully at professionally typseset material, you will find that there is a *third* type of dash, of a length in-between the hyphen and the "long dash", that is used, for example, when you are denoting a range of numbers, such as 1993–2001. There is also a "double-dash", usually used when a person is "cut off" in dialogue:

> I mused, "This seminar is such as load of——"
>
> "Shut up!" cried Dr. Costella.

You might also be thinking that a *minus sign* is a type of dash—probably because you have used a dash on a typewriter or word processor for the purposes of a minus sign. However, in LaTeX, minus signs should *only* be inserted in the same way as other mathematical material; they do not really belong in a plain text document. I will therefore defer a consideration of minus signs to Seminar 2.

It is important that you understand the difference between the four types of dashes: what they are called, when you should use them, and how you specify them to LaTeX. To ensure no confusion, I will treat each in turn.

Hyphens

The shortest dash is the *hyphen*, -. It is used in the following situations:

1. To denote a prefix or suffix, such as "pre-" or "-ology".

2. To attach a prefix to a word, if omitting the hyphen would lead to ambiguity, such as "un-ionised"; if omitting the hyphen would produce a double letter, such as "non-negative"; or for prefixes that are themselves complete words, such as "self-evident", "well-known" and "three-divergence".

3. To "bind" successive words together in a closer way than normal spacing would suggest, to disambiguate meaning, such as

   > We performed a least-squares fit ...

   rather than

   > We performed a least squares fit ...

   This is usually done when a group of words ("least squares" in the above) is being used as an adjective (for the noun "fit" in the above). For example, if one wrote

   > He is a run of the mill physicist.

then the eye would, at first, see “He is a run”, and jump to the conclusion that “run” is the object of the verb “is”; but then this parsing of the sentence would have to be thrown out when “of the mill” is seen. If we instead write

> He is a run-of-the-mill physicist.

then the eye “processes” the adjectival phrase “run-of-the-mill” as a single unit, as it indeed should be.

4. To separate parts of a person’s name that are hyphenated, such as “Olivia Newton-John”.

5. In written-out numbers of two or more words, such as “twenty-five”.

6. When scientific adjectives are hyphenated together, such as “positive-definite”.

7. End-of-line hyphenation breaks of polysyllabic words. This is a task that is performed by the typesetter (in our case, TEX).

To insert a hyphen into your LATEX code, you press the dash key `-` on your keyboard *once*. For example, the code

```
The self-energy is positive-definite.
```

produces the output

> The self-energy is positive-definite.

To ensure that you are not using the hyphen incorrectly, occasionally check your LATEX code against this short checklist:

- You should *never* leave a space both before and after a hyphen. If you have done so, you probably need an *em dash* instead (*see* later section).

- The only time you leave a space *before* a hyphen is when you are explicitly writing about a suffix, such as in

  ```
  There are so many words ending in -ology these
  days that they need a new science called Ologyology.
  ```

which produces the output

> There are so many words ending in -ology these days that they need a new science called Ologyology.

- You only leave a space *after* a hyphen if you are explicitly writing about a prefix, or are using the "distributive law" convention for multiple prefixes; for example,

```
The use of the prefix non- seems to have
increased in the pre- and post-war periods.
```

  which produces the output

  > The use of the prefix non- seems to have increased in the pre- and post-war periods.

- You should *never* have need to put a hyphen between numbers; this should always be an *en dash* (*see* next section). For example,

```
We should wait 10-20 minutes.   % WRONG!
We should wait 10--20 minutes.  % correct.
```

- You should *not* place a hyphen between the names of different people; you should always use an *en dash* (*see* next section). For example,

```
It is just the Aharonov-Bohm effect.   % WRONG!
It is just the Aharonov--Bohm effect.  % correct.
```

- You should not have a hyphen between two nouns of equal standing; use an *en dash* instead (*see* next section). For example,

```
Consider the electron-photon interaction.   % WRONG!
Consider the electron--photon interaction.  % correct.
```

- You should never be using a plain hyphen as a minus sign. For example,

```
The temperature went down to -10 degrees.  % WRONG!
```

  We will see how to produce minus signs correctly in Seminar 2.

- If a hyphen "doesn't look long enough", then you most likely require an *em dash* (*see* later section).

En dashes

The second-longest dash is called an *en dash*, because it is about as wide as a capital letter 'N', as you can see from the following comparison:

–
N

You use an en dash in the following situations:

1. Any time you have need to place a dash between two numbers. This is usually when you are indicating a *range*, such as "10–12 weeks" or "1994–6"; but it is also applicable when the two numbers relate to different quantities; for example, to refer to "page 3–27" of a document that has pages numbered chapter by chapter.

2. Between the names of *different people*, such as in the "Bargmann–Michel–Telegdi spin precession equation", or the "Dirac–Pauli representation".

3. Between nouns of equal standing; for example, in "proton–neutron cross-section". However, this does *not* apply if the first noun is being used as an adjective, such as in "the density-distribution function". Thus,

   ... the electron–neutrino interaction ...

   is referring to *two* particles (an electron and a neutrino), whereas

   ... the electron-neutrino interaction ...

   is referring to *one* particle, the *electron neutrino*—"electron" serving here as a qualifying adjective, to distinguish it from the muon neutrino and the tauon neutrino. (Presumably the rest of the sentence would indicate *what* the electron neutrino is interacting with.)

You insert an en dash into your text by pressing the dash key on your keyboard *twice in a row*. For example, the code

```
The Aharonov--Bohm electron--field interaction
should be measurable with a run-time of 10--12 weeks.
```

produces the output

> The Aharonov–Bohm electron–field interaction should be measurable with a run-time of 10–12 weeks.

The following checklist may help you avoid common errors with en dashes:

- You should never have a space before *or* after an en dash.
- You should *not* use an en dash to simulate a minus sign, even though it might look quite similar to one.
- All dashes between numbers should be en dashes.
- *Do not* use en dashes between numbers that are written out; use the word "to" or "through" instead. For example,

```
Wait ten--twelve weeks for a reply.    % WRONG!
Wait ten to twelve weeks for a reply.  % correct.

It appears on pages three--seven.         % WRONG!
It appears on pages three through seven.  % correct.
```

- Do *not* use en dashes to hyphenate a person's name, such as in "Olivia Newton-John".
- Do not go "en dash crazy" and replace normal hyphens by en dashes wherever they appear. Each has specific purposes.

Em dashes

The next longest form of dash is the *em dash*, so called because it is about as wide as a capital letter 'M':

—
M

Em dashes are *only* used as punctuation marks in your text. They are produced by pressing the dash key on your keyboard *three times in a row*. For example, the code

```
It is, in the minds of many, a dead subject---a vein
that was mined out long ago.
```

produces the output

> It is, in the minds of many, a dead subject—a vein that was mined out long ago.

A subtle question arises about whether one should place a *space* before and after an em dash. Some publishers do; some don't. For example, the above piece of text, if published in a World Scientific journal, would appear as

> It is, in the minds of many, a dead subject — a vein that was mined out long ago.

which you can produce yourself simply by inserting a space before and after the dash:

```
It is, in the minds of many, a dead subject --- a vein
that was mined out long ago.
```

To allow a more powerful control over the spacing (or otherwise) of em dashes, CostelLaTeX provides the command

```
\emdash
```

that *automatically* inserts a space before and after the em dash—or not—depending on what sort of *document conventions* are specified (these will be described in detail in a later seminar). To ensure its correct functioning, *do not* leave a space before `\emdash`. For example, the code

```
This works\emdash really!
```

produces the output

> This works—really!

or

> This works — really!

depending on the document conventions in force.

It is more difficult to describe the correct general usage of em dashes than the others, since then we are getting into complex questions of grammar and syntax.

Double-dashes

CostelLaTeX provides the command

```
\doubledash
```

for a dash that is twice as long as an em dash. For example, the output at the start of this section was produced using the code

```
I mused, ``This seminar is such as load of\doubledash''

``Shut up!'' cried Dr.~Costella.
```

Except to denote the interruption of dialogue, the double-dash should be used rarely, if at all.

1.6.8 Unbreakable spaces

If you looked closely at Fig. 1.6 earlier, you may have wondered what the "tilde" characters (~) in the input were there for—for example, the line

```
(1)~completely un-understandable,
```

The answer is that ~ is another of LaTeX's special commands: it signifies that a space should be produced in the output, but that *an end-of-line break cannot occur at that point*.

You can see why I used these *unbreakable spaces* where I did, in Fig. 1.6, by considering Fig. 1.5: Imagine if LaTeX happened to place "(1)" at the end of one line of text, and "completely un-understandable" on the next. The "(1)" would be left sticking out like the canine's proverbials—looking something like an equation number, in fact—rather than a numbering of Feynman's first point of argument. By placing an unbreakable space between "(1)" and "completely", I can be sure that, no matter *what* text I insert into the paragraph, the string "(1) com" would always remain a single unit. (LaTeX could choose to break the word "completely" after "com-", but the reader would still understand that the number referred to the following text.)

Of course, using ~ like this was "defensive programming" on my part: I anticipated a bad line break before it actually occurred, based on previous experience. Usually, however, you won't think to use a ~ instead of a space until LaTeX *does* in fact break a line where you would wish it wouldn't. That's fine: you will build up a knowledge of where you should use them. But, as you gain experience, you should try and insert unbreakable spaces in *every* place that you wouldn't want a line break—even if LaTeX isn't breaking your line there anyway. The reason is that, if you were to suddenly *change the document style* one day, you might find that one of these "sleeping" bad line breaks gets activated; you will then waste time editing the file, re-processing it, and re-printing it.

As a guide, you should use unbreakable spaces in the following situations; the reasons should be obvious, if you consider what the output would look like if a line break appeared there:

- When referring to "Fig. 1", "Table 2", "Eq. (3)", *etc.* We will see in a later seminar how to generate the *number* of the figure, table, equation, *etc.*, automatically; later still I will provide a command that includes the unbreakable space for you.

- Between the words and initials in a person's name. For example, use

  ```
  John~P.~Costella and Bruce~H.~J.~McKellar
  ```

  rather than

  ```
  John P. Costella and Bruce H. J. McKellar.
  ```

  If necessary, you may need to help LaTeX hyphenate a name properly (*see* Sec. 1.6.15), so that it can break a line *somewhere*. (Incidentally, you should *always* place a space after an initial; *never* run initials together. Some journals, unfortunately, set a bad example.)

- When part of the name of an object is a word of only one letter, or is a number; for example,

  ```
  ANSI~C
  Physical Review~D
  Terminator~II
  Revision~12.6
  5~October, 1994
  ```

1.6.9 Normal spaces

An unbreakable space is a relatively easy concept to understand. A little more subtle are what might be referred to as "normal spaces"—the spaces that appear between words and sentences.

The first subtlety arises with any typeset material whatsoever. Consider Figs. 1.2 and 1.3. You will notice that there is a *little more space* left after the end of a sentence than between words; typeset text looks just that little bit better when this is done. This, of course, was also the motivation behind the old "typewriter rule of thumb" that stated you typed *two* spaces after the end of a sentence—which then took a whole lot of unlearning when proportional-font word processors came into existence.

Since LaTeX is a professional typesetting system, it must also place a little extra space after each sentence. Unfortunately, it is not easy for a simple machine like LaTeX to know whether your full stops are ending a sentence, or simply denoting abbrevations or initials. It therefore makes the following assumptions, which work most of the time:

1. A full stop after a capital letter is not the end of a sentence (it is usually someone's initial).

2. A full stop after any other character *is* the end of a sentence (usually true, but often not).

LaTeX then provides the following ways of overriding either of these two rules:

1. The code `\@.` produces an end-of-sentence full stop, even if it follows a capital letter. For example, the code

   ```
   I usually write programs in FORTRAN\@.
   I am therefore a masochist.
   ```

   produces the correct output

   > I usually write programs in FORTRAN. I am therefore a masochist.

   rather than the incorrectly-spaced result

   > I usually write programs in FORTRAN. I am therefore a masochist.

if \@ were omitted.

2. The code \␣ (where by "␣" I am denoting a space) will produce an ordinary (*i.e.*, inter-word) space, regardless of where it appears. For example, the code

   ```
   McKellar et al.\ found similar results
   to the ones obtained above.
   ```

   produces the correct output

   > McKellar et al. found similar results to the ones obtained above.

   rather than the incorrectly-spaced result

   > McKellar et al. found similar results to the ones obtained above.

   if \␣ were replaced by ␣. You must also use this "ordinary space" command if the full stop is followed by parentheses or quotes; for example, in

   ```
   Fermions (electrons, etc.)\ have half-integral spin.
   ```

   and

   ```
   Use ``et al.''\ for the omitted authors.
   ```

These rules apply identically to question marks (?), exclamation marks (!) and colons (:). Note that a colon used normally in the middle of a sentence *should* have a little extra space after it. You would only override this extra space if the colon were a part of a special notation, such as in

```
Wipe the c:\ disk and reinstall DOS\@.
That'll get rid of the virus.
```

which produces

> Wipe the c: disk and reinstall DOS. That'll get rid of the virus.

You should note that the command \␣ can be used to produce an ordinary space in the output in any situation, not just after the punctuation marks listed above. We will have need to use it a number of times shortly, when other subtleties of spaces become apparent.

1.6.10 Ellipses

You may have noticed that I have several times omitted text in the examples given, and denoted the omitted text by "three dots", such as in

> … the electron-neutrino interaction …

The proper name for this set of "three dots" is an *ellipsis*. You should note that you do *not* obtain an ellipsis by typing three full stops in a row: the code

```
... the electron-neutrino interaction ...  % WRONG!
```

would produce the output

> ... the electron-neutrino interaction ...

which is not spaced correctly.

The *correct* way to insert an ellipsis is to use the command `\ldots`. For example, you would write

```
Go on \ldots, it won't hurt a bit!
```

to produce

> Go on …, it won't hurt a bit!

Note that if the character adjacent to an ellipsis is a letter, then you should put a space between them; but if the adjacent character is a punctuation mark, you should not.

A problem now arises. Imagine that we wanted (for some unknown reason) to produce the output

> a…b

We cannot just write

```
a\ldotsb
```

because LaTeX would complain that it doesn't know what "`\ldotsb`" means. The problem arises because LaTeX allows commands that are made up of *a backslash followed by lowercase and uppercase letters*, such as `\ldots`. To figure out where the command name ends, it simply looks for the first non-letter character it finds. This would be well and good if we were writing

```
0\ldots1
```

because 1 is not a letter, so LaTeX does not get confused. But how do we follow a command name with a letter?

LaTeX's solution to this problem is to deem that a *space* following a command name ends the command name, but *without creating a space in the output*. In other words, we type

```
a\ldots b
```

to produce

a. . . b

Note that the space after `\ldots` *does not* appear in the output. If it did, there would be no way to put alphabetic characters immediately after command names.

Now another problem arises: how to produce text like

He . . . is a mongrel.

If we simply type

```
He \ldots is a mongrel.
```

then we get the output

He . . . is a mongrel.

because the space after `\ldots` serves only to end the command name, not to put a space in the output.

The solution is to use the "ordinary space" command, `\␣`, described in the previous section. In other words, we type

```
He \ldots\ is a mongrel.
```

to get our desired result. Note that LaTeX now knows that `\ldots` is the first command, since the `\` character following it is not a letter. It then reads `\␣` as the next command. It then reads the rest of the sentence.

1.6.11 Comments

You may have noticed that I often placed comments to the right of the sample source code, such as the "`% WRONG!`" in

```
I went down the street.  % WRONG!
```

As you may have guessed, the `%` symbol is a special command that tells LaTeX to simply *ignore* everything else on that line. It is logically equivalent to grabbing the next line (from its first non-space character), and gluing it onto the end of the current line, starting right on top of the `%` symbol itself. Thus, you can write

```
Professor Opat will talk about antidisestablish%
    mentarianism at 1:00~pm today.
```

to produce the output

> Professor Opat will talk about antidisestablishmentarianism at 1:00 pm today.

LaTeX has simply glued the end of the word "antidisestablishmentarianism" onto its start, since no space appears before the `%`. However, LaTeX *does* ignore all spaces at the start of a line following a `%` command.

You can leave a blank line in your source file (well, *almost* blank)—say, for visual clarity—without starting a new paragraph, by typing a `%` as the first character:

```
This will not
%
start a new paragraph.
```

produces

> This will not start a new paragraph.

Note, however, that you *cannot* split a command name over two lines. For example,

```
He went down to the shop \ldo%
ts\ and bought some lollies.    % WRONG!
```

is incorrect because `\ldots`, a command name, may not be split by a `%` symbol.

1.6.12 Commenting out slabs of source code

From the previous section, we can see that you can "comment out" a line of your LaTeX source code (*i.e.*, tell LaTeX to ignore it) by placing a `%` command at the start of the source line.

This procedure is, however, very tedious for larger slabs of text. Instead, CostelLaTeX allows you to simply preface the code to be commented out by

```
\comment{
```

and place a matching

```
}
```

after the end of it. You do, however, need to ensure that all braces appearing inside the commented text are matched.

For example, the code

```
This is
\comment{
a command \sneakycommand
\e{Yelp!}
that isn't working right
}
working OK.
```

produces the output

> This is working OK.

1.6.13 Emphasising text

You may have noticed that I tend to use *italicised* text for emphasis quite frequently. You should always use italics for emphasis in printed matter, never underlining.

CostelLaTeX provides the command `\e{}`, which emphasises the text enclosed in the braces. For example, the code

```
I told you that he \e{did}n't!
```

produces the output

> I told you that he *did*n't!

ò	\`o	õ	\~o	ǒ	\v o	o̧	\c o
ó	\'o	ō	\=o	ő	\H o	ọ	\d o
ô	\^o	ȯ	\.o	o͡o	\t{oo}	o̲	\b o
ö	\"o	ŏ	\u o				

Table 1.1: Accented foreign characters in text mode. The letter 'o' is used for illustration only; any letter may be accented.

œ	\oe	å	\aa	ł	\l	¿	?`
Œ	\OE	Å	\AA	Ł	\L	¡	!`
æ	\ae	ø	\o	ß	\ss		
Æ	\AE	Ø	\O				

Table 1.2: Special foreign characters and punctuation symbols.

1.6.14 Foreign characters

LaTeX can produce most characters required for printing text or names from European languages.

However, these commands are only suitable for *small fragments* of foreign text; you should *not* try to use them to write complete foreign-language documents. One reason is that English-language implementations of LaTeX are set up to hyphenate words according to English conventions; these algorithms will incorrectly hyphenate foreign words, in general. More importantly, residents of other countries often have access to a more streamlined preprocessing system for LaTeX, which converts extra ANSI character codes (typeable directly from the keyboard) to LaTeX's foreign-language codes. This functionality, however, lies outside the scope of these seminars.

To produce accented foreign characters, in plain text, use one of the commands listed in Table 1.1. To produce other special characters and punctuation symbols not found in English, use the commands listed in Table 1.2.

Note that, to place accents above the letter 'i' or 'j', you need to first remove their dots. The commands `\i` and `\j` produce a dotless 'i' and 'j' repectively. For example, the code

```
\'El est\'a aqu\'\i.
```

produces the output

> Él está aquí.

You should also note that the commands listed in Tables 1.1 and 1.2 are for use in *plain text only*. Accented mathematical symbols require a different set of commands, and will be described in Seminar 2.

1.6.15 Helping LaTeX with hyphenation

LaTeX can hyphenate practically any English word. Sometimes, it can't, and may complain that it can't find a good place to break a line. Sometimes, it hyphenates someone's name, in a way that doesn't look right; this is because it is only designed to hyphenate *English words* correctly.

To tell LaTeX how to hyphenate an individual word in your text, you insert `\-` commands wherever you think it should be allowed to break it. For example, you might write

```
John P.~Cost\-ella and Bruce H.~J.~Mc\-Kel\-lar
```

1.6.16 Footnotes

You can place a footnote in your text by using the command `\footnote{}`. For example, the code

```
Fermions\footnote{Fermions are particles
with half-intgral spin, \ie they obey
Fermi--Dirac statistics.} are very nice particles.
```

produces the output

> Fermions[1] are very nice particles.

where you will find the footnote at the bottom of this page. Note that there is no space before the `\footnote` command.

I suggest that you think carefully before using footnotes in your document. These days, they tend to be frowned upon, except in special situations.

[1] Fermions are particles with half-intgral spin, *i.e.*, they obey Fermi–Dirac statistics.

1.6.17 Quotations

Short quotations

You may have noticed that my quote from Feynman in Fig. 1.5 was "indented", compared to the surrounding text. If you examine the source code in Fig. 1.6, you will see that this was achieved by simply telling LaTeX to put the enclosed text in a "`quote` environment". In other words, you put the command

```
\begin{quote}
```

at the start of the quotation, and

```
\end{quote}
```

at the end.

The `quote` environment is generally useful for small quotes; I have used it extensively throughout these seminar notes to "display" output of the sample pieces of code. It has the useful property that its paragraphs are *not* indented; instead, if you begin a new paragraph within a `quote` environment, a small amount of vertical space is added. For example, the code

```
I will now show you a quote.
I will waffle on a bit so that you can see how
the paragraph widths are arranged:
\begin{quote}
Hello, this is the first paragraph of a quote.
It is not very useful to you.

This is the second paragraph of the quote.
It is equally useless.
It does, however, illustrate the point.
\end{quote}
That was the quote.
It was a pretty useless quote, really, but
that's just bad luck.
```

produces the output

I will now show you a quote. I will waffle on a bit so that you can see how the paragraph widths are arranged:

> Hello, this is the first paragraph of a quote. It is not very useful to you.
>
> This is the second paragraph of the quote. It is equally useless. It does, however, illustrate the point.

That was the quote. It was a pretty useless quote, really, but that's just bad luck.

Longer quotations

Another LaTeX environment suitable for quotations is, strangely enough, called the `quotation` environment. In it, paragraphs *are* indented, and no vertical space is added between them (in other words, the output looks just like a normal set of paragraphs, except that the margins are indented). For example, the code

```
This is some code from a previous
section of these notes, that you should find
very fascinating:
\begin{quotation}
During my Ph.D., I wrote many ``macros''
that simplified or streamlined the task of
writing a large \LaTeX\ document.
By the time I had completed the thesis,
this collection was quite
comprehensive.

Since then, I have rewritten the macros
extensively, and added many more
simplifying features.
\end{quotation}
Not very interesting, I guess, but I needed to
put \e{something} in as an example.
```

produces the output

> This is some code from a previous section of these notes, that you should find very fascinating:
>
> > During my Ph.D., I wrote many "macros" that simplified or streamlined the task of writing a large LaTeX document. By the time I had completed the thesis, this collection was quite comprehensive.
> >
> > Since then, I have rewritten the macros extensively, and added many more simplifying features.
>
> Not very interesting, I guess, but I needed to put *something* in as an example.

As should be by now apparent, you can nest the `quote` and `quotation` environments inside other `quote` and `quotation` environments (for example, I have used an enclosing the `quote` environment myself when producing the above examples).

Centred quotations

There is also a `centre` environment, which is similar in form to the `quote` and `quotation` environments, except that the text is centred. For example, the code

```
... are working on a \e{third} incompatible
version of \LaTeX, which they have tentatively
dubbed ``\LaTeX~3''.
\begin{centre}
\e{If it ain't broke, don't fix it.}
\end{centre}
```

produces the output

> ... are working on a *third* incompatible version of LaTeX, which they have tentatively dubbed "LaTeX 3".
>
> *If it ain't broke, don't fix it.*

You should only use the `centre` environment rarely, for special purposes (such as the one above).

1.6.18 Bulletted and numbered lists

Creating bulletted and numbered lists with LaTeX is very easy and logical.

Bulletted lists

To create a bulletted list, you simply use the `itemise` environment. Each item is begun with the command `\item`. For example, the code

```
There are a number of ways that you may get the files:
\begin{itemise}
\item
I can give them to you on a floppy disk.
\item
I can send them to you by e-mail.
\item
They may already be on your computer,
if you are logging onto a machine maintained
by one of the departments of the University.
\end{itemise}
If the files are not already on your machine, you must
simply copy them to the directory where
``\TeX\ input files'' are stored on your computer.
```

produces the output

> There are a number of ways that you may get the files:
>
> - I can give them to you on a floppy disk.
> - I can send them to you by e-mail.
> - They may already be on your computer, if you are logging onto a machine maintained by one of the departments of the University.
>
> If the files are not already on your machine, you must simply copy them to the directory where “TeX input files” are stored on your computer.

Numbered lists

To create a numbered list, you simply use the `enumerate` environment rather than the `itemise` environment. For example, the code

```
You must follow these steps to defuse the bomb:
\begin{enumerate}
\item
Remove the bomb casing.
\item
Cut the red wire.
\item
Cut the black wire.
\item
Cut the blue and orange wires simultaneously.
\end{enumerate}
\item
Before performing step~2, you
need to unscrew the detonator cap.
I hope you did this.
```

produces the output

> You must follow these steps to defuse the bomb:
>
> 1. Remove the bomb casing.
> 2. Cut the red wire.
> 3. Cut the black wire.
> 4. Cut the blue and orange wires simultaneously.
>
> Before performing step 2, you need to unscrew the detonator cap.
> I hope you did this.

1.6.19 Verbatim mode

You will probably have noticed that I have been able to give fragments of sample code in a font that looks like an old typewriter, such as when I write "`\BeginDocument`". You may have also noticed that LaTeX does not complain that I am using commands inappropriately when I am doing so; it simply types them out verbatim.

In fact, that it precisely what I tell LaTeX to do: enter a special mode called *verbatim mode*. In this mode, it simply prints every character of my source code—including spaces and line breaks, and all the characters not normally permissible in plain text—until I tell it to exit from verbatim mode.

There are two forms of verbatim mode, which I will now describe.

Verbatim mode inside a paragraph

To print a small amount of text verbatim, in the middle of a paragraph, you use the command

```
\verb
```

followed by an almost-arbitrary symbol character (any aymbol you can type, except for `*`). The character you specify is the *delimiter* for verbatim mode: LaTeX simply keeps typing characters verbatim, until it finds a matching delimiter character. For example, the code

```
Type \verb+\BeginDocument+ to begin the document.
```

produces the output

> Type `\BeginDocument` to begin the document.

where here I have chosen to use `+` as the delimiter. As another example, the code

```
Type \verb:a+b = c: into the document.
```

produces the output

> Type `a+b = c` into the document.

where now I am using '`:`' as the delimiter character.

Note that *all* spaces in your input code are printed literally in verbatim mode. For example, the code

```
The string \verb+cat  dog+ has two spaces.
```

produces the output

> The string `cat  dog` has two spaces.

If you need to *emphasise* these spaces, place an asterisk (*) after the command name `\verb`; this will tell LaTeX to print the symbol ␣ every time it sees a space in your input code. For example, the code

```
The string \verb*+cat  dog+ has two spaces.
```

produces the output

> The string `cat␣␣dog` has two spaces.

Whole paragraphs of verbatim text

To produce whole paragraphs of verbatim text, you use the `verbatim` environment. For example, the code

```
\begin{verbatim}
   Hello   there !
Some   @#$%^&*()_-+=[]{}\|;:'",<.>/?`~  junk!!
\end{verbatim}
```

produces the output

```
   Hello   there !
Some   @#$%^&*()_-+=[]{}\|;:'",<.>/?`~  junk!!
```

Note that the `verbatim` environment does not actually indent the text; this is done in the above because I have used the `quote` environment for all displayed code and examples, in these seminar notes. In other words, the code that actually appears in these seminar notes, for the above output, is

```
\begin{quote}
\begin{verbatim}
   Hello   there !
Some   @#$%^&*()_-+=[]{}\|;:'",<.>/?`~  junk!!
\end{verbatim}
\end{quote}
```

You can put any characters you like in a `verbatim` environment, and all spaces and Enter characters are interpreted literally. The only thing you are *not* allowed to put in a `verbatim` environment is the fourteen-character string `\end{verbatim}`, which acts as the delimiter for the end of the environment.

1.6.20 Internet mail system inoculation

Many scientific documents are transmitted digitally over the Internet. Unfortunately, the electronic mail system in the Internet reserves the word `From`, at the start of any line, for denoting the start of the information header of a piece of e-mail. It therefore automatically changes any such occurrence of `From`, in the body of the mail, to `>From`. LaTeX then translates the `>` character to the symbol "¿". The result is that, if a LaTeX paper contains, for example, the code

```
From the preceding discussion, we can see that...
```

then it will be received, via e-mail, as

```
>From the preceding discussion, we can see that...
```

and, after being LaTeXed, will produce the output

> ¿From the preceding discussion, we can see that...

To help you avoid this problem, CostelLaTeX provides the simple command `\From`, which, when LaTeXed, simply produces the word "From". Thus, the above code need simply be changed to

```
\From\ the preceding discussion, we can see that...
```

In general, one need simply change `From␣` wherever it appears in a document to `\From\␣`.

You should also be careful not to place the tilde (~) character at the start of a source line, or following punctuation at the start of a source line. Such a problem would rarely arise for normal documents, but did cause some headaches when I was distributing the first version of CostelLaTeX!

1.6.21 Typing in your text

You will notice that I haven't yet told you exactly *how* you should type your words, sentences and paragraphs into your text editor. Hopefully, you have already picked up most of the rules, from the examples I have given. Here they are explicitly:

1. When you need to type a *space*, you can type any of the following:

- a single space;
- multiple spaces—any number of them;
- you can hit ENTER to start a new line;
- you can put spaces before and/or after hitting ENTER.

All of these actions will produce a *single logical space*, in LaTeX's mind. However, you cannot leave a space where one doesn't belong (such as before a comma), nor can you leave a completely blank line if you only intend to insert a space.

2. To start a new paragraph, you leave *one or more completely blank lines*. Paragraphs are usually typeset with indented first lines; note that you *do not* place any special characters in your file to make this indenting happen: LaTeX handles indenting for you, based on the document style you have chosen.

That's all the rules there are. They are summarised well by the example in the LaTeX manual:

```
The ends  of words and sentences are marked
  by   spaces. It  doesn't matter how many
spaces    you type; one is as good as 100.

One   or more   blank lines denote the  end
of  a paragraph.
```

which produces the output

> The ends of words and sentences are marked by spaces. It doesn't matter how many spaces you type; one is as good as 100.
>
> One or more blank lines denote the end of a paragraph.

LaTeX ignores the "niceness" of your input file; it only cares about the logical concepts of end-of-word, end-of-sentence, and end-of-paragraph.

There are, however, several points of "style" that you should heed:

- *Do not* keep typing off the right-hand margin of your text editor, *even if* it appears to "wrap" your words nicely onto the next line. You *must* hit the ENTER key, at the end of a word, before the line gets too long; otherwise, you may not be able to transfer your files to other people. LaTeX may even fail if your "line length" (the number of characters you type before hitting ENTER) exceeds 255. In other words, type your LaTeX source code like you would an *e-mail message* on a standard network terminal.

- *Always* keep line lengths below 80 characters wide. Some people's text editors cannot handle wider lines.
- Try and keep your line lengths below 65 characters. Some e-mail systems (most notably QuickMail on the Macintosh) play strange formatting tricks on lines that exceed this length.
- There is no minimum line length. You can have as little as one word on a line.
- *Do not* use the TAB key on your keyboard at all. It is usually harmless, but can wreak havoc with some people's text editors.
- It is good practice to start new sentences on new lines. Although you don't have to, it makes cutting and pasting easier.
- If you insert text into a line, you might have to break it (by pressing ENTER) to keep the line lengths under your chosen maximum length. *Do not* go and try and make the input lines all nice and the same length again. It is a waste of time. LaTeX source code often contains lines of only one or two words of text; leave it how it lands. LaTeX is only looking at the *logical* structure of your source file.

1.6.22 So many rules for plain text?

You might think that all these rules I have explained—just to produce a *plain text* document—must make LaTeX a complicated system. But almost all of these rules are common to *all* professional typesetting systems: they are rules of typography and style that all published material conforms to. En dashes and em dashes, left quotes and right quotes—all these things exist in Microsoft Word just as they do in LaTeX; the only difference is that the average Word user never gets taught the correct way to use them.

Take a closer look at the next published book you read. You'll see all the rules at work, regardless of how it was typeset. LaTeX just makes it easier to apply the rules, logically and consistently.

Seminar 2: Mathematical material

2.1 General philosophy

Inserting mathematical material into a CostelLaTeX document is quite simple. You will learn how to do so most quickly if you first understand the *general philosophy* behind LaTeX's treatment of maths.

2.1.1 Text mode

In Seminar 1, I described how to write plain text documents with LaTeX. You can think of LaTeX as being in a special "brain state"—which I will call *text mode*—when it was reading your code for these documents.

LaTeX begins your document in text mode. You must ensure it is returned to text mode before you start a new chapter or section, and before the end of the document.

2.1.2 Maths mode

To insert mathematical material, you must tell LaTeX to enter a different "brain state": *maths mode*. While in this mode, LaTeX interprets the characters and commands you type differently than when it is in text mode.

2.1.3 Four categories of characters and commands

The characters and commands that you type into LaTeX can, in full generality, be divided into four categories:

1. Those that you can only use in text mode.
2. Those that you can only use in maths mode.
3. Those that you can use in both text mode *and* maths mode, but which perform *different tasks* in the different modes.
4. Those that you can in both text mode and maths mode, which perform *exactly the same task* in both modes.

There are actually only a small number of characters and commands that fall into our last category above, that work identically in both text *and* maths mode:

- the "commenting" command: `%`
- the "insert a bit of space" command: `\,`
- the command for producing an ellipsis: `\ldots`
- a rarely-used *underlining* command: `\underline{}`
- the ten decimal digits:

  `1 2 3 4 5 6 7 8 9 0`

- the punctuation commands

  `. : ; , ? ! ( ) [ ] / @ \# \$ \% \& \_ \{ \}`

All of the other characters and commands that you will use in your documents behave differently (whether radically or subtly) in text and maths mode; thus, you should take careful note of the precise purpose and usage of each piece of notation.

2.2 How characters act in maths mode

LaTeX interprets the very *characters* you type in a different way, depending on whether you are in maths mode or text mode. I will now list these differences.

2.2.1 Spaces

You will recall from Seminar 1 that LaTeX has only two concepts of "space" in text mode:

1. A *logical space* is obtained by typing one or more spaces, and/or pressing ENTER.
2. A *new paragraph* is begun by leaving a blank line.

In maths mode, the rules are slightly different:

1. All "logical spaces" are ignored.
2. You may not leave a blank line.

Thus, you can leave an arbitrary number of spaces in your source code in maths mode, and they will be ignored. For example, the code

```
$abc=def$
```

and

```
$   a  b c   = d e   f $
```

both produce the output

$$abc = def.$$

(We shall see shortly that using `$` is one way to begin and end maths mode.) This ignoring of spaces is *different* from text mode, where you may *only* leave a logical space where one belongs; for example,

```
H e  wal ked out   , and bought a news paper  .  % WRONG!
```

produces the incorrect output

H e wal ked out , and bought a news paper .

Most importantly, you may not leave a space before a punctuation symbol, such as a full stop or comma, in text mode, but doing so in maths mode is harmless.

2.2.2 Alphabetic characters

LaTeX automatically interprets all of the 52 uppercase and lowercase alphabetic characters as *symbols* in maths mode, rather than the letters of an English word. The most obvious effect of this is the fact that it *italicises* mathematical symbols. For example, the code

```
Let $x$ and $y$ be reals.
```

produces

> Let x and y be reals.

However, Latin symbols are *not* simply equivalent to italicised letters, and it is poor form to write words in maths mode. Most importantly, the rules for the spacing of letters are different in the two modes. The standard example is the following:

```
The nine symbols $different$ are
\e{different} from normal text.
```

which produces the telling output

> The nine symbols $different$ are *different* from normal text.

If you need to write a word—such as "if" or "GeV"—in an equation, then you need to reenter text mode; this will be discussed in Sec. 2.3.6.

2.2.3 Spacing of symbols

You may already have noticed that LaTeX tends to place symbols quite close to each other, except operation symbols; for example, in

$$abc = def + ghi.$$

LaTeX is generally very good at determining how much space should surround mathematical symbols. You should *not* try and override its choices, in the vast majority of situations. *Do not* try and "space out" equations; the way LaTeX does it is the correct way. If they look too cramped, then wait a few weeks before complaining. You will soon get used to it.

2.3 Entering and leaving maths mode

2.3.1 Inline and displayed maths

When you tell LaTeX to enter maths mode, you must specify whether you want *inline maths* or *displayed maths*.

Inline maths, also known as *in-text maths*, is any mathematical material that is incorporated into the plain text of your document. For example, if I begin to talk about x and y, or the temperature $-40°\mathrm{C}$, then I am using inline maths: these symbols are typeset in this body of this paragraph, as if they were themselves words.

Displayed maths is any mathematical material that you wish to have "displayed"; *i.e.*, it is placed on a line of its own, usually centred, with an amount of vertical space before and after it; for example, the equation

$$i\hbar\frac{\partial}{\partial t}\psi = H\psi.$$

Note that inline maths may not be assigned an equation number; displayed maths may be assigned one if so desired. Furthermore, because inline maths should fit within the vertical space of a *single line* of text, some commands produce slightly different results for inline maths than with displayed maths, as we will shortly see.

2.3.2 Inline maths mode

The simplest way to enter and leave inline maths mode is to type the symbol `$`. For example, the code

```
Let $x$ be a real number.
```

produces the output

Let x be a real number.

2.3.3 Displayed maths mode

In CostelLaTeX, you enter displayed maths mode by issuing the `\beq` command, and end it with the `\eeq` command. For example, the code

```
I told him that
\beq
a=b+c,
\eeq
but he didn't believe me.
```

produces the output

> I told him that
>
> $$a = b + c,$$
>
> but he didn't believe me.

Note that you should *always* punctuate sentences appropriately, even if the punctuation appears at the end of displayed maths, as with the comma above. *Do not* try and leave any space between the end of the displayed maths and the punctuation symbol; there should be none (just as there is no space between the last word of a sentence and the full stop). Whenever constructing a sentence containing mathematical material, you should remember that, grammatically speaking, quantities such as a and b are just nouns (like "dog" or "apple"), and operations such as $=$ and $+$ are verbs ("equals" and "plus"). Thus, the above example is in all ways equivalent to

> I told him that a equals b plus c, but he didn't believe me.

2.3.4 Changing displayed maths to inline

If, at some stage, you decide that you would rather have some piece of displayed maths inline, then you simply need preface it by the `\inline` command. For example, the code

```
I told him that
\inline
\beq
a=b+c,
\eeq
but he didn't believe me.
```

produces the output

> I told him that $a = b + c$, but he didn't believe me.

Of course, if you knew you were going to have the maths inline from the start, then it would have been quicker and simpler to use `$` characters.

2.3.5 Changing inline maths to displayed

To change inline maths that uses the simple `$...$` notation to displayed maths, you need to change the two `$` commands to `\beq` and `\eeq` respectively.

2.3.6 Reentering text mode from within maths mode

If you are in an equation, and you need to write a word (such as "if" or "and"), or some units (such as "cm" or "GeV"), then it is poor form to simply type the letters of the word in as if they were individual mathematical symbols. For example, the code

```
\beq
d=16.5 cm   % WRONG!
\eeq
```

would produce the output

$$d = 16.5cm$$

which implies that d is the product of the number 16.5, the quantity c, and the quantity m.

Instead, you need to temporarily reenter *text mode*. To do so, you simply put the text inside the command

```
\text{}
```

For example, the above code should be written

```
\beq
d=16.5\text{ cm}
\eeq
```

producing the correct output

$$d = 16.5 \text{ cm}$$

Note that we needed to put a *space* in before `cm`, in order that a space be put in the output. It is *not* sufficient to simply leave a space *before* the `\text{}` command, because at that point we are still in maths mode, which ignores all spaces:

```
\beq
d=16.5 \text{cm}    % WRONG!
\eeq
```

If the text that you are adding should in fact be a *superscript or subscript* on some quantity, then use the command

```
\supertext{}
```

or

```
\subtext{}
```

respectively. For example, the code

```
\beq
k\subtext{Dirac}
=p\subtext{Schr\"odinger}\supertext{spin half}
\eeq
```

produces the output

$$k_{\text{Dirac}} = p_{\text{Schrödinger}}^{\text{spin half}}$$

2.3.7 Unbreakable, unstretchable inline maths

When you use inline maths, LaTeX sometimes decides that it needs to break the current line of the paragraph somewhere inside the maths. At other times, it needs to "stretch" the spaces between the terms of your inline maths, in the same way that it stretches the spaces between words, so that each line of text is fully justified (*i.e.*, flush with each margin).

LaTeX generally does these things in quite an intelligent way (for example, only breaking or stretching around symbols such as $=$ or $+$). On rare occasions, however, the effect is ugly.

To enter *unbreakable* (and unstretchable) inline maths mode, place the maths inside a

```
\ubm{}
```

command. For example, the code

```
The formula \ubm{a=b+c} will never be
broken or stretched by \LaTeX.
```

produces

> The formula $a = b + c$ will never be broken or stretched by LaTeX.

You should only need to use `\ubm{}` rarely, if at all. Note that LaTeX may be unable to find a good place to break your lines of text if the maths inside a `\ubm{}` command is too long.

2.4 Breaking a long equation across lines

At some stage you will find, to your horror, that your displayed maths will not fit horizontally on the page, no matter how nice a notational scheme you have set up. The first example of this phenomenon that you will find in my Ph.D. thesis is

$$\begin{aligned}\dot{z}^{\alpha}\dot{z}_{\alpha} = c_1^2 + 4(c_1\cdot c_2)\tau + \Big\{6(c_1\cdot c_3) + 4c_2^2\Big\}\tau^2 + \Big\{8(c_1\cdot c_4) + 12(c_2\cdot c_3)\Big\}\tau^3 \\ + \Big\{10(c_1\cdot c_5) + 16(c_2\cdot c_4) + 9c_3^2\Big\}\tau^4 \\ + \Big\{12(c_1\cdot c_6) + 20(c_2\cdot c_5) + 24(c_3\cdot c_4)\Big\}\tau^5 + \mathrm{O}(\tau^6).\end{aligned}$$

When this happens, it is necessary for *you* to decide how to break your equation over several lines. LaTeX does not make these decisions for you, and neither does CostelLaTeX. The reason is that a good or bad breaking of a long equation can mean the difference between clarity and obfuscation; programs like LaTeX simply cannot be given a sufficient amount of intelligence to make such important judgments.

However, once you have decided how to break your equation into multiple lines, implementing your decision is easy.

2.4.1 The left, centre and right columns

As a simple example, let us return to the code

```
\beq
a=b+c
\eeq
```

which produces the output

$$a = b + c.$$

Let us assume that we want to add another three terms, $d + e + f$, starting a second line on the right-hand side of the equation (ignoring the fact that the first line has plenty of room for these terms!). To do so, we must first tell CostelLaTeX that we are considering the equation to be configured as three "columns": the left-hand side of the equation, the centre column (that contains the = sign), and the right-hand side of the equation. To do so, we insert the command

```
\tabc
```

("tab to the centre column") between `a` and `=`; and the command

```
\tabr
```

("tab to the right column") between `=` and `b`. In other words, our code now looks like

```
\beq
a\tabc=\tabr b+c
\eeq
```

If we LaTeX this, we find that nothing has changed in the output:

$$a = b + c.$$

Now, since the centre column of an equation often contains just an equals sign, CostelLaTeX provides a shorthand command

```
\tabs=
```

("tab using the symbol `=`"), which is equivalent in all ways to

```
\tabc=\tabr
```

Thus, we really only needed to add `\tabs` before the equals sign:

```
\beq
a\tabs=b+c
\eeq
```

This works for any simple symbol, such as `<` and `>`, not just for equals signs. However, always remember that you can revert to using `\tabc` and `\tabr` explicitly, on the rare occasion that you need to put something more complicated in the centre column of the equation.

2.4.2 Moving to the next line

We now need to tell CostelLaTeX to "move to the next line". We do this with one of the commands

```
\nextl
\nextc
\nextr
```

which automatically positions us in the left, centre or right column of the next line respectively. In our case, we want to move to the right column of the next line immediately, since that is where we want to add the three new terms (leaving the left and centre columns of the second line blank). Our code now reads

```
\beq
a\tabs=b+c
\nextr+d+e+f
\eeq
```

and produces the desired output

$$\begin{aligned} a &= b + c \\ &\quad + d + e + f. \end{aligned}$$

2.4.3 Adding another equality

Let us now assume that we want to add another equality to our equation—say, we have proved that a is also equal to $g + h + i$. We simply add the command `\nextc` at the end of the equation, put in another equals sign, tab to the right column with `\tabr`, and add the terms `g+h+i`:

```
\beq
a\tabs=b+c
\nextr+d+e+f
\nextc=\tabr g+h+i
\eeq
```

Again, as another simplification, CostelLaTeX provides the shorthand command

```
\nexts=
```

which is equivalent in all respects to

```
\nextc=\tabr.
```

Thus, our code is most simply written

```
\beq
a\tabs=b+c
\nextr+d+e+f
\nexts=g+h+i
\eeq
```

and produces the output

$$\begin{aligned} a &= b + c \\ &\qquad + d + e + f \\ &= g + h + i. \end{aligned}$$

2.4.4 Adding another equation

Finally, let us assume that we further wanted to list an equation for the quantity j, in the same "set of equations" as that above for a. We then need simply add `\nextl` at the end of the above code, and add the equation for j:

```
\beq
a\tabs=b+c
\nextr+d+e+f
\nexts=g+h+i,
\nextl 2\p j\tabs=k+l+m+n
\eeq
```

which produces the output

$$\begin{aligned} a &= b + c \\ &\qquad + d + e + f \\ &= g + h + i, \\ 2\pi j &= k + l + m + n. \end{aligned}$$

Note that the left-hand side of an equation is always right-justified, and the right-hand side left-justified, so that the equals signs are always aligned, with the quantities on either side snuggled up to them.

2.4.5 Preventing equation breaks across pages

Sometimes, LaTeX breaks the lines of an equation across two pages. Often, this is most useful, in order to maintain “filled” pages (especially if equations with many lines are present). However, it can sometimes be inconvenient or ugly.

CostelLaTeX provides commands that enable you to have full control over LaTeX's ability to break multiline equations. The `\equnbreakable` command disallows LaTeX from breaking any subsequent equation across multiple pages; this remains in force until cancelled by the `\eqbreakable` command. (The default is to allow breaking.)

The `\equnbreakable` and `\eqbreakable` commands may also be used *inside* an equation, to control precisely where LaTeX *may* break a long equation. For example, the code

```
\beq
\equnbreakable
a\tabs=-b+c+d
\nextr-e-f-g,
\eqbreakable
\nextl
\equnbreakable
h\tabs=-i-j+k
\nextr+l+m-n
\nextr-o+p+q,
\eeq
```

allows the equation

$$\begin{aligned} a &= -b + c + d \\ &\quad - e - f - g, \\ h &= -i - j + k \\ &\quad + l + m - n \\ &\quad - o + p + q, \end{aligned}$$

to be broken after the second line (the end of the equation for a), but not anywhere else. It should be noted that, when appearing within an equation, the `\equnbreakable` and `\eqbreakable` commands are "local" to that equation; *i.e.*, once the equation ends, the "breakability status" returns to what it was before the equation began. Thus, equations appearing after the code

```
\equnbreakable
\beq
\eqbreakable
\eeq
```

will *not* be breakable, because the `\eqbreakable` command only applies within the (in this case empty) equation.

2.5 Numbered equations

LaTeX and CostelLaTeX provide commands that make the numbering and cross-referencing of equations simple and intuitive.

2.5.1 Numbering a displayed equation

If you want a displayed equation to be numbered, then you simply need to preface it by the command

```
\eqn{label},
```

where `label` is the "name" that you want to give to the equation. For example, the code

```
I told him that
\eqn{toldhim}
\beq
a=b+c,
\eeq
but he didn't believe me.
```

produces the output

I told him that

$$a = b + c, \tag{2.1}$$

but he didn't believe me.

Note that LaTeX automatically keeps track of the "equation number counter" for you: you do *not* have to insert the equation numbers yourself. For example, if we followed the above code with the further code

```
The equation
\eqn{another}
\beq
d\tabs=e+f
\nextr+g+h
\eeq
is also numbered.
```

we would obtain the output

The equation

$$\begin{aligned} d &= e + f \\ &\quad + g + h \end{aligned} \tag{2.2}$$

is also numbered.

In general, when an equation number appears in an example in these seminar notes, I will let LaTeX generate the number automatically. The precise equation number that will appear in your document will of course depend on how many equations appear before the one in question!

Note also that the precise *format* for equation numbering is determined by the document style you are using. For example, for the style that I have used for these seminar notes, the equation number is preceded by the seminar number: Eq. (2.1) is the first equation in the second seminar; Eq. (2.2) is the second equation in the second seminar; the first equation in next seminar's notes will be Eq. (3.1), *etc.* However, had I used a *different* document style, the equations may have simply been numbered sequentially—(1), (2), (3), *etc.*—throughout the whole series of seminars. Or, if I so wished, I could use a document style that used both the seminar number *and* the section number in the equation numbering scheme: the above equations would then be (2.5.1) and (2.5.2), since this section is number 2.5.

You must always remember that the precise numbering format has been chosen as the most suitable, by the designer of the document style you are using. In general, I recommend that you do not try to change the equation numbering style. If you definitely need to do so, see me and I will tell you how to do it.

2.5.2 Referring to a numbered equation

The reason that we give numbered equations labels is so that we can *refer* to them in other parts of our document. To refer to an equation called `label`, we simply use the command

```
\eq{label}.
```

For example, the code

```
I again told him that
\eqn{toldagain}
\beq
a=b+c;
\eeq
he still claims that \eq{toldagain} is incorrect.
```

produces the output

I again told him that

$$a = b + c; \tag{2.3}$$

he still claims that (2.3) is incorrect.

If you are typing in this example yourself, note that it is now necessary to run LaTeX on your file *twice*. This is because, the first time through, LaTeX compiles the list of labels and corresponding numbers, as it processes your document. It is only on the second pass that LaTeX links up all the cross-references to these labels.

2.5.3 Placing a brace next to the equation number

If so desired, one may "brace" the lines of an equation such as (2.2), by placing the command `\eqbracenumbers` anywhere before the equation. For example, the code

```
\eqbracenumbers
The equation
\eqn{braced}
```

```
\beq
d\tabs=e+f
\nextr+g+h
\eeq
is now braced.
```

produces the output

The equation

$$\left.\begin{aligned} d &= e + f \\ &\quad + g + h \end{aligned}\right\} \tag{2.4}$$

is now braced.

The `\eqbracenumbers` command applies to all equations following it, and remains in effect until cancelled by an `\eqnobracenumbers` command. It is most commonly used when a number of equations are collected together. For example, the code

```
\eqbracenumbers
\eqn{Maxwell}
\beq
\tabc
  \del\vdot\bE=\rh,
\nextc
  \del\cross\bB-\pard_t\bE=\bJ,
\nextc
  \del\vdot\bB=0,
\nextc
  \del\cross\bE+\pard_t\bB=\boldzero,
\eeq
```

produces the output

$$\left.\begin{gathered} \nabla\cdot\boldsymbol{E} = \rho, \\ \nabla\times\boldsymbol{B} - \partial_t\boldsymbol{E} = \boldsymbol{J}, \\ \nabla\cdot\boldsymbol{B} = 0, \\ \nabla\times\boldsymbol{E} + \partial_t\boldsymbol{B} = \mathbf{0}, \end{gathered}\right\} \tag{2.5}$$

where the use of the centre column for each equation means that each will be centred horizontally.

An individual equation may have its brace suppressed, while otherwise still leaving the `\eqbracenumbers` command in effect, by preceding the equation by the `\eqnnobrace` command. This is usually done for single-line equations in the midst of many braced multiline equations. (For technical reasons, this cannot be automatically performed by COSTELLATEX.)

2.5.4 Punctuating a braced equation number

Often, braced equations have the punctuation symbols of the individual lines (the commas above) grouped together in the one single punctuation symbol, *after* the brace. This is accomplished in COSTELLATEX by placing an `\eqpunct{}` command between the `\eqn{}` and `\beq` commands:

```
\eqbracenumbers
\eqn{MaxwellOnePunct}
\eqpunct.
\beq
\tabc
  \del\vdot\bE=\rh
\nextc
  \del\cross\bB-\pard_t\bE=\bJ
\nextc
  \del\vdot\bB=0
\nextc
  \del\cross\bE+\pard_t\bB=\boldzero
\eeq
```

which produces the output

$$\left.\begin{array}{c} \nabla\cdot\boldsymbol{E}=\rho \\ \nabla\times\boldsymbol{B}-\partial_t\boldsymbol{E}=\boldsymbol{J} \\ \nabla\cdot\boldsymbol{B}=0 \\ \nabla\times\boldsymbol{E}+\partial_t\boldsymbol{B}=\boldsymbol{0} \end{array}\right\}. \tag{2.6}$$

2.5.5 Numbering individual lines of an equation

When you number multiline displayed maths, you must in general decide how many numbers you want appended, and where you want them appended:

1. You may want one number to cover the entire multiline maths, *e.g.*, if it is describing a single equation.

2. You may want a number on *each* line of the maths, *e.g.*, if you are writing down a set of equations.

3. You might want numbers on only *selected* lines of the displayed maths.

The commands described in the previous sections have, so far, assumed choice 1. To change to either choice 2 or 3, you need simply remove the `\eqn{}` command from *before* the equation, and instead place new `\eqn{}` commands at the *end of each line* that you want to be numbered. For example, if we only want to number the scalar equations of Eq. (2.5), we simply modify the code to

```
\beq
\tabc
  \del\vdot\bE=\rh,
  \eqn{Gauss}
\nextc
  \del\cross\bB-\pard_t\bE=\bJ,
\nextc
  \del\vdot\bB=0,
  \eqn{NoMonopoles}
\nextc
  \del\cross\bE+\pard_t\bB=\boldzero,
\eeq
```

to produce

$$\nabla\cdot\boldsymbol{E} = \rho, \tag{2.7}$$
$$\nabla\times\boldsymbol{B} - \partial_t\boldsymbol{E} = \boldsymbol{J},$$
$$\nabla\cdot\boldsymbol{B} = 0, \tag{2.8}$$
$$\nabla\times\boldsymbol{E} + \partial_t\boldsymbol{B} = \mathbf{0},$$

as desired.

2.5.6 Having equations numbered by default

It may be noted that COSTELLaTeX assumes that every equation number is given a corresponding label. This encourages good LaTeX practice. After all, if an equation is going to be referred to later (which is why one usually numbers it), then it will need a label. Alternatively, if the equation will not necessarily be referred to again, but *is* of sufficient importance that someone *may* wish to have it numbered (for example, if another author may be likely to refer to "Eq. (2.7) of Costella's notes"), then, again, the equation is clearly of sufficient importance that a suitable label for it should be able to be thought up.

On the other hand, some authors prefer to have *all* equations numbered, whether or not they will ever be referred to again; and some journals encourage this practice. To have this happen in COSTELLaTeX, one need simply use the command `\eqnbydefault`. From that point, all equations will, by default, be numbered. If you wish to then label a particular equation, preface the equation by the regular `\eqn{}` command; to suppress the equation number, preface it by the command `\eqnnone`. COSTELLaTeX assumes that multiline equations should, by default, be given one single number covering the entire equation. If you instead wish a particular equation to have individually numbered lines, preface it by the command `\eqnmulti`; you are then responsible for inserting `\eqn{}` commands after the lines you wish numbered.

To have subsequent equations being again numbered only on demand, rather than by default, simply issue the command `\eqnondemand`.

2.5.7 Appending letters to equation numbers

Often, it is convenient to have a set of equations numbered with appended letters, such as

$$\nabla\cdot\boldsymbol{E} = \rho, \tag{2.9a}$$

$$\nabla\times\boldsymbol{B} - \partial_t\boldsymbol{E} = \boldsymbol{J}, \tag{2.9b}$$

$$\nabla\cdot\boldsymbol{B} = 0, \tag{2.9c}$$

$$\nabla\times\boldsymbol{E} + \partial_t\boldsymbol{B} = \boldsymbol{0}, \tag{2.9d}$$

rather than with distinct numbers. You begin a "lettered" set of equations by issuing the command

```
\eqnbeginletters{label},
```

where `label` is an equation label that can be used to refer to the equations as a set (*e.g.*, to refer to "Eqs. (2.9)" above). You end the lettered set of equations by issuing the command

```
\eqnendletters.
```

Between these two commands, you simply issue the usual `\eqn{}` commands. For example, the equations (2.9a), (2.9b), (2.9c) and (2.9d) were generated by the code

```
\eqnbeginletters{coll}
\beq
\tabc
  \del\vdot\bE=\rh,
  \eqn{letter1}
\nextc
  \del\cross\bB-\pard_t\bE=\bJ,
  \eqn{letter2}
\nextc
  \del\vdot\bB=0,
  \eqn{letter3}
\nextc
  \del\cross\bE+\pard_t\bB=\boldzero.
  \eqn{letter4}
\eeq
\eqnendletters
```

Although rarely needed, COSTELLaTeX also allows you to insert symbols before and after the equation number letter, to produce lettered equation numbers such as (1-a) or (1[a]). To do so, issue the command

```
\eqnsetletterbrackets{leftsym}{rightsym},
```

and all lettered equation numbers following the command will put `leftsym` between the number and the letter, and `rightsym` after the letter. For example, the code

```
\eqnsetletterbrackets{[}{]}
\eqnbeginletters{brac}
\beq
a\tabs=b+c,
\eqn{firstbrac}
\nextl
d\tabs=e+f.
\eqn{secondbrac}
\eeq
\eqnendletters
They were \eq{firstbrac} and \eq{secondbrac}
of \eq{brac}.
```

produces the output

$$a = b + c, \tag{2.10[a]}$$
$$d = e + f. \tag{2.10[b]}$$

They were (2.10[a]) and (2.10[b]) of (2.10).

2.5.8 Copying an equation number

It is sometimes useful to number an equation with the same number as an existing equation number. For example, if one begins a paper with some stated result, labelled Eq. (1), and then proceeds to prove that result, then the last line of the proof should be the original Eq. (1), and should be numbered so.

CostelLaTeX provides a simple command, `\eqncopy{}`, which performs this task. For example, the code

```
We wish to prove
\eqn{orig}
\beq
a=b+c.
\eeq
We prove it by noting that
\eqn{step1}
```

```
\beq
b=b
\eeq
and
\eqn{step2}
\beq
c=c,
\eeq
and hence
\eqncopy{orig}
\beq
a=b+c,
\eeq
as required.
A corollary of \eq{orig} is that
\eqn{corollary}
\beq
a=b+c+d-d.
\eeq
```

produces the output

We wish to prove

$$a = b + c. \tag{2.11}$$

We prove it by noting that

$$b = b \tag{2.12}$$

and

$$c = c, \tag{2.13}$$

and hence

$$a = b + c, \tag{2.11}$$

as required. A corollary of (2.11) is that

$$a = b + c + d - d. \tag{2.14}$$

Command	Result	Command	Result
`\eqnprime{label}`	$(1')$	`\eqnstar{label}`	(1^{*})
`\eqndprime{label}`	$(1'')$	`\eqndstar{label}`	(1^{**})
`\eqntprime{label}`	$(1''')$	`\eqntstar{label}`	(1^{***})

Table 2.1: Commands for numbering an equation with an existing number, appended by primes or stars. For the purpose of illustration, it is assumed that the original equation is Eq. (1).

Note that the equation with the copied number does not have any new label attached, since it is simply a copy of the original number.

Unfortunately, for technical reasons, the `\eqncopy{}` command may not be used inside a multiline equation; it may only appear outside an equation.

2.5.9 Primed and starred equation numbers

As with copied equation numbers, it is also sometimes convenient to be able to label an equation with an existing number, but appended by a symbol or symbols—such as primes or stars—to denote that it is different from, or has been changed from, the original equation. CostelLaTeX provides the commands listed in Table 2.1, which add one, two or three primes or stars to the end of the equation number.

There is also provided a more general `\eqnappend{}{}` command, in which the appended symbols may be specified. For example, the code

```
This is the original equation:
\eqn{orig}
\beq
a=b+c.
\eeq
This is the primed equation:
\eqnprime{orig}
\beq
a\neq b+d.
\eeq
This equation has a number appended with $\al$:
```

```
\eqnappend{orig}{$\al$}
\beq
d=e+f.
\eeq
This is the next equation:
\eqn{next}
\beq
x=y+z.
\eeq
```

produces the output

This is the original equation:

$$a = b + c. \tag{2.15}$$

This is the primed equation:

$$a \neq b + d. \tag{2.15$'$}$$

This equation has a number appended with α:

$$d = e + f. \tag{2.15α}$$

This is the next equation:

$$x = y + z. \tag{2.16}$$

To refer to an equation whose number was generated by one of the above commands, simply remove the character `n` from the command name. For example, the code

```
They were equations \eqprime{orig}
and \eqappend{orig}{$\al$}.
```

produces

They were equations (2.15$'$) and (2.15α).

α	`\al`	θ	`\tht`	o	`o`	τ	`\ta`
β	`\be`	ϑ	`\vth`	π	`\p`	υ	`\ups`
γ	`\ga`	ι	`\io`	ϖ	`\vp`	ϕ	`\ph`
δ	`\de`	κ	`\ka`	ρ	`\rh`	φ	`\vph`
ϵ	`\veps`	λ	`\la`	ϱ	`\vrh`	χ	`\ch`
ε	`\eps`	μ	`\m`	σ	`\si`	ψ	`\ps`
ζ	`\ze`	ν	`\n`	ς	`\vs`	ω	`\om`
η	`\et`	ξ	`\x`				

Table 2.2: Lowercase Greek symbols.

$\varGamma$	`\Ga`	$\varLambda$	`\La`	$\varSigma$	`\Si`	$\varPsi$	`\Ps`
$\varDelta$	`\De`	$\varXi$	`\X`	$\varUpsilon$	`\Ups`	$\varOmega$	`\Om`
$\varTheta$	`\Th`	$\varPi$	`\Py`	$\varPhi$	`\Ph`		

Table 2.3: Uppercase Greek symbols.

2.6 Mathematical symbols

LaTeX and CostelLaTeX can produce practically any mathematical symbol you require. In this section, I list the commands that you use to generate these symbols. Note that they may only be used in maths mode.

2.6.1 Greek symbols

To insert a Greek symbol in maths mode, use one of the commands listed in Tables 2.2 and 2.3. For example,

```
Let $\al$ and $\be$ be complex constants,
and $\Om$ the dimension of the space.
```

produces the output

> Let α and β be complex constants, and $\varOmega$ the dimension of the space.

Note that if an uppercase Greek letter is typographically the same as an uppercase Latin letter (*e.g.*, the uppercase Greek chi and Latin ex), then there is no extra command provided; simply use the Latin symbol.

$\mathcal{A}$	`\sA`	$\mathcal{F}$	`\sF`	$\mathcal{K}$	`\sK`	$\mathcal{O}$	`\sO`	$\mathcal{S}$	`\sS`	$\mathcal{W}$	`\sW`
$\mathcal{B}$	`\sB`	$\mathcal{G}$	`\sG`	$\mathcal{L}$	`\sL`	$\mathcal{P}$	`\sP`	$\mathcal{T}$	`\sT`	$\mathcal{X}$	`\sX`
$\mathcal{C}$	`\sC`	$\mathcal{H}$	`\sH`	$\mathcal{M}$	`\sM`	$\mathcal{Q}$	`\sQ`	$\mathcal{U}$	`\sU`	$\mathcal{Y}$	`\sY`
$\mathcal{D}$	`\sD`	$\mathcal{I}$	`\sI`	$\mathcal{N}$	`\sN`	$\mathcal{R}$	`\sR`	$\mathcal{V}$	`\sV`	$\mathcal{Z}$	`\sZ`
$\mathcal{E}$	`\sE`	$\mathcal{J}$	`\sJ`								

Table 2.4: Script symbols.

2.6.2 Script symbols

LaTeX provides "script" versions of the twenty-six Latin uppercase symbols A through Z. To use any of these script symbols, you must first issue the command

```
\UseScriptSymbols
```

You can then use any of the commands listed in Table 2.4. For example, the code

```
\UseScriptSymbols
Consider the operators $\sA$ and $\sB$.
```

produces the output

> Consider the operators $\mathcal{A}$ and $\mathcal{B}$.

CostelLaTeX requires that you use the `\UseScriptSymbols` command (and other similar commands listed in the following sections) so that special-purpose symbols are only loaded if they are required. This is necessary, to conserve resources on the "small memory" version of LaTeX (*e.g.*, the standard emTeX on the PC), since loading every symbol would exhaust the available memory. Note, however, that the "big memory" version of LaTeX (*e.g.*, most Unix versions, and emTeX's `blatex` version) can easily handle arbitrarily large documents, with all symbols loaded.

Note also that you only need to issue the `\UseScriptSymbols` command *once* in your document. If you like, you can put it straight after the `\include{costella}` command at the start of the document.

$\boldsymbol{a}$	`\ba`	$\boldsymbol{f}$	`\bff`	$\boldsymbol{k}$	`\bk`	$\boldsymbol{o}$	`\bo`	$\boldsymbol{s}$	`\bs`	$\boldsymbol{w}$	`\bw`
$\boldsymbol{b}$	`\bb`	$\boldsymbol{g}$	`\bg`	$\boldsymbol{l}$	`\bl`	$\boldsymbol{p}$	`\bp`	$\boldsymbol{t}$	`\bt`	$\boldsymbol{x}$	`\bx`
$\boldsymbol{c}$	`\bc`	$\boldsymbol{h}$	`\bh`	$\boldsymbol{m}$	`\bm`	$\boldsymbol{q}$	`\bq`	$\boldsymbol{u}$	`\bu`	$\boldsymbol{y}$	`\by`
$\boldsymbol{d}$	`\bd`	$\boldsymbol{i}$	`\bi`	$\boldsymbol{n}$	`\bn`	$\boldsymbol{r}$	`\br`	$\boldsymbol{v}$	`\bv`	$\boldsymbol{z}$	`\bz`
$\boldsymbol{e}$	`\bfe`	$\boldsymbol{j}$	`\bj`								

Table 2.5: Bold Latin lowercase symbols.

$\boldsymbol{A}$	`\bA`	$\boldsymbol{F}$	`\bF`	$\boldsymbol{K}$	`\bK`	$\boldsymbol{O}$	`\bO`	$\boldsymbol{S}$	`\bS`	$\boldsymbol{W}$	`\bW`
$\boldsymbol{B}$	`\bB`	$\boldsymbol{G}$	`\bG`	$\boldsymbol{L}$	`\bL`	$\boldsymbol{P}$	`\bP`	$\boldsymbol{T}$	`\bT`	$\boldsymbol{X}$	`\bX`
$\boldsymbol{C}$	`\bC`	$\boldsymbol{H}$	`\bH`	$\boldsymbol{M}$	`\bM`	$\boldsymbol{Q}$	`\bQ`	$\boldsymbol{U}$	`\bU`	$\boldsymbol{Y}$	`\bY`
$\boldsymbol{D}$	`\bD`	$\boldsymbol{I}$	`\bI`	$\boldsymbol{N}$	`\bN`	$\boldsymbol{R}$	`\bR`	$\boldsymbol{V}$	`\bV`	$\boldsymbol{Z}$	`\bZ`
$\boldsymbol{E}$	`\bE`	$\boldsymbol{J}$	`\bJ`								

Table 2.6: Bold Latin uppercase symbols.

2.6.3 Bold symbols

CostelLaTeX provides simple commands for boldfaced Latin, Greek and script symbols. To generate these symbols, you must first issue the commands

```
\UseBoldLatinSymbols
\UseBoldGreekSymbols
\UseBoldScriptSymbols
```

respectively. You can then use the commands listed in Tables 2.5, 2.6, 2.7, 2.8 and 2.9. For example, the code

$\boldsymbol{\alpha}$	`\bal`	$\boldsymbol{\zeta}$	`\bze`	$\boldsymbol{\lambda}$	`\bla`	$\boldsymbol{\varpi}$	`\bvp`	$\boldsymbol{\upsilon}$	`\bups`
$\boldsymbol{\beta}$	`\bbe`	$\boldsymbol{\eta}$	`\bet`	$\boldsymbol{\mu}$	`\bmu`	$\boldsymbol{\rho}$	`\brh`	$\boldsymbol{\phi}$	`\bph`
γ	`\bga`	$\boldsymbol{\theta}$	`\bth`	$\boldsymbol{\nu}$	`\bnu`	$\boldsymbol{\varrho}$	`\bvr`	$\boldsymbol{\varphi}$	`\bvph`
$\boldsymbol{\delta}$	`\bde`	$\boldsymbol{\vartheta}$	`\bvth`	$\boldsymbol{\xi}$	`\bxi`	$\boldsymbol{\sigma}$	`\bsi`	$\boldsymbol{\chi}$	`\bch`
$\boldsymbol{\epsilon}$	`\bveps`	$\boldsymbol{\iota}$	`\bio`	$\boldsymbol{o}$	`\bo`	$\boldsymbol{\varsigma}$	`\bvs`	$\boldsymbol{\psi}$	`\bps`
ε	`\beps`	$\boldsymbol{\kappa}$	`\bka`	π	`\bpi`	$\boldsymbol{\tau}$	`\bta`	$\boldsymbol{\omega}$	`\bom`

Table 2.7: Bold Greek lowercase symbols.

$\boldsymbol{\Gamma}$	\bGa	$\boldsymbol{\Lambda}$	\bLa	$\boldsymbol{\Pi}$	\bPi	$\boldsymbol{\Upsilon}$	\bUps	$\boldsymbol{\Psi}$	\bPs
$\boldsymbol{\Delta}$	\bDe	$\boldsymbol{\Xi}$	\bXi	$\boldsymbol{\Sigma}$	\bSi	$\boldsymbol{\Phi}$	\bPh	$\boldsymbol{\Omega}$	\bOm
$\boldsymbol{\Theta}$	\bTh								

Table 2.8: Bold Greek uppercase symbols.

$\boldsymbol{\mathcal{A}}$	\bsA	$\boldsymbol{\mathcal{F}}$	\bsF	$\boldsymbol{\mathcal{K}}$	\bsK	$\boldsymbol{\mathcal{O}}$	\bsO	$\boldsymbol{\mathcal{S}}$	\bsS	$\boldsymbol{\mathcal{W}}$	\bsW
$\boldsymbol{\mathcal{B}}$	\bsB	$\boldsymbol{\mathcal{G}}$	\bsG	$\boldsymbol{\mathcal{L}}$	\bsL	$\boldsymbol{\mathcal{P}}$	\bsP	$\boldsymbol{\mathcal{T}}$	\bsT	$\boldsymbol{\mathcal{X}}$	\bsX
$\boldsymbol{\mathcal{C}}$	\bsC	$\boldsymbol{\mathcal{H}}$	\bsH	$\boldsymbol{\mathcal{M}}$	\bsM	$\boldsymbol{\mathcal{Q}}$	\bsQ	$\boldsymbol{\mathcal{U}}$	\bsU	$\boldsymbol{\mathcal{Y}}$	\bsY
$\boldsymbol{\mathcal{D}}$	\bsD	$\boldsymbol{\mathcal{I}}$	\bsI	$\boldsymbol{\mathcal{N}}$	\bsN	$\boldsymbol{\mathcal{R}}$	\bsR	$\boldsymbol{\mathcal{V}}$	\bsV	$\boldsymbol{\mathcal{Z}}$	\bsZ
$\boldsymbol{\mathcal{E}}$	\bsE	$\boldsymbol{\mathcal{J}}$	\bsJ								

Table 2.9: Bold script symbols.

```
\UseBoldLatinSymbols
Let $\bv$ denote the velocity of the electron,
and $\bV$ the velocity of the system as a whole.
\UseBoldGreekSymbols
The angular velocity of the precession of
the particle is $\vom$,
which may be small compared to the
rotation $\vOm$ of the field lines.
\UseBoldScriptSymbols
Let $\bsA$ be an arbitrary vector operator.
```

produces the output

> Let $\boldsymbol{v}$ denote the velocity of the electron, and $\boldsymbol{V}$ the velocity of the system as a whole. The angular velocity of the precession of the particle is $\boldsymbol{\omega}$, which may be small compared to the rotation $\boldsymbol{\Omega}$ of the field lines. Let $\boldsymbol{\mathcal{A}}$ be an arbitrary vector operator.

There is also a command `\boldzero` which produces the symbol $\mathbf{0}$, which is useful when boldface is used to denote three-vectors.

There is a very important caveat about boldfaced symbols that you must be aware of: TeX and LaTeX are, in general, *unable to correctly generate boldfaced symbols in subscripts or superscripts*. This is a limitation of TeX that is unavoidable (basically, standard TEX simply does not come with any fonts for boldfaced symbols that are small enough!).

You must, furthermore, be extremely careful of an apparent solution to this problem, that has tripped many people up: while it *is* possible for one to add extra fonts to LATEX that contain small boldfaced symbols, you must remember that colleagues on different machines, journal editors, and people retrieving papers from bulletin boards will *not* in general have these fonts. Thus, if you most definitely need to place boldfaced symbols in superscripts or subscripts, then you should change your notation from the outset, and not use boldface at all. Note that three-vectors never have this problem: you can always write

$$\exp\Big\{i(\boldsymbol{k}\cdot\boldsymbol{x}-\omega t)\Big\},$$

rather than put the argument in an exponent; indeed, many journals' editorial policies strongly recommend this method of expression, as it is both easier to read, and less troublesome to typeset.

2.6.4 Feynman slashes

At this point, I will reveal most strongly my natural bias towards my own field of research, by listing a number of CostelLATEX commands that are of great use in theoretical physics, but probably not anywhere else: Feynman's covariant "slash" notation of the Dirac algebra,

$$\not{x} \equiv x^\mu \gamma_\mu.$$

CostelLATEX lets you produce the "slash" of any Latin, Greek or script symbol, by first issuing one of the commands

```
\UseSlashedLatinSymbols
\UseSlashedGreekSymbols
\UseSlashedScriptSymbols
```

and then using the commands listed in Tables 2.10, 2.11, 2.12, 2.13, 2.14 and 2.15. For example, the somewhat meaningless code

```
\UseSlashedLatinSymbols
\UseSlashedGreekSymbols
```

$\not{a}$	`\asl`	$\not{h}$	`\hsl`	$\not{o}$	`\osl`	$\not{u}$	`\usl`
$\not{b}$	`\bsl`	$\not{i}$	`\isl`	$\not{p}$	`\psla`	$\not{v}$	`\vsl`
$\not{c}$	`\csl`	$\not{j}$	`\jsl`	$\not{q}$	`\qsl`	$\not{w}$	`\wsl`
$\not{d}$	`\dsl`	$\not{k}$	`\ksl`	$\not{r}$	`\rsl`	$\not{x}$	`\xsl`
$\not{e}$	`\esl`	$\not{l}$	`\lsl`	$\not{s}$	`\ssl`	$\not{y}$	`\ysl`
$\not{f}$	`\fsl`	$\not{m}$	`\msl`	$\not{t}$	`\tsl`	$\not{z}$	`\zsl`
$\not{g}$	`\gsl`	$\not{n}$	`\nsl`				

Table 2.10: Slashed Latin lowercase symbols.

$\not{A}$	`\Asl`	$\not{H}$	`\Hsl`	$\not{O}$	`\Osl`	$\not{U}$	`\Usl`
$\not{B}$	`\Bsl`	$\not{I}$	`\Isl`	$\not{P}$	`\Psl`	$\not{V}$	`\Vsl`
$\not{C}$	`\Csl`	$\not{J}$	`\Jsl`	$\not{Q}$	`\Qsl`	$\not{W}$	`\Wsl`
$\not{D}$	`\Dsl`	$\not{K}$	`\Ksl`	$\not{R}$	`\Rsl`	$\not{X}$	`\Xsl`
$\not{E}$	`\Esl`	$\not{L}$	`\Lsl`	$\not{S}$	`\Ssl`	$\not{Y}$	`\Ysl`
$\not{F}$	`\Fsl`	$\not{M}$	`\Msl`	$\not{T}$	`\Tsl`	$\not{Z}$	`\Zsl`
$\not{G}$	`\Gsl`	$\not{N}$	`\Nsl`				

Table 2.11: Slashed Latin uppercase symbols.

$\not{\alpha}$	`\alsl`	$\not{\theta}$	`\thsl`	$\not{o}$	`\osl`	$\not{\tau}$	`\tasl`
$\not{\beta}$	`\besl`	$\not{\vartheta}$	`\vthsl`	$\not{\pi}$	`\pisl`	$\not{\upsilon}$	`\upssl`
$\not{\gamma}$	`\gasl`	$\not{\iota}$	`\iosl`	$\not{\varpi}$	`\vpsl`	$\not{\phi}$	`\phsl`
$\not{\delta}$	`\desl`	$\not{\kappa}$	`\kasl`	$\not{\rho}$	`\rhsl`	$\not{\varphi}$	`\vphsl`
$\not{\varepsilon}$	`\vepssl`	$\not{\lambda}$	`\lasl`	$\not{\varrho}$	`\vrsl`	$\not{\chi}$	`\chsl`
$\not{\epsilon}$	`\epssl`	$\not{\mu}$	`\musl`	$\not{\sigma}$	`\sisl`	$\not{\psi}$	`\pssl`
$\not{\zeta}$	`\zesl`	$\not{\nu}$	`\nusl`	$\not{\varsigma}$	`\vssl`	$\not{\omega}$	`\omsl`
$\not{\eta}$	`\etsl`	$\not{\xi}$	`\xisl`				

Table 2.12: Slashed Greek lowercase symbols.

$\not{\Gamma}$	`\Gasl`	$\not{\Lambda}$	`\Lasl`	$\not{\Sigma}$	`\Sisl`	$\not{\Psi}$	`\Pssl`
$\not{\Delta}$	`\Desl`	$\not{\Xi}$	`\Xisl`	$\not{\Upsilon}$	`\Upssl`	$\not{\Omega}$	`\Omsl`
$\not{\Theta}$	`\Thsl`	$\not{\Pi}$	`\Pisl`	$\not{\Phi}$	`\Phsl`		

Table 2.13: Slashed Greek uppercase symbols.

$\not\mathcal{A}$	\sAsl	$\not\mathcal{H}$	\sHsl	$\not\mathcal{O}$	\sOsl	$\not\mathcal{U}$	\sUsl
$\not\mathcal{B}$	\sBsl	$\not\mathcal{I}$	\sIsl	$\not\mathcal{P}$	\sPsl	$\not\mathcal{V}$	\sVsl
$\not\mathcal{C}$	\sCsl	$\not\mathcal{J}$	\sJsl	$\not\mathcal{Q}$	\sQsl	$\not\mathcal{W}$	\sWsl
$\not\mathcal{D}$	\sDsl	$\not\mathcal{K}$	\sKsl	$\not\mathcal{R}$	\sRsl	$\not\mathcal{X}$	\sXsl
$\not\mathcal{E}$	\sEsl	$\not\mathcal{L}$	\sLsl	$\not\mathcal{S}$	\sSsl	$\not\mathcal{Y}$	\sYsl
$\not\mathcal{F}$	\sFsl	$\not\mathcal{M}$	\sMsl	$\not\mathcal{T}$	\sTsl	$\not\mathcal{Z}$	\sZsl
$\not\mathcal{G}$	\sGsl	$\not\mathcal{N}$	\sNsl				

Table 2.14: Slashed script symbols.

$\not\ell$	\ellsl	$\not\partial$	\pardsl

Table 2.15: Extra slashed symbols.

```
\UseSlashedScriptSymbols
\beq
f(p,Q,\eps,\La,\sA)
=\psla\Qsl(\pardsl\sAsl)\epssl\Lasl.
\eeq
```

produces the output

$$f(p,Q,\varepsilon,\Lambda)=\not p\not Q(\not\partial\not\mathcal{A})\not\varepsilon\not\Lambda.$$

I suppose that, if you were working in some field of endeavour other than physics, you might be able to find other uses for the slash notation. But perhaps not.

2.6.5 Operation symbols

Various operation symbols available in LaTeX and CostelLaTeX are listed in Table 2.16. The difference between `\times` and `\cross`, and between `\cdot` and `\vdot`, is that the latter command in each pair does not leave as much space around the symbol, and is often more suitable for cross-products and dot-products, especially in lengthy expressions.

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

Table 2.16: Operation symbols.

2.6.6 Relation symbols

The commands for various relation symbols are listed in Table 2.17. To produce a slash through one of these symbols, simply precede it by the command `\not`. For example, the code

```
If $x\not<y$ then $x\not\leq y-1$.
```

produces the output

If $x \not< y$ then $x \not\leq y - 1$.

2.6.7 Arrow symbols

The commands for various arrows are listed in Table 2.18.

2.6.8 Primes

In maths mode, the ' character produces a prime. For example, the code

```
Consider $x$, $x'$, $x''$ and $x'''$.
```

produces the output

Consider x, x', x'' and x'''.

$<$	<	$>$	>	$=$	=	$\propto$	\propto
$\lesssim$	\applt	$\gtrsim$	\appgt	$\neq$	\neq	$\parallel$	\parallel
$\leq$	\leq	$\geq$	\geq	$\equiv$	\id	$\models$	\models
$\prec$	\prec	$\succ$	\succ	$\equiv$	\equiv	$\perp$	\perp
$\preceq$	\preceq	$\succeq$	\succeq	$\sim$	\sim	$\mid$	\mid
$\ll$	\ll	$\gg$	\gg	$\sim$	\like	$\bowtie$	\bowtie
$\subset$	\subset	$\supset$	\supset	$\simeq$	\simeq	$\bowtie$	\Join
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\simeq$	\appeq	$\smile$	\smile
$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\approx$	\approx	$\frown$	\frown
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\cong$	\cong		
$\in$	\in	$\ni$	\ni	$\doteq$	\doteq		
$\vdash$	\vdash	$\dashv$	\dashv	$\asymp$	\asymp		

Table 2.17: Relation symbols.

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftrightarrow$	\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	$\leadsto$	\leadsto	$\rightarrow$	\goesto

Table 2.18: Arrow symbols.

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

Table 2.19: Commands for common mathematical functions.

2.6.9 Mathematical function names

Since function names, such as “cos”, are *words*—and *not* the product of three separate quantities c, o and s, LaTeX provides explicit commands that produce the correct text-mode output for many common function names. These are listed in Table 2.19. For example, the code

```
\beq
\log xy=\log x+\log y.
\eeq
```

produces the output

$$\log xy = \log x + \log y.$$

If you want to use a function name that does not appear in Table 2.19, simply use the `\text{}` command For example, the code

```
\beq
\text{Ei}(x).
\eeq
```

produces the output

$$\text{Ei}(x).$$

Some of the commands in Table 2.19 can have subscripts attached; for example, the code

```
\beq
\lim_{n\goesto\infty}a_n=0
\eeq
```

produces the output

$$\lim_{n\to\infty} a_n = 0.$$

There are two additional LaTeX commands for producing the "mod" function: `\bmod` for a binary relation and `\pmod` for a parenthesised expression. For example, the code

```
\beq
\tabc
  \gcd(m,n)=a\bmod b,
\nextc
  x\equiv y\pmod{a+b},
\eeq
```

produces the output

$$\begin{aligned} \gcd(m,n) &= a \bmod b, \\ x \equiv y &\pmod{a+b}. \end{aligned}$$

2.6.10 Miscellaneous symbols

The commands for various miscellaneous symbols are listed in Table 2.20.

2.7 Mathematical structures

In this section, I describe how to create most of the various mathamatical "structures" that you may need in your documents: subscripts and superscripts, fractions, brackets and braces, arrays, sums and products, and so on.

Integrals will be described in Sec. 2.8.

<table>
<tr><td>∞</td><td>`\infty`</td><td>$\prime$</td><td>`\prime`</td><td>$\forall$</td><td>`\forall`</td><td>$^\circ$</td><td>`\degrees`</td></tr>
<tr><td>$\hbar$</td><td>`\hbar`</td><td>$\emptyset$</td><td>`\emptyset`</td><td>$\exists$</td><td>`\exists`</td><td>$^\circ$C</td><td>`\degreesC`</td></tr>
<tr><td>ƛ</td><td>`\labar`</td><td>$\surd$</td><td>`\surd`</td><td>$\neg$</td><td>`\neg`</td><td>$\Diamond$</td><td>`\Diamond`</td></tr>
<tr><td>∂</td><td>`\pard`</td><td>∇</td><td>`\del`</td><td>$\flat$</td><td>`\flat`</td><td>$\triangle$</td><td>`\triangle`</td></tr>
<tr><td>∂</td><td>`\partial`</td><td>∇</td><td>`\nabla`</td><td>$\natural$</td><td>`\natural`</td><td>$\clubsuit$</td><td>`\clubsuit`</td></tr>
<tr><td>ℓ</td><td>`\ell`</td><td>$\top$</td><td>`\top`</td><td>$\sharp$</td><td>`\sharp`</td><td>$\diamondsuit$</td><td>`\diamondsuit`</td></tr>
<tr><td>$\wp$</td><td>`\wp`</td><td>$\bot$</td><td>`\bot`</td><td>$\backslash$</td><td>`\backslash`</td><td>$\heartsuit$</td><td>`\heartsuit`</td></tr>
<tr><td>$\Re$</td><td>`\Re`</td><td>$\|$</td><td>`\|`</td><td>$\imath$</td><td>`\imath`</td><td>$\spadesuit$</td><td>`\spadesuit`</td></tr>
<tr><td>$\Im$</td><td>`\Im`</td><td>$\angle$</td><td>`\angle`</td><td>$\jmath$</td><td>`\jmath`</td><td>$\mho$</td><td>`\mho`</td></tr>
<tr><td>$\aleph$</td><td>`\aleph`</td><td>$\Box$</td><td>`\Box`</td><td></td><td></td><td></td><td></td></tr>
</table>

Table 2.20: Miscellaneous symbols.

2.7.1 Subscripts and superscripts

To produce a superscript in maths mode, you use the ^ command. This is a command that takes an *argument*; *i.e.*, you need to "feed" the command the symbols that you wish to be subscripted. LaTeX handles arguments according to these two rules:

1. If the argument can be typed with a *single character*, or as a *single command*, then you simply type the argument directly after the command name.

2. If the argument does not fall into the first category, then you place it in braces, directly after the command name.

For example, the code

```
The quantities $e^x$, $x^2$, $2^n$ and $x^\al$
are simpler to type than the quantities
$e^{2x}$, $x^{25}$, $2^{3n}$ and $x^{\al\be}$.
```

produces the output

> The quantities e^x, x^2, 2^n and x^α are simpler to type than the quantities e^{2x}, x^{25}, 2^{3n} and $x^{\alpha\beta}$.

Subscripts are similarly obtained, using the _ command: the code

```
Let $x_1$, $y_\epsilon$ and $z_{10}$ be
completely meaningless constants.
```

produces the output

> Let x_1, y_ϵ and z_{10} be completely meaningless constants.

You can place both a subscript *and* a superscript on a symbol: the code

```
The square of $x_1$ is $x_1^2$.
Writing $x^2_1$ achieves the same result.
```

produces the output

> The square of x_1 is x_1^2. Writing x^2_1 achieves the same result.

Note that you can also have superscripts or subscripts *inside* other superscripts or subscripts. For example, the code

```
The function $e^{-x^2}$ is of great use.
```

produces the output

> The function e^{-x^2} is of great use.

A trickier thing in LaTeX is to place ordered indices such that their subscripted or superscripted position indicates that they are covariant or contravariant; for example, $T_{\alpha\beta}{}^\mu{}_\nu$. To do so, you use code of the form

```
$T_{\al\be}{}^\m{}_\n$
```

The empty `{}` objects serve as new (invisible) symbols for LaTeX to "hang" the following subscript or superscript on.

2.7.2 Fractions

To produce a fraction in inline maths, you should generally use the forward slash: `/`. For example, the code

```
Letting $x/y$ approach the value $1/2$, we find
that the quantity $xI/y\Om$ diverges.
```

produces the output

> Letting x/y approach the value $1/2$, we find that the quantity $xI/y\Omega$ diverges.

Note that standard conventions dictate that a single inline fraction bar *may* be used without parentheses for multiple factors: all factors to the left are in the numerator, and all those to the right are in the denominator. You should not generally use more than one forward-slash in an inline fraction.

In *displayed* material, you should generally use a *built-up fraction*, produced by the command

```
\f{}{}
```

which takes *two* arguments: the numerator and denominator respectively. For example, the code

```
\beq
\f12
\eeq
```

produces the output

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

(where LaTeX simply reads the first argument to `\f{}{}` as `1`, and the second argument as `2`), and the code

```
\beq
f(x)=\f{e^x}{(1-a/x)^{1/\be}}.
\eeq
```

produces the output

$$f(x) = \frac{e^x}{(1-a/x)^{1/\beta}}.$$

(where here we have needed to put the numerator and denominator in braces). Note that it is bad style to use a built-up fraction within another built-up fraction, or within a subscript or superscript; use the forward slash `/` instead.

On rare occasions, you may have need to use a "small" built-up fraction—one in which the numerator and denominator are set in half-sized type. You obtain this using the command

```
\smallfrac{}{}.
```

For example, the code

```
Consider a spin-$\smallfrac12$ particle.
```

produces the output

Consider a spin-$\frac{1}{2}$ particle.

However, in almost all situations it is preferable to rewrite or rephrase your maths, to avoid the use of a "small" built-up fraction, as it often creates reproduction, legibility or editorial problems, when faxed to others, or submitted as a manuscript to a journal.

2.7.3 Stacking a symbol above a relation symbol

The command

```
\stackrel{}{}
```

places a small copy of the first argument above the (relation symbol) second argument. For example, the code

```
\beq
A\stackrel{a'}\goesto B\stackrel{b'}\goesto C
\eeq
```

produces the output

$$A \stackrel{a'}{\to} B \stackrel{b'}{\to} C.$$

Both arguments to `\stackrel{}{}` are processed in maths mode. If you need to place some *text* above a relation symbol, use the command

```
\stacktext{}{}.
```

For example, the code

```
\beq
\vec x\stacktext{def}=(x_1,x_2,\ldots,x_n)
\eeq
```

produces

$$\vec{x} \stackrel{\text{def}}{=} (x_1, x_2, \ldots, x_n).$$

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

Table 2.21: Maths mode accents. The symbol a is used for illustrative purposes only; the accents can be applied to any symbol.

2.7.4 Maths mode accents

The commands illustrated in Table 2.21 are "maths mode accents", similar in form to some of those used on foreign characters (*see* Sec. 1.6.14), but applicable in maths mode.

If you want to apply an "accent" to the symbols i or j, you must first remove their dots. The commands `\imath` and `\jmath` produce a dotless i and j respectively. For example, the code

```
Consider $\vec\imath$ and $\tilde\jmath$.
```

produces the output

Consider $\vec\imath$ and $\tilde\jmath$.

LaTeX supplies the commands `\widehat` and `\widetilde`, that attempt to produce a suitably wide version of `\hat` or `\tilde` respectively. For example, the code

```
Look at $\widehat{a+b}$ and $\widetilde{c+d}$.
```

produces the output

Look at $\widehat{a+b}$ and $\widetilde{c+d}$.

However, I recommend that you only use these commands if absolutely necessary; if you need such notation often, you probably need to change your notation to something more suitable, and typographically manageable.

2.7.5 Overlining an expression

The command `\overline` puts a horizontal line above its argument. For example, the code

```
The average is $\overline{x^2+1}$.
```

produces the output

> The average is $\overline{x^2+1}$.

However, you should *not* use `\overline` on a single symbol; use `\bar` instead (the typographical spacing will in general be incorrect if you use `\overline`).

2.7.6 Horizontal braces

Horizontal braces can be placed above or below an expression with the `\overbrace{}` and `\underbrace{}` commands respectively. For example, the code

```
\beq
\overbrace{a+\underbrace{b+c}+d}
\eeq
```

produces the output

$$\overbrace{a+\underbrace{b+c}+d}.$$

To place a number above an overbrace or under an underbrace, use a superscript or subscript respectively. For example,

```
\beq
\underbrace{a+\overbrace{b+\cdots+y}^{24}+z}_{26}
\eeq
```

produces

$$\underbrace{a+\overbrace{b+\cdots+y}^{24}+z}_{26}.$$

2.7.7 The logical remove-space command

In Seminar 1, I introduced the *logical space* command,

```
\,
```

which inserts a small amount of space into the output of your document. I warned that this command should only be used in a small and well-defined set of situations.

In an even smaller number of situations, it is necessary to *remove* a small amount of space. For example, consider the code

```
$x/\log x$,
```

which produces

$$x/\log x.$$

There is too much space after the fraction bar. To correct it, you use the `\!` command (which removes the same amount of space that `\,` adds): the code

```
$x/\!\log x$
```

produces the more suitable output

$$x/\!\log x.$$

I advise you to use the `\,` and `\!` commands sparingly. If you *do* happen to find some context in which it is necessary to use these commands frequently, it may be better to set up macros that encapsulate them (to be described in Seminar 5), in case you find a better way to perform the same task at a later date.

2.7.8 Equation options or ranges

Sometimes you may need to list a set of "options" or "ranges" in an equation. For example,

$$x = \left\{ \begin{array}{ll} y & \mbox{if } y > 0, \\ z + y & \mbox{otherwise.} \end{array} \right.$$

To produce such equations, you use a combination of the two commands

```
\options{}
\option{}
```

You use `\options{}` to encompass the entire set of options, and use an `\option{}` command for each option or range. You can have as many options or ranges as you like. For example, the above equation was produced using the code

```
\beq
x=\options{
\option{y}{if $y>0$,}
\option{z+y}{otherwise.}
}
\eeq
```

Note that the first argument of `\option{}{}` is processed in maths mode (since it is generally a mathematical expression), whereas the second argument is processed in text mode (since it is generally an English phrase). Hence, in the second argument only, spaces in your code are non-trivial, and mathematical material must be placed inside `$...$` notation.

2.7.9 Specifying errors

To specify an error on a numerical quantity, when the error is taken to be the same in the positive and negative directions, you write, for example,

$$12.6 \pm 0.3,$$

which is simply produced using the `\pm` command:

```
$12.6\pm0.3$.
```

If, on the other hand, the errors are *not* symmetrical, then you need to write, for example,

$$3.6\,{}^{+\,1.3}_{-\,0.7}.$$

You produce this notation by using the command

```
\plusminus{}{}.
```

For example, the above output was produced with the code

```
$3.6\plusminus{1.3}{0.7}$.
```

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

Table 2.22: Commands for sums, products, *etc.*

2.7.10 Sums and products

To produce a sum, product, or in fact any of a number of similar contructions, you use one of the commands listed in Table 2.22, with an optional subscript and/or superscript with the expressions to appear beneath or above the symbol respectively. For example, the code

```
One can show that
\beq
\sum_{i=1}^na_i=1.
\eeq
```

produces the output

One can show that

$$\sum_{i=1}^n a_i = 1.$$

The size of the symbols of Table 2.22, and the positions of their subscripts and/or superscripts, are changed for inline maths, so that the resulting expression fits within the text of the paragraph. For example, the code

```
One can show that
\inline
\beq
\sum_{i=1}^na_i=1.
\eeq
```

produces the output

One can show that $\sum_{i=1}^n a_i = 1$.

2.7.11 Parentheses, brackets and braces

Parentheses, brackets and braces are generally referred to as *binding symbols*, as they are usually used to override the default precedence rules of operation symbols to force the correct "binding" of operands to operators. Modulus signs, and Dirac bras and kets, are also examples of binding symbols.

CostelLaTeX provides two different sets of commands to generate binding symbols:

- *Single-line binding symbols* must be contained on a single line of an equation, but they automatically size themselves to encompass the enclosed object.
- *Multiline binding symbols* may be broken across an arbitrary number of lines, and are automatically of matching size. You specify how large you want them to be.

Single-line binding symbols

The single-line binding symbol commands are listed and illustrated in Table 2.23; they enclose their arguments in delimiters of a size chosen by LaTeX that is large enough to enclose the object. For example, the code

```
\beq
2\braces{\f{\pard z}{\pard t}+1}
=3\parens{1-\f zt}
\eeq
```

produces the output

$$2\left\{\frac{\partial z}{\partial t}+1\right\}=3\left(1-\frac{z}{t}\right).$$

CostelLaTeX also provides the commands

```
\braket{}{}
\bracket{}{}{}
```

These commands are illustrated in Table 2.24; they attempt to create an appropriately-sized Dirac bra–ket or bra–operator–ket product symbol respectively. For example, the code

(x)	`\parens x`	$\left(\frac{1}{2}\right)$	`\parens{\f12}`
$[x]$	`\brackets x`	$\left[\frac{1}{2}\right]$	`\brackets{\f12}`
$\{x\}$	`\braces x`	$\left\{\frac{1}{2}\right\}$	`\braces{\f12}`
$\langle x\rangle$	`\angles x`	$\left\langle\frac{1}{2}\right\rangle$	`\angles{\f12}`
$\lvert x\rvert$	`\bars x`	$\left\lvert\frac{1}{2}\right\rvert$	`\bars{\f12}`
$\lVert x\rVert$	`\dbars x`	$\left\lVert\frac{1}{2}\right\rVert$	`\dbars{\f12}`
$\langle x\rvert$	`\bra x`	$\left\langle\frac{1}{2}\right\rvert$	`\bra{\f12}`
$\lvert x\rangle$	`\ket x`	$\left\lvert\frac{1}{2}\right\rangle$	`\ket{\f12}`

Table 2.23: Single-line binding symbols. Each command is illustrated with both simple and fraction-sized arguments.

$\langle x\|x\rangle$	`\braket xx`
$\left\langle\frac{1}{2}\middle\|\frac{1}{2}\right\rangle$	`\braket{\f12}{\f12}`
$\langle x\|x\|x\rangle$	`\bracket xxx`
$\left\langle\frac{1}{2}\middle\|\frac{1}{2}\middle\|\frac{1}{2}\right\rangle$	`\bracket{\f12}{\f12}{\f12}`

Table 2.24: Single-line Dirac-bracket structures.

```
\UseScriptSymbols
\beq
\tabc
  \angles\sO\id\bracket\ps\sO\ps,
\nextc
  \braket\ps\ps\id1.
\eeq
```

produces the output

$$\begin{gathered}\langle\mathcal{O}\rangle \equiv \langle\psi|\mathcal{O}|\psi\rangle\,. \\ \langle\psi|\psi\rangle \equiv 1.\end{gathered}$$

The `\braket{}{}` and `\bracket{}{}{}` comands only work reliably when all of the arguments are of the same height; if this is not the case, then multiline binding symbols should be used.

Multiline binding symbols

Often, one needs to split equations across multiple lines. In some cases, one cannot avoid likewise splitting parentheses, brackets or braces across these multiple lines. Let us take, as a somewhat contrived example, the equation

$$\begin{aligned} x = a\Big\{b + c\Big[d + e \\ + f + g + h + i\Big]\Big\}.\end{aligned}$$

To produce such an equation, you use code of the form

(	`\leftparen`	)	`\rightparen`
[	`\leftbracket`	]	`\rightbracket`
{	`\leftbrace`	}	`\rightbrace`
$\langle$	`\leftangle`	$\rangle$	`\rightangle`
$\vert$	`\leftbar`	$\vert$	`\rightbar`
$\Vert$	`\leftdbar`	$\Vert$	`\rightdbar`
	`\leftnone`		`\rightnone`

Table 2.25: Multiline binding symbol commands.

```
\beq
\eqindent{40}
x\tabs=a
  \bigbind\leftbrace b+c
  \medbind\leftbracket d+e
\nextr
  +f+g+h+i\rightbracket\rightbrace.
\eeq
```

The command

```
\eqindent{40}
```

tells COSTELLaTeX that you would like all lines begun with a `\nextr` command to be indented 40 units (each unit is about one-tenth of the width of the letter 'x'); this value is set to zero by `\beq` at the start of any equation, so must be set inside the equation.

The binding symbols themselves are produced by using the `\left`x and `\right`x commands, taken from Table 2.25. The symbol produced by each `\left`x command is of the *binding symbol size* currently in effect; COSTELLaTeX ensures that the symbol produced by the matching `\right`x command is automatically of the same size. There are four pre-defined binding sizes that are built into COSTELLaTeX, as well as a `\sizebind{}` command which lets you specify any arbitrary size; these are illustrated in Table 2.26.

Note that the left and right binding symbols need *not* be of the same "type". For example, you can produce the equation

$$\frac{1}{\sqrt{2}}\Bigl(\Bigl\langle\psi^{(+)}\Bigr| + \Bigl\langle\psi^{(-)}\Bigr|\Bigr)$$

Output	Size	Sizing command
$\{\}$	10	`\stdbind`
$\Big\{\Big\}$	20	`\medbind`
$\bigg\{\bigg\}$	25	`\bigbind`
$\Bigg\{\Bigg\}$	30	`\biggerbind`
$\left\{\rule{0pt}{50pt}\right\}$	50	`\sizebind{50}`

Table 2.26: Illustrations of the multiline binding symbol sizing commands.

with the code

```
\beq
\f1{\sqrt2}\bigbind\leftparen
  \medbind\leftangle\ps^{(+)}\rightbar
  +\leftangle\ps^{(-)}\rightbar
\rightparen
\eeq
```

COSTELLaTeX also provides the commands `\midbar` and `\middbar`, which produce a vertical bar or double-bar of the appropriate height, but which do not affect the current "nesting level" of the binding symbol sizes. For example, the equation

$$\left|\left\langle\psi^{(+)}\middle|x_\mu\middle|\psi^{(-)}\right\rangle\right|^2=1$$

is produced by the code

```
\beq
\bigbind\leftbar
  \medbind\leftangle
  \ps^{(+)}\midbar x_\m\midbar\ps^{(-)}
```

```
  \rightangle
\rightbar^2=1.
\eeq
```

The `\lonebar` and `\lonedbar` commands are also provided, the former of which is useful for denoting where a particular quantity is being evaluated: the equation

$$f'(x)\Big|_{\text{fixed point}} = 0$$

is simply produced by the code

```
f'(x)\medbind\lonebar\subtext{fixed point}=0.
```

2.7.12 Arrays

Producing an array is done in two steps:

1. You set up a rectangular array of mathematical quantities.

2. You place a set of parentheses, or perhaps bars or double-bars, around this array of quantities.

The first step is performed with the `array` environment. The best way to see this environment at work is to consider the example given in the LaTeX manual: the code

```
\beq
\begin{array}{clcr}
  a+b+c & uv    & x-y & 27   \\
  a+b   & u + v & z   & 134  \\
  a     & 3u+vw & xyz & 2,978
\end{array}
\eeq
```

produces the output

$$\begin{array}{clcr}
a+b+c & uv & x-y & 27 \\
a+b & u+v & z & 134 \\
a & 3u+vw & xyz & 2,978
\end{array}$$

The `l`, `c` and `r` characters in the extra argument of `\begin{array}` specify the alignment of the corresponding columns of the array. The `&` command specifies that you want to move to the *next column* of the array; the `\\` command specifies that you want to move to the leftmost column of the *next line* of the array.

To place (say) parentheses around the array, simply use the `\parens{}` command; for example, the code

```
\beq
I\id\parens{\begin{array}{cc}1&0\\0&1\end{array}}
\eeq
```

produces the output

$$I \equiv \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right).$$

2.7.13 Extra ellipses for arrays

Instead of the standard ellipsis `\ldots`, you will usually want to use the extra ellipsis commands `\cdots`, `\vdots` and `\ddots` inside arrays. For example, the code

```
\beq
I\id\parens{
\begin{array}{cccc}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{array}
}.
\eeq
```

produces the output

Example source code	Output using rationalised units	Output using irrationalised units
`$\RatFourPi$`	4π	
`$\IrratFourPi$`		4π
`$\RatOneOnFourPi{q^2}$`	$\frac{q^2}{4\pi}$	q^2
`$\IrratOneOnFourPi{q^2}$`	q^2	$\frac{q^2}{4\pi}$
`$\RatFourPiPower3$`	$(4\pi)^3$	
`$\IrratFourPiPower3$`		$(4\pi)^3$

Table 2.27: Examples of commands for rationalised and irrationalised units.

$$I \equiv \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

2.7.14 Rationalised and irrationalised units

Sometimes, you may have the misfortune of having to write documents in a field in which irrationalised units have not been eradicated. If you write some LaTeX code for such an audience, you may need to put in a substantial amount of effort re-editing it for a rational audience.

To avoid this problem, CostelLaTeX provides commands that let you automatically change your system of units, simply by changing a single command. To do so, you first need to use the commands listed in Table 2.27 throughout your document. You then use one of the commands

```
\RatUnits
\IrratUnits
```

Command	Size
`\eqstdheight`	10
`\eqmedheight`	20
`\eqbigheight`	25
`\eqbiggerheight`	30
`\eqsizeheight{`x`}`	x

Table 2.28: Equation line height-setting commands.

at the start of your document (or use the default value for the document style you are using) to specify that the following maths should use the rationalised or irrationalised units respectively.

2.7.15 Setting the heights of equation lines

Often, one produces multliline equations in which some lines are "taller" than other (usually because fractions or binding symbols are used in the taller lines, and not in the shorter ones). LaTeX only leaves enough vertical space as is necessary for each line of the equation. The result is often visually effective; but at other times (especially when printing equations with many lines) it can be a distraction to have a mismatch of the vertical spacing between lines. For example, the equation

$$\begin{aligned} Z = a &+ \frac{1}{2}b + \frac{3}{4}c \\ &+ \frac{5}{6}d + e + 2f \\ &+ g + 3h + i \\ &+ 2j + 3k + 5l \\ &+ \frac{7}{8}m + \frac{1}{9}n \end{aligned}$$

has two "tall" lines, and three "short" lines.

CostelLaTeX provides commands for setting the minimum heights of all subsequent lines in an equation. They are listed in Table 2.28; the heights produced are identical to those for the analogous binding symbol sizing commands of Table 2.26. For example, the code

Command	Size
`\textstdheight`	10
`\textmedheight`	20
`\textbigheight`	25
`\textbiggerheight`	30
`\textsizeheight{`x`}`	x

Table 2.29: Text height-setting commands.

```
\beq
\eqbigheight
Z\tabs=a+\f12b+\f34c
\nextr+\f56d+e+2f
\nextr+g+3h+i
\nextr+2j+3k+5l
\nextr+\f78m+\f19n
\eeq
```

produces the result

$$
\begin{aligned}
Z = a &+ \frac{1}{2}b + \frac{3}{4}c \\
&+ \frac{5}{6}d + e + 2f \\
&+ g + 3h + i \\
&+ 2j + 3k + 5l \\
&+ \frac{7}{8}m + \frac{1}{9}n.
\end{aligned}
$$

Note that all equation lines are set back to standard height at the beginning of a new equation; hence, it is necessary to use the height-changing commands *after* the `\beq` command.

There are also five analogous commands for setting the minimum text height in *any* mode, listed in Table 2.29. You might, for example, use these commands to vertically "space out" the rows of an array (or, equivalently, a table; *see* Seminar 5), by making LaTeX think that the text is taller than it really is.

2.8 Integrals

A good test of any technical document processing system is its ability to handle the typographical nightmare of mathematical *integrals*. LaTeX and CostelLaTeX provide powerful commands that should cover most of the integral constructions you will need to produce, with a minimum of effort.

2.8.1 The integral sign

To produce a single integral sign, you use the command

```
\Int.
```

For example, the code

```
\beq
y\id\Int dx
\eeq
```

produces the output

$$y \equiv \int dx.$$

Note that `\Int` is the *only* integral command to have an uppercase `I`; the remaining commands to be described in this section start with a lowercase `i`. CostelLaTeX needed to define `\Int` differently, because LaTeX's own integral command is called `\int`. I strongly recommend you use the commands described here, rather than bare LaTeX's `\int` command, for your integral constructions.

2.8.2 The integration variables

You will note that, in the above example, we simply typed in the notation for the integration variable, `dx`, explicitly. CostelLaTeX's integral commands include many that automatically include, and correctly format, the integration variable or variables for you. The most trivial of these is the command

```
\dvar{}.
```

For example, the code

```
$\dvar x$
```

produces the output

$$dx.$$

Of course, you would never use this command in practice, because it requires seven keystrokes, compared to only two to type `dx`. However, it serves the purpose of illustrating the general form of these commands.

More important is the case of multi-dimensional integrals, in which the integration variable is denoted by some sort of vector (rather than a different symbol for each dimension). In such cases, you may often wish to write such things as

$$d^3x.$$

Unfortunately, the "obvious" way of doing this, via the code

```
$d^3x$
```

produces the somewhat lopsidedly-spaced output

$$d^3x.$$

Instead, you can use the command

```
\dnvar{}{}.
```

For example, the code

```
$\dnvar3x$
```

produces the output

$$d^3x,$$

and the code

```
$\dnvar{2n}{\tilde x}$
```

produces the output

$$d^{2n}\tilde{x}.$$

2.8.3 Integrating over the integration variable

There is a command

```
\intd{}
```

which produces a single integral sign, and automatically includes the integration variable for you. For example, the code

```
$\intd x$
```

produces the output

$$\int\!dx.$$

You will notice that the integration variable has been placed a little closer to the integral sign, than when we explicitly typed it in in Sec. 2.8.1. This "snuggling up" makes integrals just that little bit better looking (and less wasteful of space).

2.8.4 Integrating over a vector integration variable

To specify a *single* integral sign, but with an integration variable of, say, d^3r, rather than simply dx, you use the command

```
\intdn{}{}.
```

For example, the code

```
$\intdn3r$
```

produces the output

$$\int\!d^3r.$$

2.8.5 Double integrals

If you are denoting a double integral, you have six choices of notation:

1. You could simply denote it, say,

$$\int dx,$$

where the quantity x is known to be two-dimensional, and hence the integral sign and the dx are understood to be two-dimensional in nature. The code for this notation is that described in Sec. 2.8.3:

```
\intd x.
```

2. You could denote it

$$\int d^2x,$$

where again the integral sign is understood to represent a double integral. The code for this notation is that described in Sec. 2.8.4:

```
\intdn2x.
```

3. You could denote it

$$\iint d^2x,$$

to make fully explicit the two-dimensional nature of the integral, with the code

```
\dintdtwo x.
```

4. You may decide to denote it

$$\iint dx,$$

where now dx is understood to be two-dimensional, with the code

```
\dintd x.
```

5. You may need to denote it

$$\iint dx\, dy,$$

if you want or need to specify notationally distinct integration variables. The code for this notation is

```
\dintdd xy.
```

6. You may decide to split the previous notation into two distinct parts,

$$\int dx \int dy,$$

if you wanted to emphasise the fact that the x integral should be performed outermost, or if you are able to factorise the integrand. The code for this notation is just

```
\intd x\intd y.
```

The notation you choose for any particular purpose is up to you; each has its strengths and weaknesses.

If you simply want to produce a double-integral sign alone,

$$\iint,$$

then you use the code

```
\dint.
```

An example of this may be if you wanted to put the integration variable in the numerator of the integrand. For example, the code

```
\UseBoldLatinSymbols
\beq
\dint\f{\dnvar2x}{\bars{\bx-\bx_0}^2}
\eeq
```

produces the output

$$\iint \frac{d^2x}{|\boldsymbol{x} - \boldsymbol{x}_0|^2}.$$

2.8.6 Triple integrals

For triple integrals, you have the same six choices of notation as for double integrals. The commands are completely analogous:

```
\intd{}
\intdn3{}
\tintdthree{}
\tintd{}
\tintddd{}{}{}
\intd{}\intd{}\intd{}
\tint
```

2.8.7 Quadruple integrals

Likewise, for quadruple integrals, you have the same choices of notation:

```
\intd{}
\intdn4{}
\qintdfour{}
\qintd{}
\qintdddd{}{}{}{}
\intd{}\intd{}\intd{}\intd{}
\qint
```

2.8.8 Multi-dimensional integrals in general

If you want to denote a multi-dimensional integral of more than four dimensions, or if you need to denote an integral expression for some *arbitrary* number of dimensions, then you can use one of the following four choices of notation:

1. You can denote it, say,

$$\iint\cdots\int dx,$$

with the code

```
\mintd x.
```

2. You can denote it

$$\idotsint d^n x,$$

with the code

```
\mintdn nx.
```

3. You can denote it, say,

$$\idotsint dx_1\, dx_2 \cdots dx_n,$$

with the code

```
\mintmd{x_1}{x_2}{x_n}.
```

Note that this command does not give you any choice of which integration variables are listed: the first, second and last variables must be provided as the three arguments to the command.

4. You can devise your own notation using the symbol

$$\idotsint,$$

obtained with the command

```
\mint.
```

2.8.9 Summary of integral commands

The various one-, two-, three-, four- and multi-dimensional integral commands are summarised, by example, in Table 2.30.

2.8.10 Upper and lower limits

You can add a lower or upper limit to any of the commands listed in Table 2.30 by prefacing it with the command

```
\llim{}
```

Output desired	Code required	Output desired	Code required
dx	`\dvar x`	d^3x	`\dnvar3x`
$\int$	`\Int`	$\iint$	`\dint`
$\int dx$	`\intd x`	$\iint dx$	`\dintd x`
$\int d^3x$	`\intdn3x`	$\iint d^2x$	`\dintdtwo x`
$\int d^nx$	`\intdn nx`	$\iint dx\,dy$	`\dintdd xy`
$\iiint$	`\tint`	$\iiiint$	`\qint`
$\iiint dx$	`\tintd x`	$\iiiint dx$	`\qintd x`
$\iiint d^3x$	`\tintdthree x`	$\iiiint d^4x$	`\qintdfour x`
$\iiint dx\,dy\,dz$	`\tintddd xyz`	$\iiiint dw\,dx\,dy\,dz$	`\qintdddd wxyz`
$\int\!\!\int\cdots\int$	`\mint`	$\int\!\!\int\cdots\int dx$	`\mintd x`
$\int\!\!\int\cdots\int d^nx$	`\mintdn nx`	$\int\!\!\int\cdots\int da\,db\cdots dz$	`\mintmd abz`

Table 2.30: Illustrations of integral commands.

or

```
\ulim{}
```

respectively. For example, the code

```
\beq
2\llim{-\infty}\intd x\,f(x)
\eeq
```

produces the output

$$2\int_{-\infty} dx\, f(x),$$

and the code

```
\beq
2\ulim\infty\intd x\,f(x)
\eeq
```

produces the output

$$2\int^{\infty} dx\, f(x).$$

Note that you should, in general, place a *logical-space* command `\,` between any differentials like dx, and any other quantities.

If you are using both lower *and* upper limits, you can use the command

```
\lims{}{}.
```

For example, the code

```
\beq
\lims0\infty\intd x
\eeq
```

produces the output

$$\int_0^\infty dx.$$

You can also use limits on any of the multiple-integral symbols: the limits are placed on the last integral symbol. For example, the code

```
\beq
\lims{-\infty}\infty\tintdthree x
\eeq
```

produces the output

$$\iiint_{-\infty}^{\infty} d^3x.$$

2.8.11 Domain of integration

To specify a *domain of integration* (*i.e.*, a description of the region over which the integral is to be performed), you preface any of the integral commands described so far by the command

```
\domain{}.
```

Note carefully that the argument of `\domain{}` is processed in *text mode*, as it is, in general, a *description*, which may or may not contain mathematical material. For example, the code

```
\domain{all space}\tintdthree r=\infty
```

produces the output

$$\iiint\limits_{\text{all space}} d^3r = \infty.$$

2.8.12 Closed integrals

To turn any of the integral commands described so far into "closed" integrals (*i.e.*, by placing a loop in the centre of the respective integral sign), simply preface the command by `\closed`. For example, the code

```
\beq
\tabr
  \domain{$\Ga$}\intd x
  +\domain{$\pard\Ga$}\closed\intd x
  +\domain{$\Ga$}\dintd x
  +\domain{$\pard\Ga$}\closed\dintd x
  +\domain{$\Ga$}\tintd x
  +\domain{$\pard\Ga$}\closed\tintd x
\nextr
  +\domain{$\Ga$}\qintd x
  +\domain{$\pard\Ga$}\closed\qintd x
  +\domain{$\Ga$}\mintd x
  +\domain{$\pard\Ga$}\closed\mintd x
\eeq
```

produces the output

$$\begin{array}{r}\displaystyle\int\limits_{\mathit{\Gamma}} dx + \oint\limits_{\partial\mathit{\Gamma}} dx + \iint\limits_{\mathit{\Gamma}} dx + \oiint\limits_{\partial\mathit{\Gamma}} dx + \iiint\limits_{\mathit{\Gamma}} dx + \oiiint\limits_{\partial\mathit{\Gamma}} dx \\ \displaystyle + \iiiint\limits_{\mathit{\Gamma}} dx + \oint\!\!\!\!\iiiint\limits_{\partial\mathit{\Gamma}} dx + \idotsint\limits_{\mathit{\Gamma}} dx + \oint\!\!\!\!\idotsint\limits_{\partial\mathit{\Gamma}} dx.\end{array}$$

2.8.13 Tightened relation symbols

When using the `\domain{}` command, you may find that LaTeX puts too much space on either side of relation symbols, such as $<$ and $>$. For example, the code

```
\beq
\domain{$r<\eps$}\intdn3r=\f43\p\eps^3
\eeq
```

produces the output

$$\int\limits_{r\ <\ \varepsilon} d^3r = \frac{4}{3}\pi\varepsilon^3.$$

To "tighten up" the spacing around the relation symbol, simply preface it by the command

```
\tightrel.
```

For example, the modified code

```
\beq
\domain{$r\tightrel<\eps$}\intdn3r=\f43\p\eps^3
\eeq
```

produces the more pleasing output

$$\int\limits_{r<\varepsilon} d^3r = \frac{4}{3}\pi\varepsilon^3.$$

2.8.14 Barred integrals

To produce a single integral sign with a "bar" through the middle, you use the command

```
\intbar.
```

For example, the code

```
\beq
\text{li}(x)\id\lims0x\intbar\f{dt}{\ln t}
\eeq
```

produces the output

$$\mathrm{li}(x) \equiv \fint_0^x \frac{dt}{\ln t}.$$

Likewise, there is a command

```
\intbard{}
```

that functions identically to `\intbar{}`, but with a bar through the integral sign. For example, the code

```
\beq
\text{Ei}(x)\id\lims{-x}\infty\intbard t\,\f{e^{-t}}t
\eeq
```

produces the output

$$\text{Ei}(x) \equiv \displaystyle{-}\!\!\!\!\!\!\int_{-x}^{\infty} dt\, \frac{e^{-t}}{t}.$$

2.9 Theorems, axioms, and their friends

2.9.1 Decisions you need to make

To use such things as theorems, corollaries, *etc.*, in your document, you first need to do the following:

1. Sit down and figure out exactly which of these environments you want, and list them on a piece of paper.
2. For each of these environments, decide what name you want to appear at the start of each one, before its number; *e.g.*, "Theorem", if you want theorems to appear as "Theorem 1", *etc.*
3. For each of these enironments, invent a simple name, such as `thm` or `axm`. This is the name that you will use in the `\begin{}` and `\end{}` commands. This name cannot be the same as an existing LaTeX or CostelLaTeX environment, or as one you have already used.
4. Decide whether you want the numbering for these environments to include the chapter number, section number, *etc.*, or whether you just want them numbered 1 through n.
5. Decide which environments should share a single numbering scheme (*e.g.*, Axiom 1 is followed by Theorem 2 and Corollary 3), and which should be independent (*e.g.*, the example following Corollary 3 is Example 1).

2.9.2 Defining the environments

Based on your written-out plans, you then *define* your chosen environments, at the start of your document, according to the following rules:

- To define an environment simply numbered 1 through n for the whole document, use a command of the form

  ```
  \newtheorem{thm}{Theorem}
  ```

  The first argument is the name of the new environment; *i.e.*, in the above, the command for beginning a theorem would be `\begin{thm}`. The second argument is the text that will appear at the start of the environment output, followed by its number.

- To instead have an environment numbered by chapter (*e.g.*, Theorems 1.1, 1.2, *etc.*, in Chapter 1; Theorems 2.1, 2.2., *etc.*, in Chapter 2; *etc.*), add `[chapter]` to the end of the above command:

  ```
  \newtheorem{thm}{Theorem}[chapter]
  ```

  To have them numbered by section, change `chapter` to `section`; to have them numbered by subsection, change `chapter` to `subsection`.

- To have an environment numbered in the *same sequence* as one you have already defined, add that previously-defined environment name, in brackets, between the first and second arguments:

  ```
  \newtheorem{conj}[thm]{Conjecture}
  ```

  Because the numbering follows that of the original environment, you do not re-specify any `[chapter]`, *etc.*, appearing in the original environment's definition.

2.9.3 Examples

Let us assume that I have put precisely the two latter example definitions above at the start of the file for these seminar notes:

```
\newtheorem{thm}{Theorem}[chapter]
\newtheorem{conj}[thm]{Conjecture}
```

(The chapter number is, for these notes, simply the seminar number.) We can now construct a theorem simply by writing the code

```
\begin{thm}
All engineers hate theorems, because theorems
imply proofs, and engineers do not have enough
time to worry about proving proofs.
\clabel{engineer}
\end{thm}
```

which produces the output

> **Theorem 2.1** *All engineers hate theorems, because theorems imply proofs, and engineers do not have enough time to worry about proving proofs.*

The command

```
\clabel{}
```

provides the theorem with a *label*, just as we labelled equations in Sec. 2.5.1. You *refer* to objects labelled by `\clabel{}` by using the command

```
\cref{}.
```

For example, we could refer to Theorem 2.1 above with code of the form

```
For example, we could refer to
Theorem~\cref{engineer} above
with code of the form
```

Let us now construct a conjecture—say, Fermat's "last theorem" conjecture. To put Fermat's name after the conjecture number, we add the desired text in square brackets after the `\begin{}` command: the code

```
\begin{conj}[Fermat]
There do not exist integers $n>2$, $x$, $y$ and
$z$ such that $x^n+y^n=z^n$.
\clabel{Fermat}
\end{conj}
```

produces the output

> **Conjecture 2.2 (Fermat)** *There do not exist integers $n > 2$, x, y and z such that $x^n + y^n = z^n$.*

Note that the number of this conjecture (2.2) follows that of the previous theorem (2.1), since that is how we defined the `conj` environment.

2.9.4 Proofs

Unfortunately, there are no standard LaTeX commands for denoting the start and end of a proof. You will probably want to define your own commands for doing so. For now, type the following at the start of your document:

```
\newenvironment{prf}{{\bf Proof:}\ }{\hfill$\Box$}
```

This now lets you write things such as the following:

```
\begin{prf}
This is a very small proof, because there is not much
that I really want to prove, since I am an engineer.
\end{prf}
```

which produces the output

> **Proof:** This is a very small proof, because there is not much that I really want to prove, since I am an engineer. $\Box$

Defining a proof environment in this way (rather than using explicit commands each time) is strongly recommended, not only because it saves you typing, but also because it lets you uniformly change the visual format of your proofs simply by changing the definition at the start of your document.

Seminar 3: Organising the document

In these days of the World Wide Web and the information superhighway, the ability to access terabytes of information is trivial. As any Web "surfer" will tell you, the real challenge is to *find what it is you are looking for*, amongst an exponentially increasing background of junk.

The documents that you write are no different. In large part, it doesn't matter if what you write is highly valuable or useless trash; if your intended audience cannot quickly and easily *access* the information that is of interest to them, then your work will not be read.

The most crucial determining factor, of course, is the actual *location* of the document as a whole. If it is not in the published literature; if it is in an inappropriate journal; if it is not available as a digital preprint; if it is not available on the World Wide Web;—all of these downfallings will reduce your audience, drastically and immediately. But even if you *have* guaranteed the accessibility of the document as a whole, you must still ensure that it possesses an *internal* structure that is logical, efficient, and—above all else—user-friendly.

This is, of course, the reason you are using a TEX-based system in the first place: documents typeset to a professionally quality are just that much less tiresome to read. LATEX goes further than this, in being a complete *document processing system*, that practically *forces* you to create documents with a consistent and logical visual layout. CostelLATEX completes this path, smoothing over a few of the remaining rough edges in LATEX, allowing you to create truly modular documents with the minimum possible fuss.

Let us now proceed to do so.

3.1 Dividing your document into parts

Some simple documents fit on a single page: a paper-mail letter; a seminar notice for pinning up on a noticeboard; a transparency for a lecture; and so on. Nine times out of ten, it is more efficient for you to produce such documents using something *other* than LATEX. Microsoft Word, for example, is ideal.

So let us assume that we are only considering "serious" documents here: reports, books, theses, journal articles, review articles, lecture and seminar notes, *etc.* Clearly, any such document cannot be simply presented as n continuous pages of information, without a break, or any other clue as to the contents; a reader would then need to read the entire n pages, to find any one piece of information—not unlike the sequential accessibility of a magnetic tape-drive.

Rather, a reader should be able to have *random access* to the wisdom contained in your document. One way to do so might be to follow the same procedure that BHP does when it looks for ore: someone could "extract samples" of your document—every few pages, say—and make a brief list of the topic being discussed in each sample. A potential reader could then simply read through this summary of "extractions", and look for a description somewhat resembling the topic that they are interested in. Assuming that the samples are representative—and that your writing is fairly well correlated over the inter-sample distance,—then the reader would in most cases be able to "jump" into roughly the correct position of your document for the information they sought.

This suggestion is, of course, being made in jest. However, *you would be surprised how close it is to reality.* How many documents have you had the misfortune to refer to, that seem to have been created in the following manner?

1. The author starts writing.

2. At some later date, the author stops writing.

3. The author then realises that the document is one long, unbroken slab of material, that no colleague seems able to read, without falling asleep.

4. In a state of panic, the author speed-reads through the document, page by page, saying, "Hmm, I'll put a new paragraph here; I'll start a new section here; and—what the hell—I'll lash out and start a whole new chapter here."

Do not write your documents this way. You'll waste dozens or hundreds of hours at the end of your writing—probably when you can't afford to waste the time—trying to figure out how to break the thing up into chapters and sections. More importantly, the document will be badly organised.

3.1.1 Subject-oriented writing

In recent decades, computer programmers have realised that the traditional, task-oriented way of writing programs is not optimal. When the program gets large, its interconnections can often become unmanageable. Changes to the code then often have *non-local* effects: parts of code far removed from that changed, which relied in some way on assumptions about the structure of the original code, now find that those assumptions are incorrect.

For this reason, practically all serious programming languages used today are *object-oriented*. The philosophy is as follows: Break your complex task up into a set of operations, on a set of fundamental objects. For each object, define completely its properties, and the ways that it can interact with other objects. Try and eliminate unnecessary interactions between the object and other objects; seek the minimally-complicated external interface possible. Then write code to bring this object to life, keeping all but its external interface hidden from the rest of the virtual world.

The great advantage of this philosophy, of course, is that it is *modular*: if you have debugged the code for the individual objects (which shouldn't be *too* unwieldy, since they are each supposed to be in some sense "fundamental"), then you know that any *future* bugs must be due to an incorrect use of an object, or an incorrect interfacing between them.

I am now going to tell you that you should learn from the computer programmers, and try and implement a philosophy of *subject-oriented writing*. If, when you are typing your document out for the first time, you realise that you are talking about something different than what you were half a page up, then *analyse the page immediately*, and try to locate where the "topic transition" occurred. Then put the new topic into, say, a new section. Put it into a brand new disk file even. It doesn't matter if a topic is short; these seminar notes, for example, are contained in over *230 individual disk files*, each considering a particular topic. Some of those files only contain a single sentence, because I only had one sentence to say on that topic.

If you follow this procedure, then creating a complete document is simple, based on the following rules:

1. Your document should resemble a *fractal*: it should have structure *at all levels*.

2. For printed material, structure your document as a *tree*: the document

itself is the "root branch"; each branch contains sub-branches, until you get down to the level of paragraphs.

3. Links to other, distant parts of the tree—like spider-webs—are performed by *cross-references*.

4. Links to information outside your document are "spider-webbed", by citation, to your *bibliography*, which is like a "switchboard" to the outside world.

5. Rearrange the document tree whenever necessary, to *minimise the total amount of spider-webbing*.

6. Every branch of the tree should be *self-contained*. If you take a given branch (with all its sub-branches) out of the tree, converting attaching links to hypothetical citations, then the extracted branch *should make sense as a standalone document*.

The only practical restriction on you for printed documents—due more to historical convention than anything else—is that you cannot have an arbitrarily deep level of "nesting" of branches; for any particular style of document, you must ultimately "trim the hedge" to a certain uniform depth.

If you follow the above steps in creating you document, then you will find that CostelLaTeX does everything that it can to make your life easier.

3.1.2 Sections

All technical documents are broken up into *sections*. Even letters published in *Physical Review Letters* have an Acknowedgments section and a References section at the end.

A section should focus on one particular topic or aspect of the work you are presenting. Generally, section $n + 1$ should follow on from the topic considered in section n.

To begin a new section, you use the command

```
\newsect{}{}
```

where the first argument contains a *label* for the section (similar to the equation labels that we created using the `\eqn{}` command), and the second argument contains the text that you want for the *heading* of the section. For example, the code

```
\newsect{NewtWig}{The Newton--Wigner method}
Let us now transform the relativistic wave equation...
```

starts a new section, headed "The Newton–Wigner method", and gives it the section label `NewtWig`. The section heading will *automatically* be placed in a suitable bold font, enlarged, with suitable spacing before and after it, and with the section number at the start (just as with these seminar notes). LaTeX does all these things for you.

I recommend that you place each new section in a new disk file. To do so, you simply create a normal CostelLaTeX document. For example, the above code could go in the file

```
\include{costella}
\BeginDocument
\newsect{NewtWig}{The Newton--Wigner method}
Let us now transform the relativistic wave equation...

\EndDocument
```

If you LaTeX this document, you will produce the output for just this section.

3.1.3 Putting the sections together

I have suggested that you place each section in an individual disk file. When you LaTeX each file, you obtain the output of each section separately. How then do you put them all together?

The answer is simple: you create a *master document*, which simply includes all of the files containing the sections. A master document is just another standard CostelLaTeX document. To include the sections, you use the command

```
\Include{}
```

(note the uppercase '`I`'!) where the argument contains the name of the disk file (without the `.tex`). For example, imagine that we had saved the above section in the disk file `newtwig.tex`, and had written two more sections, contained in the files `foldy.tex` and `transfrm.tex`. You could create a complete document, containing these three sections, and a short introduction, by typing

```
\include{costella}
\BeginDocument
\newsect{Intro}{Introduction}
This document considers a few aspects of
the Dirac equation.

\Include{newtwig}
\Include{foldy}
\Include{transfrm}
\EndDocument
```

That's all there is to it!

Note that sections need *not* be contained in separate files, if you don't want them to; here, the first section (the introduction) is simply typed into the master document; the other three sections are `\Include`d. How you divide your document up into files is, in principle, completely arbitrary, and independent of the way you divide it up into sections. However, I suggest that it is simplest and most logical to have the two structures similar.

There is another subtlety to do with disk files that I must warn you about. This is the fact that much of the world uses IBM-PC compatible computers. Currently, MS-DOS has two features that you should be aware of:

- Filenames are restricted to "8+3" format: from one to eight characters, optionally followed by a full stop, optionally followed by up to three characters.

- Case is ignored in all filenames.

This can cause problems when transferring files to or from other file systems, like Unix or Macintoshes. It also has ramifications for the way that you write the filename in the `\Include{}` command. If you want to ward off future hassles, you should therefore follow these rules religiously:

1. Use *only* lowercase letters, numerical digits, and the character `@`, in filenames.

2. Use no more than *eight* characters in filenames.

3. *Always* follow the filename by the extension `.tex`.

4. *Never* use uppercase characters in the argument of an `\Include{}` command.

5. Never include the extension `.tex` in the the argument of an `\Include{}` command.

Note that various file systems will let you avoid various of these rules; it will therefore take discipline on your part to stick to them.

3.1.4 Chapters

If you are writing a long document—a report, a thesis, a set of lecture notes, *etc.*—then you will usually find that there are so many distinct topics to be considered that there are too many sections.

In this situation, you should generally see if you can group related sections together into *chapters*. Unlike sections, chapters do not necessarily have to "follow on" from one to the next; each has its own distinct flavour. Of course, it is best if you *can* arrange your chapters so that they follow on logically, with later chapters building on material presented in earlier chapters; but sometimes a new chapter will consider a completely new topic.

To start a new chapter, you first need to ensure that you are using an appropriate *document style*. CostelLaTeX provides the following four classes of document style:

1. *Article style*. This is the default style. It allows you to have sections, but not chapters. If you are writing a report that is short enough, and focussed enough, to not need chapters, then use this style.

2. *Report style*. This is the style that I recommend to you for reports, theses, lecture notes, *etc.* These seminar notes use report style. To do so, you place the command

   ```
   \Report
   ```

   after `\include{costella}`.

3. *Preprint style.* I recommend this style for *all* papers that you write, except perhaps for huge review articles. Preprint style does not contain chapters. You use the command

   ```
   \Preprint
   ```

   after `\include{costella}` to use the preprint style. If you want to produce a double-spaced version of your manuscript, with tables and figures at the end of the document, and the Abstract and References sections on separate pages, to send to a journal, then change the `\Preprint` command to

   ```
   \Submission.
   ```

4. *Book style.* Use this if you are actually writing a book, by using the command

   ```
   \Book
   ```

   after `\include{costella}`. Book style not only lets you have chapters, but also lets you group chapters into *parts* (to be discussed shortly).

Thus, to use chapters, you must first be using the report or book style. When you have inserted the appropriate command to select one of these styles, then you can start a new chapter with the command

```
\newchap{}{}
```

where, again, the first argument is the chapter label, and the second is the heading. For example, the code

```
\newchap{DiracEqn}{The Dirac Equation}
```

begins a new chapter labelled `DiracEqn`, with the heading "The Dirac Equation".

It is up to you whether you decide to have all non-trivial words in chapter headings capitalised or not. I prefer to, but some people don't.

If you wanted to create a chapter out of the example sections we created earlier, rather than having them being a complete document, then you need simply change the code to

```
\include{costella}
\Report
\BeginDocument
\newchap{DiracEqn}{The Dirac Equation}
This chapter considers a few aspects of
the Dirac equation.

\Include{newtwig}
\Include{foldy}
\Include{transfrm}
\EndDocument
```

where here I have eliminated the “Introduction” section, making the introductory text simply a normal paragraph under the chapter heading.

3.1.5 Putting the chapters together

If you followed the instructions given in Sec. 3.1.3 for including sections in a master document, then the good news is that you follow *exactly the same procedure* to bring the chapters together. For example, the document

```
\include{costella}
\Report
\BeginDocument
\Include{diraceqn}
\Include{schroeqn}
\Include{kgeqn}
\EndDocument
```

brings the three files

```
diraceqn.tex
schroeqn.tex
kgeqn.tex
```

together into a single document. Ordinarily, these latter two files will also be chapters in their own right; but, again, you are in general free to do what you like: they could simply contain more sections for the first chapter, if that is what they contained.

Note that a file `\Include`d in another file can *itself* `\Include` files, as with the file `diraceqn.tex` above. CostelLaTeX keeps track of all of this "file housekeeping" for you. However, for practical reasons, you are currently limited to four levels of "nesting" of files within files. This should be sufficient for most purposes; if you need a greater depth of nesting, please contact me.

3.1.6 Putting files into subdirectories

If you are writing a long document, you may be reticent to create hundreds of files, for the various chapters, sections, and so on, simply from the point of view of logistics. For example, you may wonder how I navigate my way around the 230+ files for these seminar notes!

The issue is actually more serious than that. Imagine that you have decided to start each chapter of your document with an introduction, and end each with a summary. The natural names for files containing these sections would be

```
intro.tex
summary.tex
```

But if you use these names for chapter 1, you'd have to invent *different* names for chapter 2. Calling them

```
intro1.tex
intro2.tex
intro3.tex
...
```

is not a good solution either—if at some stage you decide to switch the order of chapters 1 and 2, then you could easily get confused by the fact that chapter 1 includes `intro2.tex`, and chapter 2 includes `intro1.tex`!

Consider further if you wanted to include a set of lecture notes that you have written with those that someone *else* has written, to form a set of notes for the course as a whole. If even one of your filenames clash, you'd have to edit the documents and rename the files, in order to bring them together.

Instead of this, you can put your files into *subdirectories*. For example, I could create the directories

```
de
sch
kg
```

and put the files for the Dirac equation chapter into `de`, the files for the Schrödinger equation chapter into `sch`, and the files for the Klein–Gordon equation chapter into `kg`. The master file for the document as a whole would stay where it is. To include the component files, you must now simply use the command

```
\Subdirectory{}{}
```

where the first argument contains the name of the subdirectory (no uppercase letters or slashes please!), and the second contains the `\Include{}` statements for the files in that subdirectory. In other words, the overall master file for our example document above would now need to look like

```
\include{costella}
\BeginDocument
\Subdirectory{de}{\Include{diraceqn}}
\Subdirectory{sch}{\Include{schroeqn}}
\Subdirectory{kg}{\Include{kgeqn}}
\EndDocument
```

It's as easy as that! Note that, within each of the `\Include`d files, included files or figures are automatically looked for in their own directory. You can also put `\Subdirectory` commands within these included files; *i.e.*, create sub-subdirectories—say, to include a lot of files in the one section.

Again, there is currently a practical limit placed on the nesting of subdirectories: they can only be four levels deep. If you need nestings deeper than this, please see me.

3.1.7 Commands at the start of the document

You might be wondering about something that seems to make multiple-file documents more trouble than they are worth: I have on several occasions mentioned that you have to place certain commands after the

```
\include{costella}
```

command and before the

```
\BeginDocument
```

command of your document. If you split your document into a large number of smaller component documents, then this implies that you have to add these commands to the start of every one of these files!

Since this would be a very inefficient way to write documents, CosTELLaTeX always looks for a “special” file, called

```
macros.tex
```

whenever it processes your document. If it doesn’t find it, it doesn’t care. However, if it *does* find it, it then *automatically* includes its contents, just before `\BeginDocument` is executed. Note that *you do not put any explicit command into your document* to include `macros.tex`; this is done automatically.

For our example document above, we could save ourselves a lot of trouble by creating a `macros.tex` file containing simply the line

```
\Report
```

This saves us from typing it into the file for every chapter, section, *etc.* More importantly, it lets us *change* our decision—say, to change the document style to `\Book`—by making *one* editing change, rather than hundreds.

Note that you do *not* place any of the usual commands

```
\include{costella}
\BeginDocument
\EndDocument
```

into `macros.tex`; if you do, your document will not be processed correctly.

Note also that, if you use the `\Subdirectory` command, you must place an appropriate copy of `macros.tex` in *all* subdirectories, if you want to be able to LaTeX the individual files in those directories. This is sometimes a hindrance, but can actually provide quite a powerful degree of flexibility: you can specify *different* document conventions for component documents, than for the document as a whole. This can be useful when you are merging directories of files from two or more different authors; each directory contains the document choices that each author has individually made, whereas the overall `macros.tex` file contains choices that they have mutually decided on for the merged document.

To stop CostelLaTeX from reading in the file `macros.tex`, issue the command

```
\NoMacros
```

after `\include{costella}`.

3.1.8 Parts

If you are using the `\Book` document style, then you have the option of grouping chapters together into even higher-level affiliations, termed *parts*.

You begin a new part with the command

```
\newpart{}{}
```

where again the first argument is the label for the part, and the second is its heading. For example,

```
\newpart{Wave}{Wave Equations}.
```

You should note that, unlike commands like `\newchap` and `\newsect`, starting a new part does *not* reset the chapter number to 1; chapters must always be numbered sequentially through the document.

3.1.9 Subsections

Often, if you are working on a document of greater than about ten pages in length, you will find that some or many of your sections contain too much material. I strongly recommend that you then further subdivide your sections, into *subsections*. You do so with the command

```
\newssect{}{}
```

which is completely analogous to `\newsect`.

I myself prefer to put each subsection into its own file; this makes it much easier to rearrange them (you only need reorder the file list in the master document for the section, rather than cut-and-paste within the section file itself). However, this is a matter of choice.

3.1.10 Subsubsections

Sometimes, and *only* sometimes, it can be useful to further subdivide your subsections into *subsubsections*. You do so with the command

```
\newsssect{}{}
```

which again is completely analogous to `\newsect`.

You should think carefully before using subsubsections. In the `\Article` style they are probably appropriate, since that gives you a total of three levels of hierarchy. On the other hand, in the `\Report` or `\Book` styles, numbering them would be unwieldy (imagine subsection 3.5.2.9!), and so they are not numbered in these styles; this consequently means that the labels you attach to them are *useless* in these styles.

What I have done

I myself have used subsubsections a few times in these seminar notes. This, in fact, is one. You must decide for yourself whether you risk confusing the reader by adding another level of nesting in your document.

3.1.11 Subsubsubsections and subsubsubsubsections

You can guess what these are. I cannot imagine what you would use them for, but they are provided by LaTeX. I don't recommend you use them very often, if at all. If you do, the commands required are

```
\newssssect{}{}
\newsssssect{}{}
```

For example, the code

```
\newsssect{SubSub}{A subsubsection}
This is still acceptable.
You can put useful material in a
subsubsection, although it sometimes
isn't numbered.

\newssssect{Arrgh}{A subsubsubsection}
```

```
This is somewhat more dubious.
I don't really know why you would want to
use this construction very often.

\newsssssect{StopIt}{A subsubsubsubsection}
This is definitely going too far.
If you manage to find any good excuse to use
such things in your document, let me know
and I will eat this page.
```

produces the following:

A subsubsection

This is still acceptable. You can put useful material in a subsubsection, although it sometimes isn't numbered.

A subsubsubsection This is somewhat more dubious. I don't really know why you would want to use this construction very often.

A subsubsubsubsection This is definitely going too far. If you manage to find any good excuse to use such things in your document, let me know and I will eat this page.

3.1.12 Title sections and title pages

The generation of a *title section* or *title page* can be troublesome. In general, you have three choices:

1. Use the standard commands. This produces an acceptable title page or section, but doesn't allow you to produce fancy tricks (university logos, *etc.*).

2. Copy one from someone else. Beware when you try to modify it that you understand how it was created!

3. Figure out yourself, using Lamport's book, how to create the visual effects you want.

Command	Description
`\PreprintNumber{}`	The preprint number.
`\Title{}`	The document title.
`\Author{}`	The name of the author.
`\Authors{}`	Same as `\Author{}`.
`\Address{}`	The address of the author.
`\Date{}`	The date to appear on the document.
`\MakeTitle{}`	Makes the title, based on the above data.

Table 3.1: Commands for generating a title page or section.

I will only teach you the commands for choice 1; they are listed in Table 3.1. The current version of COSTELLATEX only supports multiple `\Author` and `\Address` commands in the `\Preprint` and `\Submission` styles; in the other styles, you must have a single author line, and a single address line.

If you are writing a paper, and you want to put *footnotes* into any of the lines of the title section, you should really use *endnotes*, which are footnotes listed at the end of the paper. The commands you will need to use are

```
\TitleNoteText{}
\TitleNote
```

The rules for doing so are somewhat subtle, because the footnote symbols themselves must be put into the *arguments* of the commands of Table 3.1:

- For every endnote you want in a command of Table 3.1, you must first use the command

  ```
  \TitleNoteText{}
  ```

  with the argument containing the endnote text, *before* the titling command.

- You then use the command

  ```
  \TitleNote
  ```

  to produce the footnote symbol itself.

For example, you might have the title block code

```
\BeginDocument
\TitleNoteText{Work done under duress.}
\Title{The results of my hard word\TitleNote}
\TitleNoteText{E-mail address:
  hbbear@channel9.tv.au.}
\Author{Humphrey B.~Bear\TitleNote}
\TitleNoteText{Has since been sacked.}
\Address{Channel 9\TitleNote}
\Date{Yesterday}
\MakeTitle
```

The text of each endnote *precedes* the line on which the corresponding footnote symbol will appear.

3.1.13 Abstracts

To begin and end the Abstract section, you use the commands

```
\BeginAbstract
\EndAbstract
```

For example,

```
\BeginAbstract
In this paper, we show that Einstein's
theory of relativity is wrong.
We also prove that the world is, in fact, flat.
\EndAbstract
```

3.1.14 Table of contents

To add a table of contents to your document, simply insert the command

```
\TableOfContents
```

where you would like it to appear (normally after the title section). LaTeX will automatically collect information for the table of contents from the headings used for your sectioning units (chapters, sections, *etc.*), and will format it up for you.

Note that, if subsequent `\TableOfContents` commands are found in files that are `\Included`, they are ignored by CostelLaTeX.

3.1.15 Acknowledgments section

The command `\Ack` will begin an unnumbered "Acknowledgments" section, in any document style.

3.1.16 Bibiography

I will describe in Seminar 4 how to create a Bibliography section (which is called the References section in shorter document styles).

3.1.17 Appendices

Material that is useful for the reader to refer to, but which does not need to be explicitly and linearly read in order to understand the arguments being presented, can often be usefully put into *appendices*.

If you are using a document style that has chapters, then issuing the command

```
\Appendix
```

will tell COSTELLaTeX that all chapters from that point should be *appendices*. They will be numbered A, B, C, *etc.*, rather than 1, 2, 3, *etc.*, as is appropriate for chapters. The command

```
\newapp{}{}
```

is then in all ways equivalent to

```
\newchap{}{}.
```

If, on the other hand, you are using a document style that does *not* have chapters, then issuing the command

```
\Appendix
```

will tell COSTELLaTeX that all *sections* from that point should be appendices. They will be numbered A, B, C, *etc.*, rather than 1, 2, 3, *etc.*, as is appropriate for sections, in these document styles. The command

```
\newapp{}{}
```

is then in all ways equivalent to

```
\newsect{}{}.
```

3.2 Cross references

On a number of occasions, we have seen that you can *label* objects in your source code (equations, chapters, sections, *etc.*), and then cross-refer to them at some other point in your document.

You may have wondered whether all these labels need be *unique*, throughout the document as a whole. The answer is, in general, *no*: equation labels are never confused with chapter labels, by CostelLaTeX, even if the label itself is the same. Likewise, labels in *different sections, chapters*, *etc.*, can be the same, without causing your code to crash CostelLaTeX. This "local name space" philosophy means that your LaTeX code can be completely *modular*: including your document with another, into a meta-document, will *never* mean that you have to change your code; the labels will not clash.

Of course, if labels are "local" to a given section or chapter, then there must be some way to cross-reference an object in a *different* section or chapter. I will now explain how you do this, by first explaining to you how CostelLaTeX keeps a track of your document structure.

3.2.1 Section paths

The key idea behind CostelLaTeX's section labelling system is that the logical structure behind your document is in the form of a *tree*.

Such a tree sectioning structure may be thought of analogously to the tree *directory* structure on the file system of your computer system. Just as any directory can be uniquely specified by its *path*—from the root directory, down the branches,—so too can any sectional division of a document be specified by its "section path": one specifies the part, chapter, section, *etc.*, that it is contained in, down to the level of the sectioning division that one is describing.

Of course, most LaTeX documents do not make use of *parts*, and many do not contain *chapters*. In these cases, CostelLaTeX simply considers the first branch from the "root" to be chapters or sections respectively.

3.2.2 Naming the section tree

Of course, if one is to navigate one's way through a tree structure, one needs to have a way to uniquely identify the appropriate branch at each level. In a file system, this is done via *directory names*. Likewise, in CostelLaTeX this is done by ensuring that every sectioning division is given a unique *label*. This is why all of the commands listed in the previous section for starting new chapters, sections, *etc.*, forced you to give a label to each new division; otherwise, CostelLaTeX would not be able to keep track of where it was.

Now, just as two subdirectories of a given directory cannot have the same name, so too (for example) two sections of a given chapter must have different labels. Similarly, just as subdirectories of *different* directories may have the same name (or even the same name as the parent directory), so too can (for example) sections in different chapters have the same label. For example, a chapter `Feynman` may contain a section `Feynman`, but it may not have a second section also called `Feynman`; but a chapter `Dirac` *could* of course also contain a section called `Feynman`.

You should always keep the file-system analogy in your mind uppermost, when considering section paths.

3.2.3 Cross-referencing using paths

Although it hasn't affected you, all of the sections, subsections, equations, tables and figures in these seminar notes have *automatically* been labelled by CostelLaTeX according to their full section path in the document! That this functionality has been transparent so far is due to the fact that CostelLaTeX assumes, by default, that the object you are cross-referencing is *in the current chapter, section, etc.* On the rarer occasions that this is not the case, you must simply tell CostelLaTeX those parts of the path that are different from the current path.

Let us proceed by example. If you look in the code for these notes, you will find that this section (Sec. 3.2) is labelled `SectLabl`, and that this subsection (Sec. 3.2.3) is labelled `CrossRef`. Let us now create a numbered equation, (3.1), with the label `Fermat`, with the code

```
\eqn{Fermat}
\beq
a^n+b^n=c^n.
\eeq
```

to produce

$$a^n + b^n = c^n. \tag{3.1}$$

To refer to this equation number in the above sentence, we of course simply used the code

```
Let us now create a numbered equation, \eq{Fermat}, ...
```

Now, if we were to LaTeX the file `sectlabl.tex`, then, on the first pass, the label `Fermat` is not defined until after it is referred to, so LaTeX issues a warning of the form

```
LaTeX Warning: Reference ‘eq:SectLabl-CrossRef-Fermat’
on page x undefined.
```

But we see here that the label that LaTeX has been asked to find is not simply `Fermat`, but rather

```
eq:SectLabl-CrossRef-Fermat
```

The part "`SectLabl-CrossRef-`" of this label is CostelLaTeX's way of (internally) keeping track of section paths; the "`eq:`" out the front is CostelLaTeX's way of ensuring that the labels of equations, tables, figures, and sectioning units won't clash.

You don't need to worry about the exact form of the label above—it is automatically handled internally by CostelLaTeX. When we want to refer to equation (3.1) from within this section, we need simply issue the command `\eq{Fermat}`. Let us, however, start a new subsection, and then refer back to equation (3.1).

3.2.4 Cross-referencing to a different path

This subsection (Section 3.2.4) is labelled `DiffPath`. If we tried to refer to Eq. (3.1) simply by means of the code

```
\eq{Fermat}
```

we would find that LaTeXwould issue a warning of the form

```
LaTeX Warning: Reference ‘eq:SectLabl-DiffPath-Fermat’
on page x undefined.
```

no matter how many times we LaTeXed the file `sectlabl.tex`. This is because CostelLaTeX is still telling LaTeX to look in the *current section* for `Fermat`, whereas it is actually in a different section.

To tell CostelLaTeX where the desired cross-referenced equation is located, we need simply specify only as much of the path as is *different* from the current path. In the current example, the path to the equation `Fermat` is

```
Section: SectLabl
Subsection: CrossRef
```

whereas the path to the current section is

```
Section: SectLabl
Subsection: DiffPath
```

The only difference is the *subsection* that we are in. We therefore issue the command `\diffssect{CrossRef}`, standing for "specify a different subsection called `CrossRef`", before the `\eq{Fermat}` command. In other words, the code

```
... equation \diffssect{CrossRef}\eq{Fermat} ...
```

is what is used in the first paragraph of this section.

As another example, imagine that we wanted to refer to a table that is in a different part, chapter, section, subsection and subsubsection of a document. We would then need code of the form

```
These values are listed in Table~%
\diffpart{partlabel}
\diffchap{chaplabel}
\diffsect{sectlabel}
\diffssect{subsectlabel}
\diffsssect{subsubsectlabel}
\tbl{tablelabel}.
```

It may seem that such a verbose way of referring to a table does not constitute much of an advance. However, in practice, one generally finds that the vast majority of objects that one refers to *are* in fact located in the same section; and the majority of exceptions are contained in closely-related sections. The benefits obtained by the freeing up of short, convenient labels, regardless of document length, outweighs the small inconvenience of having to use an occasional `\diff`x`{}` command.

As another example, imagine that we wished to refer to various subsections of these notes. To refer to the previous subsection (Sec. 3.2.3), we need simply issue the code

```
Section~\ssect{CrossRef}
```

since CostelLaTeX automatically applies the current path, down to the *section* level, before appending the subsection label (`CrossRef`).

3.2.5 Referring to objects in a parent section

Just imagine we wanted to refer to an equation in Sec. 3.2, which is the parent section (`SectLabl`) of the current subsection (`Parent`). We could not just issue a command of the form

```
\eq{equationlabel}
```

because that would look for the label

```
eq:SectLabl-Parent-equationlabel
```

Instead, we need to tell CostelLaTeX to look in the parent section, by issuing instead the commands

```
\parentsect\eq{equationlabel}
```

A list of these "parenting" commands is given in Table 3.2.

3.2.6 Referring to objects in a daughter section

To refer to objects in a daughter sectioning division, simply issue `\diff`x commands as required. For example, if you wish to refer, in the text at the start of some chapter, to an equation `equationlabel` in subsection `subsectlabel` of section `sectlabel` of that chapter, you would issue the code of the form

```
... as will be shown in equation~%
\diffsect{sectlabel}
\diffssect{subsectlabel}
\eq{equationlabel} ...
```

Command	Object is in the parent...
`\parentpart`	part
`\parentchap`	chapter
`\parentsect`	section
`\parentssect`	subsection

Table 3.2: Commands for specifying that an object is in a parent sectioning division.

Command	Refers to the current...
`\thispart`	part
`\thischap`	chapter
`\thissect`	section
`\thisssect`	subsection
`\thissssect`	subsubsection

Table 3.3: Commands for referring to the current sectioning division.

3.2.7 Referring to the current sectioning division

Occassionally, you may have need to refer to the current sectioning division, *i.e.*, the current chapter, section, *etc.*

CostelLaTeX provides the commands listed in Table 3.3 that perform this task. For example, to tell you that this subsection is numbered 3.2.7, I issue the code

```
For example, to tell you that this subsection is
numbered \thisssect, ...
```

3.2.8 Cross-referencing commands

Although already illustrated extensively throughout this section, I list explicitly in Table 3.4 the commands that are used to cross-refer to the number of a sectioning division, equation, table or figure. Note that the `\Part{}` command has a capital first letter; this was necessary because LaTeX reserves the command name `\part{}` for its own purposes.

Command	Object
`\Part{label}`	Part
`\chap{label}`	Chapter
`\sect{label}`	Section
`\ssect{label}`	Subsection
`\sssect{label}`	Subsubsection
`\eq{label}`	Equation
`\tbl{label}`	Table
`\fig{label}`	Figure

Table 3.4: Commands for cross-referring to the number of an object.

Command	Result	Command	Result
`\PART`	`Part~`	`\PARTS`	`Parts~`
`\Chap`	`Ch.~`	`\Chaps`	`Chs.~`
`\App`	`App.~`	`\Apps`	`Apps.~`
`\Sect`	`Sec.~`	`\Sects`	`Secs.~`
`\Ssect`	`Sec.~`	`\Ssects`	`Secs.~`
`\Sssect`	`Sec.~`	`\Sssects`	`Secs.~`
`\Eq`	`Eq.~`	`\Eqs`	`Eqs.~`
`\Tbl`	`Table~`	`\Tbls`	`Tables~`
`\Fig`	`Fig.~`	`\Figs`	`Figs.~`

Table 3.5: Commands for producing the text previous to a cross-reference, together with the equivalent code under default document conventions.

CostelLaTeX also provides commands that produce the *text* before a cross-referencing number, such as the "Sec." in

See Sec. 3.4.2 for details.

that automatically change format if you are using a document style that prefers to refer to these objects in a different way (for example, writing "Section" or "Sect." rather than "Sec."). These commands are listed in Table 3.5. For example, the code

```
Refer to \Tbl\tbl{xref} and \Eq\eq{result}.
```

would, under default conventions, produce output of the form

> Refer to Table 5.2 and Eq. (5.18).

3.2.9 Referring to a page number

To refer to the number of the *page* that an object lies on, rather than the number of the object itself, preface the cross-referencing command for the object by the `\Page` command. For example, the code

```
This subsection begins on page~\Page\thisssect.
```

produces the output

> This subsection begins on page 158.

3.2.10 Draft mode

One slight inconvenience with using LaTeX is that, to make a cross-reference to an object some distance away from the point of reference, one often has to hunt through the text source file to find out the exact label that one has used for the object.

CostelLaTeX simplifies this process, by providing a *draft mode*, in which all objects that are assigned labels have those labels printed out *in the actual output document*. Draft mode may be begun at any point in a document, by issuing the command `\DraftMode`; it may be likewise turned off at any point of the document by issuing the command `\NoDraftMode`. For example, the code

```
\DraftMode
Here is equation~\eq{Fermat}:
\eqn{Fermat}
\beq
a^n+b^n=c^n.
\eeq
\NoDraftMode
```

produces the output

Here is equation (3.2):

$$a^n + b^n = c^n. \tag{3.2}$$

⟦eq:Seminar3-SectLabl-Draft-Fermat⟧

Note that the printed label is positioned so that it does not affect the horizontal alignment of the equation itself. COSTELLaTeX attempts to achieve such a result as far as possible, but naturally there may arise situations in which this will not be the case.

3.2.11 Cross-referencing an object in another file

When you split your document into component files, you can cross-refer to an object in another file in the same way as you would for an object in the current file.

However, when you LaTeX the component file alone, LaTeX will of course complain that it cannot find the object you are cross-referencing. You can suppress this whingeing by prefacing the cross-referencing command by one of the commands

```
\far
\farfar
\farfarfar
```

You use the first command if LaTeX *will* be able to find the cross-reference, when the *parent* document of the current document is LaTeXed. You use `\farfar` if it will not be until *its* parent (*i.e.*, the grandparent of the current document) is LaTeXed that the reference will be found, and `\farfarfar` if it will not be until the *great-grandparent* file is LaTeXed that the reference will be found. For example, if you want to refer to an Appendix C, from the file that starts Chapter 7, then the code

```
This is discussed in Appendix~\far\chap{Discussion}.
```

would produce the output

This is discussed in Appendix C.

Counter	Command
Part	`\setpartnumber{}`
Chapter	`\setchapnumber{}`
Section	`\setsectnumber{}`
Subsection	`\setssectnumber{}`
Subsubsection	`\setsssectnumber{}`

Table 3.6: Commands for setting current counter values.

when the master document is LaTeXed; but, if the component file is LaTeXed alone, then LaTeX will not look for the cross-reference, but will rather simply produce the output

> This is discussed in Appendix ⟦far⟧.

Note that it doesn't ultimately matter if LaTeX "complains" or not about missing references, as long as the final document provides all the relevant links.

3.2.12 Setting sectioning numbers

Sometimes, one would like to specify to LaTeX the number of the chapter, section, *etc.*, that one would like to be the "current" number, when a document is split over several source files, so that when the chapter or section is LaTeXed alone, the numbers used are those that will be applicable when the document as a whole is processed. (This is not necessary, of course, but may sometimes be useful.)

CostelLaTeX provides the commands listed in Table 3.6 that perform this task. If a file containing these commands is LaTeXed alone, the corresponding counters are set. If the file is, on the other hand, processed as a part of a *master document*, then the commands of Table 3.6 have no effect. (This is done so that an accidental old numbering of a section—harmless in itself—does not destroy the integrity of the complete document.)

A slight subtlety arises when you want to specify the number of an object *before* that object is defined. For example, if I want the next chapter of a document to be Chapter 1, then, according to LaTeX logic, the *current* chap-

Counter	Command
Part	`\setnextpartnumber{}`
Chapter	`\setnextchapnumber{}`
Section	`\setnextsectnumber{}`
Subsection	`\setnextssectnumber{}`
Subsubsection	`\setnextsssectnumber{}`
Equation	`\setnexteqnumber{}`
Table	`\setnexttblnumber{}`
Figure	`\setnextfignumber{}`

Table 3.7: Commands for setting counter values for the following object.

ter needs to be set to Chapter 0. CostelLaTeX takes care of this subtlety for you, by providing the `\setnext`x`number` commands listed in Table 3.7.

3.3 Document conventions files

One of the major improvements of LaTeX over plain TeX is its usage of *document style files*. These are short TeX descriptions of standardised, professionally designed document layouts. By simply specifying a different style file, you can change the visual appearance of your document, *without changing any of the logical commands in your source text*.

CostelLaTeX extends this philosophy to *all* aspects of a document, not just its visual layout: the spelling and punctuation conventions employed; the format of the bibliography entries and citations; and so on. In fact, if you discover a convention that varies from journal to journal, and this document aspect is *not* automatically adjusted by CostelLaTeX, then simply let me know and I will endeavour to cover it in the next version of the package.

As with LaTeX's document styles, you, the author, must simply use the set of *standardised commands* supplied by CostelLaTeX, in the body of your document. You then specify, at the start of the document, what document conventions you would like (*e.g.*, Physical Review D, Australian Journal of Physics, Wiley International book, *etc.*). CostelLaTeX takes care of the rest.

Of course, you may decide to *not* make use of all of CostelLaTeX's convention-independent commands in your documents. That is perfectly fine. Those aspects of your document will then be convention-dependent. You must decide for yourself—depending on the range of possibilities of target audience, and your previous LaTeX experience—whether using the convention-independent commands will bring about a net gain or loss of productivity in the long run.

3.3.1 Using the default conventions

If you want to use what I consider to be a good set of default document conventions, then you need do nothing. CostelLaTeX will automatically load these conventions unless you specify otherwise.

3.3.2 Specifying a different document conventions file

To specify that you want to use a different set of document conventions—usually those applicable to a particular journal—then you need to do the following:

1. Make sure you have a copy of the appropriate document conventions file in the TeX inputs directory on your system. It will have a filename ending with `.tex`, as do all CostelLaTeX files.

2. Insert the command

   ```
   \SetConventions{convfile}
   ```

   immediately after `\include{costella}`, where `convfile.tex` is the name of the conventions file.

That's all there is to it!

3.3.3 Asking the user to specify the conventions file

If you are writing a paper, you may wish to have the choice of LaTeXing it according to several different conventions. For example, if you are sending a paper to the American Journal of Physics, you would use the `amjphys` conventions file for your manuscript submission; but you may choose to use the *default* conventions for generating a preprint (with British spelling and a more standardised bibiliography layout).

To let you generate output according to either convention, without needing to repeatedly edit the top lines of the source code, you simply use the command

```
\AskConventions{default or amjphys}
```

instead of `\SetConventions{}`. When you LaTeX this file, you will then see the following:

```
You may enter the document conventions desired.
Choices: default or amjphys

\Cconventions=
```

CostelLaTeX then waits for you to type in your choice.

3.3.4 Where are they?

You may be wondering where all the document conventions files are, in the CostelLaTeX package. The answer is that, currently (version 1.12), there are only four:

```
default
defsuper
austjph
amjphys
```

The last two are for the Australian and American Journals of Physics respectively (they are good test cases for debugging purposes); `defsuper` is a default file for superscript citation conventions. The reason that there are currently only four of these files is that I needed to wait until I had finalised the interface between CostelLaTeX and its conventions files, before making dozens of the latter; otherwise, every change to the interface would have meant dozens of changes to individual files.

I will be gradually building up a collection of conventions files, for most of the commonly-read journals in Physics, Maths and Statistics. Please tell me your "favourite" journals, and I will endeavour to have them included in the package promptly.

3.3.5 Document style options

Although COSTELLaTeX encapsulates document style options in its document conventions files, I have included some commands that let you choose options that are appropriate for *any* document. These commands should appear before `\BeginDocument`.

- To override the default type size of the text of your document, you can use one of the commands `\TenPoint`, `\ElevenPoint` or `\TwelvePoint`.
- To specify that you want the document formatted for *double-sided* printing, use the command

  ```
  \DoubleSided{}
  ```

  where the argument contains the amount of "inside gutter" you want to leave for binding (*e.g.*, `1cm`). You can also use the command

  ```
  \SingleSided{}
  ```

  to specify the left-side gutter for single-sided documents.
- To specify double-spaced or 1.5-spaced text for your document, issue the command `\DoubleSpaced` or `\ThreeHalvesSpaced` respectively. You can also use the override command `\SingleSpaced` for document conventions that default to double-spaced or 1.5-spaced.

If you are writing a thesis or report, I recommend using 12 point, 1.5-spaced text. (Thesis guidelines for the University of Melbourne explicitly allow this.)

3.3.6 SI and cgs units

There is a command

```
\SIcgs{SI expression}{cgs expression}
```

that will output either the SI or the cgs expression, depending on the document conventions in force. For example, the code

```
The depth is \SIcgs{1.5~m}{150~cm}.
```

produces the output

The depth is 1.5 m.

if SI conventions are in force, and

The depth is 150 cm.

if cgs conventions are in force.

3.4 Spelling conventions

CostelLaTeX treats *spelling conventions*—the thorny question of the Americanisation of the written word—in the same way as it treats document conventions. If you use the set of standardised commands provided with CostelLaTeX, you can have the spelling conventions in your document converted from British to American and back (or to your own personal choice of conventions) with a single preamble command.

3.4.1 Spelling commands

The CostelLaTeX spelling commands are listed in Tables 3.8, 3.9 and 3.10.

3.4.2 Using no spelling commands

If you don't want to use CostelLaTeX's spelling commands, you need do nothing: the spelling commands have not been loaded unless you specifically ask for them.

3.4.3 Using default document spelling conventions

If you want to use the *default* spelling conventions appropriate to the document conventions file you have already chosen, then simply issue the command

```
\UseSpellingCommands
```

before the `\BeginDocument` command in your document.

Command	American	British	JPCspell	Example
\aeae	ae	æ	æ	vit\aeae
\aee	e	æ	æ	\aee sthetics
\aes	as	æ	æ	formul\aes
\AEAE	AE	Æ	Æ	VIT\AE
\AEE	E	Æ	Æ	\AEE STETICS
\AES	AS	Æ	Æ	FORMUL\AES
\att	at	att	att	form\att ed
\aums	ums	a	a	moment\aums
\coe	coe	coe	coe	\coe fficient
\Coe	Coe	Coe	Coe	\Coe fficient
\coi	coi	coi	coi	\coi ncident
\Coi	Coi	Coi	Coi	\Coi ncident
\coo	coo	coo	coo	\coo rdinate
\Coo	Coo	Coo	Coo	\Coo rdinate
\el	el	ell	ell	trav\el ing
\ett	et	ett	ett	riv\ett ed
\exes	exes	ices	ices	ind\exes
\ices	ixes	ices	ices	append\ices
\me		me	me	program\me
\metris	meteriz	metris	metris	para\metris e
\noone	no one	no one	no one	\noone
\Noone	No one	No one	No one	\Noone
\oee	e	œ	œ	\oe sophagus
\OEE	E	Œ	Œ	\OEdipus
\our	or	our	our	col\our
\pree	pree	pree	pree	\pree minent
\re	er	re	re	cent\re
\red	ered	red	red	cent\red
\rea	rea	rea	rea	\rea lign
\Rea	Rea	Rea	Rea	\Rea lign
\ree	ree	ree	ree	\ree ngage
\Ree	Ree	Ree	Ree	\Ree ngage
\rei	rei	rei	rei	\rei nstate
\Rei	Rei	Rei	Rei	\Rei nstate
\reo	reo	reo	reo	\reo pen
\Reo	Reo	Reo	Reo	\Reo pen
\reu	reu	reu	reu	\reu se
\Reu	Reu	Reu	Reu	\Reu se
\ring	ering	ring	ring	cent\ring
\s	z	s	s	quanti\s e
\scep	skep	scep	scep	\scep tical
\ses	sises	ses	ses	the\ses
\ue		ue	ue	catalog\ue
\uses	uses	i	i	cact\uses
\uss	us	uss	uss	foc\uss ing

Table 3.8: Spelling commands for parts of words.

Command	American	British	JPCspell	Example
`\ala`	a la	à la	à la	`\ala`
`\Ala`	A la	À la	À la	`\Ala`
`\role`	role	rôle	rôle	`\role`
`\Role`	Role	Rôle	Rôle	`\Role`
`\naive`	naive	naïve	naïve	`\naive`
`\Naive`	Naive	Naïve	Naïve	`\Naive`
`\naivete`	naivety	naïveté	naïveté	`\naivete`
`\Naivete`	Naivety	naïveté	naïveté	`\Naivete`

Table 3.9: Spelling commands for foreign accenting.

Command	English	American
`\bill`	bill	check
`\boot`	boot	trunk
`\bonnet`	bonnet	hood
`\chemist`	chemist	drugstore
`\footpath`	footpath	sidewalk
`\fullstop`	full stop	period
`\maths`	maths	math
`\Maths`	Maths	Math
`\quarter`	quarter	fourth

Table 3.10: Commands for words that are different in American.

3.4.4 Specifying the spelling conventions

Although this is not recommended, you can specify precisely the spelling conventions that you want used for your document, overriding the default choice in the document conventions file, by using the `\SetSpelling{}` command.

For example, to use the American Journal of Physics document conventions, but British spelling conventions, you would start your document with

```
\include{costella}
\SetConventions{amjphys}
\SetSpelling{british}
\BeginDocument
...
```

There is also provided the `\AskSpelling{}` command, which works like the `\AskConventions{}` command.

3.4.5 Latin phrases and abbreviations

Authors often establish commands for Latin iations such as "e.g." and "etc.", both to avoid having to punctuate the abbreviation repeatedly, and also to avoid common mistakes (such as writing "eg." rather than "e.g.,").

It is also conventional to place Latin phrases and abbreviations in the italic typeface; but some journals require them in the plain roman face.

To simplify the usage of such phrases and abbreviations, in a convention-independent way, CostelLaTeX provides the commands listed in Table 3.11. (The ␣ symbol denotes a space in the output.)

CostelLaTeX also provides the commands `\see` and `\See`, which produce the output "*see*" and "*See*" respectively. While these words are plain English, they are often italicised, to more easily catch the eye and direct the reader to the cross-reference.

3.4.6 Harvard and separator commas

In addition to sabotaging spelling, the Americans have also white-anted punctuation conventions.

a priori	`\apriori`	*Etc.*	`\Etc`
A priori	`\Apriori`	*ibid.*	`\ibid`
a posteriori	`\aposteriori`	*i.e.,*␣	`\ie`
A posteriori	`\Aposteriori`	*I.e.,*␣	`\Ie`
cf.	`\cf`	*loc. cit.*	`\loccit`
Cf.	`\Cf`	*op. cit.*	`\opcit`
e.g.,␣	`\eg`	*per se*	`\perse`
E.g.,␣	`\Eg`	*Per se*	`\Perse`
et al.	`\etal`	*viz.*	`\viz`
etc.	`\etc`	*Viz.*	`\Viz`

Table 3.11: Latin phrases and abbreviations.

The Harvard comma

The Physical Review Style Guide insists that "Harvard (serial) commas" must be used. These apply to lists of more than two objects, when written out with commas and a final "and". The standard convention is to write, for example,

> I like the numbers 1, 2, 3 and 4.

The infamous Harvard comma is inserted after the second-last object:

> I like the numbers 1, 2, 3, and 4.

To do this in a document-independent way, CostelLaTeX provides the commands `\hcand` and `\hcor`. For example, the above output would be produced by the code

```
I like the numbers 1, 2, 3\hcand4.
```

Note that there are no spaces around `\hcand`, except for the end-of-command space if you are following the command with a word (the usual case):

```
You can see cats, dogs\hcor pigs.
```

The separator comma

This is the comma used for phrases such as

> The four-momentum, p, is conserved.

The Americans insist that such commas should be omitted (except when the symbol is separated from the noun by an intervening phrase), and so require that you write

> The four-momentum p is conserved.

To write such a sentence in a convention-independent way, use the CosTELLaTeX command `\sepc`:

```
The four-momentum\sepc$p$\sepc is conserved.
```

Note that sometimes you need to use a *normal* comma after the symbol, if the sentence would have had a comma there anyway; for example,

```
The momentum\sepc$p$, the energy\sepc$E$\hcand
the angular momentum\sepc$j$\sepc are all conserved.
```

Seminar 4: Bibliographies

CostelLaTeX provides a powerful set of commands that lets you create bibliographies that automatically *change their format*, depending on which document conventions file you choose. It also provides a set of accompanying commands that let you cite these bibliography entries in a convention-independent way.

4.1 Creating your bibliography

4.1.1 Are we going to learn BibTeX?

LaTeX has a companion program, BibTeX, that can manage bibliography databases for you in quite an elegant and efficient manner.

Unfortuately, BibTeX is not used by many people at all. From what I can gather, this is because BibTeX is simply *too* inflexible: to produce a one-off entry in your bibliography, or to change the format around to suit the style of a particular journal, requires too much work (either by the author, or the organisation producing the suitable BibTeX style files). Rather, most authors simply type their bibliography in as plain LaTeX code—perhaps setting up a few macros to simplify the procedure; LaTeX is only invoked to handle the automatic numbering and cross-referencing functions.

It can also be inconvenient sending BibTeX bibliography files with your documents over the Internet—since on the one hand you would not want to send a database containing every paper you have ever read with every document you write, but, on the other hand, if you have a separate BibTeX file for every document, you have eliminated the very reason for using BibTeX.

4.1.2 CostelLaTeX's approach

CostelLaTeX provides a set of commands that allow you the sort of format-independence enjoyed by BibTeX users, while still retaining the power of being able to type one-off references in directly. CostelLaTeX's document conventions files automatically set up the appropriate bibliography and citation formats for you.

A side-benefit is that you can trivially copy-and-paste references from one paper to another, even if the two papers use completely different bibliography formats. Such portability was previously only available to users of BibTeX.

4.1.3 Beginning the bibliography

To begin the bibliography section of your document, use the command

```
\BeginBib.
```

The `\BeginBib` command will generally appear after the other sections of your document (usually after the `\Ack` section).

4.1.4 Specifying the start of an item

You tell CostelLaTeX that you want to start a new bibliography item (*i.e.*, a new reference) by using the command

```
\BibItem.
```

4.1.5 Specifying the year

The next thing you need to do is specify the *year of publication* of the reference. (CostelLaTeX deems that all references should have a year, even if it is not printed in a few situations.) You do so with code of the form

```
\BibYear{1994}.
```

4.1.6 Specifying the authors

You specify the names of the author of the reference in one of two ways:

1. If the author in question does *not* have any hyphenated given names, you use one of the commands

   ```
   \BibNameOne
   \BibNameTwo
   \BibNameThree
   \BibNameFour
   ```

Author's name	No. of initials	Command used
A. Einstein	1	`\BibNameOne A{Einstein}`
J. P. Costella	2	`\BibNameTwo JP{Costella}`
B. H. J. McKellar	3	`\BibNameThree BHJ{McKellar}`
H. J. I. E. Costella	4	`\BibNameFour HJIE{Costella}`

Table 4.1: Examples of commands for specifying authors with one, two, three and four initials.

where the `One`, `Two`, `Three` or `Four` specifies *how many initials* the author has. These commands are illustrated in Table 4.1.

2. If the author *has* hyphenated given names, you use the `\BibSurname`, `\BibInitial` and `\BibHyphenate` commands. These can also be used for authors falling into category 1, but they require more typing.

For examples of category 1, to specify that Bruce McKellar, Tien Kieu and I are co-authors of a reference, I would use the code

```
\BibNameTwo JP{Costella}
\BibNameTwo TD{Kieu}
\BibNameThree BHJ{McKellar}
```

CostelLaTeX will automatically output these names in the appropriate format for your chosen document conventions file. The default format is

J. P. Costella, T. D. Kieu and B. H. J. McKellar

but with different journal conventions it may appear as

Costella, J. P., Kieu, T. D., and McKellar, B. H. J. (1994)

or

Costella J P (1994), Kieu T D and McKellar B H J

CostelLaTeX handles this formatting for you automatically. (You can now see why the year needed to be specified before the authors.)

If I then wanted to add "D.-D. Wu" to the list of authors, after Tien, I would add commands so that the author block of this reference now looked like the following:

```
\BibNameTwo JP{Costella}
\BibNameTwo TD{Kieu}
\BibSurname{Wu}
\BibInitial D
\BibHyphenate
\BibInitial D
\BibNameThree BHJ{McKellar}
```

4.1.7 Including the ever-popular Al

The command

```
\BibEtAl
```

will specify that there are omitted authors, and will place the string "et al." (italicised if appropriate) after the name of the first author.

CostelLaTeX will automatically do this, in most document styles, if you specify too many authors.

4.1.8 Citation labels

CostelLaTeX will *automatically* generate citation labels for you, based on the surname of the first author and the year of publication. For example, given the above commands, CostelLaTeX would automatically invent the label

```
Costella1994
```

for the reference in question.

If you have more than one reference with the same first author surname and year, then you need to place `\BibAppend` commands before their `\BibYear` commands. I recommend you only use the letters 'a', 'b', 'c', *etc.*, for this purpose. For example, your bibliography might look like

```
\BeginBib
\BibItem
\BibAppend a
\BibYear{1994}
\BibNameTwo JP{Costella}
```

```
...
\BibItem
\BibAppend b
\BibYear{1994}
\BibNameTwo JP{Costella}
...
\EndBib
```

CostelLaTeX then assigns these two references the labels `Costella1994a` and `Costella1994b` respectively. The appended letters also appear in the citations if you use a bibliography style that does not use numbers; *e.g.*, papers are referred to as (Costella, 1994a), *etc.*

4.1.9 Choosing your own label

I mentioned above that CostelLaTeX automatically generates citation labels for you. It will, however, stand aside and let you invent the label, for any or all of your references, if you so desire. All you need to do is place the command `\BibLabel{}` before `\BibYear`. For example, the bibliography entry

```
\BibItem
\BibLabel{bruce}
\BibYear{1992}
\BibNameThree BHJ{McKellar}
...
```

would have the label `bruce`, rather than the default `McKellar1992`.

You *must* use the `\BibLabel` command if the first author of the reference has foreign accents or spaces in their surname; otherwise, the automatically-generated label will crash LaTeX. For example,

```
\BibItem
\BibLabel{Schrodinger1932}
\BibYear{1932}
\BibNameOne E{Schr\"odinger}
...
\BibItem
```

```
\BibLabel{Stephenson1995}
\BibYear{1995}
\BibNameTwo GJ{Stephenson Jr.}
...
```

4.1.10 Specifying the reference's contents

You may now, if you like, specify any contents for the reference whatsoever, by using the command

```
\BibText{}.
```

For technical reasons, this command has a somewhat counterintuitive format. The best way to learn it is by example: the bibliography entry

```
\BibItem
\BibYear{1971}
\BibNameTwo HB{Bear}
\BibText Pp{ersonal communication}
```

would, with standard document conventions, produce the entry

[1] H. B. Bear, personal communication,

but under other conventions (for example, for the Australian Journal of Physics) would produce the entry

Bear, H. B. (1971). Personal communication.

You can see that the first two arguments of the `\BibText` command—`B` and `b` in the above example—are two copies of the first character of the entry: the first gets used if the entry requires an uppercase starting letter; the second for a lowercase starting letter.

Note that the *year* of a reference does not automatically get added using `\BibText`—unless it is part of the document convention's way of identifying references,—despite the fact that you are required to specify a year using the `\BibYear` command for *all* references.

4.1.11 Specifying a journal

CostelLaTeX provides several sets of constructions for reference types that you are likely to use often. (If you find I have missed one or two, let me know and I will add them.) You use the corresponding commands instead of the all-powerful `\BibText` command.

The most common reference is a *journal article*. You start such a reference with the command

```
\BibJournal{},
```

where the agument specifies the name of the journal. (When specifying journal abbreviations, you need not worry about full stops being interpreted wrongly; the normal "extra space after full stop" rule is suppressed in bibliographies.)

To specify the *series* of the journal (*i.e.*, A, B, C, D, *etc.*), you must use the separate command

```
\BibSeries{}
```

after `\BibJournal{}`. For example, you would specify Physical Review D with the code

```
\BibJournal{Phys. Rev.}
\BibSeries D
```

Note that you should *always* place a full stop at the end of any abbreviated word, *even if it is the last word in the journal title.*

You then need to specify at least the volume number and start page for the journal article. You use code of the form

```
\BibVolume{123}
\BibStartPage{4567}
```

after `\BibJournal` to do this. Your journal article would then, under default conventions, appear in the bibliography in the form

[2] H. B. Bear, *Phys. Rev. D* **123** (1971) 4567.

There are two optional commands for journal articles:

```
\BibEndPage{4570}
\BibTitle{On the quality of honeypots}
```

The information in these commands is used for some journals (*e.g.*, the American Journal of Physics), but ignored in others. Placing this information in your bibliography, even if your chosen journal does not require it, lets you cut-and-paste the reference to a different paper, destined for a journal that *does* require the information, at some later date.

4.1.12 Personal communications

To specify that a reference is a personal communication, you can use the simplified command

```
\BibPersComm.
```

This command is in fact completely equivalent to typing

```
\BibText Pp{ersonal communication}
```

explicitly.

4.1.13 Preprints

You specify that the reference is a preprint with the command

```
\BibPreprint.
```

The following commands are all optional for preprints:

```
\BibNumber{UM--P--95/12}
\BibTitle{The Foldy--Wouthuysen transformation}
\BibInstit{University of Melbourne}
```

4.1.14 Books

You specify a book with the command

```
\BibBook{},
```

where the argument contains the title of the book. For example, you might specify

```
\BibBook{The Principles of Quantum Mechanics}.
```

Note that you should *make a firm decision* whether you want to capitalise all non-trivial words in the titles of books, or not, and then stick to this decision. My recommendation is that you do.

The following commands are all optional for books, although the first two are highly recommended:

```
\BibPublisher{Oxford University Press}
\BibCity{Oxford}
\BibEdition{4th}
\BibPartOfBook{Ch.~4}
```

4.1.15 Contributions to books or proceedings

To specify that a given reference is contained *in* a book, *i.e.*, is a contribution to a book, rather than being the complete book itself, you use the command

```
\BibInBook{},
```

where the argument contains the title of the book that the reference is contained in. For example, you might use

```
\BibInBook{Electromagnetism: Paths to Research}
```

to produce references such as

... P. Pearle, in *Electromagnetism: Paths to Research* ...

You can also use this command for contributions to *conference proceedings*; to make its use intuitive, there is a completely equivalent command

```
\BibProceedings{},
```

where the argument should contain the title of the proceedings volume.

The following commands are optional (but suggested) for `\BibInBook{}` or `\BibProceedings{}`:

```
\BibTitle{The title of my contribution}
\BibEditor{I. M. A. Editor}
\BibEditors{A. B. First and C. D. Second}
\BibPublisher{Worldwide Pathetic}
\BibCity{Dudsville}
\BibPartOfBook{Ch.~3}
```

4.1.16 Ph.D. theses

You specify that the reference is a Ph.D. thesis with the command

```
\BibPhDThesis.
```

The following commands are all optional for Ph.D. theses, but the first is strongly suggested:

```
\BibInstit{The University of Melbourne}
\BibTitle{Single-Particle Electrodynamics}
\BibUnpublished
```

4.1.17 Specifying that the reference continues on

If you want to place multiple references within the same numbered reference (*e.g.*, to list *all* papers on a certain topic as reference [3]), you use the command

```
\BibMore
```

after each one, instead of a new `\BibItem` for each.

The command

```
\BibMoreInfo Ii{nformation}
```

simply provides more information about the reference; for example,

```
\BibMoreInfo Aa{vailable by anonymous-ftp from
ftp.ph.unimelb.edu.au, directory pub/theses/costella}
```

Note that the arguments of `\BibMoreInfo` require the same counterintuitive fudging as those of `\BibText` with regard to the case of the first letter of the textual information.

The command `\BibMoreSameAuthor(s)` (where the final `s` is optional) acts like `\BibMore`, but it furthermore tells CostelLaTeX that the authors are the same as the preceding reference. Normally, the authors will not be reprinted in this circumstance.

The command `\BibMoreSameJournal` command specifies that the reference has the same authors *and* is in the same journal as the preceding reference. In many cases, the journal name is not reprinted. You must of course specify at least the volume and start page for the new reference.

4.1.18 Ending the bibliography section

To end the bibliography section, use the command

```
\EndBib.
```

This will usually be the second-last command in your document (the last being `\EndDocument`).

4.1.19 An example bibliography

To illustrate the bibliography commands described in this section, I include an excerpt from the bibliography code for a paper to appear shortly in the American Journal of Physics:

```
\BeginBib
%
\BibItem
\BibYear{1994}
\BibNameTwo JP{Costella}
\BibPhDThesis
\BibTitle{Single-Particle Electrodynamics}
\BibInstit{The University of Melbourne}
\BibUnpublished
\BibMoreInfo Aa{vailable by anonymous-ftp from
  ftp.ph.unimelb.edu.au, directory pub/theses/costella,
```

```
  or on the
  WWW:~http:/$\!$/www.ph.unimelb.edu.au%
  /$\sim$jpc/homepage.htm}
%
\BibItem
\BibYear{1949}
\BibNameTwo TD{Newton}
\BibNameTwo EP{Wigner}
\BibJournal{Rev. Mod. Phys.}
\BibTitle{Localized states for elementary systems}
\BibVolume{21}
\BibStartPage{400}
\BibEndPage{406}
%
\BibItem
\BibYear{1958}
\BibNameThree PAM{Dirac}
\BibBook{The Principles of Quantum Mechanics}
\BibEdition{4th}
\BibPublisher{Oxford University Press}
\BibCity{Oxford}
%
\BibItem
\BibYear{1964}
\BibNameTwo JD{Bjorken}
\BibNameTwo SD{Drell}
\BibBook{Relativistic Quantum  Mechanics}
\BibPublisher{Mc\-Graw-Hill}
\BibCity{New York}
%
\BibItem
\BibAppend a
\BibYear{1952}
\BibNameTwo LL{Foldy}
\BibJournal{Phys. Rev.}
\BibTitle{The electromagnetic properties of
  Dirac particles}
```

```
\BibVolume{87}
\BibStartPage{688}
\BibEndPage{693}
%
\BibItem
\BibAppend b
\BibYear{1952}
\BibNameTwo LL{Foldy}
\BibJournal{Phys. Rev.}
\BibTitle{The electron--neutron interaction}
\BibVolume{87}
\BibStartPage{693}
\BibEndPage{696}
%
\EndBib
```

4.2 Citing a reference in your bibliography

Hand-in-hand with COSTELLATEX's bibliography commands are its commands for handling *citations* in a journal-independent way.

It requires some discipline to use all of the following commands properly, but doing so will let you change your document conventions without needing to change any of the text.

4.2.1 Parenthetical citations

Most citations you insert into your documents will be *parenthetical* (*i.e.*, "in brackets"): grammatically, they are not a part of the sentence at all, but are rather more like a footnote (which is precisely why journals used to use footnotes for them). For example, if you write

> The Smith function [15] is important.

or

> The Smith function[15] is important.

or

The Smith function (Smith and Jones, 1990) is important.

then in each situation the sentence would be perfectly correct if the citation were omitted:

The Smith function is important.

To insert such a parenthetical citation into your document, you simply use the command

```
\Cite{}
```

where the argument contains the *citation label* of the reference you want to cite (*see* Secs. 4.1.8 and 4.1.9). For example, if the corresponding entry in your bibliography looks like

```
\BibItem
\BibYear{1990}
\BibNameOne J{Smith}
\BibNameThree KPR{Jones}
...
```

then you would produce the above citations with code of the form

```
The Smith function\Cite{Smith1990} is important.
```

Note that *all of the above forms of citation are automatically generated*, depending on what document conventions are in force, from the one simple command. This power does, however, carry with it some subtleties; for example, the space before "[15]" is *automatically generated*, even though we didn't put any space in the source code. The reason, of course, is that in the *superscript* convention, an explicit space would be typographically incorrect, and hence the CostelLaTeX command `\Cite{}` has been defined so that it *must* be attached to the end of a word.

Note also that CostelLaTeX *automatically* extracts the authors' surnames and year of publication from the bibliography, if author–year citation conventions (the third example above) are in force.

4.2.2 Punctuation next to a parenthetical citation

There is a subtle issue of convention that arises whenever you need to place a *punctuation symbol* after the word that a parenthetical citation is attached to. The problem is that standard typographical conventions dictate that the punctuation should come *after* a citation of the form "[12]", such as in

> This has been proven [12].

but *before* one of the form "[12]", such as in

> This has been proven.[12]

This is an issue that you need to address, regardless of the method you use to prepare your manuscripts.

Obviously, if you wrote a paper using the former "square-bracket" convention, always putting an explicit punctuation symbol *after* the citation commands, and then at some point wanted to print it using the "superscript" convention, then you would need to edit every citation, to switch the order of the two commands.

As you might have already guessed, CostelLaTeX's commands are designed to *automatically* handle this change of notation for you, provided that you use the command

```
\punct{}
```

immediately before the `\Cite{}` command of a parenthetical citation attached to a word that is followed by a punctuation symbol. For example, the code for the above output would be

```
This has been proven\punct.\Cite{Smith1986}
```

Note that the `\punct{}` command reads in one argument: the full stop that follows it. As a further example, the code

```
We use the Smith function\punct,\Cite{Smith1990} but
we do not believe that it is correct.
```

produces

> We use the Smith function [15], but we do not believe that it is correct.

or

> We use the Smith function,[15] but we do not believe that it is correct.

or

> We use the Smith function (Smith and Jones, 1990), but we do not believe that it is correct.

depending on the document conventions in force.

4.2.3 Explicit citations

If a citation is used as a noun in a sentence, rather than being a parenthetical reference, then it is referred to as being an *explicit* citation. For example, in the sentence

> See [15] for details.

the citation "[15]" is the object of the verb "see". In general, if you want to check whether a given citation is parenthetical or explicit, then omit the citation, and if the remaining words do not make a proper sentence, then the citation is explicit. For the above example, we see immediately that

> See for details.

makes no sense at all, and hence the citation is explicit.

The reason that explicit citations must be distinguished from parenthetical ones is that they must take a substantially *different* form, using superscript or author–year conventions. For example, in a journal using superscripts for citations, the above would need to be written

> See Ref. 15 for details.

rather than using a superscript at all. Likewise, in a journal using the author–year convention for citations, it would appear as

> See Smith and Jones (1990) for details.

where you will note the parentheses now surround *only* the year—not the authors themselves, as they do for a parenthetical reference.

To let CostelLaTeX produce explicit citations for you correctly, regardless of the document conventions in force, you need to use the command

```
\CiteExplicit{}
```

rather than `\Cite{}`. For example, the above output would be obtained using the code

```
See \CiteExplicit{Smith1990} for details.
```

Note that, since the citation now acts like a word or phrase in the sentence, there is *always* a space before it, and there are *never* subtleties of punctuation. Thus, you place a space in your code before `\CiteExplicit{}`, just like you would any other word, and you place punctuation after it. For example,

```
This result is proved in \CiteExplicit{Smith1990}.
```

4.2.4 Citing multiple references at once

You can cite multiple references in the one citation, simply by enclosing the relevant `\Cite{}` commands within the commands

```
\CiteBeginMulti
\CiteEndMulti
```

For example, the code

```
The result has been found%
\CiteBeginMulti
\Cite{Smith1902}
\Cite{Brown1945}
\Cite{Jones1977}
\CiteEndMulti
to depend on the trial wavefunction.
```

produces the output

> The result has been found [11, 13, 20] to depend on the trial wavefunction.

or

> The result has been found[11,13,20] to depend on the trial wavefunction.

or

> The result has been found (Smith, 1902; Brown and Hodges, 1945; Jones and Graham, 1977) to depend on the trial wavefunction.

depending on the document conventions in force. Note that there should be no space before the `\CiteBeginMulti` command (hence the need for the `%` command after the word `found` in the example above), but that spaces after the individual `\Cite{}` commands in such a structure *are* allowed (and ignored by COSTELLaTeX).

Multiple citations possess the same subtleties with regard to punctuation as single ones; if the word that the citation is attached to is to be followed by a punctuation symbol, then you must *precede* the citation by the `\punct{}` command. For example, the code

```
The integral is known to vanish%
\punct;\CiteBeginMulti
\Cite{Frogston1966}
\Cite{Smiggles1974}
\CiteEndMulti
we will not rederive this result here.
```

would produce the output

> The integral is known to vanish [7, 19]; we will not rederive this result here.

or

> The integral is known to vanish;[7,19] we will not rederive this result here.

or

> The integral is known to vanish (Frogston, 1966; Smiggles and Biggles, 1974); we will not rederive this result here.

as appropriate.

4.2.5 Explicit multiple citations

If your multiple-reference citation is an *explicit* one (rather than parenthetical), then you need to use the commands

```
\CiteExplicitBeginMulti
\CiteExplicitEndMulti
```

rather than `\CiteBeginMulti` and `\CiteEndMulti`. For example, the code

```
For more details, see
\CiteExplicitBeginMulti
\Cite{Smith1951}
\Cite{Jones1952a}
\Cite{Jones1952b}
\Cite{Brown1958}
\CiteExplicitEndMulti
.
```

produces the output

> For more details, see [21, 22, 23, 24].

or

> For more details, see Refs. 21, 22, 23, 24.

or

> For more details, see Smith (1951), Jones (1952a), Jones (1952b), Brown and Johnston (1958).

depending on the document conventions in force. Note that, just as with `\CiteExplicit{}`, the multiple-citation is considered to be a phrase; thus, you *do* put a space before `\CiteExplicitBeginMulti`, and you put any punctuation symbols required directly after `\CiteExplicitEndMulti`.

You must be careful, however, if using an explicit multiple-citation in the *middle* of a sentence: since the command `\CiteExplicitEndMulti` takes no arguments, there are consequently no braces to complete the command, and hence you must use a "normal-space" command to indicate that a space should follow. For example, the code

```
This is proved in
\CiteExplicitBeginMulti
\Cite{Whitlam1975}
\Cite{Fraser1983}
\Cite{Hawke1992}
\CiteExplicitEndMulti\
as well as being anticipated in \Cite{Keating1996}.
```

produces

> This is proved in [24, 25, 30] as well as being anticipated in [31].

or

> This is proved in Refs. 24, 25, 30 as well as being anticipated in Ref. 31.

or

> This is proved in Whitlam (1975), Fraser (1983), Hawke (1992) as well as being anticipated in Keating (1996).

4.2.6 Appending a note to an explicit citation

Although of sometimes dubious value, bare LaTeX lets you append a "note" to a single explicit citation. CostelLaTeX provides this functionality with the command

```
\CiteNote{}{}.
```

For example, the code

```
See \CiteNote{Zulu1980}{Ch.~4} for details.
```

produces the output

> See [45, Ch. 4] for details.

or

> See Ref. 45, Ch. 4 for details.

or

See Zulu (1980), Ch. 4 for details.

In most cases, you would be better off writing the "note" into the text of your sentence, as many journals with square-bracket citation conventions do not accept extra material being placed in the brackets.

4.2.7 Extracting authors' surnames

CostelLaTeX provides four commands that automatically extract the surnames of the authors of a given reference from the bibliography for you, so you don't have to retype them:

```
\CiteAuthor{}
\CiteAuthorOnly{}
\CiteAuthors{}
\CiteAuthorsOnly{}
```

The latter two commands are identical to the former two; they are simply more intuitive when referring to more than one author.

The `\CiteAuthor(s){}` command lists the authors of the given bibliography entry, and then cites it. For example, the code

```
This was shown by \CiteAuthors{bj} to be true.
```

would produce

This was shown by Bjorken and Drell [34] to be true.

or

This was shown by Bjorken and Drell[34] to be true.

or

This was shown by Bjorken and Drell (1964) to be true.

as appropriate.

The `\CiteAuthor(s)Only{}` command only extracts the authors' surnames; it does not produce a citation. For example, the code

```
\CiteAuthorsOnly{bj} also found another result.
```

produces

Bjorken and Drell also found another result.

in all document styles.

4.2.8 Repeated punctuation symbols

There is another, "second-order" subtlety with punctuation appearing next to a citation, that arises when using commands such as `\CiteAuthor{}`: what happens if the list of authors ends with "*et al.*"? In square-bracket conventions, there is no problem: the code

```
This was done by \punct.\CiteAuthors{Smith1907}
```

produces the output

This was done by Smith *et al.* [49].

However, for the *superscript* notation style, the result of the above code would be

This was done by Smith *et al.*.[49]

where the first full stop comes from "*al.*", and the second is the full stop ending the sentence.

In practice, a repeated punctuation symbol such as this is replaced by a *single* punctuation symbol:

This was done by Smith *et al.*[49]

It would be nice if CostelLaTeX were able to be smart enough to do this for you. Unfortunately, it cannot. Instead, you need to use the command

```
\repeatpunct{}
```

rather than `\punct{}` to place a full stop after a `\CiteAuthor(s){}` command for which the list of authors ends in "*et al.*" In other words, the code for the above citation needs to be modified to

```
This was done by \repeatpunct.\CiteAuthors{Smith1907}
```

4.2.9 Including Al for citations only

You will recall, from Sec. 4.1.7, that placing the command

```
\BibEtAl
```

in a bibliography entry will truncate the list of authors to "first author *et al.*" However, for some purposes (*e.g.*, review articles), it is sometimes necessary to list *every* author in the bibliography, even if there are many. A problem then arises if you want to use a command such as `\CiteAuthor{}` for this reference: every time you do, you will get all of the many authors' surnames listed!

To avoid this problem, you can use the command

```
\BibCiteEtAl
```

after the last author in the given entry in the bibliography. This command tells CostelLaTeX to still list all of the authors in the bibliography, but instead use "first author *et al.*" when using a command like `\CiteAuthor{}`.

Seminar 5: Tables and figures

In this last seminar, I will describe how to include tables (Sec. 5.1) and figures (Sec. 5.2) in your document. For the benefit of those who want to get into the nitty gritty of defining their own macros, Sec. 5.3 provides a brief overview of some of the commands that CostelLaTeX provides to simplify the "programming of LaTeX". Finally, Sec. 5.4 describes in simple and straightforward terms how you define your own macros, to encapsulate complex tasks into a set of simple commands.

5.1 Tables

LaTeX and CostelLaTeX make the task of creating tables relatively painless. There are two parts to this process:

1. Using `\BeginTbl` and `\EndTbl` to create a *floating table environment*, which automatically floats to the top or bottom of the page (or perhaps to a page of its own), which has a caption saying "Table n", with your desired caption text after it, and which is given its own identifying label. Floating table environments could, in principle, contain anything (*e.g.*, plain text); they are *not* restricted to containing only tabulated material (although this is generally what you would use them for).

2. Creating a `tabular` environment inside the floating table environment. This is what actually lets you put text and maths in a tabular form. It doesn't *have* to go inside a floating table environment (*e.g.*, it could just appear in your document like a displayed equation), but in most situations it should be.

I will describe each of these processes in turn, and will then show how they are put together.

5.1.1 Floating table environments

To begin a floating table environment, you use the command

    \BeginTbl.

This is a pretend table.

Table 5.1: A pretend table.

To end it, you use the command

```
\EndTbl{}{}
```

where the first argument is the *label* that you wish to refer to the table by, and the second argument is the text that you want to appear in the table's caption. For example, the code

```
Refer to the accompanying table.%
\BeginTbl
This is a pretend table.
\EndTbl{pretend}{A pretend table.}
It lists nothing of use at all.
```

produces the output

> Refer to the accompanying table. It lists nothing of use at all.

Note that the floating table environment should be attached to the end of a word, *without any intervening space*, just like the `\footnote{}` command.

You can stop COSTELLaTeX from "floating" most of your tables to the top or bottom of pages, by using the command

```
\TblNoFloat
```

anywhere before the table or tables in question. This allows LaTeX the option of placing tables in the text, at the position they appear. Depending on the situation, however, individual tables may still be floated to the following page, or to a page of their own, depending on how much space is available on the current page.

You can restore normal (floating) behaviour by issuing the command

```
\TblFloat.
```

0	Nothing	2	More than 1
1	Something	3	Even more

Table 5.2: A sample table.

5.1.2 Tabulating your material

The `tabular` environment is practically identical to the mathematical `array` environment that was described in Sec. 2.7.12. The only difference is the following:

- An `array` environment must appear in maths mode, and its entries are processed in maths mode.
- A `tabular` environment must appear in text mode, and its entries are processed in text mode.

Thus, `array` environments belong in equations, and `tabular` environments belong in tables.

As an example of the `tabular` environment, the code

```
\BeginTbl
\begin{tabular}{clcl}
0 & Nothing   & 2 & More than 1 \\
1 & Something & 3 & Even more
\end{tabular}
\EndTbl{label}{A sample table.}
```

produces Table 5.2.

There are many other formatting tricks you can play with tables, mainly to do with horizontal and vertical lines. These are all listed in Lamport's LaTeX manual. I will only describe two of these commands, because, in general, you should try to make your tables *as simple as possible*. You should look at all of the many, many tables appearing in these seminar notes. Many of them contain no lines; most of the remainder contain a *single horizontal line* separating the column headings from the data. Rarely, a vertical line is useful. You should *never* try to produce a fully "boxed in" table, *i.e.*, one that looks like a computer speadsheet; it is simply too visually complicated. Professionally typeset tables use as few "gimmicks" as possible.

	0	1
0	0	0
1	0	1

Table 5.3: The Boolean 'and' operation.

The two commands I will describe, then, are as follows: If you put a `|` character (called a "pipe" or "vertical bar" character; it will in general be in some obscure corner of your keyboard) in the extra argument of `\begin{tabular}`, a vertical line will be placed between the corresponding columns of the table. Similarly, if you place the command `\hline` after the `\\` of a row of the table, a horizontal line will be placed under that row. For example, the code

```
\BeginTbl
\begin{tabular}{c|cc}
  & 0 & 1 \\ \hline
0 & 0 & 0 \\
1 & 0 & 1
\end{tabular}
\EndTbl{boolean}{The Boolean 'and' operation.}
```

produces Table 5.3.

5.1.3 Appending letters to table numbers

CostelLaTeX allows you to append letters to the numbers of individual tables, to produce numbers such as Table 1(a). The commands for doing so are completely analogous to those described in Sec. 2.5.7 for appending letters to equation numbers; Table 5.4 lists the relevant command names.

5.2 Figures

CostelLaTeX provides a powerful set of commands that lets you include arbitrary figures in your LaTeX document. The following sections describe the steps that you must take to do so.

Function	Equation command	Table command
Begin appended letters	`\eqnbeginletters`	`\TblBeginLetters`
End appended letters	`\eqnendletters`	`\TblEndLetters`
Set brackets used	`\eqnsetletterbrackets`	`\TblSetLetterBrackets`

Table 5.4: Commands for appending table numbers with letters.

5.2.1 Floating figure environments

The first step in including a figure is to create a *floating figure environment*, completely analogous to the floating table environment of Sec. 5.1.1. The commands are the same; the only difference is that you change `Tbl` in the command name to `Fig`. Here is a list of the commands you use:

```
\BeginFig
\EndFig{}{}
\FigNoFloat
\FigFloat
\FigBeginLetters{}
\FigEndLetters
\FigSetLetterBrackets{}{}
```

5.2.2 Leaving space for a manually-pasted figure

Although not the most technologically elegant solution, at some point in time you will want to simply leave enough space in your floating figure environment for a figure that you will paste in by hand. To do so, you use the command

```
\FigLeaveSpace{}
```

where you specify how much *vertical space* you want to leave as the argument. For example, the code

```
This is shown in the accompanying diagram.%
\BeginFig
\FigLeaveSpace{2cm}
\EndFig{manual}{Space left for a
manually-pasted figure.}
Note the whiteness of the blank space left.
```

Figure 5.1: Space left for a manually-pasted figure.

produces the output

> This is shown in the accompanying diagram. Note the whiteness of the blank space left.

Note that dimensions can be in millimetres, centimetres or inches; for example,

```
7.65in
12.8cm
45.8mm
```

If you are working on any but the simplest of projects, it may be worth examining the option of scanning in manually-pasted figures using a digital scanner, and including them in your document via a PostScript file. This would make sense if, for example, you might wish to send the document over the Internet, or if you might want to put it on the World Wide Web at some stage in the future.

5.2.3 Including PostScript figures

The commands described in Sec. 5.2.1 set up a floating figure environment for you, complete with "Figure n:", a caption, and an identifying label. But they don't actually give you any way of *drawing* a figure.

I have some bad news and some good news.

The bad news is that there is no viable and simple way of including diagrams, photographs, *etc.*, into a LaTeX document, so that *every possible user* of LaTeX, around the world, can view and print the output.

The good news is that you can include practically *any* type of figure via a *PostScript file*. PostScript is a page description language, created by Adobe, and originally made popular by the Apple laser printer range. It is now the standard format for the electronic interchange of figures throughout academia. (A "figure" could, for these purposes, be defined as simply any material that your computer can print on a page.)

If you have a PostScript printer *(note: not all laser printers are PostScript printers!)*, then you can print PostScript files directly. If not, then you will need to use a program such as GhostView (freeware) to convert the PostScript description of a page to one that your printer can understand. (Practically all Windows-connected printers can print PostScript files in this way.) You can also use GhostView to view a PostScript file directly on your screen. (A DVI viewer will simply leave a blank space wherever a PostScript file is to be inserted.)

5.2.4 How do I get a PostScript figure?

If you can print something on a Macintosh or Windows computer, from any application program whatsoever, then you can get a PostScript file of it.

Simply go to the printer setup dialogue box and ask the computer to print to a PostScript file, rather than the printer. (Windows users without a PostScript printer should install a fictitious PostScript printer for this purpose.) You should generally save these PostScript files with an extension of `.ps`.

If you are having trouble getting PostScript files out of your machine, ask a local guru (or me, if you can't find one) for some help.

PostScript files are plain text, so they can be sent to other people via e-mail without problem. (As long as they are not too huge!)

You will save yourself a lot of trouble later if you ensure that your PostScript files are formatted for *A4 paper*, not the American Letter size of paper. You should also try and make sure that the object you are printing will be *centred horizontally* on the page; use Print Preview (if your application has this facility) to set the margins, *etc.*, so that this is the case. However, it does not really matter whether or not the object is *vertically* centred on the page or not.

5.2.5 Including the figures

Once you have the PostScript files for your figures, including them in your document is simple:

1. Decide how many images "across the page" you would like (*i.e.*, how many "columns" of images you want). Issue one of the commands

```
\FigOneWide
\FigTwoWide
\FigThreeWide
\FigFourWide
```

2. Use the `\BeginFig` and `\EndFig` commands to set up a floating figure environment.

3. For each image you want in the floating figure environment, insert a command

```
\Figure{filename.ps}
```

   where `filename.ps` is the name of the PostScript file in question.

That's all there is to it!

Well, you might have to crop some space off the edges of the images ...

5.2.6 Cropping excess space from your figures

If you have created your PostScript files by "printing" to a file from an application program, then each file will be in the shape of an A4 page. If you take a look at the output of these PostScript files, you will most likely will find that there is some amount of (now unwanted) *white space* around the figure. Some of this will be simply be the margins left by the application program, to ensure that the figure is in the area of page printable by your printer. There may also be too much white space to the left or right of the figure. Hopefully, you have been able to centre your figures horizontally on the page, but if not, the left and right areas of white space won't be equal. Also, in general, unless the object you are printing just happens to be the shape of an A4 page, there will be some blank space above and below the figure.

On the other hand, CostelLaTeX assumes that the figure you are including contains *only* the amount of surrounding white space that is necessary to separate the various figures in the one floating figure environment. (If there is only one figure, it assumes that the figure fills the entire space allocated to it.) To bring COSTELLaTeX's expectations into line with reality, you need to tell it how much of this "white space" it has to crop from the edges of the figure. To do so, you do the following:

1. If you are including multiple figures in the one floating figure environment, ensure that they have been produced with the one application program, with consistent page layout settings. It is too much trouble if each figure is on a different place on the page; COSTELLaTeX does not support such complicated constructions.

2. Ensure that your PostScript files are all formatted for A4 paper.

3. Out of the figures that you want to include in the one floating figure environment, choose the one that is largest out of the lot.

4. LaTeX the file `figgrid.tex` (provided with the COSTELLaTeX package). If you use a command line version of LaTeX, you need simply type

   ```
   latex figgrid
   ```

5. LaTeX will then ask you for the name of your PostScript file. Tell it the name of the file for the largest figure just chosen.

6. Convert the output `figgrid.dvi` to a PostScript file, say, `figgrid.ps`, using a program such as `dvips`.

7. View the file `figgrid.ps` using a program such as GhostView, or print it on your PostScript printer. You should see a full-page version of your figure, with an overlying *cropping grid*. An example of this, using the Abstract page of my Ph.D. thesis as the relevant PostScript file, is shown in Fig. 5.2.

8. Read off suitable values for cropping of the top and bottom of the figure. Don't forget to leave a bit of space around each figure. For Fig. 5.2, suitable values would be

   ```
   \FigCropTop{0.2}
   \FigCropBottom{0.18}
   ```

9. If your figure is not centred horizontally, you will need to shift it left or right. Fig. 5.2 could be shifted left by about 0.02 units (although this is being a tad pedantic), so issue the command

   ```
   \FigShiftLeft{0.02}
   ```

10. Read off a value for left–right cropping. Note that the scale is non-linear; this is due to technical reasons to do with LaTeX's mathematical prowess. For Fig. 5.2, a reasonable value would be

    ```
    \FigCropLR{0.2}
    ```

The above commands should be placed *after* the `\BeginFig` command for the floating figure environment, and *before* the first `\Figure{}` command appearing therein.

5.2.7 What if I have missing figures?

Just say that you want figures three-abreast on the page. What happens if you only have eight figures in total?

Well, you cannot just leave the ninth spot empty; my macros are not smart enough to figure out what to do. For each empty image on the last "row" of images, you must issue one of the commands

```
\FigEmpty
\FigOmit
```

The former pretends that a figure exists at that position (it leaves a blank space for it). The latter tells CostelLaTeX to shift the remaining figures inwards so that they are horizontally centred.

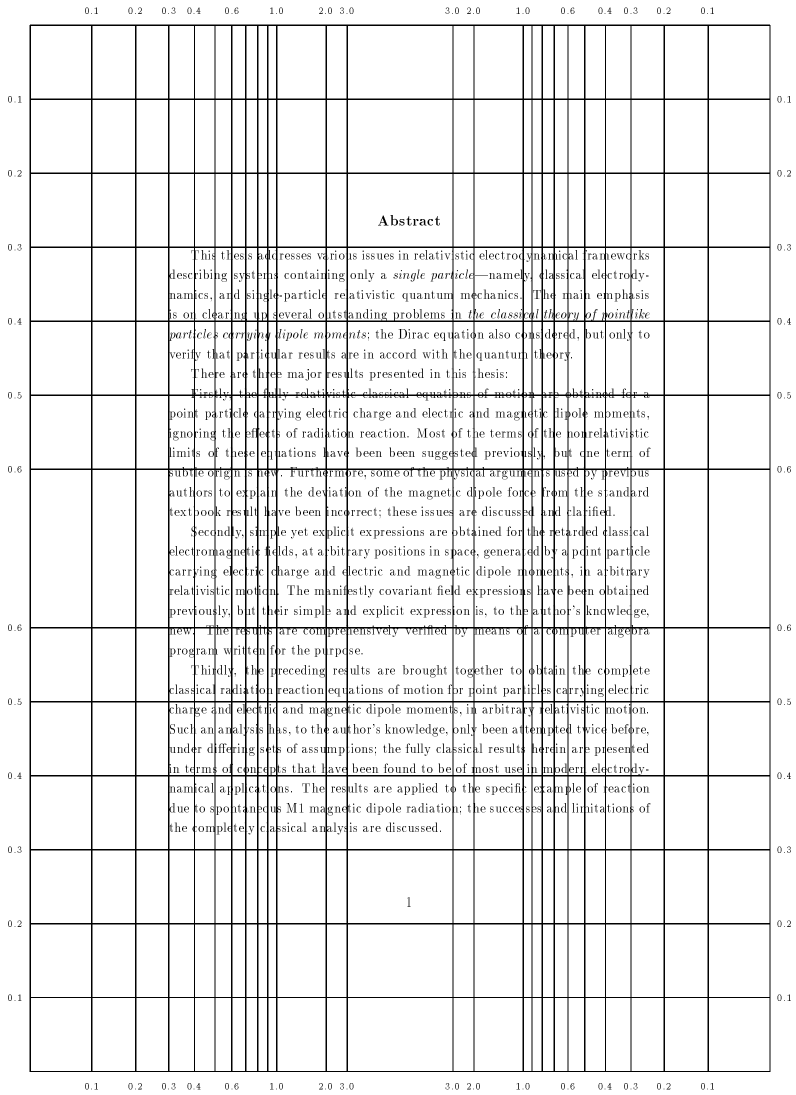
5.2.8 Lettering individual figures

To have CostelLaTeX automatically letter your figures (a), (b), (c), *etc.*, within each floating figure environment, simply issue the command

```
\FigLetter
```

anywhere in your document before the `\BeginFig` command for that figure.

To stop CostelLaTeX from lettering the individual figures in each floating figure environment, issue the command



Abstract

This thesis addresses various issues in relativistic electrodynamical frameworks describing systems containing only a *single particle*—namely, classical electrodynamics, and single-particle relativistic quantum mechanics. The main emphasis is on clearing up several outstanding problems in *the classical theory of pointlike particles carrying dipole moments*; the Dirac equation also considered, but only to verify that particular results are in accord with the quantum theory.

There are three major results presented in this thesis:

Firstly, the fully relativistic classical equations of motion are obtained for a point particle carrying electric charge and electric and magnetic dipole moments, ignoring the effects of radiation reaction. Most of the terms of the nonrelativistic limits of these equations have been been suggested previously, but one term of subtle origin is new. Furthermore, some of the physical arguments used by previous authors to explain the deviation of the magnetic dipole force from the standard textbook result have been incorrect; these issues are discussed and clarified.

Secondly, simple yet explicit expressions are obtained for the retarded classical electromagnetic fields, at arbitrary positions in space, generated by a point particle carrying electric charge and electric and magnetic dipole moments, in arbitrary relativistic motion. The manifestly covariant field expressions have been obtained previously, but their simple and explicit expression is, to the author's knowledge, new. The results are comprehensively verified by means of a computer algebra program written for the purpose.

Thirdly, the preceding results are brought together to obtain the complete classical radiation reaction equations of motion for point particles carrying electric charge and electric and magnetic dipole moments, in arbitrary relativistic motion. Such an analysis has, to the author's knowledge, only been attempted twice before, under differing sets of assumptions; the fully classical results herein are presented in terms of concepts that have been found to be of most use in modern electrodynamical applications. The results are applied to the specific example of reaction due to spontaneous M1 magnetic dipole radiation; the successes and limitations of the completely classical analysis are discussed.

1

Figure 5.2: PostScript file of the Abstract page of my Ph.D. thesis, with cropping grid overlaid. The procedure for obtaining such output for your own PostScript file is described in the text.

```
\FigDontLetter.
```

5.2.9 Captioning individual figures

To add a caption to an individual figure within a given floating figure environment, simply preface its `\Figure{}` command by the command

```
\FigCaption{},
```

where the argument contains the text that you want as the caption for that figure.

5.2.10 Labelling individual figures

To give an individual figure within a given floating figure environment its own *label* (distinct from the label for the set of figures as a whole), preface its `\Figure{}` command by the command

```
\FigLabel{}
```

where the argument contains the label. For example, imagine that you use the code

```
\BeginFig
\FigTwoWide
\FigLetter
%
\FigLabel{first}
\FigCaption{First image}
\Figure{first.ps}
%
\FigLabel{second}
\FigCaption{Second image}
\Figure{second.ps}
%
\EndFig{overall}{Overall caption.}
```

You would refer to the overall figure environment by the code

```
... Fig.~\fig{overall} ...
```

which would produce output of the form

... Fig. 5.1 ...

You could alternatively refer to the images individually, with code of the form

```
... Figs.~\fig{first} and~\fig{second} ...
```

which would produce the output

... Figs. 5.1(a) and 5.1(b) ...

5.2.11 Example figure

Because figures take up so much room, I have not illustrated each of the above commands with separate examples. I now present an example covering most of the commands above.

I have used DVIPS to create five PostScript files, from the first five pages of my Ph.D. thesis. I called them `page`x`.ps`, where $x = 0, 1, 2, 3, 4$. I then created Fig. 5.3 with the following code:

```
\BeginFig
\FigLetter
\FigThreeWide
\FigCropTop{0.2}
\FigCropBottom{0.18}
\FigShiftLeft{0.02}
\FigCropLR{0.2}
%
\FigLabel{title}
\FigCaption{title page}
\Figure{page0.ps}
%
\FigLabel{abs}
\FigCaption{abstract}
\Figure{page1.ps}
%
\FigLabel{decl}
```

Single-Particle Electrodynamics

Mr. John Paul Costella

30 June 1994

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Melbourne (School of Physics)

This thesis is less than 100,000 words in length

Produced on acid-free paper

Abstract

This thesis addresses various issues in relativistic electrodynamical frameworks describing systems containing only a *single particle*—namely, classical electrodynamics, and single-particle relativistic quantum mechanics. The main emphasis is on clearing up several outstanding problems in *the classical theory of pointlike particles carrying dipole moments*; the Dirac equation also considered, but only to verify that particular results are in accord with the quantum theory.

There are three major results presented in this thesis:

Firstly, the fully relativistic classical equations of motion are obtained for a point particle carrying electric charge and electric and magnetic dipole moments, ignoring the effects of radiation reaction. Most of the terms of the nonrelativistic limits of these equations have been been suggested previously, but one term of subtle origin is new. Furthermore, some of the physical arguments used by previous authors to explain the deviation of the magnetic dipole force from the standard textbook result have been incorrect; these issues are discussed and clarified.

Secondly, simple yet explicit expressions are obtained for the retarded classical electromagnetic fields, at arbitrary positions in space, generated by a point particle carrying electric charge and electric and magnetic dipole moments, in arbitrary relativistic motion. The manifestly covariant field expressions have been obtained previously, but their simple and explicit expression is, to the author's knowledge, new. The results are comprehensively verified by means of a computer algebra program written for the purpose.

Thirdly, the preceding results are brought together to obtain the complete classical radiation reaction equations of motion for point particles carrying electric charge and electric and magnetic dipole moments, in arbitrary relativistic motion. Such an analysis has, to the author's knowledge, only been attempted twice before, under differing sets of assumptions; the fully classical results herein are presented in terms of concepts that have been found to be of most use in modern electrodynamical applications. The results are applied to the specific example of reaction due to spontaneous M1 magnetic dipole radiation; the successes and limitations of the completely classical analysis are discussed.

1

Copyright Declaration

I understand that my Ph.D. thesis titled *Single-Particle Electrodynamics* will be lodged in the University Library, will be made available to any persons entitled to use the Library and may be photocopied or microfilmed by or on behalf of the Librarian for use for research or private study pursuant to the provisions of the Copyright Act 1968. I agree that any persons to whom a copy is furnished under such conditions may, subject to acknowledging the source of the quotation, quote extracts from the thesis in any paper or written work prepared by the user but may not without my permission publish in a paper or other written work extracts which in total would constitute an infringing copy for the purpose of the Copyright Act.

2

(a) title page (b) abstract (c) declaration

Acknowledgments

This thesis describes work carried out at the School of Physics of the University of Melbourne, from 1 January 1991 through 30 June 1994, towards the degree of Doctor of Philosophy. This work was financially supported, throughout this period, by an Australian Postgraduate Research Allowance; and additionally, for the first twelve months, by a Dixson Research Scholarship. It was also supported indirectly by the Australian Research Council, via the resources of the Theoretical Particle Physics Group of the School of Physics of the University of Melbourne.

Some of this work was carried out during a two-month visit, in early 1992, to the United States Department of Energy's Institute for Nuclear Theory at the University of Washington, which is supported by US-DOE Grant #DOE/ER40561. I would like to offer my heartfelt thanks to all at the Institute for Nuclear Theory for their generosity and helpfulness. The following institutions also provided generous support for technical visits to their establishments in 1992: Los Alamos National Laboratory; CEBAF; the University of California at San Diego; and TRIUMF.

I would like to thank the following people for innumerable discussions and unending assistance throughout my studies: J. Anandan, D. P. Barber, S. D. Bass, M. V. Berry, J. Boatman, L. A. Bursill, J. H. Choi, G. J. Clerk, J. A. Daicic, A. J. Davies, C. P. Dettman, J. Filshie, N. E. Frankel, D. Fryberger, W. Haxton, X.-G. He, P. Herczeg, K. C. Hines, L. L. Hollenberg, B. Holstein, V. D. Hopper, K. Jones, G. C. Joshi, I. B. Khriplovich, S. A. Karataglidis, A. G. Klein, K. Lekamge, J. Millener, J. M. Namyslowski, M. Narielvala, K. A. Nugent, G. I. Opat, V. S. Pantelidis, J. Pasupathy, W. Rowlands, E. Smart, J. Spencer, G. J. Stephenson Jr., M. J. Thomson, S. N. Tovey, L. Wilets, L. Wolfenstein, D.-D. Wu, T. Zier, D. A. Zubanov.

I would like to thank Peter O'Brien for putting up with my frequent cries for computational help and violations of disk-space limits.

3

I apologise to Dr. Geoff Taylor for never submitting the final write-up of my first and only experimental physics project, in 1989, and promise to recite the Particle Data Book summary tables as repentance.

I would particularly like to thank Assoc. Prof. Bill Wignall for agreeing to talk some Quantum Mechanics with a bright-eyed student in 1987, and for regularly deflecting me onto topics of interest and importance ever since.

I thank Veronica Pantelidis for her moral support, from the other side of the planet, and I hope I may meet her and the two Nicks in person some day.

I thank Henry and Miriam Cieslak for being such excellent neighbours, and especially for knocking on my door with plates of food, in the past few weeks, when I forgot to eat.

I thank Russell Petty for keeping me sane!

To my Ph.D. supervisor, Professor Bruce McKellar, I owe an unrepayable debt, for agreeing, in 1990, to supervise an unknown Engineering student who had not even taken the prerequisite undergraduate course in Quantum Mechanics. On reading this thesis, I see the products of Bruce's guidance on every page, and am reminded of the many long but rewarding afternoons on which we struggled to unlock Nature's secrets.

I must thank my various girlfriends, during this time, for reminding me that relationships are not analytical functions of time: no matter how many derivatives you measure, you never know what's going to happen next.

The lifelong friendship of Roger Behrend, and the mathematical guidance he has provided, is most deeply appreciated.

Most of all, I would like to thank my Mum, for being the best Mum in the world.

4

(d) acknowledgments (e)

Figure 5.3: The first five pages of my Ph.D. thesis, in miniature format.

```
\FigCaption{declaration}
\Figure{page2.ps}
%
\FigLabel{ack}
\FigCaption{acknowledgments}
\Figure{page3.ps}
%
\Figure{page4.ps}
%
\FigOmit
%
\EndFig{Thesis}{The first five pages of
my Ph.D.\ thesis, in miniature format.}
```

5.2.12 Placing your figures inline

Normally, using the commands `\BeginFig` and `\EndFig` results in your figure being placed in a floating figure environment. Occasionally, this is undesirable—say, if you wanted to include the University of Melbourne logo above the title of your document. If you issue the command

```
\FigInline
```

command immediately before a `\BeginFig` command, then the figure will be inserted as if it were a paragraph of its own, centred horizontally; *i.e.*, it will act like a displayed equation, typographically speaking.

Note, however, that PostScript figures included with `\FigInline` *must* be A4-shaped (*i.e.*, height/width $= \sqrt{2}$); otherwise, the vertical spacing of the figure among the surrounding text will be incorrect.

5.3 Programming LaTeX

In writing the CostelLaTeX macro package, I was effectively "programming in LaTeX".

I do not think that LaTeX was meant to be programmed in. Sometimes, I think it would be easier programming an electric toaster: official LaTeX has no `if..then` statements, no `for` loops, no division operation.

Short command	Full command name
`\nc`	`\newcommand`
`\rnc`	`\renewcommand`
`\prot`	`\protect`
`\nlen`	`\newlength`
`\slen`	`\setlength`
`\alen`	`\addtolength`
`\nctr`	`\newcounter`
`\sctr`	`\setcounter`
`\actr`	`\addtocounter`
`\hsp`	`\hspace`
`\vsp`	`\vspace`
`\ec`	`\ignorespaces`

Table 5.5: Shorter names for commonly-used LaTeX commands.

The reason I chose this path, rather than using lower-level (and more powerful) TeX commands, was so that I could write an "unmacro" program that removed all LaTeX macros from a set of source files, and spit out pure LaTeX code. (This program *has* now been written, and will be released shortly.) If I had used low-level TeX commands, these would survive the unmacroing process, and would pollute the resulting LaTeX output.

If you ever choose to program LaTeX yourself (which I don't recommend you do, except to define simple document-specific macros), you may find it easier using a few of the commands described in the following sections.

Note that CostelLaTeX defines many *internal* commands, all with names of the form `\C....` Do not use any of these internal commands; they are likely to change without notice.

5.3.1 Shorter names for common commands

CostelLaTeX provides shorter names for various commands in LaTeX that are frequently used, as listed in Table 5.5.

5.3.2 Defining empty commands

CostelLaTeX provides the two commands

```
\nce{}
\ncep{}
```

that define new commands, taking no or one argument respectively, that do nothing. They are generally used for defining "hooks" that are redefined by various conditions or included files. For example, the code

```
\nce\test
\ncep\testone
```

is in all ways equivalent to the code

```
\nc\test{}
\nc\testone[1]{}
```

5.3.3 Clearing commands

CostelLaTeX provides three commands for "clearing" previously-defined commands:

```
\cnc{}
\cncp{}
\cncptwo{}
```

for commands that take no, one and two parameters respectively. For example, after the code

```
\nc\test{This is a test.}
\nc\testone[1]{This is a test with #1.}
\nc\testtwo[2]{This is a test with #1 and #2.}
\cnc\test
\cncp\testone
\cncptwo\testtwo
```

the commands `\test{}`, `\testone{}` and `\testtwo{}{}` would all do nothing (but they would still absorb the appropriate number of arguments).

5.3.4 Scope delimiters

In LaTeX, the "scope" of declarations is generally delimited by braces (*i.e.*, `{` and `}`). There are some occasions, however, in which it is not permissible to use these symbols. For example, if one would like to write a pair of macros, one of which contains a `{`, and the other the matching `}`, one finds that LaTeX will not allow it, because the braces are parsed immediately.

CostelLaTeX provides the commands `\bscope` and `\escope`, which effectively act like the `{` and `}` commands, and which may of course be used at will in macro constructions. For example, the code

```
\nc\bsmaller{\bscope\footnotesize}
\nc\esmaller\escope
This is regular \bsmaller and smaller \esmaller text.
```

would produce the output

> This is regular and smaller text.

5.3.5 Saving and restoring a counter's value

Sometimes, to play fancy tricks with LaTeX, one may want to "save" the value of some counter (*e.g.*, the `equation` counter), and then "restore" it later. CostelLaTeX provides the commands

```
\savecounter{ctrname}
\restorecounter{ctrname}
```

that perform these tasks. These commands cannot be nested.

5.3.6 Dummies

Sometimes one needs to "trick" LaTeX into thinking that a piece of text exists at a certain location, whereas in reality no text exists there. For this purpose, an empty `\mbox{}` command is generally appropriate.

To make this usage of `\mbox{}` explicit, CostelLaTeX defines a command `\dummy` that does this job. It is used several times internally, in the file `costella.tex`, but is also available for general use.

5.3.7 Preamble and postamble commands

Because the `\documentstyle` and `\begin{document}` commands are encapsulated by CostelLaTeX, and because both of these commands are only actually executed in the `\BeginDocument` command, it is not trivial inserting a true LaTeX "preamble" command (one that must occur between `\documentstyle` and `\begin{document}`) into a CostelLaTeX document.

To do so, place a `\Preamble{}` command before `\BeginDocument`, with *all* your preamble commands inside its argument. (Do not use `\Preamble{}` more than once.) For example, to change the page numbering style for your document, you might put code such as the following into your `macros.tex` file:

```
\Preamble{
\rnc\thepage{\arabic{chapter}--\arabic{page}}
}
```

There is a similar, "delayed execution" command `\Postamble{}`; the contained commands are executed after `\begin{document}`. This is necessary when you need to place such commands in the `macros.tex` file, since this file is read in before `\begin{document}` is executed.

5.4 Defining your own macros

The CostelLaTeX package is nothing but a set of "macros"—commands that I devised, and described to LaTeX. I wrote them, and subsequently revised them, to basically remove more and more of the tedious, repetitive or inflexible aspects of my LaTeXing work.

This is the key to maximising your productivity using LaTeX. If you find yourself typing the same sequence of keys repetitively—say, "the University of Melbourne", or perhaps "splodget density-distribution function", then simply define new, short LaTeX commands that are "expanded" by LaTeX into these strings of text. The CostelLaTeX command for doing this is `\nc`. For example, the code

```
\nc\tUoM{the University of Melbourne}
\nc\sddf{splodget density-distribution function}
At \tUoM, we like to measure the \sddf, because
```

```
\tUoM\ is famous for good measurements of the \sddf.
```

defines two new commands, `\tUoM` and `\sddf`, that can be used just like any other LaTeX or CostelLaTeX command. The output of the above code is

> At the University of Melbourne, we like to measure the splodget density-distribution function, because the University of Melbourne is famous for good measurements of the splodget density-distribution function.

All you have to do is make sure that the command names you invent do not clash with any others already used by LaTeX or CostelLaTeX; if they do, LaTeX will stop and tell you about it.

In general, you should place your macro definitions up the top of your document. The best place is in the file `macros.tex`—this is why it is called `macros.tex`!

A subtlety does arise, however, when it comes to sharing "modules" of your documents with other people. If you intend to do so, it might be wise if you sit down with your intended co-authors and decide on a set of procedures for defining new macros; after all, you are likely to need the same functionality as your co-authors do. It *is* possible, nevertheless to have your macros "local" to your part of an overall document, avoiding any possible clashes with other macros defined in other modules. Come and talk to me if you need to consider such subtleties in your work.

We now turn to one of LaTeX's nice features: the ability to define macros with *arguments*. As an example, imagine that you want to denote the *dual* of a quantity by placing a tilde above it. A smart thing to then do would be to put the definition

```
\nc\dual[1]{\tilde{#1}}
```

in your `macros.tex` file. The `[1]` tells LaTeX that the new command `\dual` takes *one argument*. The `#1` in `\tilde{#1}` represents this argument. You could then, in your document, use the code

```
Consider $\dual x$, $\dual y$ and $\dual\al$.
```

This would produce the output

> Consider $\tilde{x}$, $\tilde{y}$ and $\tilde{\alpha}$.

You might think that `\dual` is only a marginally shorter command name than `\tilde`; what is the point? The point is that, if after writing a hundred pages of mathematics with this command, you one day wake up and decide you should really denote duals of quantities by preceding them with an asterisk, you simply need to change the line in your `macros.tex` file to

```
\nc\dual[1]{{}^{\ast}#1}
```

The above code would then *automatically* produce the new output

Consider ${}^{*}x$, ${}^{*}y$ and ${}^{*}\alpha$.

You've just saved yourself a hell of a lot of mucking around.

LaTeX lets you have up to *nine* arguments for any command; if this is insufficient for your purposes (say, for producing a 4×4 array), you can always break up the task into smaller macros (say, by having a command for setting up the array as a whole, and a separate command for describing a row of the array).

You can encapsulate tedious or complicated constructions into multi-argument commands. For example, you might define the command

```
\nc\comb[2]{({}^{#1}_{#2})}
```

You can then produce the output

The number of ways of winning is $\binom{6}{1} + \binom{6}{2} + \binom{6}{3}$.

with the simple code

```
The number of ways of winning is
$\comb61+\comb62+\comb63$.
```

The beauty of this process is that you don't even have to know exactly *why* the original command definition works; once you have tucked it away into your `macros.tex` file, it acts just like any other LaTeX command. In fact, the CostelLaTeX package is nothing more than a set of macros that you yourself could have written, given enough time and patience; all I have done is sat down and written them once and for all.

Now you can do the same. Happy LaTeXing!