Topics on Computer Science

D. T. McGuiness, PhD

B.Sc Academic Writing

Lecture Book

2025.WS

(2025, D. T. McGuines, Ph.D)

Current version is 2025.WS.

This document includes the contents of Academic Writing, official name being *Presenting and Moderating in English*, taught at MCI in the Mechatronik. This document is the part of the module MECH-B-2-SWE-ENG-ILV taught in the B.Sc degree.

All relevant code of the document is done using *SageMath* where stated and Python v3.13.5.

This document was compiled with LuaTEX v1.22.0, and all editing were done using GNU Emacs v30.1 using AUCTEX and org-mode package.

This document is based on the following books and resources shown in no particular order:

LaTeX Tutorials: A primer by Indian TeX Users Group , Indian TEX Users Group *The LaTeX Companion Part I* by Frank Mittelbach , Addison-Wesley *The LaTeX Companion Part II* by Frank Mittelbach , Addison-Wesley *LaTeX: A Document Preparation System* by Leslie Lamport , Addison-Wesley *The TeXBook* by Donald Knuth , Addison-Wesley *Guide to LaTeX* by Helmut Kopka, Patrick W. Daly , Addison-Wesley *LaTeX Wikibook* by Wikibook , Wikibooks

The document is designed with no intention of publication and has only been designed for education purposes.

The current maintainer of this work along with the primary lecturer

is D. T. McGuines, Ph.D. (dtm@mci4me.at).

Contents

List of Figures

List of Tables

Part I

Typesetting with LaTeX

Chapter 1

Introduction

Table of Contents

1.1 Introduction

It would be a great disservice to say LaTeX is just a system for typesetting mathematics. Its applications span writing business and personal letters, newsletters, articles, and books covering the whole range of the sciences and humanities, right up to full-scale volumes and encyclopaedias and reference works on all topics [2].

Nowadays, distributions of LaTeX exist for practically every type of computer and Operating System (OS). This chapter hopes to provide a strong justification of its use along with a wealth of information about its many present-day uses. First, however, let's look at some background information.

Figure 1.1: The unofficial mascot of TeX the CTAN Lion [1].

1.2 A Historical Overview

1.2.1 Inception

In May 1977, Donald Knuth[1] of Stanford University [3] started work on the text-processing system that is now known as TEXand METAFONT [4], [5]. In the framework of the TEXbook [6], He writes:

[1] An American computer scientist, mathematician, and professor emeritus at Stanford University. He is the 1974 recipient of the Association for Computing Machinery (ACM) Turing Award. Knuth has been called the "*father of the analysis of algorithms*".

> *"TEX a new typesetting system intended for the creation of beautiful books–and especially for books that contain a lot of mathematics. By preparing a manuscript in TeX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic: quality is comparable to that of the world's finest printers"*

[2] A description language used to define raster fonts.

In 1979, Gordon Bell wrote in a foreword to an earlier book, TEX and METAFONT,[2] New Directions in Typesetting [7]:

> *"Donald Knuth's Tau Epsilon Chi (TEX) is potentially the most significant invention in typesetting in this century. It introduces a standard language in computer cryptography and in terms of importance could rather be introduction of the Gutenberg press"*

[3] The current version can be seen in CTAN.

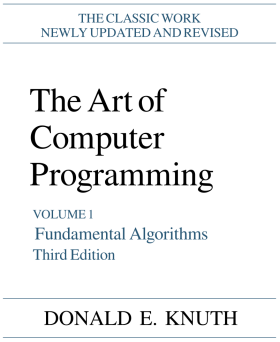
In the early 1990s, Donald Knuth officially announced TEX would **NOT** undergo any further development [8] in the interest of stability.[3] The 1990s saw a flowering of experimental projects which extended TEX in various directions which many of these are coming to fruition in the early 21st century, making it an exciting time to be involved in automated typography [2].

[4] created simply to ensure the rapid completion and typographic quality of his then-current work on *The Art of Computer Programming*

The development of TEX from its birth as one of Donald Knuth's "personal productivity tools"[4] was largely influenced and improved by the American Mathematical Society (AMS) on behalf of US research mathematicians.

Information — **The Art of Computer Programming**

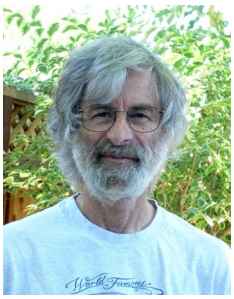
Donald Knuth is known for working on the Art of Computer Programming rather than LATEX and therefore it is worth taking a slight detour here and look at it.



It is a comprehensive, multi-volume monograph presenting programming algorithms and their analysis. As of 2025 it consists of published volumes 1, 2, 3, 4A, and 4B, with more expected to be released in the future. The Volumes 1-5 are intended to represent the central core of computer programming for sequential machines with the subjects of Volumes 6 and 7 are important but more specialised. The books are known to be hard to read but contain a detailed and scientific look at the performance of algorithms.

1.2.2 Adding Macros: LaTeX

While Knuth was developing TeX, in the early 1980s, Leslie Lamport[5] started work on the document preparation system now called LaTeX, which used TeX's typesetting engine and macro system to implement a declarative document description language based on that of a system called Scribe by Brian Reid [9]. The appeal of such a system is that a few high-level LaTeX declarations, or commands, allow the user to easily compose a large range of documents without having to worry much about their typographical appearance.



[5] An American computer scientist and mathematician, known for his seminal work in distributed systems, and as the initial developer of the document preparation system LaTeX and the author of its first manual.

> The details of the layout can be left for the document designer to specify elsewhere.

The 2nd edition of LaTeX: A Document Preparation System [10] begins with:

> *"LaTeX is a system for typesetting documents. Its first widely available version, mysteriously numbered 2.09, appeared in 1985."*

This release of a **stable** and **well-documented** LaTeX led directly to the rapid spread of LaTeX-based document processing beyond the mathematical community.

LaTeX was the first widely used language for describing the logical structure of a large range of documents and hence introducing the philosophy of logical design.

> The essence of "logical design" is that the author should be concerned only with the logical content of his or her work and not its visual appearance.

Back then, LaTeX was described variously as TeX for the common men. Its use spread very rapidly during the next decade. By 1994 Leslie could write,

> *LaTeX is now extremely popular in the scientific and academic communities, and it is used extensively in industry [2]*

The worldwide availability of LaTeX allowed fast adoption in TeX and in its use for typesetting a range of languages. LaTeX 2.09 came with documentation worth translating because of its clear structure and straightforward style. There were two (2) pivotal conferences[6] which established clearly the widespread adoption of LaTeX in Europe and led directly to internationalisation with the work led by Johannes Raman [11] on more general support for languages and switching between them.

[6] Exeter UK, 1988, and Karlsruhe Germany, 1989

> In context of typography, the word *language* does **NOT** refer exclusively to the variety of natural languages and dialects across the universe as it also has a wider meaning.

For typography, "language" covers a lot more than just the choice of "characters that make up words", as many important distinctions derive from other cultural differences that affect traditions of written communication. Therefore, important typographic differences are **NOT** necessarily in line with national groupings but rather arise from different types of documents and distinct publishing communities.

1.2.3 Reworking Fonts and the LaTeX Project

[7] A Senior IT professional, working with computer typesetting and TEX since the mid 80's, leading the LATEX development since the beginning of the 90's [12].

Another important contribution to the reach of LATEX was the pioneering work of Frank Mittelbach.[7] and Rainer Schopf on a complete replacement for LATEX's interface to font resources, the New Font Selection Scheme (NFSS). They were also heavily involved in the production of the AMS-LATEX system that is added advanced mathematical typesetting capabilities to LATEX.

As a reward for all their efforts, which included a steady stream of bug reports, and fixes for Lamport, by 1991 Mittelbach and Schopf had been allowed to take over the technical support and maintenance of LATEX with on of their first acts was to consolidate International LATEX as part of the kernel of the system, based on the typographical standards of Europe [2].

Very soon Version 2.09 was formally frozen and, although the change-log entries continue for a few months into 1992, plans for its demise as a supported system were already far advanced as something new was badly needed. The worldwide success of LATEX by the early 1990s led in a sense to too much development activity to improve and add functionality by numerous developer. This finally culminated with the announcement in 1994 of the new standard LATEX, christened LATEX 2_ε. However this was not enough and another unification effort went underway to unify the branching dialects of LATEX under one branch.

The development of this "New Standard LATEX" and its maintenance system was started in 1993 by the LATEX3 Project Team [13]. Although the major changes to the basic LATEX kernel and the standard document classes (systems in 2.09) were completed by 1994, substantial extra support for coloured topography, generic graphics, and fine positioning control were added later, largely by David Carlisle.[8] Additional work by Mark Purtill on extensions of NFSS to better support variable font encoding and scalable fonts [2], [14].

[8] Principal Technical Consulant and Group Leader of the XML Technologies group along with a core member of the LATEX group. He is currently active in the TEX stack exchange, answering anyone's questions, the author of this book included.

Although the original goal for this new version was consolidation of the wide range of models carrying the LATEX nature, what emerged was a substantially more powerful system with both a robust mechanism (via LATEX packages) for extension and, importantly, a solid technical support and maintenance system. This provides robustness via standardisation and maintainability of both the code base and the support systems.

> This system remains the current standard LATEX system.

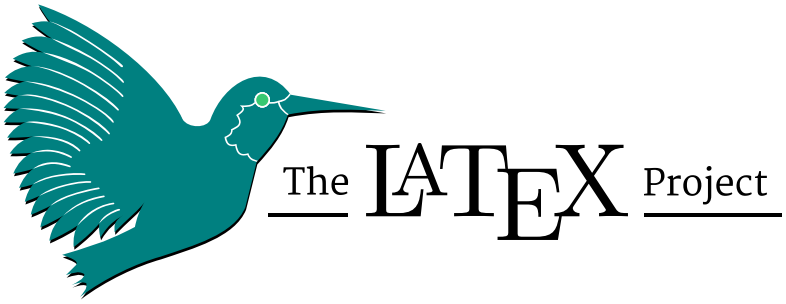


Figure 1.2: The LATEX Project logo, found on its official website [15].

1.3 Current State

Before diving into how to program and type wonderful documents, let's have an overview of the vast array of files used by a typical LaTeX system with its many components. This overview will also involve some descriptions of how the various components interact with each other and of course with the LaTeX compiler. Most users[9] will never need to know anything of this software environment which supports their work, but this section will be a useful general reference and an aid to understanding some of the more technical parts of the lecture book [2].

> The following description assumes familiarity with a standard computer file system in which a file extension is used to denote the file type.[10]

When processing a document, LaTeX reads and writes several files, some of which are further processed by other smaller applications.[11] These files, in a non-exhaustive list, are listed in **Table** 1.1.

The most obviously important files in any LaTeX documentation project are the input source files. Typically, there will be a master file that uses other subsidiary files. These files most often have the extension `.tex`, which are known as plain text files as they can be edited with a basic text editor.

> Often, several graphical images are included in the typeset document utilising the `graphicx` which we will talk later.

For LaTeX to operate, it also needs several files containing structure and layout definitions:

- *class* files with the extension `.cls`;
- *package* files with the extension `.sty`

Many of these aforementioned files are provided by the basic system set-up, but others may be supplied by individual users as customisation. LaTeX is distributed with five (5) standard document classes out-of-the-box:

`article`, `report`, `book`, `slides`, and `letter`.

These document classes can further be customised by the contents of other files specified either by class options (`.cls`) or by loading additional packages (`.sty`). In addition, many LaTeX documents will implicitly input language definition files of the `babel` system with the extension `.ldf` and encoding definition files of the `inputenc/fontent` packages with the extension `.def`.

The information LaTeX needs about the glyphs[12] is found in TeX font metric files (`.tfm`).

> This does **NOT** include information about the shapes of glyphs, only about their dimensions.

Information about which font files are needed by LaTeX is stored in font definition files (.fd).[13]

[9] For anyone who wants to learn on how to make good documents, it is worth understanding the internal cogs and gears of the software.

[10] i.e., a `.doc` extension would tell it is a Word file, or `.txt` file would imply it is a text file.

[11] For example, if you were to have a glossary of acronyms in your document, an additional `.glo` would be generated during its compilation.

[12] a glyph is "*the specific shape, design, or representation of a character*".

[13] Both types are loaded automatically when necessary.

	File Type	Common File Extension(s)
Document Input	text	`.tex .dtx .ltx`
	Bibliography	`.bbl`
	Index/Glossary	`.ind .glo`
Graphics	Internal	`.tex`
	External	`.ps .eps .tif .png .jpg .gif .pdf`
Other Input	Layout and Structure	`.cls .sty`
Internal	Auxiliary	`.aux`
	Tables of Content	`.toc`
	List of figures/tables	`.lof .lot`
Output	Results	`.pdf .dvi`
	Transcript	`.log`
Bibliography	Input and Ouput	`.aux .bib`
	Database, Style, Transcript	`.bib .bst .blg`

Table 1.1: An overview of all important files being generated during a LaTeX compilation process (Adapted from [2]).

The output from LaTeX itself is a collection of internal files mentioned previously, plus one (1) very important file that contains all the information produced by TeX about the typeset form of the document.

1.3.1 Files During Compilation

TeX's own particular representation of the formatted document is that of a device-independent file (`.dvi`). TeX positions glyphs and rules with a precision far better than 0.01 µm.[14]

[14] This is one of the reasons why the document generated by TeX is considered device independent as no device can accurately produce a document better that what TeX describes internally.

Some variants of the TeX program, such as pdfTeX, can produce device-independent file formats including the Portable Document Format (PDF) (`.pdf`), which is the native file format of Adobe Acrobat.

`.dvi` specifies **ONLY** the names/locations of fonts and their glyphs. It does **NOT** contain any rendering information for those glyphs whereas `.pdf` file format can contain such rendering information.

Some of the *internal* files contain code needed to pass information from one LaTeX run to the next,

such as for cross-references.[15] and for typesetting particular elements of the document such as the table of contents (`.toc`), the lists of figures (`.lof`), and of tables (`.lot`).

[15] the *auxiliary* file extension.

> There can be others which can be specific to particular packages or to other parts of the system.

Finally, LaTeX generates a transcript file of its activities with the extension `.log`. This file contains a lot of information, such as:

- the names of the files read,
- numbers of the pages processed,
- `warning` and `error` messages,

and other pertinent data that is especially useful when debugging errors.[16]

[16] It is an infamous joke among programmers, that the error messages of LaTeX can be cryptic. I personally disagree. While the errors may **NOT** give exact information, it does give you enough information.

Information about citations in document is written by LaTeX to a `.aux` file. This information is used first to extract the information from a database (i.e., `.bib`) and then sort it which is then written to a bibliography file (`.bbl`) [2].

Because of the limitations of TeX, especially its failure to handle graphics, it is often necessary to complete the formatting of some elements of the typeset document after TeX has positioned everything and written this information to the `.dvi` file. This is normally done by TeX's `\special` primitive which simply puts this information at the correct place in the `.dvi` file.

> This information may be simply the name of a graphs file to be input; or it may be instructions in a graphics language.

Once the document has been successfully processed by TeX, we will probably want to take a look at the format. This is commonly done on screen, but detailed inspection of printed output should always be performed via printing on paper at the highest available resolution. A current favourite approach is to use a `.pdf` file, especially when electronic distribution of the formatted document is required [2].

Occasionally you will find that some applications will produce much better quality screen output than others; this is due to limitations of the different technologies and the availability of suitable font resources.

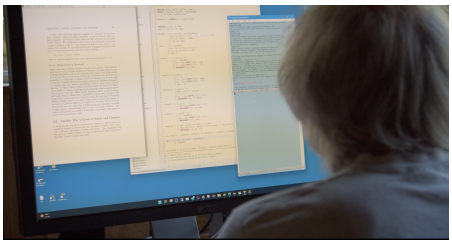


Figure 1.3: Once your document is compiled, it is always a good practice to see the generated file and look if everything is fine. Here Lamport (author of LaTeX is running Emacs with a Latex and message buffer open. Along with a PDF viewer [16]. It is worth mentioning, the preferred editor of this document's author is also Emacs.

1.3.2 A Hello, World!

[17] When I say technically, I mean given enough time and perseverance it is possible to do any kind of computation like other languages. However, this doesn't mean it is easy.

LaTeX is a programming language. It is a full fledged language, technically[17] with similar capabilities to C, C++, Python. To start it is always good to write the simplest programme possible and that is generally a `Hello, World!` programme.

Below you can see the LaTeX version of it.

C.R. 1 latex

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

\begin{document}

If one merely wishes to type in ordinary text, without complicated
mathematical formulae or special effects such as font changes, then one merely
has to type it in as it is, leaving a completely blank line between successive
paragraphs.

You do not have to worry about paragraph indentation: all paragraphs will be
indented with the exception of the first paragraph of a new section.

One must take care to distinguish between the `left quote' and the `right quote'
on the computer terminal.  Also, one should use two `single quote' characters
in succession if one requires ``double quotes''.  One should never use the
(undirected) `double quote' character on the computer terminal, since the
computer is unable to tell whether it is a `left quote' or a `right quote'.
One also has to take care with dashes: a single dash is used for hyphenation,
whereas three dashes in succession are required to produce a dash of the sort
used for punctuation---such as the one used in this sentence.

\end{document}
```

For anyone curious, this is a passage from a semi-famous document about LaTeX [17]. While the language may look unwieldy, please don't worry as by the end of this course you will learn how to craft well typed and designed documents.

1.4 Installation

As with most software which is free and open-source, there are many ways in which we can install LaTeX on our computers [18].

> The core of LaTeX is a back end software package that complies the LaTeX code (.tex file) and creates the final document (PDF).

There are a various programs available to compile your document depending on your OS.

> *The LaTeX Project provides information about how to install LaTeX on Windows, macOS, and Linux, as well as online (Overleaf) services.*

We will look at the instruction on how to install the most common LaTeX programs for Windows and macOS in the next section.

Aside from the back end software required to run LaTeX It is **highly recommend** to use a text editor.[18]

[18] It would **NOT** be the best practice to use notepad to do the majority of your academic writing on.

A text editor, to put it simply, is a front-end software which can be used to create `.tex` file used by the compiler and provides numerous benefits such as:

- Syntax-highlighting,
- Auto-completion,
- Auto-reference inserting,
- Easy navigation between files.

Text editor programs such as TeXwork, TexStudio, TexMaker, and TexShop provide a friendly interface for users and are usable out of the box with minimal configuration.

Often these front end programs include help menus, wizards for creating LaTeX objects, drop down menus for inserting symbols or altering text, and many other features that will support both new and experienced LaTeX users.

Information — **Advanced Editors**

Of course, these are **NOT** the only editors capable of working with LaTeX. As it is written in **plain text**, any editor can technically work with it. For anyone who has a pre-defined work-flow or have an affinity to working with a specific editor they are welcome to use their own editors.

For this course, we will look at **TeXStudio**, a well-crafted and supported editor.

1.4.1 Windows

For a Windows computer it is **strongly recommend** to use the common MiKTeX distribution. The MiKTeX/about page contains several links including how to install, deploy, and update MiKTeX.[19] A basic set of installation instruction can be found below, but if you are having trouble be sure to check the MiKTeX page:

[19] The logo for MiKTeX

- Download MiKTeX by clicking Download.
- Run the installer. During the installation process, you will be prompted to select the paper size. It is recommend that you choice A4. This can be changed later if needed. In additon, it is recommended to choose Yes to the option, Install missing packages on-the-fly.
- Once the installation is complete it is recommend that you check for any updates. The

update wizard can be found through the Windows start menu under the name MiKTeX. Once the window opens up, click on Updates on the left side. If not clicked, please click on Check for updates. It may take a few seconds but once done, please then click on Update now.

We should have a working LaTeX distribution on our computer. Next we need to install a nice editor and as mentioned we shall install TexStudio.

- Download TeXStudio by clicking Download.
- Run the installer.

The installation should be very straightforward. Once the installation is finished. Please click on the icon to start the application which you will be greeted with the following window

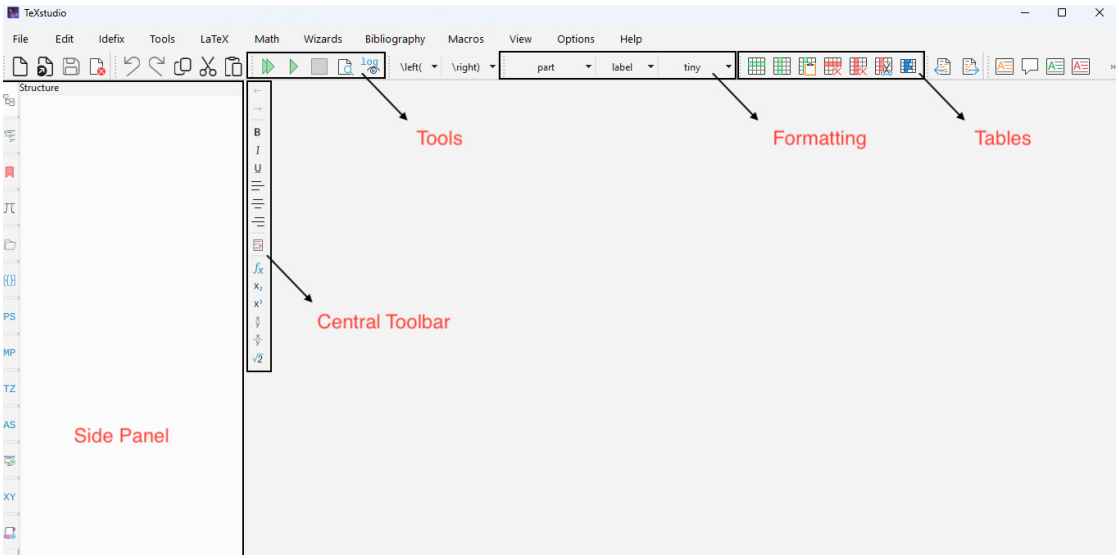


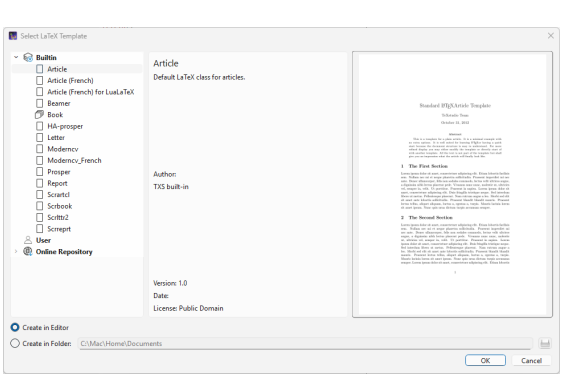
Figure 1.4: Startup of TeXStudio on Windows 11.

On the left we have the side panel which is currently showing an empty structure view. On the lower right you see the messages panel which can be switched to the log panel, the preview panel, or the search results panel. The toolbar allows easy access to some often needed functions, three of them are marked as we will use them soon. The central toolbar offers access to some common latex commands as we will see [19].

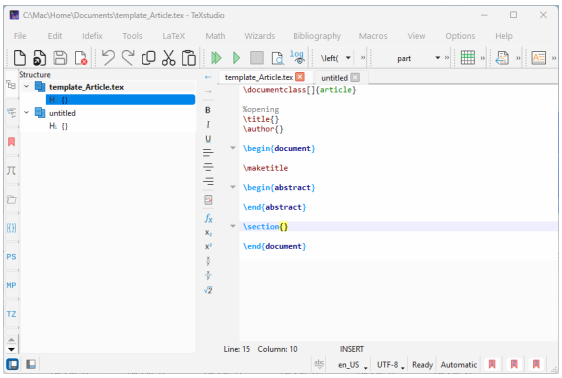
1.4.2 Creating a Document

To start easy, lets use their "Quick start" option. Inside the TeXStudio, please click on the menu File > New From Template. You will be greeted with the following menu in **Fig.** 1.5a:

Here we choose the `article` class. Once chosen, we can see some boilerplate code being entered into our editor.



(a) Default installed LaTeX templates.



(b) Template entered to the editor.

As a final part of this chapter, to see if everything is working, please click on the green triangle button to run the LaTeX compiler and then click on the document with the magnifier to see your output.

Information — Typesetting before TeX

While, of course, they existed but were highly specialised, and usually quite expensive. Here is a non-exhaustive list of some several phases in the history of maths composition [22]:

Hand Set

Anything before the late 1800s.

MonoType

A system for printing by hot-metal typesetting from a keyboard. For maths typesetting you would need a "4-line system" [23], which can be seen below:

$$\int \cosh^{-1}\left(\frac{n_1^2-\sigma_1^2}{1-\sigma_1^2}\right)^{1/2}\frac{d\sigma_1}{\sigma_1^2}$$

A

$$\sum_k{}' K_0\left(\frac{|\boldsymbol{r}-\boldsymbol{r}_k|}{L}\right)=\frac{2\pi L^2}{\sigma}$$

B

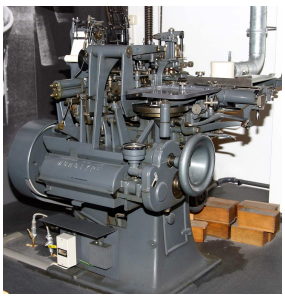


Figure 1.6: A Mono-type caster [20].

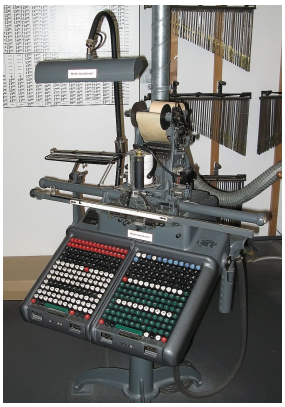


Figure 1.7: Monotype-Taster, Modell "Taster D" 1965 [21].

Typewriter

The "direct type" methods were of two (2) distinct kinds.

- Traditional one-symbol-per-key typewriters,
- A machine with interchangeable "fonts".

Phototypesetting

Uses photography to make columns of type on a scroll of photographic paper [24]. The job had to be carefully edited beforehand to make sure that no additional symbols would be needed, as changing the symbol set in the middle of a job could run the risk of exposing the output medium, which would mean starting over.

Chapter 2

Looking at the Basics

Table of Contents

2.1 The Process of LaTeX

For those who chose to skip the introduction chapter and just want to get into the heart of the topic, Here is a short rundown:

LaTeX is a typesetting program and is an extension of the original program TeX written by Donald Knuth.

But then what is a typesetting program?

To answer this, let us look at the various stages in the preparation of a document using computers:

1. The text is entered into the computer by the user or some script,
2. the input text is formatted into lines, paragraphs and pages,
3. the output text is displayed on the computer screen,
4. the final output is printed.

[1] think of Microsoft Word, Apple Pages basically, we call them "What You See Is What You Get" or WYSIWYG.

For word processors[1] all these operations are integrated into a single application package. But a typesetting language like LaTeX is concerned **only** with the second page. So to typeset a document using LaTeX we type the text of the document and the necessary formatting commands in a text editor and then compile it. After, compilation, the document can be viewed using a previewer or printed using a printer.

TeX is also a full programming language, so by learning the quirks and the nuances of the language, people can write code for additional features such as:

- Drawing scientific diagrams,
- Doing mathematical calculations,
- Automating document creation,
- Embedding lua-script, and
- Generating slides.

> In fact LaTeX itself has a significant collection of additional features, and the collective effort is continuing, with more and more people writing extra packages which are all house in CTAN.

2.1.1 A Small Example

[2] Make sure your TeXStudio is open and all its settings configured which was covered in the previous chapter.

Let us see LaTeX in action by typesetting a short document. We will start with our text editor[2] and type in the lines below **exactly** as shown in the example below:

C.R. 1

latex

```
1 \documentclass{article}
2 \begin{document}
3   This is my \emph{first} document prepared in \LaTeX.
4 \end{document}
```

> Be very careful with the \ character (called the backslash). We use this character to denote something is a **command** and is different from /, called slash.

[3] Instead of `myfile` you can use any name you wish, but be sure to have `.tex` at the end as the extension.

Once you have written the text, save the file as `myfile.tex`.[3] Once you press the run button in your TeXStudio, you will see a PDF output with the following text.

> This is my *first* document prepared in LaTeX.

[4] that is, the file you have typed, which is a `.tex` file

Now let us take a closer look at the source file[4] and see what is going on.

The first line `\documentclass{article}` tells LaTeX that what we want to produce is an article.

If we want to write a book, this must be changed to `\documentclass{book}`. We will look into these a bit later.

The whole document we want to typeset should be included between `\begin{document}` and `\end{document}`. Think of this as a wrapper which contains all the text we **want** to print out to the user.

Any part which is before `\begin{environment}` is called the **preamble**.

Looking back to our example, this is just one (1) line. Now compare this line in the source and the output. The first three words (`This is my`) are produced as typed. Then `\emph{first}`, becomes *first* in the output.[5] Therefore, `\emph` is a **command** to LaTeX to typeset the text within the braces in italic.

[5] It is a common practice to emphasise words in print using italic letters

Following this, the next three words come out without any change in the output. Finally, the input `\LaTeX` comes out in the output as LaTeX. Therefore, as we see, our source is a mixture of text to be typeset and a couple of LaTeX commands `\emph` and `\LaTeX`. The first command changes the input text in a certain way and the second one generates new text.

Now let's look at the file again and add one more sentence given below.

C.R. 2

```latex
This is my \emph{first} document prepared in \LaTeX. I typed it
on \today.
```

What do you get in the output? What new text does the command `\today` generate?

2.1.2 What is the Point of LaTeX

So, why all this trouble and why not simply use a word processor?

The answer lies in the motivation behind LaTeX. Donald Knuth says that his aim in creating LaTeX is to beautifully typeset technical documents especially those containing a lot of Mathematics. It is very difficult to produce complex mathematical formulas using a word processor.[6]

[6] as the author of this document who had to write many technical documents in Word can attest.

If you want your document to look really beautiful then LaTeX is the natural choice.

2.2 Basics of Typesetting

We have seen that, to typeset something in LaTeX, we type in the text to be typeset together with some commands. Words **must** be separated by spaces[7] and lines maybe broken arbitrarily by the language.

[7] does not matter how many.

The end of a paragraph is specified by a blank line in the input. In other words, whenever you want to start a new paragraph, just leave a blank line and proceed. For example, the first two (2) paragraphs above were produced by the input:

C.R. 3 latex

```latex
\noindent We have seen that to typeset something in \LaTeX, we type in the
text to be typeset together with some \LaTeX\ commands.
Words must be separated by spaces (does not matter how many)
and lines maybe broken arbitrarily.

\noindent The end of a paragraph is specified by a \emph{blank line}
in the input. In other words, whenever you want to start a new
paragraph, just leave a blank line and proceed.
```

If you noticed the first line of each paragraph starts without an indentation from the left margin of the text. This is caused by the command `\noindent`. To allow indentation, simply remove the command.

There is an easier way to suppress paragraph indentation for **all** paragraphs of the document in one go, but such tricks can wait.

2.2.1 The Use of Spaces

You might have noticed that even though the length of the lines of text we type in a paragraph are different, in the output, all lines are of equal length, aligned perfectly on the right and left. LaTeX does this by adjusting the space between the words automatically.

In traditional typesetting, a little extra space is added to periods which end sentences and LaTeX also follows this custom. But how does LaTeX know whether a period ends a sentence or not?

It assumes that every period **NOT** following an upper case letter ends a sentence.

But this does not always work, for there are instances where a sentence does end in an upper case letter. For example, consider the following:

Carrots are good for your eyes, since they contain Vitamin A. Have you ever seen a rabbit wearing glasses?

The right input to produce this is

C.R. 4 latex

```latex
Carrots are good for your eyes, since they contain Vitamin A\@. Have
you ever seen a rabbit wearing glasses?
```

We use the command `\@` before the period to produce the extra space after the period.

On the other hand, there are instances where a period following a lowercase letter does **NOT** end a sentence. For example [25]:

> The numbers 1, 2, 3, etc. are called natural numbers. According to Kronecker, they were made by God; all else being the work of Man.

To produce this (without extra space after etc.) the input should be:

C.R. 5
latex
```
The numbers 1, 2, 3, etc.\ are called natural numbers. According to
Kronecker, they were made by God; all else being the works of Man.
```

Here, we use the command \.[8]

[8] that is, a backslash and a **space** here and elsewhere, we sometimes use to denote a space in the input, especially when we draw attention to the space

There are other situations where the command \(which always produce a space in the output) is useful.

For example, type the following line and compile it:

C.R. 6
latex
```
\documentclass{article}

\begin{document}
I think \LaTeX is fun.
\end{document}
```

We get:

> I think LaTeXis fun.

What happened to the **space** we typed between LaTeX and is?

You see, LaTeX gobbles[9] up all spaces after a command. To get the required sequence in the output, change the input as "I think LaTeX is fun".

[9] In this context, gobble means it eats up all the white-space.

Again, the command (\) comes to the rescue.

2.2.2 Quotations

Have you noticed that in typesetting, opening quotes are different from closing quotes?

Look at the LaTeX output below:

C.R. 7
latex
```
Note the difference in right and left quotes in
'single quotes' and ''double quotes''.
```

This is produced by the input

Note the difference in right and left quotes in ‘single quotes’ and “double quotes”.

If your keyboard does not have a left quote key, you can use `\lq` command to produce it. The corresponding command `\rq` produces a right quote.

2.2.3 Dashes

In text, dashes are used for various purposes and they are distinguished in typesetting by their lengths;

- Short dashes are used for hyphens,
- Longer dashes are used to indicate number ranges and still longer dashes used for parenthetical comments.

Let’s have a look at the following TEX output:

X-rays are discussed in pages 221–225 of Volume 3—the volume on electromagnetic waves.

This is produced from the input:

C.R. 8

latex
```
X-rays are discussed in pages 221--225 of
Volume 3---the volume on electromagnetic waves.
```

Please observe that:

- one dash (`-`) produces a hyphen in the output (-),
- two dashes (`--`) produces a longer dash in the output (–),
- three dashes (`---`) produce the longest dash (—) in the output.

2.2.4 Accents

Sometimes, especially when typing foreign words in English, or just writing documents in non-english, we need to put different types of accents over the letters. The table below shows the accents available in LATEX.

Each column shows some of the accents and the inputs to generate them.

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

Table 2.1: Some accent modifier commands.

> The letters i and j need special treatment with regard to accents, since they should not have their customary dots when accented.

The commands \i and \j produce dot-less i and j as ı and ȷ.

> Él está aquí

we need to type:

C.R. 9

```latex
\'{E}l est\'{a} aqu\'{\i}
```

Some symbols from non-English languages are also available in LaTeX, as shown in the table below:

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

Table 2.2: Commands to print out control symbols

2.2.5 Special Symbols

We have seen that input \LaTeX produces LaTeX in the output and \ produces a space. Therefore, TeX uses the symbol \ for a special purpose, as briefly mentioned previously:

> To indicate the program that what follows is **NOT** text to be typeset but an instruction to be carried out.

So what if you want to get \ in your output?[10] Well, the command \ produces \ in the output. As you have notices \ is a symbol which has a special meaning for TeX and cannot be produced by direct input. As another example of such a special symbol, see what is obtained from the input below:

[10] improbable as it may be

C.R. 10
```latex
Maybe I have now learnt about 1% of \LaTeX.
```

You only get:

> Maybe I have now learnt about 1

What happened to the rest of the line? You see, TEX uses the per cent symbol (%) as the comment character; that is a symbol which tells TEX to consider the text following as comments and **NOT** as text to be typeset. This is especially useful for a LATEX programmer to explain a particularly sticky bit of code to others. But then, how do you get
a percent sign in the output? Just type \% as in

C.R. 11
```latex
Maybe I have now learnt about 1\% of \LaTeX.
```

which would give us:

> Maybe I have now learnt about 1% of LATEX.

The symbols \ and % are just two (2) of the ten (10) characters TEX reserves for its internal use. The complete list is:

C.R. 12
```text
~ # $ % ^ & _ \ { }
```

We have seen how LATEX uses two of these symbols and others we will see as we proceed. Also, we have noted that \ is produced in the output by the command \textbackslash and % is produced by \%. What about the other symbols? The table below gives the inputs to produce these symbols.

Output	Command	Output	Command
~	\textasciitilde	&	\&
#	\#	_	_
$	\$	\	\textbackslash
%	\%	{	\{
^	\textasciicircum	}	\}

Table 2.3: Commands to print out additional control symbols.

As we can see from the aforementioned table that except for three (3), all special symbols are produced by preceding them with a \. Of the exceptional three, we have seen that änd äre used for producing accents. So what does \\ do?

> It is used to break lines.

For example,

C.R. 13
latex
```
This is the first line. \\ This is the second line.
```

produces

> This is the first line.
> This is the second line

We can also give an optional argument to \\ to i**increase the vertical distance** between the lines. For example,

C.R. 14
latex
```
This is the first line. \\[10pt]
This is the second line.
```

gives

> This is the first line.
>
> This is the second line

Now there is an extra 10 points of space between the lines.[11]

[11] 1 point (1pt) is about 0.013 of an inch.

2.2.6 Position of Text

We have seen up to now, that LaTeX aligns text in its own way, regardless of the way text is formatted in the input file.[12] Now suppose you want to typeset something like this

[12] To a certain extent LaTeX does **NOT** care about how many spaces there are between words or whether there are numerous spaces between paragraphs.

> MCI Mechatronics
>
> Welcomes You!
>
> This is to certify that you have chosen to become great engineers by the time your degree is finished!!
>
> The Lecturer
> D. T. McGuiness

Which is produced by the following command:

C.R. 15
latex
```
\begin{center}
  MCI Mechatronics\\[.75cm]
  Welcomes You!
\end{center}
\noindent This is to certify that you have chosen to
become great engineers by the time your degree is finished!!
\begin{flushright}
```

C.R. 16 latex
```
  The Lecturer\\
  D. T. McGuiness
\end{flushright}
```

Let's have a look at the comments. Here, the commands:

C.R. 17 latex
```
\begin{center} ... \end{center}
```

typesets the text between them exactly at the center of the page and the commands

C.R. 18 latex
```
\begin{flushright} ... \end{flushright}
```

typesets text flush with the right margin. The corresponding commands

C.R. 19 latex
```
\begin{flushleft} ... \end{flushleft}
```

places the enclosed text flush with the left margin.[13]

[13] Change the `flushright` to `flushleft` and see what happens to the output.

These examples are an illustration of a LaTeX construct called an environment, which is of the form

C.R. 20 latex
```
\begin{name} ... \end{name}
```

where name is the name of the environment. We have seen an example of an environment at the very beginning of this chapter, namely the `document` environment.

2.3 Fonts

[14] collectively called type.

The actual letters and symbols[14] which LaTeX produces are characterised by their style and size. For example, in this lecturebook *emphasized* text is given in *italic* style and the example inputs are given in `typewriter` style. We can also produce smaller and bigger type.

> A set of types of a particular style and size is called a font.

2.3.1 Type Style

In LaTeX, a type style is specified by family, series and shape. They are shown in the **Table ??**
Any type style in the output is a combination of these three (3) characteristics. For example, by default we get roman family, medium series, upright shape type style in a LaTeX output. The `\textit` command produces roman family, medium series, italic shape type. Again, the command `\textbf` produces roman family, boldface series, upright shape type.
We can combine these commands to produce a wide variety of type styles. For example, the input:

C.R. 21 latex
```
\textsf{\textbf{sans serif family, boldface series, upright shape}}
\textrm{\textsl{roman family, medium series, slanted shape}}
```

	Style	Command
Family	roman	`\textrm{roman}`
	sanf serif	`\textsf{sans serif}`
	typewriter	`\texttt{typewriter}`
Series	medium	`\textmd{medium}`
	boldface	`\textbf{boldface}`
Shape	upright	`\textup{upright}`
	italic	`\textit{italic}`
	slanted	`\textsl{slanted}`
	small cap	`\textsc{small cap}`

Table 2.4: Different type-styles supported by LaTeX.

which produces the following output:

sans serif family, boldface series, upright shape
roman family, medium series, slanted shape

It may be possible that some of these type styles may **NOT** be available in your computer. In that case, LATEX gives a warning message on compilation and substitutes another available type style which it thinks is a close approximation to what you had requested.

We can now tell the whole story of the `\emph` command. We have seen that it usually, that is when we are in the middle of normal (upright) text, it produces italic shape. But if the current type shape is slanted or italic, then it switches to upright shape. Also, it uses the family and series of the current font. Thus

C.R. 22
latex
```
\textit{A polygon of three sides is called a \emph{triangle} and a
    polygon of four sides is called a \emph{quadrilateral}}
```

A polygon of three sides is called a triangle *and a polygon of four sides is called a* quadrilateral

whereas the following input:

C.R. 23
latex
```
\textbf{A polygon of three sides is called a
  \emph{triangle} and a polygon of four sides is called a
  \emph{quadrilateral}}
```

produces:

> **A polygon of three sides is called a triangle and a polygon of four sides is called a quadrilateral**

Each of these type style changing commands has an alternate form as a declaration. For example, instead of `\textbf{boldface}` you can also type `{\bfseries boldface}` to get boldface.

> Note that that not only the name of the command, but its usage also is different.

For example, to typeset:

> By a **triangle**, we mean a polygon of three sides.

if we were to write:

C.R. 24

```latex
By a \bfseries{triangle}, we mean a polygon of three sides.
```

we would get:

> By a **triangle, we mean a polygon of three sides.**

[15] and no more

Therefore to make the declaration act upon a specific piece of text,[15] the declaration and the text **should be enclosed in braces**. **Table** 2.5 completes the one given earlier, by giving also the declarations to produce type style changes.

	Style	Command	Declaration
Family	roman	`\textrm{roman}`	`{\rmfamily roman}`
	sanf serif	`\textsf{sans serif}`	`{\sffamily sans serif}`
	typewriter	`\texttt{typewriter}`	`{\ttfamily typewriter}`
Series	medium	`\textmd{medium}`	`{\mdseries medium}`
	boldface	`\textbf{boldface}`	`{\bfseries boldface}`
Shape	upright	`\textup{upright}`	`{\upshape upright}`
	italic	`\textit{italic}`	`{\itshape italic}`
	slanted	`\textsl{slanted}`	`{\slshape slanted}`
	small cap	`\textsc{small cap}`	`{\scshape small cap}`

Table 2.5: Different type-styles supported by LaTeX along with their declaration versions.

These declaration names can also be used as environment names. Therefore to typeset a long passage in, say, sans serif, just enclose the passage within the commands:

C.R. 25

```latex
\begin{sffmily} ... \end{sffamily}
```

2.4 Type Size

Traditionally, type size is measured in (printer) points. The default type that TEX produces is of 10 pt size. There are some declarations (ten, to be precise) provided in LATEX for changing the type size. They are given in **Table** 2.6:

Size	Command	Size	Command
size	`\tiny size`	size	`\large size`
size	`\scriptsize size`	size	`\Large size`
size	`\footnote size`	size	`\LARGE size`
size	`\small size`	size	`\huge size`
size	`\normalsize size`	size	`\Huge size`

Table 2.6: Different type-styles supported by LATEX along with their declaration versions.

Note that the `\normalsize` corresponds to the size we get by default and the sizes form an ordered sequence with `\tiny` producing the smallest and `\Huge` producing the largest. Unlike the style changing commands, there are no command-with-one-argument forms for these declarations.

Chapter 3

The Document

Table of Contents

3.1 The Document Class

It is time to describe how an entire document with chapters and sections and other additional content can be produced with LaTeX. We have seen previously, that all LaTeX files should begin by specifying the kind of document to be produced, using the command `\documentclass{...}`.

We've also noted that for a short article[1] we write `\documentclass{article}`, and for books, we write `\documentclass{book}`. There are, of course, other document classes available in LaTeX such as `report` and `letter`. All of them share some common features and there are features specific to each.

[1] Short can be misleading as the document can actually turn out to be quite long.

In addition to specifying the type of document,[2] we can also specify some options which modify the default format.

[2] which we must do, since LaTeX has no default document class

Therefore the actual syntax of the `\documentclass` command is:

C.R. 1

```latex
\documentclass[options]{class}
```

Note that **options** are given in **square brackets** and not braces.

Options are specified within square brackets (`[...]`), after which mandatory arguments are given within braces (`{...}`).

3.1.1 Font Size

We can select the size of the font for the normal text in the entire document with one of the options:

```
10pt 11pt 12pt
```
C.R. 2 text

Therefore, if we wanted to have a document with size 11pt, we write:

```
\documentclass[11pt]{article}
```
C.R. 3 latex

The default is `10pt` and so this is the size we get, if we do **NOT** specify any font-size option.

3.1.2 Paper Size

We know that LaTeX has its own method of breaking lines to make paragraphs, which we looked previously. It also has methods to make vertical breaks to produce different pages of output. For these breaks to work properly, it must know the **width** and **height** of the paper used. The various options for selecting the paper size are given below:[3]

[3] As you can see, the measurements are given in inches instead of centimetres. This is because the software was designed in America. However, there are packages and document classes for use with European standards, which we will have a look at it later.

Document	Dimensions (in)	Document	Dimensions (in)
`letterpaper`	11 × 8.5	`a4paper`	11 × 8.5
`legalpaper`	14 × 8.5	`a5paper`	11 × 8.5
`executivepaper`	10.5 × 7.25	`b5paper`	11 × 8.5

Table 3.1: A short-list of possible document classes for the standard documentclass.

Normally, the longer dimension is the vertical one-that is, the height of the page. The default is `letterpaper`.

The default option abides by the American standards. To use a European standard, you need to declare `a4paper` in the `documentclass` options such as:
`\documentclass[]{a4paper}`

3.1.3 Page Formats

There are options for setting the contents of each page in a single column or in two (2) columns.[4] This is set by the options

[4] as in most dictionaries and journal publications.

C.R. 4
```
onecolumn twocolumn
```
text

and the default is `onecolumn`.

There is also an option to specify whether the document will be finally printed on just one side of each paper or on both sides. The names of the options are:

C.R. 5
```
oneside twoside
```
text

One of the differences is, with the `twoside` option, page numbers are printed on the right on odd-numbered pages and on the left on even numbered pages, so when these printed back to back, the numbers are always on the outside, for better visibility and conform to typographical traditions.

> LaTeX has no control over the actual printing. It only makes the format for printing.

The default is `oneside` for article, report and letter and twoside for book.

In the `report` and `book` class there is a provision to specify the different chapters. Chapters always begin on a new page, leaving blank space in the previous page, if necessary. With the book class there is the additional restriction that chapters begin only on odd-numbered pages, leaving an entire page blank, if need be.

Such behaviour is controlled by the options,

C.R. 6
```
openany openright
```
text

The default is `openany` for `reportclass` so that chapters begin on "any" new page and `openright` for the book class so that chapters begin only on new right, that is, odd numbered, page.

There is also a provision in LaTeX for formatting the "title" (the name of the document, author(s) and so on) of a document with special typographic consideration. In the article class, this part of the document is printed along with the text following on the first page, while for report and book, a separate title page is printed. These are set by the options:

C.R. 7
```
notitlepage titlepage
```
text

As noted above, the default is `notitlepage` for article and titlepage for report and book. As with the other options, the default behaviour can be overruled by explicitly specifying an option with the document class command.

3.2 Page Style

Having decided on the overall appearance of the document through the `\documentclass` command with its various options, we next see how we can set the style for the individual pages. In LaTeX, each page has a head and foot usually containing such information as the current page number or the current chapter or section. Just what goes where is set by the command:

C.R. 8
```latex
\pagestyle{...}
```

where the mandatory argument can be any one of the following styles:

C.R. 9
```text
plain empty headings myheadings
```

The behaviour pertaining to each option is as follows:

plain
: The page head is empty and the foot contains just the page number, centred with respect to the width of the text. This is the default for the article class if no `\pagestyle` is specified in the preamble.

empty
: Both the head and foot are empty. In particular, no page numbers are printed.

headings
: This is the default for the book class. The foot is empty and the head contains the page number and names of the chapter section or subsection, depending on the document class and its options as given below:

Class	Option	Left Page	Right Page
`book`, `report`	one-sided	-	chapter
	two-sided	chapter	section
`article`	one-sided	-	section
	two-sided	section	subsection

Table 3.2: The behaviour of the `headings` option.

myheadings
: The same as headings, except that the section information in the head are **NOT** predetermined, but to be given explicitly using the commands `\markright` or `\markboth` as described below.

Moreover, we can customise the style for the current page only using the command:

C.R. 10
```latex
\thispagestyle{style}
```

where style is the name of one of the styles above. For example, the page number may be suppressed for the current page alone by the command `\thispagestyle{empty}`. Note that only the printing of the page number is suppressed. The next page will be numbered with the next number and so on.

3.2.1 Heading Declarations

As mentioned previously, in the page style `myheadings`, we have to specify the text to appear on the head of every page. It is done with one of the commands

C.R. 11
```latex
\markboth{left head}{right head}
\markright{right head}
```

where `left head` is the text to appear in the head on left-hand pages and right head is the text to appear on the right-hand pages. The `\markboth` command is used with the `twoside` option with even numbered pages considered to be on the left and odd numbered pages on the right. With `oneside` option, all pages are considered to be right-handed and so in this case, the command `\markright` can be used.

These commands can also be used to override the default head set by the headings style.

These give only a limited control over the head and foot, since the general format, including the font used and the placement of the page number, is fixed by LaTeX. Better customisation of the head and foot are offered by the package **fancyhdr**, which is included in most LaTeX distributions.

3.3 Page Numbering

The style of page numbers can be specified by the command:

C.R. 12
```latex
\pagenumbering{...}
```

The possible arguments to this command and the resulting style of the numbers are given below:

The default value is arabic.

This command resets the page counter. Therefore for example, to number all the pages in the Preface[5] with lowercase Roman numerals and the rest of the document with Indo-Arabic

[5] an introduction to a book, typically stating its subject, scope, or aims.

Command	Description
arabic	Indo-Arabic numerals
roman	lowercase Roman numerals
Roman	upper-case Roman numerals
alph	lowercase English letters
Alph	uppercase English Letters

Table 3.3: Different number styles.

numerals, declare `\pagenumbering{roman}` at the beginning of the Preface and issue the command `\pagestyle{arabic}` immediately after the first `\chapter` command.

We can make the pages start with any number we want by the command

C.R. 13
```latex
\setcounter{page}{number}
```

where number is the page number we wish the current page to have.

3.4 Formatting Lengths

Each page that LaTeX produces consists not only of a head and foot as discussed above but also a body containing the actual text. In formatting a page, LaTeX uses the width and heights of these parts of the page and various other lengths such as the left and right margins. The values of these lengths are set by the paper size options and the page format and style commands.

For example, the page layout with values of these lengths for an odd page and even in this book are separately shown below. These lengths can all be changed with the command `\setlength`. For example,

C.R. 14
```latex
\setlength{\textwidth}{15cm}
```

makes the width of text 15 cm. The package geometry gives easier interfaces to customise page format.

3.5 Parts of a Document

We now turn our attention to the contents of the document itself. Documents are divided into chapters, sections and so on. There may be a title part[6] and an abstract. All these require special

[6] sometimes even a separate title page.

typographic considerations and LaTeX has a number of features which automate this task.

3.5.1 The Title

The "title" part of a document usually consists of the name of the document, the name of author(s) and sometimes a date. To produce a title, we make use of the commands:

C.R. 15

```latex
\title{document name}
\author{author names}
\date{date text}

\maketitle
```

Note that after specifying the arguments of `\title`, `\author` and `\date`, we must issue the command `\maketitle` for this part to be typeset. By default, all entries produced by these commands are centred on the lines in which they appear. If a title text is too long to fit in one line, it will be broken automatically. However, we can choose the break points with the `\\` command.

If there are several authors and their names are separated by the `\and` command, then the names appear side by side. Therefore we can do the following:

C.R. 16

```latex
\title{Title}
\author{Author 1\\
            Address line 11\\
            Address line 12\\
            Address line 13
            \and
            Author 2\\
            Address line 21\\
            Address line 22\\
            Address line 23}
    \date{Month Date, Year}
```

If instead of `\and`, we use (plain old) `\\`, the names are printed one below another. We may leave some of these arguments empty; for example, the command `\date{ }` prints no date. Note, however, that if you simply omit the `\date` command itself, the current date will be printed. The command

C.R. 17

```latex
\thanks{footnote text}
```

can be given at any point within the `\title`, `\author` or `\date`. It puts a marker at this point and places the footnote text as a footnote. (The general method of producing a footnote is to type `\footnote{footnote text}` at the point we want to refer to.) As mentioned earlier, the "title" is printed in a separate page for the document classes book and report and in the first page of the document for the article class.[7]

[7] this behaviour can be modified by the options `titlepage` or `notitlepage`.

3.5.2 Abstract

In the document classes `article` and `report`, an abstract of the document in special format can be produced by the commands

C.R. 18
latex
```
\begin{abstract}
\end{abstract}
```

Note that we have to type the abstract ourselves. In the report class this appears on the separate title page and in the article class it appears below the title information on the first page.

> This command is **NOT** available in the book class.

3.5.3 Dividing the Document

A book is usually divided into chapters and, chapters are divided into sections, sections into subsections and so on. LaTeX provides the following hierarchy of sectioning commands in the book and report class:

C.R. 19
latex
```
\chapter
\section
\subsection
\subsubsection
\paragraph
\subparagraph
```

Except for `\chapter` all these are available in article class also. For example, the heading at the beginning of this chapter was produced by

C.R. 20
latex
```
\chapter{The Document}
```

and the heading of this section was produced by

C.R. 21
latex
```
\section{Dividing the document}
```

To see the other commands in action, suppose at this point of text let's type:

C.R. 22
latex
```
\subsection{Example}
In this example, we show how subsections and subsubsections
are produced (there are no subsubsubsections). Note how the
subsections are numbered.
\subsubsection{Subexample}
Did you note that subsubsections are not numbered? This is so in the
\texttt{book} and \texttt{report} classes. In the \texttt{article}
```

C.R. 23

```latex
class they too have numbers. (Can you figure out why?)
\paragraph{Note}
Paragraphs  and subparagraphs do not have numbers. And they have
\textit{run-in} headings.
Though named paragraph we can have several paragraphs of text
within this.
\subparagraph{Subnote}
Subparagraphs have an additional indentation too.
```

which gives us the following output:

3.5.4 Example

In this example, we show how subsections and subsubsections are produced (there are no subsubsubsections). Note how the subsections are numbered.

Subexample

Did you note that subsubsections are not numbered? This is so in the `book` and `report` classes. In the `article` class they too have numbers. (Can you figure out why?)

Note Paragraphs and subparagraphs do not have numbers. And they have *run-in* headings. Though named "paragraph" we can have several paragraphs of text within this.

Subnote Subparagraphs have an additional indentation too.

3.5.5 Additional Information

In the `book` and the `report` classes, the `\chapter` command shifts to the beginning of a new page and prints the word Chapter and a number and beneath it, the name we have given in the argument of the command. The `\section` command produces two (2) numbers[8] indicating the chapter number and the section number followed by the name we have given.

[8] separated by a dot (i.e., Chapter.2)

It does not produce any text like "Section".

Subsections have three (3) numbers indicating

the chapter, section and subsection.

`\subsubsections` and commands below it in the hierarchy do **NOT** have any numbers. In the article class, `\section` is highest in the hierarchy and produces single number like `\chapter` in book.[9] In this case, subsubsections also have numbers, but none below have numbers.

[9] It does not produce any text like "Section", though.

Each sectioning command also has a "starred" version which does not produce numbers. Thus `\section*{name}` has the same effect as `\section{name}`, but produces no number for this section.

Some books and longish documents are divided into parts also. LaTeX also has a `\part` command for such documents. In such cases, `\part` is the highest in the hierarchy, but it does not affect the numbering of the lesser sectioning commands. You may have noted that LaTeX has a specific format for typesetting the section headings, such as the font used, the positioning, the vertical space before and after the heading and so on. All these can be customised but require knowledge on LaTeX.

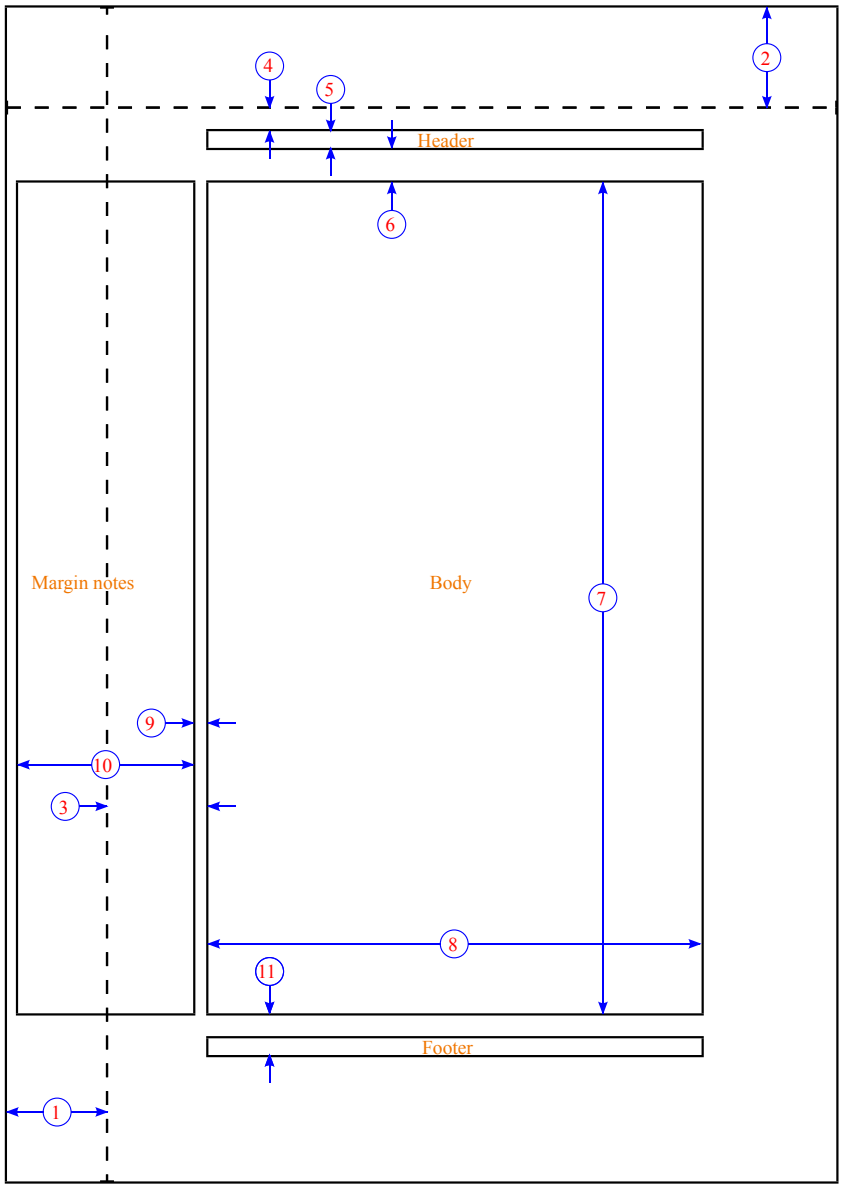


Figure 3.1: Page dimensions of a book class.

Chapter 4

Displaying Content

Table of Contents

4.1 Borrowing Text

There are many instances in a document when we want to visually separate a portion of text from its surrounding material. One method of doing this is to typeset the distinguished text with added indentation.

It is called displaying.

LaTeX has various constructs for displaying text depending the nature of the displayed text. We will start with quotations.

Quotations are often used in a document, either to add weight to our arguments by referring to a higher authority or because we find that we cannot improve on the way an idea has been expressed by someone else. If the quote is a one-liner, we can simply include it within double-quotes and be done with it. But if the quotation is several lines long, it is better to display it. Look at the following example:

C.R. 1

latex

```
Some mathematicians elevate the spirit of Mathematics  to a kind of
intellectual aesthetics. It is best voiced  by Bertrand Russell in the
following lines.
\begin{quote}
  The true spirit of delight, the exaltation, the sense of being more than man,
```

C.R. 2
```latex
  which is the touchstone of the highest excellence, is to be found in Mathematics
  as surely as in poetry.. . . Real life is, to most men, a long second best, a
  perpetual compro- mise between the ideal and the possible; but the world of pure
  reason knows no compromise, no practical limitations, no barriers to the creative
  activity embody- ing in splendid edifices the passionate aspiration after the
  perfect, from which all great work springs.
\end{quote}
```

Note that here we give instructions to TEX to typeset some material in a separate paragraph with additional indentation on either side and indicate the start and end of material requiring special treatment, by means of the commands

C.R. 3
```latex
\begin{quote} ... \end{quote}
```

This is an example of what is known in LATEX as an environment. Environments are used to delimit passages requiring special typographic treatments and to give instructions to LATEX on how to typeset it. The general form of an environment is of the form

C.R. 4
```latex
\begin{name} ... \end{name}
```

where name is the name of the environment and signifies to LATEX the type of typographic treatment required. The quoted part in this example is a single paragraph. If the quotation runs into several paragraphs, we must use the quotation environment, by enclosing the quotation within `\begin{quotation}` and `\end{quotation}`.

> As usual, paragraphs are separated by blank lines while typing the source file.

4.2 Making Lists

Lists are needed to keep some semblance of order in a chaotic world and LATEX helps us to typeset them nicely. Also, there are different kinds of lists available by default and if none of them suits your need, there are facilities to tweak these or even design your own. Let us first have a look at the types of lists LATEX provides.

4.2.1 Using Bullet Lists

The itemize environment gives us a bullet-list. For example it produces something like this:

One should keep the following in mind when using TEX

- TEX is a typesetting language and not a word processor

- TEX is a program and and not an application
- There is no meaning in comparing TEX to a word processor, since the design purposes are different

C.R. 5
latex
```
One should keep the following in mind when using \TeX
\begin{itemize}
\item \TeX\ is a typesetting language and not a word processor
\item \TeX\ is a program and and not an application
\item There is no meaning in comparing \TeX\ to a word processor, since the design
  purposes are different
\end{itemize}
```

The `\begin{itemize} ... \end{itemize}` pair signifies we want a bullet-list of the enclosed material. Each item of the list is specified by an `\item` command. We can have lists within lists. For example:

One should keep the following in mind when using TEX

- TEX is a typesetting language and not a word processor
- TEX is a program and and not an application
- Theres is no meaning in comparing TEX to a word processor, since the design purposes are different
- TEX is the natural choice in one of these situations
  - If we want to typeset a document containing lot of Mathematics
  - If we want our typed document to look beautiful

Being a program, TEX offers a high degree of flexibility.

C.R. 6
latex
```
One should keep the following in mind when using \TeX
\begin{itemize}
    \item \TeX\ is a typesetting language and not a  word processor
    \item \TeX\ is a program and and not an application
    \item Theres is no meaning in comparing \TeX\ to a word processor, since the design
          purposes are different
    \item \TeX\ is the natural choice in one of these situations
          \begin{itemize}
              \item If we want to typeset a document containing lot of Mathematics
              \item If we want our typed document to look beautiful
          \end{itemize}
\end{itemize}
Being a program, \TeX\ offers a high degree of flexibility.
```

The itemize environment supports four (4) levels of nesting. The full list of labels for the items[1] is as shown below

[1] 'bullets' for the first level, 'dashes' for the second and so on

- The first item in the first level
- the second item in the first level
  - The first item in the second level
  - the second item in the second level
    - The first item in the third level
    - the second item in the third level
      - The first item in the fourth level
      - the second item in the fourth level

The source of this is as follows:

C.R. 7 latex

```latex
\begin{itemize}
\item The first item in the first level
\item the second item in the first level
  \begin{itemize}
  \item The first item in the second  level
  \item the second item in the second level
    \begin{itemize}
    \item The first item in the third  level
    \item the second item in the third level
      \begin{itemize}
      \item The first item in the fourth  level
      \item the second item in the fourth level
      \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}
```

4.2.2 Ordered Lists

When the order of the items in a list is important, we need a list which specifies this order. For example, consider this

1. prepare a source file with the extension "tex"
2. Compile it with LaTeXto produce a "dvi" file
3. Print the document using a "dvi" driver

Such a numbered list is produced by the `enumerate` environment in LaTeX. The above list was produced by the following source.

C.R. 8
latex
```
\begin{enumerate}
\item prepare a source file with the extension "tex"
\item Compile it with \LaTeX to produce a "dvi" file
\item Print the document using a "dvi" driver
\end{enumerate}
```

As in the case of itemize environment, here also four levels of nesting are supports. The example below shows the labels used for different levels.

1. The first item in the first level
2. the second item in the first level
    a) The first item in the second level
    b) the second item in the second level
        i. The first item in the third level
        ii. the second item in the third level
            A. The first item in the fourth level
            B. the second item in the fourth level

How about customizing the labels? Here there is an additional complication in that the labels for items in the same level must follow a sequence (such as 1,2,3, ... for the first level, (a), (b), (c), ... for the second and so on, by default). There is a method for doing it, but it will take us into somewhat deeper waters. Fortunately, there is a package **enumitem** which makes it easy. So if we want:

The three basic steps in producing a printed document using LaTeX are as follows:

Step 1. Prepare a source file with the extension "tex"

Step 2. Compile it with LaTeXto produce a "dvi" file

i. Use a previewer (such as "xdvi" on X Window System) to view the output

ii. Edit the source if needed

iii. Recompile

Step 3. Print the document using a "dvi" driver (such as "dvips")

which is typed as

C.R. 9
latex
```
The three basic steps in  producing a printed document
using \LaTeX\ are as follows:
\begin{enumerate}[label=Step \arabic*.]
\item Prepare a source file with the extension "tex"
\item Compile it with \LaTeX to produce a "dvi" file
  \begin{enumerate}[label=\roman*.]
  \item Use a previewer (such as "xdvi" on
```

C.R. 10 latex

```
    \textsf{X Window System}) to view the output
  \item Edit the source if needed
  \item Recompile
  \end{enumerate}
\item Print the document using a "dvi" driver
  (such as "dvips")
\end{enumerate}
```

4.2.3 Description Lists

There is a third type of list available off-the-shelf in LaTeX which is used in typesetting lists like this

Let us take stock of what we have learnt

TeX A typesetting program

Emacs A text editor and also

a programming environment

a mailer

and a lot else besides

AbiWord A word processor

which is typed as:

C.R. 11 latex

```
Let us take stock of what we have learnt
\begin{description}
\item[\TeX] A typesetting program
\item[Emacs] A text editor and also
  \begin{description}
  \item a programming environment
  \item a mailer
  \item and a lot else besides
  \end{description}
\item[AbiWord] A word processor
\end{description}
```

Note that this environment does not produce on its own any labels for the various items, but only produces as labels, whatever we give inside square brackets immediately after each `\item`. By default, the labels are typeset in boldface. Also, there is no indentation for the first level.

Chapter 5

Typesetting Mathematics

Table of Contents

5.1 True Purpose of TEX

Donald Knuth created TEX for primarily typesetting Mathematics beautifully [6]. LATEX includes all the capabilities of TEX in Mathematics typesetting, with an easier interface. Of course, LATEX, being a full language, supports extension to the language, which is why there are many packages in CTAN.

Packages like **amsmath** are worth mentioning here as they enhance and refine these interfaces significantly.

5.2 Fundamentals

A mathematical expression occurring in running text, called in-text math, is produced by enclosing it between dollar signs `$...$`. Therefore if we want to produce:

> The equation representing a straight line in Cartesian plane is of the form $ax + by + c = 0$, where a, b, c are constants.

we need to type:

C.R. 1

```latex
The equation representing a straight line in Cartesian plane
is of the form $ax + by + c = 0$, where $a$, $b$, $c$ are constants.
```

Let's see what is going on here.

[1] To be a bit pedantic, it is called *math italic*.

First, note the text within dollars is typeset in italic.[1] Again, even though we did not leave any spaces within $ax + by + c = 0$, TEX leaves spaces on either side of the addition signs ($+$) and the equality ($=$) sign.

On the other hand, even if we type `$ax + by + c = 0$`, the output **would be the same**

$$ax + by + c = 0$$

> TEX has its own spacing rules in math mode and knows to disregard user-given space.

To see another instance of this, change the last part of the code above to read

C.R. 2

```latex
... where $a, b, c$ are constants.
```

On first glance, it saves some typing. However, look at the output:

> The equation representing a straight line in Cartesian plane is of the form $ax + by + c = 0$, where a, b, c are constants.

See the difference?

There are **no spaces** after the commas, though we had such spaces in the output. So TEX swallows spaces in math mode.

As we have seen in these examples, dollar signs (`$..$`) are TEX way of distinguishing mathematical text. LATEX has other ways also of doing it,

- using `\( ... \)`
- or `\begin{math} ... \end{math}`.

Therefore either of the inputs shown below also produces the same output as above.

C.R. 3 latex

```
The equation representing a straight line in the Cartesian plane is of
the form \(ax+by+c=0\), where \(a\), \(b\), \(c\) are constants.
```

C.R. 4 latex

```
The equation representing a straight line in the Cartesian plane is
of the form \begin{math}ax+by+c=0\end{math}, where \begin{math} a
\end{math}, \begin{math} b \end{math}, \begin{math} c \end{math} are
constants.
```

Continuing on, suppose we want to display the equation[2] in the previous output as in

[2] In this context, display means to put the equation in centre-stage and not inline.

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0$$
>
> where a, b, $c$are constants.

This can be done by changing the input as follows:

C.R. 5 latex

```
The equation representing a straight line in the Cartesian plane is
of the form
$$
\begin{equation}
ax+by+c=0
\end{equation}
$$
where $a$, $b$, $c$ are constants.
```

Again `$$...$$` is the TEX way of producing displayed math. LATEX has the additional constructs `\[ ... \]` or `\begin{displaymath} ... \end{displaymath}` also to do this.

5.2.1 Superscripts and Subscripts

Now let's look at the text below:

> In the seventeenth century, Fermat conjectured that if $n > 2$, then there are no integers x, y, z for which
>
> $$x^n + y^n = z^n \tag{5.1}$$
>
> This was proved in 1994 by Andrew Wiles.

This is produced by the input

C.R. 6 latex

```
In the seventeenth century, Fermat conjectured that
if $n > 2$, then there are no integers $x$, $y$, $z$
for which
```

C.R. 7 latex

```
%
$$
  x^n + y^n = z^n
$$
%
This was proved in 1994 by Andrew Wiles.
```

[3] mathematicians call them exponents

This shows that superscripts[3] are produced by the ^ symbol. If the superscript is more than one character long, be careful to group these characters properly.

Therefore to produce:

> It is easily seen that $(x^m)^n = x^{mn}$.

we must type:

C.R. 8 latex

```
It is easily seen that $(x^m)^n=x^{mn}$.
```

> Instead of `$x^{mn}$`, if we type `$x^mn$` we end up with $x^m n$ instead of the intended x^{mn} in the output.

[4] and mathematicians do need them for their interesting theorems and whatnot.

We can have superscripts of superscripts.[4] For example let's look at the following

> Numbers of the form $2^{2^n}+1$, where n is a natural number, are called Fermat numbers.

is produced by:

C.R. 9 latex

```
Numbers of the form $2^{2^n}+1$, where $n$ is
a natural number, are called Fermat numbers.
```

> Please observe that we have grouped the superscript.

[5] a character (letter, number, or symbol) that is printed or written below the normal line of text, often slightly smaller than the surrounding text.

Now let us see how subscripts[5] are produced.

To get the following output:

> The sequence (x_n) defined by
>
> $$x_1 = 1, \quad x_2 = 1, \quad x_n = x_{n-1} + x_{n-2} \quad (n > 2)$$
>
> is called the Fibonacci sequence.

we need to type:

C.R. 10 latex

```latex
The sequence $(x_n)$ defined by
$$
x_1=1,\quad x_2=1,\quad x_n=x_{n-1}+x_{n-2}\;\;(n>2)
$$
is called the Fibonacci sequence.
```

We can see, subscripts are produced by the _ character. Note how we insert spaces by the \quad command.[6] Subscripts of subscripts can be produced as in the case of superscripts with appropriate grouping.

[6] The command \; in math mode produces what is known as a "thickspace".

We can also have superscripts and subscripts together. For example:

> If the sequence (x_n) converges to a, then the sequence (x_n^2) converges to a^2

is produced by

C.R. 11 latex

```latex
If the sequence $(x_n)$ converges to $a$, then the sequence
  $(x_n^2)$ converges to $a^2$
```

Again, we must be careful about the grouping (or the lack of it) when typesetting superscripts and subscripts together. The following inputs and the corresponding outputs showcases the problems one may encounter:

C.R. 12 latex

```latex
$$
x_m^n\qquad x^n_m\qquad {x_m}^n\qquad {x^n}_m
$$
```

> $$x_m^n \qquad x_m^n \qquad {x_m}^n \qquad {x^n}_m$$

Information — **Character Boxes**

This has to do with the way TEX internally works to produce characters, producing "boxes" to fit the output characters. The box for x_m^n can be taught as $\boxed{x_m^n}$, whereas the box for ${x_n}^m$ can be visualised as $\boxed{{\boxed{x_m}}^n}$

5.2.2 Roots

To put it simply, square-roots are produced by the \sqrt argument. Therefore $\sqrt{2}$ produces $\sqrt{2}$. This command has an optional argument to produce other roots. As an example:

> Which is greater $\sqrt[4]{5}$ or $\sqrt[5]{4}$?

is produced by

C.R. 13
latex
```
Which is greater $\sqrt[4]{5}$ or $\sqrt[5]{4}$?
```

The horizontal line above the root[7] elongates to accommodate the enclosed text. For example, `$\sqrt{x+y}$` produces $\sqrt{x+y}$. Also, you can produce nested roots as in

[7] called *vinculum* by mathematicians who like to think they are better than engineers.

The sequence

$$2\sqrt{2}\,,\quad 2^2\sqrt{2-\sqrt{2}}\,,\quad 2^3\sqrt{2-\sqrt{2+\sqrt{2}}}\,,\quad 2^4\sqrt{2-\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}\,,\ \ldots$$

converge to π.

by typing

C.R. 14
latex
```
The sequence
$$
2\sqrt{2}\,,\quad 2^2\sqrt{2-\sqrt{2}}\,,\quad 2^3
\sqrt{2-\sqrt{2+\sqrt{2}}}\,,\quad 2^4\sqrt{2-
  \sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}\,,\;\ldots
$$
converge to $\pi$.
```

The `\ldots` command above produces ..., the three dots indicating indefinite continuation, called ellipsis (more about them later). The command `\,` produces a "thinspace"[8]

[8] as opposed to a thickspace produced by `\;` , seen earlier.

> Why all this thin and thick spaces in the above input? Remove them and see the difference. A tastefully applied thinspace is what makes a mathematical expression typeset in TEX really beautiful.

The symbol π in the output produced by `$\pi$`. It is a Greek letter named "pi". Mathematicians often use letters of the Greek alphabet and a multitude of other symbols in their work. A list of available symbols in LATEX is given at the end of this chapter.

> Note that certain symbols are marked to be not available in vanilla LATEX, but only in certain packages.

5.2.3 Symbols in Mathematics

We have noted that TEX leaves some additional spaces around "binary operators" such as + and . The same is true for any symbol classified as a binary operator.[9] For example, consider the following

[9] an operator that operates on two operands. An operand is a value or a variable on which an operator performs an operation. The term "binary" indicates that these operators specifically take two operands.

For real numbers x and y, define an operation $\circ$ by

$$x \circ y = x + y - xy$$

This operation is associative.

C.R. 15
latex
```
For real numbers $x$ and $y$, define an operation $\circ$ by
$$
x\circ y = x+y-xy
$$
This operation is associative.
```

Observe there are spaces surrounding the $\circ$ symbol in the output. On the other hand suppose we want the following output:

For real numbers x and y, define an operation $\Box$ by

$$x \Box y = x^2 + y^2$$

The list of symbols show that the symbol $\Box$ is produced by \Box but that it is available only in the package **latexsym** or **amssymb**. So if we load one of these using the \usepackage command and then type:

C.R. 16
latex
```
For real numbers $x$ and $y$, define an operation $\Box$ by
$$
x\Box y = x^2+y^2
$$
```

you will only get

For real numbers x and y, define an operation $\Box$ by

$$x \Box y = x^2 + y^2$$

Notice the difference? There are **NO** spaces around $\Box$. This is because, this symbol is **NOT** by default defined as a binary operator. But we can ask TEX to consider this symbol as a binary operator by the command \mathbin before \Box as in:

C.R. 17
latex
```
For real numbers $x$ and $y$, define an operation $\Box$ by
$$
x\mathbin\Box y=x^2+y^2
$$
```

For real numbers x and y, define an operation $\square$ by

$$x \square y = x^2 + y^2$$

and this will produce the output shown first. This holds for Relations as well.

TEX leaves some space around "Relation" symbols and we can instruct TEX to consider any symbol as a relation by the command `\mathrel`. Therefore we can produce

Define the relation ρ on the set of real numbers by $x \rho y$ iff $x - y$ is a rational number.

by typing

C.R. 18
```latex
Define the relation $\rho$ on the set of real numbers by
$x\mathrel\rho y$ iff $x-y$ is a rational number.
```

5.3 Custom Commands

[10] along with other stuff

We have seen that LATEX produces mathematics[10] by means of "commands". The interesting thing is that we can build our own commands using the ones available.

For example, suppose that the following expression $(x_1, x_2, \ldots, x_n)$ occurs too frequently in a document. If we now write:

C.R. 19
```latex
\newcommand{\vect}{(x_1,x_2,\dots,x_n)}
```

Then we can type `$\vect$` anywhere after wards to produce $(x_1, x_2, \ldots, x_n)$ as in

C.R. 20
```latex
We often write $x$ to denote the vector $\vect$.
```

to get

We often write x to denote the vector $(x_1, x_2, \ldots, x_n)$.

[11] The place before `\begin{environment}`

The best place to keep such "newcommands" is the preamble,[11], so that we can use them anywhere in the document. Also, it will be easier to change the commands, if the need arises.

OK, we can now produce $(x_1, x_2, \ldots, x_n)$ with `$\vect$` vects, but how about $(y_1, y_2, \ldots, y_n)$ or $(z_1, z_2, \ldots, z_n)$? Do we have to define new commands for each of these? No, as we can also define commands with variable arguments as well. Therefore, if we change our definition of `\vect` to:

C.R. 21
```latex
\newcommand{\vect}[1]{(#1_1,#1_2,\dots,#1_n)}
```

Then we can use `$\vect{x}$` to produce $(x_1, x_2, \dots, x_n)$ and `$\vect{x}$` to produce $(a_1, a_2, \dots, a_n)$ and so on.

The form of this definition calls for some elaboration. The `[1]` in the `\newcommand` above indicates that the command is to have one (1) (variable) argument. What about the `##1`? Before producing the output, each occurrence of `##1` will be replaced by the single argument we supply to `\vect` in the input.

For example, the input `$\vect{a}$` will be changed to `$(a_1,a_2,\dots,a_n)$` at some stage of the compilation.

We can also define commands with more than one argument.[12] Therefore, for example, if the document contains not only $(x_1, x_2, \dots, x_n)$, $(y_1, y_2, \dots, y_n)$ and so on, but $(x_1, x_2, \dots, x_m)$, $(y_1, y_2, \dots, y_p)$ also, then we can change our definition of `\vect` to

[12] the maximum number is 9.

C.R. 22
```latex
\newcommand{\vect}[2]{(#1_1,#1_2,\dotsc,#1_#2)}
```

so that we can use `$\vect{x}{n}$` to produce $(x_1, x_2, \dots, x_n)$ and `$\vect{y}{p}$` to produce $(y_1, y_2, \dots, y_p)$.

5.4 Additional Math

There are some many other features of typesetting math in LaTeX, but these have better implementations in the package **amsmath** which has some additional features as well. So, for the rest of the chapter the discussion will be with reference to this package and its derivatives.

> All discussion below is under the assumption that the package **amsmath** has been loaded with the command `\usepackage{amsmath}`.

5.4.1 Single Equations

In addition to the LaTeX commands for displaying math as discussed earlier, the **amsmath** also provides the `\begin{equation*} ... \end{equation*}` construct. Thus with this package loaded, the output

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0$$

where a, b, c are constants.

can also be produced by

C.R. 23 latex

```
The equation representing a straight line in the Cartesian plane is
of the form
\begin{equation*}
  ax+by+c=0
\end{equation*}
where $a$, $b$, $c$ are constants.
```

Why the * after equation? Suppose we try it without the * as:

C.R. 24 latex

```
The equation representing a straight line in the Cartesian plane is of the form
\begin{equation}
  ax+by+c=0
\end{equation}
where $a$, $b$, $c$ are constants.
```

we get:

The equation representing a straight line in the Cartesian plane is of the form

$$ax + by + c = 0 \tag{5.2}$$

where a, b, c are constants.

This provides the equation with a number. We will discuss equation numbering in some more detail later on. For the time being, we just note that for any environment name with a star we discuss here, the unstarred version provides the output with numbers.

Ordinary text can be inserted inside an equation using the `\text` command. Therefore we can get:

Thus for all real numbers x we have

$$x \le |x| \quad \text{and} \quad x \ge |x|$$

and so

$$x \le |x| \quad \text{for all } x \text{ in } R.$$

from:

C.R. 25 latex

```
Thus for all real numbers $x$ we have
\begin{equation*}
  x\le|x|\quad\text{and}\quad x\ge|x|
\end{equation*}
```

C.R. 26

```latex
and so
\begin{equation*}
  x\le|x|\quad\text{for all $x$ in $R$}.
\end{equation*}
```

Note the use of dollar signs in the second `\text` above to produce mathematical symbols within `\text`. Sometimes a single equation maybe too long to fit into one line.[13] Look at the one below:

[13] or sometimes even two lines.

$$(a+b+c+d+e)^2 = a^2+b^2+c^2+d^2+e^2 + 2ab+2ac+2ad+2ae+2bc+2bd+2be+2cd+2ce+2de$$

This is produced by the environment `multline*`;[14] as shown below.

[14] Note the spelling carefully. It is **NOT** multiline.

C.R. 27

```latex
\begin{multline*}
  (a+b+c+d+e)^2=a^2+b^2+c^2+d^2+e^2\\
  +2ab+2ac+2ad+2ae+2bc+2bd+2be+2cd+2ce+2de
\end{multline*}
```

`multline` can be used for equations requiring more than two (2) lines, but without tweaking, the results are not very satisfactory. For example, the input:

C.R. 28

```latex
\begin{multline*}
  (a+b+c+d+e+f)^2=a^2+b^2+c^2+d^2+e^2+f^2\\
  +2ab+2ac+2ad+2ae+2af\\
  +2bc+2bd+2be+2bf\\
  +2cd+2ce+2cf\\
  +2de+2df\\
  +2ef
\end{multline*}
```

which produces:

$$(a+b+c+d+e+f)^2 = a^2+b^2+c^2+d^2+e^2+f^2 + 2ab+2ac+2ad+2ae+2af + 2bc+2bd+2be+2bf + 2cd+2ce+2cf + 2de+2df + 2ef$$

By default, the `multline` environment places the first line flush left, the last line flush right and the lines in between, centered within the display. A better way to typeset the above multiline (not

multline) equation is as follows.

$$\begin{aligned}(a+b+c+d+e+f)^2 = a^2+b^2+c^2+d^2+e^2+f^2 \\ +2ab+2ac+2ad+2ae+2af \\ +2bc+2bd+2be+2bf \\ +2cd+2ce+2cf \\ +2de+2df \\ +2ef\end{aligned}$$

This is done using the split environment as shown below.

C.R. 29 latex

```
\begin{equation*}
  \begin{split}
    (a+b+c+d+e+f)^2 & = a^2+b^2+c^2+d^2+e^2+f^2\\
                    &\quad +2ab+2ac+2ad+2ae+2af\\
                    &\quad +2bc+2bd+2be+2bf\\
                    &\quad +2cd+2ce+2cf\\
                    &\quad +2de+2df\\
                    &\quad +2ef
  \end{split}
\end{equation*}
```

Let's understand what is going on here. First note that the `split` environment cannot be used independently, but only inside some equation structure such as equation. Unlike `multline`, the `split` environment provides for alignment among the "split" lines (using the & character, as in tabular). Therefore in the above example, all the + signs are aligned and these in turn are aligned with a point a to the right of the = sign. It is also useful when the equation contains multiple equalities as in

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b) \\ &= a^2+ab+ba+b^2 \\ &= a^2+2ab+b^2\end{aligned}$$

which is produced by

C.R. 30 latex

```
\begin{equation*}
    \begin{split}
      (a+b)^2 & = (a+b)(a+b)\\
              & = a^2+ab+ba+b^2\\
              & = a^2+2ab+b^2
    \end{split}
  \end{equation*}
```

5.4.2 Groups of Equations

A group of displayed equations can be typeset in a single go using the `gather` environment. For example,

$$(a,b)+(c,d)=(a+c,b+d)$$
$$(a,b)(c,d)=(ac-bd,ad+bc)$$

can be produced by

C.R. 31

```latex
\begin{gather*}
  (a,b)+(c,d)=(a+c,b+d)\\
  (a,b)(c,d)=(ac-bd,ad+bc)
\end{gather*}
```

Now when several equations are to be considered one unit, the logically correct way of typesetting them is with some alignment. For example,

Thus x, y and z satisfy the equations

$$x+y-z=1$$
$$x-y+z=1$$

This is obtained by using the `align*` environment as shown below

C.R. 32

```latex
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z &= 1\\
  x-y+z &= 1
\end{align*}
```

We can add a short piece of text between the equations, without disturbing the alignment, using the `\intertext` command. For example, the output

Thus x, y and z satisfy the equations

$$x+y-z=1$$
$$x-y+z=1$$

and by hypothesis

$$x+y+z=1$$

C.R. 33 latex

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z & = 1\\
  x-y+z & = 1\\
  \intertext{and by hypothesis}
  x+y+z & =1
\end{align*}
```

We can also set multiple columns of aligned equations side by side as in:

Compare the following sets of equations

$$\cos^2 x + \sin^2 x = 1 \qquad \cosh^2 x - \sinh^2 x = 1$$
$$\cos^2 x - \sin^2 x = \cos 2x \qquad \cosh^2 x + \sinh^2 x = \cosh 2x$$

All that it needs are extra &'s to separate the columns as can be sen from the input Compare the following sets of equations

C.R. 34 latex

```
Compare the following sets of equations
\begin{align*}
  \cos^2x+\sin^2x & = 1        & \cosh^2x-\sinh^2x & = 1\\
  \cos^2x-\sin^2x & = \cos 2x & \cosh^2x+\sinh^2x & = \cosh 2x
\end{align*}
```

We can also adjust the horizontal space between the equation columns. For example, Compare the sets of equations

C.R. 35 latex

```
Compare the sets of equations
\begin{align*}
  \cos^2x+\sin^2x & = 1        &\qquad \cosh^2x-\sinh^2x & = 1\\
  \cos^2x-\sin^2x & = \cos 2x &\qquad \cosh^2x+\sinh^2x & = \cosh 2x
\end{align*}
```

gives

Compare the sets of equations

$$\cos^2 x + \sin^2 x = 1 \qquad \cosh^2 x - \sinh^2 x = 1$$
$$\cos^2 x - \sin^2 x = \cos 2x \qquad \cosh^2 x + \sinh^2 x = \cosh 2x$$

Perhaps a nicer way of typesetting the above is

Compare the following sets of equations

$$\begin{aligned} \cos^2 x + sin^2 x &= 1 \\ \cos^2 x - \sin^2 x &= \cos 2x \end{aligned} \quad \text{and} \quad \begin{aligned} \cosh^2 x - \sinh^2 x &= 1 \\ \cosh^2 x + \sinh^2 x &= \cosh 2x \end{aligned}$$

This cannot be produced by the equation structures discussed so far, because any of these environments takes up the entire width of the text for its display, so that we cannot put anything else on the same line. So **amsmath** provides variants `gathered`, `aligned` and `alignedat` which take up only the actual width of the contents for their display. Thus the above example is produced by the input

C.R. 36

latex

```
Compare the following sets of equations
\begin{equation*}
  \begin{aligned}
    \cos^2x+sin^2x  & = 1\\
    \cos^2x-\sin^2x & = \cos 2x
  \end{aligned}
  \qquad\text{and}\qquad
  \begin{aligned}
    \cosh^2x-\sinh^2x & = 1\\
    \cosh^2x+\sinh^2x & = \cosh 2x
  \end{aligned}
\end{equation*}
```

Another often recurring structure in mathematics is a display like this

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x \leq 0 \end{cases}$$

There is a special environment cases in **amsmath** to take care of these. The above example is in fact produced by

C.R. 37

latex

```
\begin{equation*}
  |x| =
  \begin{cases}
    x & \text{if $x\ge 0$}\\
    -x & \text{if $x\le 0$}
  \end{cases}
\end{equation*}
```

5.4.3 Numbered Equations

We have mentioned that each of the the starred equation environments has a corresponding unstarred version, which also produces numbers for their displays. Therefore our very first example of displayed equations with equation instead of `equation*` as in

C.R. 38 latex

```
The equation representing a straight line in the Cartesian plane is
of the form
\begin{equation}
  ax+by+c=0
\end{equation}
where $a$, $b$, $c$ are constants.
```

produces

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0 \tag{5.3}$$
>
> where a, b, c are constants.

Why VIII.2 for the equation number? Well, this is Equation number 2 of Chap- ter VIII, isn't it? If you want the section number also in the equation number, just give the command

C.R. 39 latex

```
\numberwithin{equation}{section}
```

We can also override the number L'IFX produces with one of our own design with the `\tag` command as in

C.R. 40 latex

```
The equation representing a straight line in the Cartesian plane is
of the form
\begin{equation}
  ax+by+c=0\tag{L}
\end{equation}
where $a$, $b$, $c$ are constants.
```

which gives

> The equation representing a straight line in the Cartesian plane is of the form
>
> $$ax + by + c = 0 \tag{L}$$
>
> where a, b, c are constants.

There is also a `\tag*` command which typesets the equation label without parentheses. What about numbering alignment structures? Except for split and aligned, all other alignment structures have unstarred forms which attach numbers to each aligned equation. For example,

C.R. 41 latex

```
\begin{align}
  x+y-z &= 1\\
  x-y+z &= 1
\end{align}
```

gives:

$$x + y - z = 1 \tag{5.4}$$
$$x - y + z = 1 \tag{5.5}$$

Here is also, you can give a label of your own to any of the equations with the \tag command. Be careful to give the \tag before the end of line character \\ though. (See what happens if you give a \tag command after a \\.) You can also suppress the label for any equation with the \notag command. These are illustrated in the sample input below:

C.R. 42

latex

```
Thus $x$, $y$ and $z$ satisfy the equations
\begin{align*}
  x+y-z & = 1\ntag\\
  x-y+z & = 1\notag\\
  \intertext{and by hypothesis}
  x+y+z & =1\tag{H}
\end{align*}
```

which gives the following output

Thus x, y and z satisfy the equations

$$x + y - z = 1$$
$$x - y + z = 1$$

and by hypothesis

$$x + y + z = 1 \tag{H}$$

What about split and aligned? As we have seen, these can be used only within some other equation structure. The numbering or the lack of it is determined by this parent structure. Thus

C.R. 43

latex

```
\begin{equation}
  \begin{split}
    (a+b)^2 & = (a+b)(a+b)\\
            & = a^2+ab+ba+b^2\\
            & = a^2+2ab+b^2
  \end{split}
\end{equation}
```

$$\begin{aligned}(a+b)^2 &= (a+b)(a+b)\\ &= a^2+ab+ba+b^2\\ &= a^2+2ab+b^2\end{aligned} \tag{5.6}$$

5.5 Additional Commands

There are more things Mathematics than just equations. Let us look at how LaTeX and in particular, the amsmath package deals with them.

5.5.1 Matrices

VIII.4.1. Matrices Matrices are by definition numbers or mathematical expressions arranged in rows and columns. The amsmath has several environments for producing such arrays. For example

The system of equations

$$\begin{aligned}x+y-z &= 1\\ x-y+z &= 1\\ x+y+z &= 1\end{aligned}$$

can be written in matrix terms as

$$\begin{pmatrix}1 & 1 & -1\\ 1 & -1 & 1\\ 1 & 1 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ z\end{pmatrix} = \begin{pmatrix}1\\ 1\\ 1\end{pmatrix}.$$

Here, the matrix $\begin{pmatrix}1 & 1 & -1\\ 1 & -1 & 1\\ 1 & 1 & 1\end{pmatrix}$ is invertible.

C.R. 44

latex

```
The system of equations
\begin{align*}
  x+y-z & = 1\\
  x-y+z & = 1\\
  x+y+z & = 1
\end{align*}
can be written in matrix terms as
\begin{equation*}
  \begin{pmatrix} 1& 1&-1\\ 1&-1& 1\\ 1&1&1
```

C.R. 45

```latex
  \end{pmatrix}
  \begin{pmatrix}
    x\\ y\\ z
  \end{pmatrix}
  =
  \begin{pmatrix}
    1\\ 1\\ 1
  \end{pmatrix}.
\end{equation*}
Here, the matrix
$\begin{pmatrix}
  1& 1&-1\\ 1&-1& 1\\ 1&1&1
\end{pmatrix}$
is invertible.
```

Note that the environment pmatrix can be used within in-text mathematics or in displayed math. Why the p? There is indeed an environment matrix (without a p) but it produces an array without the enclosing parentheses (try it). If you want the array to be enclosed within square brackets, use bmatrix instead of pmatrix. Thus

> Some mathematicians write matrices within parentheses as in $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ while others prefer square brackets as in $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

is produced by

C.R. 46

```latex
Some mathematicians write matrices within parentheses as in
$
\begin{pmatrix}
  a & b\\
  c&d
\end{pmatrix}
$
while others prefer square brackets as in
$
\begin{bmatrix}
  a & b\\
  c&d \end{bmatrix}
$
```

Some mathematicians write matrices within parentheses as in

There is also a vmatrix environment, which is usually used for determinants as in

The determinant $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ is defined by

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

which is obtained from the input

C.R. 47 latex

```
The determinant
$
\begin{vmatrix}
  a & b\\
  c & d
\end{vmatrix}
$
is defined by
\begin{equation*}
  \begin{vmatrix}
    a & b\\
    c & d
  \end{vmatrix}
  =ad -bc
\end{equation*}
```

There is a variant `Vmatrix` which encloses the array in double lines. Finally, we have a Bmatrix environment which produces an array enclosed within braces `{ }`. A row of dots in a matrix can be produced by the command `\hdotsfour`. it should be used with an argument specifying the number of columns to be spanned. For example, to get

A general $m \times n$ matrix is of the form

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

we type

C.R. 48 latex

```
A general $m\times n$ matrix is of the form
\begin{equation*}
  \begin{pmatrix}
    a_{11} & a_{12} & \dots & a_{1n}\\
    a_{21} & a_{22} & \dots & a_{2n}\\
    \hdotsfor{4}\\
    a_{m1} & a_{m2} & \dots & a_{mn}
  \end{pmatrix}
\end{equation*}
```

The command \hdotsfor has also an optional argument to specify the spacing of dots.

5.5.2 Dots

In the above example, we used the command \dots to produce a row of three (3) dots. This can be used in other contexts also. For example,

C.R. 49
latex
```
Consider a finite sequence $X_1,X_2,\dots$, its sum $X_1+X_2+\dots$
and product $X_1X_2\dots$.
```

gives

Consider a finite sequence $X_1, X_2, \ldots$, its sum $X_1 + X_2 + \ldots$ and product $X_1X_2 \ldots$.

Here the dots in all the three (3) contexts are along the "baseline" of the text. Isn't it better to typeset this as

Consider a finite sequence $X_1, X_2, \ldots$, its sum $X_1 + X_2 + \cdot$ and product $X_1X_2 \cdots$.

with raised dots for addition and multiplication?

The above text is typeset by the input

C.R. 50
latex
```
Consider a finite sequence $X_1,X_2,\dotsc$, its sum $X_1+X_2+\cdotb$
and product $X_1X_2\dotsm$.
```

Here:

- \dotsc stands for dots to be used with commas,
- \dotsb for dots with binary operations (or relations), and
- \dotsm for multiplication dots.

There is also a \dotsi for dots with integrals as in

$$\int_{A_1}\int_{A_2}\cdots\int_{A_n} f$$

5.5.3 Delimiters

How do we produce something like

$\left|\begin{smallmatrix} a & h & g \\ h & b & f \\ g & f & c \end{smallmatrix}\right| = 0$, the matrix $\left(\begin{smallmatrix} a & h & g \\ h & b & f \\ g & f & c \end{smallmatrix}\right)$ is not invertible.

Here the small in-text matrices are produced by the environment `smallmatrix`. This environment does **NOT** provide the enclosing delimiters, which we must supply as in

C.R. 51
latex
```
$
\left|\begin{smallmatrix}
  a & h & g\\ h & b & f\\ g&f&c
\end{smallmatrix}\right|
=0
$,
the matrix
$
\left(\begin{smallmatrix}
  a & h & g\\ h & b & f\\ g&f&c
\end{smallmatrix}\right)
$
is not invertible.
```

Why the `\left|...\right|` and `\left{...\right}`?

These commands `\left` and `\right` enlarge the delimiter following them to the size of the enclosed material. To see their ef- fect, try typesetting the above example without these commands. The list of symbols at the end of the chapter gives a list of delimiters that are available off the shelf. One interesting point about the `\left` and `\right` pair is that, though every `\left` should be matched to a `\right`, the delimiters to which they apply need not match. In par- ticular we can produce a single large delimiter produced by `\left` or `\right` by matching it with a matching command followed by a period. For example,

$$\left.\begin{aligned} u_x &= v_y \\ u_y &= -v_x \end{aligned}\right\} \quad \text{Cauchy-Riemann Equations}$$

is produced by

C.R. 52
latex
```
\begin{equation*}
  \left.
    \begin{aligned}
      u_x & = v_y\\
      u_y & = -v_x
    \end{aligned}
  \right\}
  \quad\text{Cauchy-Riemann Equations}
\end{equation*}
```

There are instances where the delimiters produced by `\left` and `\right` are too small or too large.

For example,

C.R. 53
latex
```
\begin{equation*}
  (x+y)^2-(x-y)^2=\left((x+y)+(x-y)\right)\left((x+y)-(x-y)\right)=4xy
\end{equation*}
```

gives

$$(x+y)^2-(x-y)^2=\left((x+y)+(x-y)\right)\left((x+y)-(x-y)\right)=4xy$$

where the parentheses are all of the same size. But it may be better to make the outer ones a little larger to make the nesting visually apparent, as in

$$(x+y)^2-(x-y)^2=\bigl((x+y)+(x-y)\bigr)\bigl((x+y)-(x-y)\bigr)=4xy$$

This is produced using the commands `\bigl` and `\bigr` before the outer parentheses as shown below:

C.R. 54
latex
```
\begin{equation*}
  (x+y)^2-(x-y)^2=\bigl((x+y)+(x-y)\bigr)\bigl((x+y)-(x-y)\bigr)=4xy
\end{equation*}
```

This is produced using the commands `\bigl` and `\bigr` before the outer parentheses as shown below:

$$(x+y)^2-(x-y)^2=\bigl((x+y)+(x-y)\bigr)\bigl((x+y)-(x-y)\bigr)=4xy$$

Apart from `\bigl` and `\bigr` there are `\Bigl`, `\biggl` and `\Biggl` commands (and their r counterparts) which (in order) produce delimiters of increasing size. (Experiment with them to get a feel for their sizes.) As another example, look at

For n-tuples of complex numbers $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$ of complex numbers

$$\left(\sum_{k=1}^{n} |x_k y_k|\right)^2 \le \left(\sum_{k=1}^{n} |x_k|\right)\left(\sum_{k=1}^{n} |y_k|\right)$$

which is produced by:

C.R. 55
latex
```
For $n$-tuples of complex numbers $(x_1,x_2,\dotsc,x_n)$ and
$(y_1,y_2,\dotsc,y_n)$ of complex numbers
\begin{equation*}
  \left(\sum_{k=1}^n|x_ky_k|\right)^2\le
  \left(\sum_{k=1}^{n}|x_k|\right)\left(\sum_{k=1}^{n}|y_k|\right)
\end{equation*}
```

Does not the output below look better?

> For n-tuples of complex numbers $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ of complex numbers
>
> $$\left(\sum_{k=1}^{n} |x_k y_k|\right)^2 \le \left(\sum_{k=1}^{n} |x_k|\right)\left(\sum_{k=1}^{n} |y_k|\right)$$

This one is produced by

C.R. 56 latex

```
For $n$-tuples of complex numbers $(x_1,x_2,\dotsc,x_n)$ and
$(y_1,y_2,\dotsc,y_n)$ of complex numbers
\begin{equation*}
  \biggl(\sum_{k=1}^n|x_ky_k|\biggr)^2\le
  \biggl(\sum_{k=1}^{n}|x_k|\biggr)\biggl(\sum_{k=1}^{n}|y_k|\biggr)
\end{equation*}
```

Here the trouble is that the delimiters produced by `\left` and `\right` are a bit too large.

5.5.4 Putting One Over Another

Look at the following text

> From the binomial theorem, it easily follows that if n is an even number, then
>
> $$1 - \binom{n}{1}\frac{1}{2} + \binom{n}{2}\frac{1}{2^2} - \cdots - \binom{n}{n-1}\frac{1}{2^{n-1}} = 0$$

We have fractions like $\frac{1}{2^{n-1}}$ and binomial coefficients like $\binom{n}{2}$ here and the common feature of both is that they have one mathematical expression over another.

Fractions are produced by the `\frac` command which takes two (2) arguments, the numerator followed by the denominator and the binomial coefficients are produced by the `\binom` command which also takes two arguments, the "top" expression followed by the "bottom" one. Thus the the input for the above example is

C.R. 57 latex

```
From the binomial theorem, it easily follows that if $n$ is an even
number, then
\begin{equation*}
  1-\binom{n}{1}\frac{1}{2}+\binom{n}{2}\frac{1}{2^2}-\dotsb
  -\binom{n}{n-1}\frac{1}{2^{n-1}}=0
\end{equation*}
```

You can see from the first paragraph above that the size of the outputs of `\frac` and `\binom` are smaller in text than in display. This default behavior has to be modified sometimes for nicer looking output. For example, consider the following output

Since (x_n) converges to 0, there exists a positive integer p such that

$$|x_n| < \tfrac{1}{2} \quad \text{for all } n \geq p$$

The second output is produced by the input:

C.R. 58

```latex
Since $(x_n)$ converges to $0$, there exists a positive integer $p$
  such that
  \begin{equation*}
    |x_n|<\tfrac{1}{2}\quad\text{for all $n\ge p$}
  \end{equation*}
```

Note the use of the command `\tfrac` to produce a smaller fraction. (The first output is produced by the usual `\frac` command.) There is also command `\dfrac` to produce a display style (larger size) fraction in text. Thus the sentence after the first example in this (sub)section can be typeset as

We have fractions like $\dfrac{1}{2^{n-1}}$ and ...

by the input

C.R. 59

```latex
We have fractions like $\dfrac{1}{2^{n-1}}$ and ...
```

As can be guessed, the original output was produced by `\frac`. Similarly, there are commands `\dbinom` (to produce display style binomial coefficients) and `\tbinom` (to produce text style binomial coefficients).

There is also a `\genfrac` command which can be used to produce custom fractions. To use it, we will have to specify six (6) things:

1. The left delimiter to be used-note that must be specified as {
2. The right delimiter-again, to be specified as }
3. The thickness of the horizontal line between the top expression and the bottom expression. If it is not specified, then it defaults to the 'normal' thickness. If it is set as 0pt then there will be no such line at all in the output.
4. The size of the output-this is specified as an integer 0, 1, 2 or 3, greater values cor- responding to smaller sizes. (Technically these values correspond to `\displaystyle`, `\textstyle`, `\scriptstyle` and `\scriptscriptstyle`.)
5. The top expression
6. The bottom expression

Thus instead of `\tfrac{1}{2}` we can also use `\genfrac{}{}{}{1}{1}{2}` and instead of `\dbinom{n}{r}`, we can also use `\genfrac{(}{)}{0pt}{0}{1}{2}` (but there is hardly any reason for doing so). More seriously, suppose we want to produce:

The Christoffel symbol $\left\{ {ij \atop k} \right\}$ of the second kind is related to the Christoffel symbol $\left[{ij \atop k} \right]$ of the first kind by the equation

$$\left\{ {ij \atop k} \right\} = g^{k1} \left[{ij \atop 1} \right] + g^{k2} \left[{ij \atop 2} \right]$$

If such expressions are frequent in the document, it would be better to define 'newcom mands' for them and use them instead of `\genfrac` every time as in the following input (which produces the same output as above). Therefore we can write the above expression as:

C.R. 60 latex

```latex
\newcommand{\chsfk}[2]{\genfrac{[}{]}{0pt}{}{#1}{#2}}
\newcommand{\chssk}[2]{\genfrac{\{}{\}}{0pt}{}{#1}{#2}}
The Christoffel symbol $\genfrac{\{}{\}}{0pt}{}{ij}{k}$ of the second
kind is related to the Christoffel symbol $\genfrac{[}{]}{0pt}{}{ij}{k}$
of the first kind by the equation
\begin{equation*}
  \chssk{ij}{k}=g^{k1}\chsfk{ij}{1}+g^{k2}\chsfk{ij}{2}
\end{equation*}
```

While on the topic of fractions, we should also mention the `\cfrac` command used to typeset continued fractions. For example, to get

$$\frac{4}{\pi} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cdots}}}$$

C.R. 61 latex

```latex
\begin{equation*}
  \frac{4}{\pi}=1+\cfrac{1^2}{2+
    \cfrac{3^2}{2+
      \cfrac{5^2}{2+\dotsb}}}
\end{equation*}
```

5.5.5 Putting Symbols Over or Under

The table at the end of this chapter gives various math mode accents such as `$\hat{a}$` to produce $\hat{a}$ and `$\dot{a}$` to produce $\dot{a}$. But what if one needs $\overset{\circ}{a}$ or $\underset{\circ}{a}$?

The commands `\overset` and `\underset` come to the rescue. Thus `$\overset{\circ}{a}$` produces $\overset{\circ}{a}$ and `$\underset{\circ}{a}$` produces $\underset{\circ}{a}$.

Basic LaTeX provides the commands `\overrightarrow` and `\overleftarrow` also to put (extensible) arrows over symbols, as can be seen from the table. The **amsmath** package also provides the commands `\underrightarrow` and `\underleftarrow` to put (extensible) arrows below mathematical expressions.

Speaking of arrows, **amsmath** provides the commands `\xrightarrow` and `\xleftarrow` which produces arrows which can accommodate long texts as superscripts or subscripts. Thus we can produce:

Thus we see that
$$0 \to A \xrightarrow{f} B \xrightarrow{g} C \to 0$$

which is generated from:

C.R. 62
latex
```
Thus we see that
\begin{equation*}
  0\xrightarrow{} A\xrightarrow{f}
  B\xrightarrow{g}
  C\xrightarrow{} 0
\end{equation*}
```

Note how the mandatory arguments of the first and last arrows are left empty to produce arrows with no superscripts. These commands also allow an optional argument (to be typed inside square brackets), which can be used to produce subscripts. For example

C.R. 63
latex
```
Thus we get
\begin{equation*}
  0\xrightarrow{} A\xrightarrow[\text{monic}]{f}
  B\xrightarrow[\text{epi}]{g}
  C\xrightarrow{} 0
\end{equation*}
```

gives:

Thus we get
$$0 \to A \xrightarrow[\text{monic}]{f} B \xrightarrow[\text{epi}]{g} C \to 0$$

By the way, would not it be nicer to make the two middle arrows the same width? This can be done by changing the command for the third arrow (the one from B) as shown below:

C.R. 64
latex
```
Thus we get
\begin{equation*}
  0\xrightarrow{} A\xrightarrow[\text{monic}]{f}
  B\xrightarrow[\hspace{7pt}\text{epi}\hspace{7pt}]{g}
  C\xrightarrow{}0
\end{equation*}
```

which produces:

Thus we get

$$0 \to A \xrightarrow[\text{monic}]{f} B \xrightarrow[\text{epi}]{g} C \to 0$$

where the lengths of the two arrows are almost the same. There are indeed ways to make the lengths exactly the same, but we will talk about it in another chapter.

Mathematical symbols are also attached as limits to such large operators as sum (Σ), product (Π) set union ($\cup$), set intersection ($\cap$) and so on. The limits are input as subscripts or superscripts, but their positioning in the output is different in text and display. For example, the input

C.R. 65

```latex
Euler not only proved that the series
$\sum_{n=1}^\infty\frac{1}{n^2}$ converges, but also that
\begin{equation*}
  \sum_{n=1}^\infty\frac{1}{n^2}=\frac{\pi^2}{6}
\end{equation*}
```

gives the output:

Euler not only proved that the series $\sum_{n=1}^\infty \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^\infty \frac{1}{n^2} = \frac{\pi^2}{6}$$

Note that in display, the sum symbol is larger and the limits are put at the bottom and top (instead of at the sides,which is usually the case for subscripts and superscripts). If you want the same type of symbol (size, limits and all) in text also, simply change the line

C.R. 66

```latex
$\sum_{n=1}^\infty\frac{1}{n^2}$
```

to

C.R. 67

```latex
$\displaystyle\sum_{n=1}^\infty\frac{1}{n^2}$
```

and we will get:

Euler not only proved that the series $\sum_{n=1}^\infty \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^\infty \frac{1}{n^2} = \frac{\pi^2}{6}$$

(Note that this also changes the size of the fraction. What would you do to keep it small?) On the other hand, to make the displayed operator the same as in the text, add the command `\textstyle` before the `\sum` within the equation. What if you only want to change the position of the limits but not

the size of the operator in text? Then change the command `$\sum_{n=1}^\infty\frac{1}{n^2}$` to `$\sum_\limits{n=1}^\infty\frac{1}{n^2}$` and this will produce the output given below.

Euler not only proved that the series $\sum_{n=1}^{\infty} \frac{1}{n^2}$ converges, but also that

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

On the other hand, if you want side-set limits in display type `\nolimits` after the `\sum` within the equation as in

C.R. 68

latex

```
Euler not only proved that the series
$\sum_{n=1}^\infty\frac{1}{n^2}$ converges, but also that
\begin{equation*}
  \sum\nolimits_{n=1}^\infty\frac{1}{n^2}=\frac{\pi^2}{6}
\end{equation*}
```

All these are true for other operators classified as “Variable-sized symbols”,except integrals. Though the integral symbol in display is larger, the position of the limits in both text and display is on the side as can be seen from the output below

Thus $\lim\limits_{x\to\infty}\int_0^x\frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$ and so by definition,

$$\int_0^{\infty} \frac{\sin x}{x}\,\mathrm{d}x = \frac{\pi}{2}$$

which is produced by:

C.R. 69

latex

```
Thus
$\lim\limits_{x\to\infty}\int_0^x\frac{\sin x}{x}\,\mathrm{d}x
=\frac{\pi}{2}$
and so by definition,
\begin{equation*}
  \int_0^\infty\frac{\sin x}{x}\,\mathrm{d}x=\frac{\pi}{2}
\end{equation*}
```

5.6 New Operators

Mathematical text is usually typeset in *italics*, and TEX follows this tradition. But certain functions in mathematics such as log, sin, lim and so on are traditionally typeset in roman. This is implemented in TEX by the use of commands like `$\log$`, `$\sin$`, `$\lim$` and so on. The symbols classified as "Log-like symbols" in the table at the end of this chapter shows such functions which are predefined in LATEX.

Having read thus far, it may be no surprise to learn that we can define our own "operator names" which receive this special typographic treatment. This is done by the `\DeclareMathOperator` command. Thus if the operator cl occurs frequently in the document, you can make the declaration

C.R. 70
latex
```
\DeclareMathOperator{\cl}{cl}
```

5.7 Fonts for Mathematics

We have noted that most mathematics is typeset in italics typeface and some mathematical operators are typeset in an upright fashion. There may be need for additional typefaces as in typesetting vectors in boldface.

In addition to these, several other math alphabets are available in various packages. Note that the command `\mathbf` produces only roman boldface and not math italic boldface. Sometimes you may need boldface math italic, for example to typeset vectors. For this, amsmath provides the `\boldsymbol` command. Thus we can get

> In this case, we define
> $$\boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}$$

from the input

C.R. 71
latex
```
In this case, we define
\begin{equation*}
  \boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}
\end{equation*}
```

If the document contains several occurrences of such symbols, it is better to make a new definition such as

C.R. 72
latex
```
\newcommand{\vect}[1]{\boldsymbol{#1}}
```

and then use `$\vect{a}$` to produce a and $\boldsymbol{a}$ to produce b and so on. the additional advantage of this approach is that if you change your mind later and want vectors to be typeset with arrows above them as `\vec{a}`, then all you need is to change the `\boldsymol` part of the definition of `\vect` to `\overrightarrow` and the change will be effected throughout the document.

Now if we change the input of the above example as

> In this case, we define
> $$\boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}$$

then we get the output

C.R. 73
latex
```
In this case, we define
\begin{equation*}
  \boldsymbol{a+b=c}
\end{equation*}
```

Note that now the symbols + and = are also in **boldface**. Thus `\boldsymbol` makes bold every math symbol in its scope.[15] There is another reason for tweaking the math fonts.

[15] provided the bold version of that symbol is available in the current math font

Recently, the International Standards Organization (ISO) has established the recognised typesetting standards in mathematics. Some of the points in it are,

- Simple variables are represented by italic letters as a, x.
- Vectors are written in boldface italic as a, x.
- Matrices may appear in sans serif as in A, X.
- The special numbers e, i and the differential operator d are written in upright roman.

The first point is the default in LaTeX and we have seen how the second point can be implemented. To fulfil the last point, it is enough if we define something like:

C.R. 74
latex
```
\newcommand{\me}{\mathrm{e}}
\newcommand{\mi}{\mathrm{i}}
\newcommand{\diff}{\mathrm{d}}
```

and then use `\mathrm{e}` for e and `\mathrm{i}` for i and `\mathrm{diff}x` for dx. Third point can be implemented using `\mathsf` but it is a bit difficult if we need them to be in *italic* also. The solution is to create a new math alphabet, say, `\mathsfs1` by the command in the preamble.

C.R. 75
latex
```
\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
```

and use it to define a command `\matr` to typeset matrices in this font by:

C.R. 76
latex
```
\newcommand{\matr}[1]{\ensuremath{\mathsfsl{#1}}}
```

so that `$\matr A$` produces A. We end this chapter with a list of common list of symbols in LaTeX. This is by no means an exhaustive list as the list can be extended either by the user of by use of packages.

Table 5.1: List of symbols used in LaTeX

Symbol	Command	Symbol	Command	Symbol	Command
$\leq$	\leq	$\exists$	\exists	$\forall$	\forall
$\geq$	\geq	$\in$	\in	$\square$	\square
$\neq$	\neq	$\subset$	\subset	$\angle$	\angle
$\nleq$	\nleq	$\subseteq$	\subseteq	Θ	\Theta
$\ngeq$	\ngeq	$\varnothing$	\varnothing	Π	\Pi
$\cong$	\cong	$\cap$	\cap	Γ	\Gamma
$\equiv$	\equiv	$\cup$	\cup	Δ	\Delta
$\sim$	\sim	$\setminus$	\setminus	Ω	\Omega
$\approx$	\approx	$\wedge$	\wedge	Σ	\Sigma
$\doteqdot$	\doteqdot	$\vee$	\vee	α	\alpha
$\times$	\times	$\Rightarrow$	\Rightarrow	β	\beta
$\cdot$	\cdot	$\rightarrow$	\rightarrow	ϵ	\epsilon
$\ast$	\ast	$\mapsto$	\mapsto	ζ	\zeta
$\div$	\div	\$	\$	η	\eta
$\pm$	\pm	&	\&	κ	\kappa
$\mp$	\mp	%	\%	λ	\lambda
$\bigcirc$	\bigcirc	\	\backslash	μ	\mu
$\oplus$	\oplus	$\sharp$	\sharp	ξ	\xi
$\otimes$	\otimes	∂	\partial	ρ	\rho
$\propto$	\propto	90°	90^\circ	τ	\tau
$\cdots$	\cdots	$\parallel$	\parallel	ϕ	\phi
$\ldots$	\dots	$\bot$	\bot	ψ	\psi
$\because$	\because	$\triangle$	\triangle	π	\pi
$\therefore$	\therefore	∇	\nabla	θ	\theta
δ	\delta	γ	\gamma	ω	\omega
σ	\sigma	∞	\infty	f'	f\;$\prime$
$\int$	\int	$\oint$	\oint	$\mathbb{Z}$	\mathbb{Z}
$\mathbb{R}$	\mathbb{R}	$\mathbb{Q}$	\mathbb{Q}	$\sqrt[3]{2}$	\sqrt[3]{2}
$\frac{2}{3}$	\frac{2}{3}	$\lceil x \rceil$	\lceil x \rceil	$\lfloor x \rfloor$	\lfloor x \rfloor

Chapter 6

Float Environments

Table of Contents

6.1 The figure Environment

Figures are really problematical to present in a document as they never split between pages. This leads to bad page breaks which in turn leave blank space at the bottom of pages. For fine-tuning that document, the typesetter[1] has to adjust the page breaks manually. Fortunately for us, LaTeX provides floating figures which automatically move to suitable locations. So the positioning of figures is the duty of LaTeX.

[1] That is us.

6.1.1 Creating floating figures

Floating figures are created by putting commands in a `figure` environment. The contents of the `figure` environment always remains in one chunk, floating to produce good page breaks. The following commands put the graphic from figure.eps inside a floating figure:

C.R. 1

latex

```
\begin{figure}
  \centering
  \includegraphics{figure.eps}
  \caption{This is an inserted EPS graphic}
  \label{fig1}
\end{figure}
```

Let's look at what is going on here:

- The optional `label` command can be used with the `ref`, and `pageref` commands to reference the caption. The `label` command must be placed immediately after the `\caption`.
- If the figure environment contains no `caption` commands, it produces an unnumbered floating figure.
- If the figure environment contains multiple `caption` commands, it produces multiple figures which float together. This is useful in constructing side-by-side graphics or complex arrangements.
- A list of figures is generated by the `listoffigures` command.
- By default, the caption text is used as the caption and also in the list of figures. The caption has an optional argument which specifies the list-of-figure entry. For example,

```latex
\caption[List Text]{Caption Text}
```
C.R. 2

  causes "Caption Text" to appear in the caption, but "List Text" to appear in the list of figures. This is useful when using long, descriptive captions.
- The figure environment can only be used in outer paragraph mode, preventing it from being used inside any box.

6.1.2 Figure Placement

The figure environment has an optional argument which allows users to specify possible figure locations. The optional argument can contain any combination of the letters: `h`, `t`, `b`, `p`.

`h` Place the figure in the text where the figure command is located. This option cannot be executed if there is not enough room remaining on the page.

`t` Place the figure at the top of the page.

`b` Place the figure at the bottom of a page.

`p` Place the figure on a page containing only floats.

> If no optional arguments are given, the placement options default to `[tbp]`.

When we input a float, LaTeX will read that float and hold it until it can be placed at a better location. Unprocessed floats are those which are read by LaTeX but have not yet been placed on the page. Though the float-placing is done by LaTeX, sometimes the user has to invoke commands to process unprocessed floats.

Following commands will do that job:

`\clearpage`
: This command places unprocessed floats and starts a new page.

`\FloatBarrier`
: This command causes all unprocessed floats to be processed. This is provided by the **placeins** package. It does not start a new page, unlike `\clearpage`.

Since it is often desirable to keep floats in the section in which they were issued, the section option

C.R. 3
```latex
\usepackage[section]{placeins}
```

redefines the `\section` command, inserting a `\FloatBarrier` command before each section.

> This option is very strict and does not allow a float from the previous section to appear at the bottom of the page, since that is after the start of a new section.

The below option

C.R. 4
```latex
\usepackage[below]{placeins}
```

is a less-restrictive version of the section option. It allows floats to be placed after the beginning of a new section, provided that some of the previous section appears on the page.

`\afterpage/\clearpage`
: The **afterpage** package provides the `\afterpage` command which executes a command at the next naturally-ocurring page break.

Therefore, using `\afterpage{\clearpage}` causes all unprocessed floats to be cleared at the next page break.

> `\afterpage{\clearpage}` is especially useful when producing small floatpage figures.

6.1.3 Customising the float placement

The following style parameters are used by LATEX to prevent awkward-looking pages which contain too many floats or badly-placed floats.

`\topnumber`
: The maximum number of floats allowed at the top of a text page (the default is 2).

`\bottomnumber`
: The maximum number of floats allowed at the bottom of a text page (the default is 1).

`\totalnumber`
: The maximum number of floats allowed on any one text page (the default is 3).

These counters prevent LaTeX from placing too many floats on a text page. These counters do **NOT** affect float pages. Specifying a `!` in the float placement options causes LaTeX to ignore these parameters. The values of these counters are set with the `\setcounter` command. For example,

C.R. 5
```latex
\setcounter{totalnumber}{2}
```

prevents more than two floats from being placed on any text page.

Figure Fractions

The commands given below control what fraction of a page can be covered by floats

> In this context, "fraction" refers to the height of the floats divided by `\textheight`.

The first three (3) commands pertain only to text pages, while the last command pertains only to float pages. Specifying a `!` in the float placement options causes LaTeX to ignore the first three parameters, but `\floatpagefraction` is always used. The value of these fractions are set by `\renewcommand`. For example,

C.R. 6
```latex
\renewcommand{\textfraction}{0.3}
```

Looking at this in more detail:

`\textfraction`
: The minimum fraction of a text page which must be occupied by text. The default is 0.2, which prevents floats from covering more than 80% of a text page.

`\topfraction`
: The maximum fraction of a text page which can be occupied by floats at the top of the page. The default is 0.7, which prevents any float whose height is greater than 70% of `\textheight` from being placed at the top of a page.

`\bottomfraction`
: The maximum fraction of a text page which can be occupied by floats at the bottom of the page. The default is 0.3, which prevents any float whose height is greater than 40% of `\textheight` from being placed at the bottom of a text page.

`\floatpagefraction`
: The minimum fraction of a float page that must be occupied by floats. Thus the fraction of blank space on a float page cannot be more than `1-\floatpagefraction`. The default is 0.5.

6.1.4 Using Graphics in LaTeX

Now we got the formalities out of the way let's look at how graphics can be handled in LaTeX documents. While LaTeX can import virtually any graphics format, Encapsulated PostScript (EPS) is the easiest graphics format to import into LaTeX. The "eps" files are inserted into the file using command `\includegraphics[]{file.eps}`.

The includegraphics command

The syntax of it is as follows:

C.R. 7
```latex
\includegraphics[options]{filename}
```

The following options are available in `\includegraphics` command:

`width` The width of the graphics,

`height` The height of the graphics,

`totalheight` The totalheight of the graphics,

`scale` Scale factor for the graphic (i.e., specifying `scale=2` makes the graphic twice as large as its natural size.)

`angle` Specifies the angle of rotation, in degrees, with a counter-clockwise (i.e., anti-clockwise) rotation being positive.

Graphics Search Path

By default, LaTeX looks for graphics files in any directory on the TEX search path. In addi- tion to these directories, LaTeX also looks in any directories specified in the `\graphicspath` command. For example,

C.R. 8
```latex
graphicspath{{dir1/}{dir2/}}
```

tells LaTeX to look for graphics files also in `dir1/` and `dir2/`.

Graphics Extensions

The `\DeclareGraphicsExtensions` command tells LaTeX which extensions to try if a file with no extension is specified in the `\includegraphics` command. For convenience, a default set of extensions is pre-defined depending on which graphics driver is selected.

C.R. 9
```latex
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

With the above graphics extensions specified, `\includegraphicsfile` first looks for `file.eps`, then `file.ps`, then `file.eps.gz`, etc. until a file is found. This allows the graphics to be specified with

C.R. 10
```latex
\includegraphics{file}
```

instead of

C.R. 11
```latex
\includegraphics{file.eps}
```

6.2 The Table Environment

The environments `tabular` and `tabular*` are the basic tools with which tables can be constructed. The syntax for these environments is [26]:

C.R. 12
```latex
\begin{tabular}[pos]{cols} rows \end{tabular}
\begin{tabular*}{width}[pos]{cols} rows \end{tabular*}
```

Both the above environments actually create a minipage. The meaning of the above arguments is as follows:

`pos` Vertical positioning arguments (see also the explanation of this argument for parboxes). It can take on the values:

`t` The top line of the table is aligned with the baseline of the current external line of text.

`b` The bottom line of the table is aligned with the external baseline.

With no positioning argument given, the table is centered on the external baseline.

`width` This argument applies only to the `tabular*` environment and determines its overall width. In this case, the `cols` argument must contain the `@`-expression `@{\extracolsep{\fill}}` somewhere after the first entry. For the other two (2) environments, the total width is fixed by the textual content.

`cols` The column formatting argument. There must be an entry for every column, as well as possible extra entries for the left and right borders of the table or for the inter-column spacings. The possible column formatting symbols are:

Without going into more unnecessary information let's have a look at an example:

Sample Tabular		
col head	col head	col head
Left	centered	right
aligned	items	aligned
items	items	items
Left items	centered	right aligned

typed as following:

C.R. 13

```latex
\begin{center}
  \begin{tabular}{|l|c|r|}
    \hline
    \multicolumn{3}{|c|}{Sample Tabular}  \\
    \hline
    col head   & col head & col head      \\
    \hline
    Left       & centered & right         \\\cline{1-2}
    aligned    & items    & aligned       \\\cline{2-3}
    items      & items    & items         \\\cline{1-2}
    Left items & centered & right aligned \\
    \hline
  \end{tabular}
\end{center}
```

6.2.1 tabularray Package

For a long time, the tabular environment was used to build tables. However, writing tables with tabular can be troublesome for beginners and really complex tables can be near impossible to write as we have seen from the previous example. Also, tables built with the tabular environment have some typographical issues and, when color is used, can be misread by PDF‘ readers. Therefore, more and more LaTeX users are calling to use the tblr environment from the tabularray package instead.

The tblr environment

The `tblr` environment, like HTML and CSS, separates the content and the style of table cells. To do so, it uses keyval arguments to define options for columns, rows or cells. A keyval argument is an argument that takes the form of:

C.R. 14

```latex
option1=value, option2=value
```

If a value contains commas, it must be surrounded by brackets in this way:

C.R. 15
```latex
mylist = {element1, element2, and element 3}, option2=value
```

The `tblr` environment takes a single mandatory argument, a keyval argument containing a list of options for styling the table. Inside the environnment, each column is separated by `&`. Finally, each row is separated by the command `\\`. A really basic table would be this:

A1	B1	C1
A2	B2	C2
A3	B3	C3

which is typed as follows:

C.R. 16
```latex
% \usepackage{tabularray}

\begin{tblr}{}
A1 & B1 & C1  \\
A2 & B2 & C2  \\
A3 & B3 & C3
\end{tblr}
```

In the mandatory argument, you can define options for columns, rows and cells using those keys:

Key	Descriptions
`cells`	Options for all cells
`cell{<Row-Number>}{<Column-Number>}`	Options for specific cells
`columns`	Options for all columns
`column{<Column-Number>}`	Option for specific columns
`rows`	Options for all rows
`row{<Row-Number>}`	Options for specific rows

The values `<Row-Number>` and `<Column-Number>` can be a single number,[2] a range (1-3) or a special value (like odd or even).

[2] 1, for the first column or row.

> The characters X, Y and Z are converted to the indexes of the last three child, respectively.

The following example use the key `bg` to set the color of the background of each cell and `fg` to set the color of the text:

C.R. 17

```latex
% \usepackage{xcolor}
% \usepackage{tabularray}

\begin{tblr}{
    row{odd}     = {bg = red},
    cell{2}{2-3} = {bg = green},
    column{1}    = {bg = blue, fg = white}
}
A1 & B1 & C1  \\
A2 & B2 & C2  \\
A3 & B3 & C3
\end{tblr}
```

This would result in the following colorful table:

A1	B1	C1
A2	B2	C2
A3	B3	C3

Some Basic Options

We already saw two options which can be used to style table cells. However, there are much more. Here are some of them:

Key	Value	Description
`bg`	Background color	Colour name
`fg`	Color of the text	Color name
`halign`	Horizontal alignment of the text	`l` (left), `c` (center), `r` (right)
`valign`	Vertical alignment of the text	`t` (top), `m` (middle), `b` (bottom), `h` (head), `f` (foot)

As a trick, you can omit the name of the keys `bg`, `halign` and `valign` and simply enter their value, separated by a comma:

C.R. 18

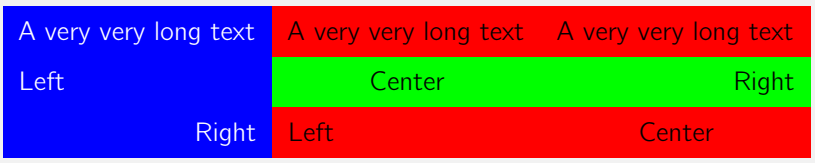
```latex
% \usepackage{color}
% \usepackage{tabularray}

\begin{tblr}{
    row{odd}     = {red},
    cell{2}{2-3} = {green},
    column{1}    = {blue, fg = white},
    cell{3}{1}   = {r},
    cell{2}{3}   = {r},
```

C.R. 19 latex

```
    cell{2}{2}   = {c},
    cell{3}{3}   = {c}
}
A very very long text & A very very long text & A very very long text  \\
Left                  & Center                & Right                 \\
Right                 & Left                  & Center
\end{tblr}
```

which is typed as following:



Adding Borders

To add borders to the table, we use other "superkeys" like `cell` or `row`. However, instead, we use `hlines` to set all horizontal borders, `hline` to set specific horizontal borders, `vlines` to set all vertical borders and `vline` to set specific vertical borders.

With those superkeys, we can set several options: the thickness of the border, the color of the border and its style (`solid`, `dashed` or `dotted`). If the brackets are empty, a normal black border will be used. See this basic example:

A1	B1	C1
A2	B2	C2
A3	B3	C3

typed as:

C.R. 20 latex

```
% \usepackage{tabularray}

\begin{tblr}{
    vlines = {},
    hlines = {}
}
A1 & B1 & C1  \\
A2 & B2 & C2  \\
A3 & B3 & C3
\end{tblr}
```

or this more complex example:

A1	B1	C1
A2	B2	C2
A3	B3	C3

implemented using:

C.R. 21

```latex
% \usepackage{color}
% \usepackage{tabularray}

\begin{tblr}{
    vline{1,4} = {3pt},
    hline{1,4} = {},
    hline{2}   = {blue,dotted}
}
A1 & B1 & C1  \\
A2 & B2 & C2  \\
A3 & B3 & C3
\end{tblr}
```

Merging Cells

It is way easier to merge columns or rows with `tblr` than `tabular`. To do so, you only have to use the `r` key to set the number of rows or `c` to set the number of cells.

> you can only use these keys with the cell superkey and they must be surrounded by distinct brackets.

Also, unlike with tabular, you don't omit ampersands. See the following example:

<table>
<tr><td colspan="4">A1</td></tr>
<tr><td rowspan="3">A2</td><td>B2</td><td>C2</td><td>D2</td></tr>
<tr><td>B3</td><td colspan="2" rowspan="2">C3</td></tr>
<tr><td>B4</td></tr>
</table>

Typed as following:

C.R. 22

```latex
% \usepackage{tabularray}

\begin{tblr}{
    vlines = {},
    hlines = {},
```

C.R. 23

latex

```
    cell{1}{1} = {c = 4}{halign = c},
    cell{2}{1} = {r = 3}{valign = m},
    cell{3}{3} = {c = 2, r = 2}{c,m}
}
A1 &    &    &    \\
A2 & B2 & C2 & D2 \\
   & B3 & C3 &    \\
   & B4 &    &
\end{tblr}
```

Chapter 7

Glossaries

Table of Contents

7.1 Introduction

A Glossary is an alphabetical list of terms in a particular domain of knowledge with the definitions for those terms. This is done to simplify typing and to reduce redundancies within the text. For example, let's look at the following text:

> *. . . Partial Differential Equations are important equations to engineers with one of the most important Partial Differential Equation is the Heat Equation which is classified as a linear Partial Differential Equation . . .*

As we can see there is a lot of redundancies which adds a lot of frivolous text which makes the text look more busy than it needs to be. With a glossary we can rewrite it as:

> *. . . Partial Differential Equations (PDEs) are important equations to engineers with one of the most important PDE is the Heat Equation which is classified as a linear PDE . . .*

Which is clearly tidier and cleaner way of writing the text. Due to this, many technical documents use terms, acronyms, and symbols in their text. It is common practice to add a glossary to make such documents more accessible.

The glossaries package can be used to create glossaries. It supports multiple glossaries, acronyms, and symbols.

7.2 Using Glossaries

To use the glossaries package, you have to load it explicitly using `usepackage{...}`:

C.R. 1
```latex
\usepackage{glossaries}
```

> If you would like to make sure your all the hyperlinks are working with your glossaries, make sure the glossaries is loaded **before** `hyperref`. Otherwise the glossary entries within the text will not be clickable.

For the glossary to show up in your **Table of Contents**, you need to specify the `toc` option when loading the packages such as:

C.R. 2
```latex
\usepackage[toc]{glossaries}
```

And of course, then we need to tell `glossaries` where to put all the glossary. To make this happen, place the following command in your document preamble in order to generate the glossary:

C.R. 3
```latex
\makeglossaries
```

[1] A general purpose programming language known for being the glue of the internet and having powerful text manipulation methods.

In addition, If you would like to use `makeglossaries` you will need to have Perl[1] installed on your computer which is not normally present by default on Microsoft Windows platforms.

> The `makeglossaries` simply provides a convenient interface to `makeindex` and `xindy` and is **NOT** essential as one could use either of those commands/packages instead.

7.2.1 Defining a Glossary Entry

To use an entry from a glossary we first need to define it. There are few ways to define an entry depending on what you define and how it is going to be used.

> A defined entry won't be included in the printed glossary unless it is used in the document.

While this may sounds like an oversight, it is actually a feature as this enables us to create a glossary of general terms for multiple documents and the based on what is used in an individual document, only the ones used will be printed.

To define a term in glossary we use the `\newglossaryentry` macro:

C.R. 4
```latex
\newglossaryentry{<label>}{<settings>}
```

where `<label>` is a unique label used to identify an entry in glossary, and `<settings>` are comma separated `key=value` pairs used to define an entry. For example, to define a entry for the word `computer`:

C.R. 5
```latex
\newglossaryentry{computer}
{
  name=computer,
  description={is a programmable machine that receives input,
               stores and manipulates data, and provides
               output in a useful format}
}
```

If we look at the example above, we have defined an entry which has the same label and entry name. This is not always have to be the case as the next entry will show where it is different:

C.R. 6
```latex
\newglossaryentry{naiive}
{
  name=na\"{\i}ve,
  description={is a French loanword (adjective, form of naïf)
               indicating having or showing a lack of experience,
               understanding or sophistication}
}
```

When you define terms, we need to remember these entries will be sorted by `makeindex` or `xindy`.

- While `xindy` is a bit more LaTeX aware, it does it by omitting macros (ï) therefore incorrectly sorting the above example as nave.
- On the other hand, `makeindex` won't fare much better either, as it doesn't understand TeX macros, it will interpret the word exactly as it was defined, putting it inside symbol class, before words beginning with `naa`.

Therefore it's needed to extend our example and specify how to sort the word:

C.R. 7
```latex
\newglossaryentry{naiive}
{
  name=na\"{\i}ve,
  description={is a French loanword (adjective, form of naïf)
               indicating having or showing a lack of experience,
               understanding or sophistication},
  sort=naiive
}
```

Here we explicitly state that this entry should be sorted as `naiive`. We can also specify plural forms, if they are not formed by adding "s":[2]

[2] Of course, for most applications it is just better to add a lowercase s to the end of glossary (i.e., PDE and PDEs).

C.R. 8 latex
```
\newglossaryentry{Linux}
{
  name=Linux,
  description={is a generic term referring to the family of Unix-like
               computer operating systems that use the Linux kernel},
  plural=Linuces
}
```

Or, for acronyms:

C.R. 9 latex
```
\newacronym[longplural={Frames per Second}]{fpsLabel}{FPS}{Frame per Second}
```

This will avoid the wrong long plural:

> Frame per Seconds instead of Frames per Second

So far, the glossary entries have been defined as key-value lists. Sometimes, a description is more complex than just a paragraph.

> For example, you may want to have multiple paragraphs, lists, figures, tables, . . .

For such glossary entries use the command `longnewglossaryentry` in which the description follows the key-value list. The computer entry then looks like this:

C.R. 10 latex
```
\longnewglossaryentry{computer}
{
  name=computer
}
  {is a programmable machine that receives input,
               stores and manipulates data, and provides
               output in a useful format}
```

Defining Symbols

When you write a long text full of mathematics (such as a thesis) you will have a bunch of letters which where used to defined to express a certain quantity. For example you might have used the letter d as the diameter of a circle. It would be a great assistance to the reader if all relevant symbols were to be compiled into a list for the reader to reference. These are called **nomenclature** and for this the `glossary` package also allows us to define symbols:

C.R. 11 latex
```
\newglossaryentry{pi}
{
  name={\ensuremath{\pi}},
  description={ratio of circumference of circle to its
```

C.R. 12
```latex
               diameter},
  sort=pi
}
```

We can also define both a name and a symbol:

C.R. 13
```latex
\newglossaryentry{real number}
{
  name={real number},
  description={include both rational numbers, such as $42$ and
               $\frac{-23}{129}$, and irrational numbers,
               such as $\pi$ and the square root of two; or,
               a real number can be given by an infinite decimal
               representation, such as $2.4871773339\ldots$ where
               the digits continue in some way; or, the real
               numbers may be thought of as points on an infinitely
               long number line},
  symbol={\ensuremath{\mathbb{R}}}
}
```

> Note that not all glossary styles show defined symbols.

Defining Acronyms

To define a new acronym you use the `\newacronym` macro:

C.R. 14
```latex
\newacronym{<label>}{<abbreviation>}{<full text>}
```

where `<label>` is the unique label identifying the acronym, `<abbrv>` is the abbreviated form of the acronym and <full> is the expanded text. For example:

C.R. 15
```latex
\newacronym{lvm}{LVM}{Logical Volume Manager}
```

> Defined acronyms can be put in separate list if you use acronym package option:
>
> `\usepackage[acronym]{glossaries}`

7.2.2 Using in the Text

When we have defined a term, we can use it in a document. There are many different commands used to refer to glossary terms.

General References

A general reference is used with \gls command. If, for example, you have glossary entries defined as those above, you might use it in this way:

C.R. 16

latex

```latex
\documentclass[]{article}

\usepackage{glossaries}

\newglossaryentry{Linux}
{name=Linux,
  description={is a generic term referring to the family of Unix-like computer operating
  ↪  systems
    that use the Linux kernel},
  plural=Linuces
}

\makeglossaries

\begin{document}

\Gls{Linux} is a great operating system as one of the great thing about \gls{Linux} is that it
↪  is open
source and which allows numerous \glspl{Linux} to flourish.

\end{document}

```

Linux is a great operating system as one of the great thing about Linux is that it is open source and which allows numerous Linuces to flourish.

Glossary

Linux is a generic term referring to the family of Unix-like computer operating systems that use the Linux kernel. 1

Figure 7.1

Command	Description
\gls{<label>}	prints the term associated with <label> passed as its argument. If the hyperref package was loaded before glossaries it will also be hyperlinked to the entry in glossary.
\glspl{<label>}	prints the plural of the defined term, other than that it behaves in the same way as gls.
\Gls{<label>}	prints the singular form of the term with the first character converted to upper case.
\Glspl{<label>}	prints the plural form with first letter of the term converted to upper case.
\glslink{<label>}{<alternate text>}	creates the link as usual, but typesets the alternate text instead. It can also take several options which changes its default behavior (see the documentation).
\glssymbol{<label>}	This command prints what ever is defined in \newglossaryentry{<label>}{symbol={...}}
\glsdesc{<label>}	prints what ever is defined in \newglossaryentry{<label>}{description={...}}

Referring Acronyms

7.2.3 Displaying the Glossary

To display the sorted list of terms you need to add the following cod

C.R. 17
```latex
\printglossaries
```

at the place you want the glossary and the list of acronyms to appear. If all entries are to be printed the command

C.R. 18
```latex
\glsaddall
```

can be inserted before \printglossaries. You may also want to use \usepackage[nonumberlist]{glossaries} to suppress the location list within the glossary.

Chapter 8

Cross-Referencing

Table of Contents

8.1 The Purpose of Cross-Referencing

Cross reference is the technical term for quoting yourself [27]. This is what you do when you say something like,

> *"As I said previously in the last chapter, . . . "*

To get serious for a moment, in a written article we may often have occasion to refer the reader to something mentioned earlier[1] in the same document. This could be:

- To tie your argument back to a section,
- To point the reader to a figure/table in some other section,
- Let the reader know additional information is in appendix,[2]
- Referencing equation to follow up on an idea.

Such cross referencing can be done by hand, but if you revise your document and insert some new sections then changing all cross references manually is no easy task and just a labour which needs to be avoided. It is always better to automate such tedious tasks.[3]

[1] or sometimes to something yet to be said

[2] A supplementary material placed at the end of a document and provides additional information which are relevant to the main document but not essential for understanding its core content.

[3] After all what's a computer for, if not to do such mundane jobs?

8.2 Making LaTeX do the Work

The basic method of using cross references, see Section 8.1 for what we mean by cross reference, in LaTeX is simple. Suppose somewhere in the second section of our article, you want to refer to the first section. We assign a key to the first section using the command:

C.R. 1
```latex
\section{section name}\label{key}
```

and at the point in the second section where the reference is to be made, we type the command

C.R. 2
```latex
\ref{key}
```

Therefore the reference "see Section 8.1 ..." in the first sentence of this section was produced by including the command `\label{intro}` in the command for the first section as follows:

C.R. 3
```latex
\section{The Purpose of Cross-Referencing}\label{intro}
```

and the command `\ref{intro}` at the place of reference in the second section as

C.R. 4
```latex
..., see Section \ref{intro} for...
```

> The `\label{key}` for a section need not be given immediately after the `\section{section name}`. It can be given anywhere within the section.

The first time we run LaTeX on a file named, say, `myfile.tex` containing cross references, the reference information is written in an auxiliary file named `myfile.aux` and at the end of the run LaTeX prints a warning:

C.R. 5
```text
LaTeX Warning: There were undefined references.

LaTeX Warning: Label(s) may have changed.
               Rerun to get cross-references right.

```

There is no need for an alarm as it is normal for this to happen. When a document is compiled for the first time, the compiler doesn't have an idea on where the references are so in the first run it collects all the information and locates all the references, and then a second run gets the references right. The same thing happens when you've changed the reference information in any way, say, by adding a new section.

While the key in `\label{key}` can be any sequence of letters, digits or punctuation characters, it is convenient to use some mnemonic.[4]

[4] such as `\label{limcon}` for a section entitled "Limits and Continuity" rather than `\label{ref98}`.

Also, when you make a reference, it's better to type `~\ref{limcon}` than `\ref{limcon}` to prevent the possibility of the reference number falling off the edge.

8.2.1 Referencing Items

In addition to sectioning commands such as `\chapter` or `\section`, reference can also be made to an `\item` entry in an enumerate environment, by attaching a `\label`. For example the input [28]:

C.R. 6

latex

```
In the classical \emph{syllogism}
\begin{enumerate}
\item All men are mortal.\label{pre1}
\item Socrates is a man.\label{pre2}
\item So Socrates is a mortal.\label{con}
\end{enumerate}
Statements (\ref{pre1}) and (\ref{pre2}) are the \emph{premises} and
statement (\ref{con}) is the conclusion.
```

In the classical *syllogism*

1. All men are mortal.
2. Socrates is a man.
3. So Socrates is a mortal.

Statements (1) and (2) are the *premises* and statement (3) is the conclusion.

Be careful about references to tables or figures (i.e., floats). For them, the `\label` command should be given after the `\caption` command or in its argument, as in the example below:

C.R. 7

latex

```
\begin{table}[h]
  \begin{center}
    \setlength{\extrarowheight}{5pt}
    \begin{tabular}{|c|c|c|c|}
      \hline
      Value of $x$ & 1 & 2 & 3\\
      \hline
      Value of $y$ & 1 & 8 & 27\\
      \hline
    \end{tabular}
    \caption{Observed values of $x$ and $y$}\label{tabxy}
  \end{center}
\end{table}
Two possible relations betweeen $x$ and $y$ satisfying
the data in Table\ref{tabxy} are $y=x3$ and
$y=6x2-11x+6$
```

Which produces the following output:

Value of x	1	2	3
Value of y	1	8	27

Table 8.1: Observed values of x and y

Two possible relations betweeen x and y satisfying the data in Table 8.1 are $y = x^3$ and $y = 6x^2 - 11x + 6$.

Think of a `\caption` command within a figure or table environment as a sort of sectioning command within the environment. Therefore, you can have several `\caption` and `\label` pairs within a single figure or table environment. You can also make forward references in exactly the same way by `\ref`-ing to the key of some succeeding `\label`.

8.2.2 Cross-Referencing in Math

Mathematical documents heavily use cross references. There are references to theorems and equations and figures and whatnot. The method of reference is exactly as before. Therefore if you've defined `\newtheorem{theorem}[subsection]`, then after typing

C.R. 8
latex
```
\begin{theorem}\label{diffcon}
  Every differentiable function is continuous
\end{theorem}
```

we get: and you can type elsewhere in the document

C.R. 9
latex
```
The converse of Theorem \ref{diffcon} is false.
```

to get:

The converse of Theorem **??** is false.

References can be made to equations as in the following examples:

C.R. 10
latex
```
\begin{equation}\label{sumsq}
  (x+y)2=x2+2xy+y2
\end{equation}
Changing $y$ to $-y$ in Equation (\ref{sumsq}) gives the following
```

gives:

$$(x+y)^2 = x^2 + 2xy + y^2 \tag{8.1}$$

Changing y to $-y$ in Equation (8.1) gives the following

If you load the package **amsmath**, you can use the command `\eqref` instead of `\ref` to make a reference to an equation. This automatically supplies the parentheses around the equation number and provides an italic correction before the closing parenthesis, if necessary.

For example,

C.R. 11
```latex
Equation \eqref{sumsq} gives the following ..........
```

produces

> Equation (8.1) gives the following ...

References can be made to individual equations in multiline displays of equations produced by such environments as align or gather (defined in the amsmath package). The `\label` command can be used within such a structure for subnumbering as in the example below:

C.R. 12
```latex
  \begin{align}
   (x+y)2&=x2+2xy+y2\label{sum}\\
   (x-y)2&=x2-2xy+y2\tag{\ref{sum}a}
\end{align}
```

$$(x+y)^2 = x^2 + 2xy + y^2 \tag{8.2}$$
$$(x-y)^2 = x^2 - 2xy + y^2 \tag{8.2a}$$

8.3 Pointing to a Page using the varioref Package

In making a reference to a table or an equation, it is more to convenient[5] to give the page number of the reference also.

[5] Of course, the convenience here is for the reader, not for the writer.

The command:

C.R. 13
```latex
\pageref{key}
```

typesets the number of the page where the command `\label{key}` was given. Therefore, for example

C.R. 14
```latex
see Table \ref{tabxy} in page \pageref{tabxy}
```

in this document produces

see Table 8.1 in page 102

To avoid the tedium of repeated by typing

```latex
\ref{key} on page \pageref{key}
```
C.R. 15 latex

you can define the **macro**:

```latex
\newcommand{\fullref}[1]{\ref{#1} on page \pageref{#1}}
```
C.R. 16 latex

and use `\fullref` for such references. However, the trouble is at times the referred object and the reference to it fall on the same page[6] so you get a reference to the page number of the very page you are reading, which looks funny.

[6] With TEX this is unfortunately not possible until the document is compiled.

This can be avoided by using the package varioref [29]. If you load this package by including the following code snippet, `\usepackage{varioref}`, in your preamble, then you can use the command:

```latex
\vref{key}
```
C.R. 17 latex

to refer to an object we've marked with `\label{key}` elsewhere in the document. The action of `\vref` varies according to the page(s) where the referred object and the references are typeset by TEX in the final output.

1. If the object and the reference are on the same page, `\vref` produces only a `\ref` suppressing `\pageref` so that only the number pointing to the object is typeset, without any reference to the page number.
2. If the object and the reference are on different pages whose numbers differ by more than one, `\vref` produces both `\ref` and `\pageref`.
3. If the object and the reference fall on pages whose numbers differ by one,[7] `\vref` produces `\ref` followed by the phrase "on the preceding page" or "on the following page" depending on whether the object or the reference occurs first. Moreover, in the next occurrence of `\vref` in a situation of the same type, the phrases are changed to "on the next page" and the "page before" respectively.

[7] that is, on successive pages

This is the default behaviour of `\vref` in the `article` class. If the article class is used with the `twoside` option or if the `book` class is used, then the behaviour in (3) above is a bit different.

1. If the object and the reference fall on the two sides of the same leaf, the behavior of `\vref` is as in (3) above.
2. If the object and the reference fall on pages forming a double spread (that is, a page of even number followed by the next page), then `\vref` produces `\ref` followed by the phrase "on the

facing page". Moreover, in the next occurence of \vref in a situation of the same type, the phrases are changed to "on the preceding page" and "on the next page" respectively.

The phrases used in the various cases considered above can be customized by redefining the commands used in generating them. For the article class without the twoside option, reference to the previous page uses the command \reftextbefore and reference to the next page uses \reftextafter.

In the case of the article class with the twoside option or the book class, the commands \reftextfaceafter and \reftextfacebefore are used in the case of reference to a page in a double spread. The default definitions of these commands are given below.

C.R. 18

```latex
\newcommand{\reftextbefore}
{on the \reftextvario{preceding page}{page before}}
\newcommand{\reftextafter}
{on the \reftextvario{following}{next} page}
\newcommand{\reftextfacebefore}
{on the \reftextvario{facing}{preceding} page}
\newcommand{\reftextfaceafter}
{on the \reftextvario{facing}{next}{page}}
```

In all these, the two (2) arguments of the command \reftextvario are phrases alternatively used in the repeated use of the reference as mentioned above. You can customize the phrases generated in various situations by redefining these with phrases of your choice in the arguments of \reftextvario.

If you want to refer only to a page number using \varioref, you can use the command:

C.R. 19

```latex
\vpageref{key}
```

to produce the page number of the object marked with \label{key}. The phrases used in the various special cases are the same as described above, except that when the referred object and the reference fall on the same page, either the phrase "on this page" or "on the current page" is produced. The command used to generate these is \reftextcurrent whose default definition is:

C.R. 20

```latex
\newcommand{\reftextcurrent}
            {on \reftextvario{this}{the current} page}
```

You can change the phrases "this" and "the current" globally by redefining this command. You can also make some local changes by using the two (2) optional arguments that \vpageref allows. Thus you can use the command:

C.R. 21

```latex
\vpageref[same page phrase][other page phrase]{key}
```

to refer to the page number of the object marked with \label{key}. The same page phrase will be

used if the object and the reference fall on the same page and the phrase other page phrase will be used, if they fall on different pages. Therefore for example, the command:

C.R. 22 latex
```
see the \vpageref[above table][table]{tabxy}
```

given in this document will produce

> see the table on page 102

8.4 The cleveref Package

Using standard cross-referencing in LaTeX only produces the label number, a name describing the label such as figure, chapter or equation has to be added manually. The **cleveref** package overcomes this limitation by automatically producing the label name and number. It further allows cross-referencing ranges of labels and multiple labels of the same or different kinds, including auto-sorting and compression of labels [30].

The package implements `\cref` for basic cross-referencing. The command is used in the same way as the standard `\ref` command. Besides the label number it also produces the reference kind.

Here is an example:

C.R. 23 latex
```
\documentclass[11pt]{article}
\usepackage{graphicx}
\usepackage{amsmath}
\usepackage{cleveref}
\begin{document}

\begin{align}
y&=a_1x+b_1\label{eqn:1}
\end{align}

\noindent
Standard equation reference (\textbackslash ref): \ref{eqn:1}\\
Cleveref equation reference (\textbackslash cref): \cref{eqn:1}

\begin{figure}[ht]\centering\rule{0.5\linewidth}{0.1\linewidth}\caption{First
 ↪  figure}\label{fig:1}\end{figure}

\noindent
Standard figure reference (\textbackslash ref): \ref{fig:1}\\
Cleveref figure reference (\textbackslash cref): \cref{fig:1}

\end{document}
```

To cross-reference multiple labels of the same or different kinds, the `\cref` command is used. Labels

are separated by commas without white-space (`\cref{ref1,ref2,etc.}`). For a range of the same label kind, the command `\crefrange{first}{last}` is available.

C.R. 24

```latex
\documentclass[11pt]{article}
\usepackage{graphicx}
\usepackage{amsmath}
\usepackage{cleveref}

\begin{document}

\begin{align}
y&=a_1x+b_1\label{eqn:1}\\
y&=a_2x+b_2\label{eqn:2}\\
y&=a_3x+b_3\label{eqn:3}\\
y&=a_4x+b_4\label{eqn:4}
\end{align}

\noindent
Range example: \crefrange{eqn:1}{eqn:4}

\begin{figure}[ht]\centering\rule{0.5\linewidth}{0.1\linewidth}\caption{First
↪  figure}\label{fig:1}\end{figure}

\noindent
Mixed references example: \cref{eqn:1,eqn:3,eqn:4,fig:1}

\end{document}
```

By default, label names are produced with a small initial letter. To capitalize the first letter at the beginning of a sentence, use `\Cref` and `\Crefrange` instead.

For capitalization of all label names throughout the document, load the package with the capitalise option.

C.R. 25

```latex
\usepackage[capitalise]{cleveref}
```

Appendix A

Supplemental Information

Table of Contents

A.1 Tables

Below are the tables which contain useful information about LaTeX.

Counter	Definition
`\arabic{counter}`	Arabic numeral
`\Roman{counter}`	Upper case Roman numeral
`\roman{counter}`	Lower case Roman numeral
`\alph{counter}`	Lower case letter
`\Alph{counter}`	Upper case letter
`\fnsymbol{counter}`	A footnote symbol

Table A.1: Different kinds of counters supported by LaTeX.

Character Code	Description
\char	Explicit denotation of a character to be typeset.
\chardef	Define a control sequence to be a synonym for a character code.
\accent	Command to place accent characters.
\if	Test equality of character codes.
\ifx	Test equality of both character and category codes.
\let	Define a control sequence to be a synonym of a token.
\uccode	Query or set the character code that is the uppercase variant of a given code.
\lccode	Query or set the character code that is the lowercase variant of a given code.
\uppercase	Convert the general text argument to its uppercase form.
\lowercase	Convert the general text argument to its lowercase form.
\string	Convert a token to a string of one or more characters.
\escapechar	Number of the character that is to be used for the escape character when control sequences are being converted into character tokens.

Table A.2: String/numerical operations supported by LaTeX.

Category code	Description	LaTeX Equivalent
0	Escape character-tells TEX to start looking for a command	c
1	Start a group	`\{`
2	End a group	`\}`
3	Math shift-switch in/out of math mode	`$`
4	Alignment tab	`&`
5	End of line	ASCII code 13 ()
6	Macro parameter	`##`
7	Superscript-for typesetting math: `$y=x^2$`	`^`
8	Subscript-for typesetting math: `$y=x_2$`	`_`
9	Ignored character	ASCII 0 <null>
10	Spacer	ASCII codes 32 (space) and 9 (tab character)
11	Letter	A...Z, a...z, (and thousands of Unicode characters)
12	Other	0...9 plus ,.;?" and many others
13	Active character	Special category code for creating single-character macros such as ~
14	Comment character-ignore everything that follows until the end of the line	%
15	Invalid character, not allowed to appear in the .tex input file	ASCII code 127 (DEL)

Glossary

ACM Association for Computing Machinery. 6

AMS American Mathematical Society. 6, 8

CTAN Comprehensive TEX Archive Network. 5, 7, 18

EPS Encapsulated PostScript. 83

NFSS New Font Selection Scheme. 8

OS Operating System. 5, 12

PDF Portable Document Format. 10, 18, 85

Bibliography

[1] 2025, *Ctan lion*, Bibby. [Online]. Available: https://ctan.org/lion?lang=en.

[2] F. Mittelbach, M. Goossens, J. Braams, D. Carlisle, and C. Rowley, *The LATEX companion*. Addison-Wesley Professional, 2004.

[3] D. E. Knuth, *Computers & Typesetting*. Addison-Wesley Reading, MA, 1986.

[4] D. E. Knuth, "Literate programming," *The computer journal*, vol. 27, no. 2, pp. 97–111, 1984.

[5] D. E. Knuth, *The METAFONT book*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[6] D. E. Knuth and D. Bibby, *The texbook*. Addison-Wesley Reading, 1984, vol. 15.

[7] D. E. Knuth, *TEX and METAFONT: New directions in typesetting*. American Mathematical Society, 1979.

[8] D. E. Knuth, *The future of tex and met afont*, 1990.

[9] B. K. Reid, "Scribe document production system user manual," Technical Report, Unilogic, Ltd, Tech. Rep., 1984.

[10] L. Lamport and A LaTex, *Document preparation system*, 1994.

[11] J. Braams, "Babel, a multilingual style-option system for use with latex's standard document styles," *TUGboat*, vol. 12, no. 2, pp. 291–301, 1991.

[12] F. Mittelbach, *Mittelbach tex stackexchange*, 2025. [Online]. Available: https://tex.stackexchange.com/users/10109/frank-mittelbach.

[13] F. Mittelbach and C. Rowley, "The latex3 project," *MAPS*, vol. 93, pp. 95–99, 1993.

[14] D. Carlisle, "Alatex tour, part 2: The tools and graphics distributions," *TUGboat*, vol. 17, no. 3, pp. 321–326, 1996.

[15] Timothy, *The latex project logo, found on its official website*, 2020. [Online]. Available: https://commons.wikimedia.org/wiki/File:LaTeX_project_logo_bird.svg.

[16] QuantaMagazine, *Deconstructing the bakery to build a distributed state machine,* 2022. [Online]. Available: https://www.youtube.com/watch?v=-v9uvhJe7_4.

[17] D. R. Wilkins, "Getting started with latex," *Copyright David R. Wilkins*, vol. 9, 1995.

[18] N. L. Services, *Getting started with latex*, 2025. [Online]. Available: https://guides.nyu.edu/LaTeX/installation.

[19] texstudio, *Texstudio guide*, 2025. [Online]. Available: https://texstudio-org.github.io/getting_started.htmlx.

[20] Manske, 2008. [Online]. Available: https://commons.wikimedia.org/wiki/File:Monotype-machine.jpg.

[21] Panoramafotos, *Monotype-taster, modell "taster d", baujahr um 1965*, 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:Monotype-Taster-1965.jpg.

[22] beeton. "Typesetting before tex and computers? - tex.stackexchange. "[Online]. Available: https://tex.stackexchange.com/questions/244082/typesetting-before-tex-and-computers.

[23] D. Rhatigan et al., *The monotype 4-line system for setting mathematics*, 2007.

[24] A. Boag, "Monotype and phototypesetting," *Journal of the Printing Historical Society*, vol. 2, pp. 57–77, 2000.

[25] J. Boniface, "Leopold kronecker's conception of the foundations of mathematics," *Philosophia Scientiae*, vol. 9, no. S2, pp. 143–156, 2005.

[26] latextables, *Tabularray*, 2025. [Online]. Available: https://www.latex-tables.com/ressources/tabularray.html.

[27] S. Nielsen, "Mediostructures in bilingual lsp dictionaries," *Lexicographica*, vol. 15, no. 1999, pp. 90–113, 1999.

[28] S. Empiricus, *Outlines of pyrrhonism*. Rowman & Littlefield, 2023.

[29] F. Mittelbach, "The varioref package," *Part of the LATEX 2ε distribution*, 1995.

[30] TomTeXblog, *Cleveref, a clever way to reference in latex*, 2013. [Online]. Available: https://texblog.org/2013/05/06/cleveref-a-clever-way-to-reference-in-latex/.