மனோன்மணியம் சுந்தரனார் பல்கலைக்கழகம்

MANONMANIAM SUNDARANAR UNIVERSITY
TIRUNELVELI-627 012

தொலைநிலை தொடர் கல்வி இயக்ககம்

DIRECTORATE OF DISTANCE AND
CONTINUING EDUCATION

M.Sc., Mathematics (I Year)

Mathematical Documentation using LaTeX

(Skill Enhancement Course – I)

Study Learning Material Prepared by
Dr. I. Valliammal, M.Sc., M.Phil., Ph.D.,
Assistant Professor of Mathematics,
Manonmaniam Sundaranar University, Tirunelveli-12

Sub. Code: SMAS21 **(For Private Circulation only)**

M.Sc., Mathematics –I YEAR

SMAS21– Mathematical Documentation using LaTeX

(Skill Enhancement Course – I)

SYLLABUS

Unit-I:

Introduction - Basics of a Latex file- Text, Symbols and Commands: Command names and arguments – Environments– Declarations – Lengths – Special characters.

Unit-II:

Document Layout and Organization: Document class – Page style – Parts of the document – Table of contents.

Unit-III:

Displayed Text: Changing font style – Centering and indenting – Lists – Generalized lists Theorem like-declarations.

Unit-IV:

Text in Boxes: Boxes - Footnotes and marginal notes. Tables: Tabular stops – Tables.

Unit-V:

Mathematical Formulas: Mathematical Environment – Main elements of math mode – Mathematical symbols – Additional Elements.

Recommended Text:

Guide to LaTeX, Helmut Kopka and Patrick W.Daly, Fourth Edition, Addison - Wesley, Pearson Education, 2004.


KNOWLEDGE IS POWER

SMAS21– Mathematical Documentation using LaTeX

CONTENTS

UNIT-1

1.1	Introduction	5
1.2	Basics of a Latex file	6
1.3	Text, Symbols and Commands	9
1.4	Command names and arguments	9
1.5	Environments	12
1.6	Declarations	13
1.7	Lengths	14
1.8	Special Characters	15

UNIT-2

2.1	Document Layout and Organization	23
2.2	Document class	23
2.3	Page style	29
2.4	Parts of the document	42
2.5	Table of contents	49

UNIT-3

3.1	Displayed Text	53
3.2	Changing font style	53
3.3	Centering and indenting	61
3.4	Lists	64
3.5	Generalized lists	69
3.6	Theorem like-declarations	73

UNIT-4

4.1	Text in Boxes	75
4.2	Boxes	75
4.3	Footnotes and marginal notes	84
4.5	Tabular stops	89
4.6	Tables	94

UNIT-5

5.1	Mathematical Formulas	101
5.2	Mathematical Environment	101
5.3	Main elements of math mode	103
5.4	Mathematical symbols	108
5.5	Additional Elements	114

Unit-I:

Introduction - Basics of a Latex file- Text, Symbols and Commands: Command names and arguments – Environments– Declarations – Lengths – Special characters.

1.1 Introduction:

LATEX is a comprehensive set of markup commands used with thepowerful typesetting program TEX for the preparation of a widevariety of documents, from scientific articles, reports, to complexbooks.

LATEX like TEX is an open software system, available free of charge.Its core is maintained by the LATEX3 Project Group but it also benefitsfrom extensions written by hundreds of user/contributors, with allthe advantages and disadvantages of such a democracy.

A LATEX document consists of one or more source files containingplain text characters, the actual textual content plus markupcommands. These include instructions which can insert graphicalmaterial produced by other programs.

It is processed by the TEX program to produce a binary file in DVI(*device independent*) format, containing precise directions for thetypesetting of each character. This in turn can be viewed on a monitor,or converted into printer instructions, or some other electronicform such as PostScript, HTML, XML, or PDF.

A variant on the TEX program called pdfTEX produces PDF outputdirectly from the source file without going through the DVI intermediary.With this, LATEX can automatically include internal linksand bookmarks with little or no extra effort, plus PDF buttons andexternal links, in addition to graphics in a wide range of common

formats.

TEX activities are coordinated by the TEX Users Group, TUG (www.tug.org) who distribute a set of CDs, called TEXLive, annually to itsmembers, containing a TEX/LATEX installation for various computertypes.

1.2 Basics of a Latex file:

1.2.1 Text and commands:

The *source file* for LATEX processing, or simply the *LATEX file*, contains the*source text* that is to be processed to produce the printed output. Splittingthe text up into lines of equal width, formatting it into *paragraphs*, andbreaking it into *pages* with page numbers and running heads are allfunctions of the processing program and not of the input text itself.

For example, words in the source text are strings of letters terminatedby some non-letter, such as *punctuation*, *blanks*, or *end-of-lines* (*hard*end-of-lines, ones that are really there, not the *soft* ones that move withthe window width); whereas punctuation marks will be transferred to theoutput, blanks and end-of-lines merely indicate a gap between words.Multiple blanks in the input, or blanks at the beginning of a line, have noeffect on the interword spacing in the output.

Similarly, a new paragraph is indicated in the input text by an emptyline; multiple empty lines have the same effect as a single one. In theoutput, the paragraph may be formatted either by indentation of the firstline, or by extra interline spacing, but this is not affected in any way bythe number of blank lines or extra spaces in the input.

The source file contains more than just text, however; it is also interspersedwith markup commands that control the formatting or indicatethe structure. It is therefore necessary for the author to be able to recognizewhat is text and what is a command. Commands consist eitherof certain single characters that cannot be used as text characters, or ofwords preceded immediately by a special character, the backslash (\).

1.2.2 Contents of a LATEX source file:

Every LATEX file contains a *preamble* and a *body*.

The preamble is a collection of commands that specify the globalprocessing parameters for the following text, such as the paper format,the height and width of the text, the form of the output page with itspagination and automatic page heads and

footlines. As a minimum, thepreamble must contain the command \documentclass to specify thedocument's overall processing type. This is the first command in the preamble.

If there are no other commands in the preamble, LATEX selects standardvalues for the line width, margins, paragraph spacing, page height andwidth, and much more. By default, these specifications are tailored tothe American norms. For European requirements, built-in options exist toalter the text height and width to the A4 standard. Furthermore, there arelanguage-specific packages to translate certain headings such as 'Chapter'and 'Abstract'.

The preamble ends with \begin{document}. Everything that followsthis command is interpreted as *body*. It consists of the actual text mixedwith markup commands. In contrast to those in the preamble, thesecommands have only a local effect, meaning they apply only to a part ofthe text, such as *indentation*, *equations*, temporary change of *font*, and soon. The body ends with the command \end{document}. This is normallythe end of the file as well.

The general syntax of a LATEX file is as follows:

\documentclass[*options*]{*class*}
    Further global commands and specifications
\begin{document}
    Text mixed with additional commands of local effect
\end{document}

A minimal LATEX file named hi.tex contains just the following lines:

\documentclass{article}
\begin{document}
    Hi!
\end{document}

1.2.3 Extending LATEX with packages:

Packages are a very important feature of LATEX. These are extensions tothe basic LATEX commands that are written to files with names that endin.sty and are loaded with the command \usepackage in the preamble.Packages can be classified by their origin:

core packages are an integral part of the LATEX basic installation and are therefore fully standard;

tools packages are a set written by members of the LATEX3 Team, andshould always be in the installation;

graphics packages are a standardized set for including pictures generatedby other programs, and for handling color; they are on the samelevel as the tools packages;

AMS-LATEX packages published by the American Mathematical Society, should be in any installation;

contributed packages have been submitted by actual users; certain ofthese have established themselves as ‘essential’ to standard LATEXusage, but all are useful.

There are over 1000 contributed packages on the included TEXLiveCD.How can one begin to get an overview of what they offer? GrahamWilliamshas compiled a list of brief descriptions which can be found online andon the TEXLive CD at

\texmf\doc\html\catalogue\catalogue.html

Documentation of contributed packages is somewhat haphazard, dependingon how much the author has put into it. The preferred methodfor distributing packages is to integrate the documentation with thecode into a single file with extension .dtx. A special program DocStrip is used to extract the actual package file or files, whileLATEXing the original .dtx file produces the instruction manual. Mostready-to-run installations will already have done all this for the user,with the resulting manuals

stored as DVI or PDF files somewhere in\texmf\doc\latex\. . . . However, we might have to generate the documentationoutput ourselves by processing the.dtx file, which should befound in \texmf\source\latex\. . . .

Some package authors write their manuals as an extra .tex file, theoutput of which may or may not be prestored in DVI or PDF form. Othersprovide HTML files. And still others simply add the instructions ascomments in the package file itself. (This illustrates some of the joys ofan open system.)

1.3 Text, Symbols and Commands:

The text that is to be the input to a LATEX processing run is written to a*source file* with a name ending in.tex, the file name extension. This file isprepared with a *text editor*, either one that handles straightforward plaintext, or one that is configured to assist the writing and processing of LATEXfiles. In either case, the contents of this file are plain ascii charactersonly, with no special symbols, no accented letters, preferably displayed ina fixed width typewriter font, with no frills like bold or italics, all in onesize. All these aspects of true typesetting are produced afterwards bythe TEX processing program with the help of *markup* commands insertedvisibly into the actual text. It is therefore vital to know how commandsare distinguished from text that is to be printed, and, of course, how theyfunction.

1.4 Command names and arguments:

A *command* is an instruction to LATEX to do something special, like printsome symbol or text not available to the restricted character set usedin the input file, or to change the current typeface or other formattingproperties. There are three types of command names:

- the single characters # $ & ~ _ ^ % { } all have special meaningsthat are explained later;
- the backslash character \ plus a single non-letter character; forexample \$ to print the $ sign; all the special characters listed abovehave a corresponding two-character command to print them literally;

• the backslash character \ plus a sequence of letters, ending with thefirst non-letter; for example, \large to switch to a larger typeface.Command names are case sensitive, so \large, \Large and \LARGEare distinct commands.

Many commands operate on some short piece of text, which thenappears as an *argument* in curly braces following the command name.For example, \emph{stress} is given to print the word stress in anemphasized typeface (here italic) as *stress*. Such arguments are said to be*mandatory*because they must always be given.

Some commands take *optional* arguments, which are normally employedto modify the effects of the command somehow. The optionalarguments appear in square braces.

In this book we present the general syntax of commands as

name[*optional*]{*mandatory*}

where typewriter characters must be typed exactly as illustrated anditalic text indicates something that must be substituted for. Optionalarguments are put into square brackets [ ] and the mandatory ones intocurly braces { }. A command may have several optional arguments, eachone in its set of brackets in the specified sequence. If none of the optionalarguments is used, the square brackets may be omitted. Any number ofblanks, or even a single new line, may appear between the command nameand the arguments, to improve legibility.

Some commands have several mandatory arguments. Each one mustbe put into a { } pair and their sequence must be maintained as given inthe command description. For example,

\rule[*lift*]{*width*}{*height*}

produces a black rectangle of size *width* and *height*, raised by an amount*lift*above the current baseline. A rectangle of width 10 mm and height3 mm is made with \rule{10mm}{3mm}. Since the optional argument *lift*is omitted, the rectangle is set on

the baseline with no lifting, asThe arguments must appear in the order specified by the syntax and maynot be interchanged.

Some commands have a so-called *-form in addition to their normalappearance. A * is added to their name to modify their functionalitysomehow. For example, the \section command has a *-form \section*which, unlike the regular form, does not print an automatic section number.For each such command, the difference between the normal and*-form will be explained in the description of the individual commands.

Command names consist only of letters, with the first non-letter indicatingthe end of the name. If there are optional or mandatory argumentsfollowing the command name, then it ends before the [or { bracket,since these characters are not letters. Many commands, however, possessno arguments and are composed of only a name, such as the command\LaTeX to produce the LATEX logo. If such a command is followed bya punctuation mark, such as comma or period, it is obvious where thecommand ends. *If it is followed by a normal word, the blank betweenthe command name and the next word is interpreted as the commandterminator:* The \LaTeX logo results in
'The LaTeX logo', that is, the blankwas seen only as the end of the command and not as spacing betweentwo words. This is a result of the special rules for blanks.

In order to insert a space after a command that consists only of a name,either an empty structure {} or a space command (\ and blank) must beplaced after the command. The proper way to 'The LaTeX logo', produce isto type either The \LaTeX{} logo or The \LaTeX\ logo. Alternatively,the command itself may be put into curly braces, as The {\TeX} logo,which also yields the right output with the inserted blank:
'The TeX logo'. Incidentally, the LaTeX 2ε logo is produced with \LaTeXe.

1.5 Environments:

An *environment* is initiated with the command \begin{*name*} and isterminated by \end{*name*}.

An environment has the effect that the text within it is treated differentlyaccording to the environment parameters. It is possible to alter(temporarily) certain processing features, such as indentation, line width,typeface, and much more. Thechanges apply only within the environment.For example, with the quote environment,

previous text
\begin{quote}
text1 \small *text2* \bfseries*text3*
\end{quote}
following text

the left and right margins are increased relative to those of the previousand following texts. In the example, this applies to the three texts *text1*,*text2*, and *text3*. After *text1* comes the command \small, which has theeffect of setting the next text in a smaller typeface. After *text2*, there is anadditional command \bfseries to switch to bold face type. Both thesecommands only remain in effect up to the \end{quote}.

> The three texts within the quote environment are indented on both sides relative to the previous and following texts. The *text1* appears in the normal typeface, the same one as outside the environment. The *text2* and *text3* appear in a smaller typeface, and *text3* **furthermore appears in bold face.**

After the end of the quote environment, the subsequent text appears inthe same typeface that was in effect beforehand.

Note that if the names of the environment in the \begin{..} \end{..}pair do not match, an error message will be issued on processing.

Most declaration command names may also be usedas environment names. In this case the command name is used *without*the preceding\ character. For example, the command \em switches toan emphatic typeface, usually *italic*, and the corresponding environment\begin{em} will set all the text in *italic* until \end{em} is reached.

A nameless environment can be simulated by a {...} pair. The effectof any command within it ends with the closing curly brace.

1.6 Declarations:

A *declaration* is a command that changes the values or meanings ofcertain parameters or commands without printing any text. The effect ofthe declaration begins immediately and ends when another declarationof the same type is encountered. However, if the declaration occurswithin an environment or a {...} pair, its scope extends only to thecorresponding \end command, or to the closing brace }. The commands\bfseries and \small mentioned in the previous section are examplesof such non-printing declarations that alter the current typeface.

Some declarations have associated arguments, such as the command\setlength which assigns a value to a *length parameter*.

Examples:

{\bfseries This text appears in bold face} The \bfseriesdeclarationchanges the typeface: **This text appears in bold face**. Theeffect of this declaration ends with the closing brace}.

\setlength{\parindent}{0.5cm} The paragraph indentation is set to0.5 cm. The effect of this declaration ends with the next encounterof the command \setlength{\parindent}, or at the latest withthe \end command that terminates the current environment.

\pagenumbering{roman} The page numbering is to be printed in Romannumerals.

Some declarations, such as the last example, are global, that is, theireffects are not limited to the current environment. The following declarationsare of this nature, the meanings of which are given later:

\newcounter *\pagenumbering* *\newlength*
\setcounter *\thispagestyle* *\newsavebox*
\addtocounter

Declarations made with these commands are effective right away andremain so until they are overridden by a new declaration of the same type.In the last example above, page numbering will be done in Roman numeralsuntil countermanded by a new *\pagenumbering{arabic}* command.

1.7 Lengths:

1.7.1 Fixed lengths:

Lengths consist of a decimal number with a possible sign in front (+ or-) followed by a mandatory dimensional unit. Permissible units and theirabbreviated names are:

cm centimeter,
mm millimeter,
in inch (1 in = 2.54 cm),
pt point (1 in = 72.27 pt),
bp big point (1 in = 72 bp),
pc pica (1 pc = 12 pt),
dd didˆot point (1157 dd = 1238 pt),
cc cicero (1 cc = 12 dd),
em a font-specific size, the width of the capital M,
ex another font-related size, the height of the letter *x*.

Decimal numbers in TEX and LATEX may be written in either the Englishor European manner, with a *period* or a *comma*: both 12.5cm and 12,5cmare permitted.

Note that 0 is not a legitimate length since the unit specification ismissing. To give a zero length it is necessary to add some unit, such as0pt or 0cm.

Values are assigned to a length parameter by means of the LATEX command\setlength along with othercommands for dealing with lengths. Its syntax is:

\setlength{\length name}{length spec}

For example, the width of a line of text is specified by the parameter\textwidth, which is normally set to a default value depending on theclass, paper type, and font size. To change the line width to be 12.5 cm,one would give:

\setlength{\textwidth}{12.5cm}

1.7.2 Rubber lengths:

Some parameters expect a *rubber* length. These are lengths that can bestretched or shrunk by a certain amount. The syntax for a rubber lengthis:

*nominal_value*plus*stretch_value*minus*shrink_value*

where the *nominal_value*, *stretch_value*, and *shrink_value*are each a length.
For example,

\setlength{\parskip}{1ex plus0.5ex minus0.2ex}

means: the extra line spacing between paragraphs, called \parskip, is tobe the height of the *x* in the current font, but it may be increased to 1.5or reduced to 0.8 times that size.

One special *rubber* length is \fill. This has the natural length of*zero*but can be stretched to any size.

1.8 Special characters:

1.8.1 Spaces:

The *space* or *blank* character has some properties different from thoseof normal characters. During processing, blanks in the input text are replaced byrubber lengths in

order to allow the line to fill up to thefull line width. As a result, some peculiar effects can occur if one is notaware of the following rules:

- one blank is the same as a thousand, only the first one counts;
- blanks at the beginning of an input line are ignored;
- blanks terminating a command name are removed;
- the end of a line is treated as a blank.

Some of the consequences of these rules are that there may be as manyblanks as desired between words or at the beginning of a line (to makethe input text more legible) and that a word may come right at the end ofa line without the spacing between it and the next word disappearing. Toforce a space to appear where it would otherwise be ignored, one mustgive the command \␣ (a \ followed by a space character, made visible hereby the symbol␣).

To ensure that certain words remain together on the same line, a *protectedspace*is inserted between them with the ~ character. Multiple protected spaces are all printed out, in contrast tonormal spaces.

Sometimes it is necessary to suppress the space that appears becauseof the new line. In this case, the last character in the line must be the*comment*character %.

Paragraphs are separated in the source text by blank lines. As forblank characters, one blank line is the same as a thousand.

Instead of a blank line, the command \par may also be used to indicatethe end of a paragraph.

1.8.2 Quotation marks:

The *quotation marks* found on the typewriter " are not used in bookprinting. Instead different characters are used at the beginning and end,such as ‘single quotes’ and “double quotes”. Single quotes are producedwith ‘ and ’, while double quotes are made by typing the respectivecharacters twice: ‘‘ for “ and ’’ for ”. Furthermore the

typewritercharacter " will also generate the double closing quote ”. However, itshould be avoided since it can lead to confusion.

1.8.3 Hyphens and dashes:

In book printing, the character that appears on the typewriter as –comesin various lengths: -, –, —. The smallest of these, the *hyphen*, is usedfor compound words such as *father-in-law* and for word division at theend of a line; the middle-sized one, the *en dash*, is used in ranges ofnumbers, for example, pages 33–36; and the largest, the *em dash*, is usedas punctuation—what is normally called the *dash*. These are generatedby typing the hyphen character one, two, or three times, so that – yields-, while -- makes –, and --- produces —. A fourth type of dash is theminus sign −, whichis entered in math mode as $-$.

1.8.4 Printing command characters:

The characters # $ ˜ _ ˆ % { } are interpretedas commands. To print them as text, one must give a commandconsisting of \ plus that character.

$ = \$ & = \& % = \% # = \# - = _

{ = \{ } = \}

1.8.5 The special characters §, †, ‡, ¶,©and £:

These special characters do not exist on the computer keyboard. Theycan however be generated by special commands as follows:

§ = \S † = \dag ‡ = \ddag ¶ = \P ©= \copyright

£ = \pounds

1.8.6 Non-English letters:

Special letters that exist in languages other than English can also begenerated with TEX. These are:

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

Å*ngstrøm*may be written as {\AA}ngstr{\o}m while *Karlstraße*canbeinput as Karlstra{\ss}e. The ‘letter’ \SS is the upper case equivalentof \ss, used for automatic conversion between upper and lower case.

1.8.7 Accents:

In non-English languages, there is a multiplicity of *diacritical marks* or*accents*, most of which can be printed with TEX:

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

The *o* above is given merely as an example: any letter may be used. With*i*and*j* it should be pointed out that the dot must first be removed. Thisis carried out by prefixing these letters with a backslash: the commands

\i and \j yield ı and ȷ. In this way ĭ and ǰ are formed by typing \u{\i} and \H{\j}.

The accent commands consisting of a non-letter may also be givenwithout the curly braces:

ò=\‘o ó=\’o ô=\^o ö=\"o õ=\~o ō=\=o ȯ=\.o

The letter accent commands should always be used with the curly braces.

1.8.8 The euro symbol:

The euro € (or €) symbol is too new to be part of the original LATEX, but itcan be produced with the help of some additional fonts and contributedpackages.

Just which package we may use depends on our installation,and whether we have access to these additional fonts.

The *Text Companion* fontscontainaeuro symbol. Since these fonts should be part of every modern LATEXinstallation, we should be able to use the euro symbol if all else fails.

The package textcomp must be loaded in the preamble withtextcomp \usepackage{textcomp}which defines many commands including \texteuro to print the symbol€. Since the European Commission originally dictated that it should only be printed in a sans serif font, it is better to issue \textsf{\texteuro} toproduce €. If we are going to use this very frequently, we might want to define ashortcut named \euro with

\newcommand{\euro}{\textsf{\texteuro}}

on defining commands.

A better solution is presented by the eurosym package by Henrikeurosym Theiling and the associated fonts that come with it, which bear the namesfeymr10, feybr10, and soon. This package defines the \euro commandto print e, which changes automatically to bold e and slanted e asneeded.

The europs package by Joern Clausen interfaces to the type 1 (PosteuropsScript) euro fonts published by Adobe. For licensing reasons, these fontsmay only be obtained from Adobe directly, even though free of charge. This package provides the command \EUR for a symbolthat varies with font family (Roman E25, sans serif E25, and typewriterE25) as well as for bold **E**25 and slanted *E25*. There is also a command\EURofc for the invariable symbol E.

Finally, the package eurosans by Walter Schmidt also addresses theeurosans Adobe euro fonts, again with the command \euro, with the same behavioras that of eurosym: always sans serif family, but changes with the otherfont attributes.

The table below summarizes the above packages:

Package	Command	Fonts	Notes
`textcomp`	`\texteuro`	Text Companion	Non standard symbol
`eurosym`	`\euro`	Eurosym	Sans serif, variable
`europs`	`\EUR` `\EURofc`	PostScript	Varies with font family Invariable, official
`eurosans`	`\euro`	PostScript	Sans serif, variable

1.8.9 Ligatures:

Certain groups of characters, when typeset, are joined together—suchcompoundcharactersare called *ligatures.* There are five ligatures that LATEX typesets automatically(if we use the Computer Modern fonts): ff, fi, fl, ffi, and ffl. If we want to prevent LATEX from forming a ligature, separate the characters with the command*textcompwordmark*. Compare *iff* with *if f*, typed as *iff* and

if\textcompwordmark f

Enclosing the second character in braces ({}) is a crude method of preventing the ligature.

1.8.10 Logos and dates:

TeX produces TEX, *LaTeX* produces LATEX, and *LaTeXe* produces LATEXe (theoriginal name of the current version of LATEX). The *AmS* command produces the logo*AMS*.

Remember to type \TeX\␣ or \TeX{} if we need a space after TEX (similarly forthe others).

- *time* is the time of day in minutes since midnight
- *day* is the day of the month
- *month* is the month of the year
- *year* is the current year

We can include these numbers in our document by using the *the* command:

Year: *the**year*; month: *the**month*; day: *the**day*

produces a result such as

Year: 2015; month: 7; day: 11

Of more interest is the *today* command, which produces today's date in the form: July 11, 2015.


KNOWLEDGE IS POWER

Unit-II:

Document Layout and Organization: Document class – Page style – Parts of the document – Table of contents.

2.1 Document Layout and Organization:

2.2 Document class:

The first command in the preamble of a LATEX file determines the globalprocessing format for the entire document.

Its syntax is:

\documentclass[*options*]{*class*}

where some value of *class* must be given, while [*options*] may be omittedif default values are acceptable.

The standard values of *class*, of which one and only one may be given,are: book, report, article, or letter. The basic differences between theseclasses lie not only in the page layouts, but also in the organization. Anarticle may contain *parts*, *sections*, *subsections*, and so on, while a reportcan also have *chapters*. A book also has chapters, but treats even andodd pages differently; also, it prints running heads on each page with thechapter and section titles.

Other classes besides the standard ones exist, as contributions forspecific journals, or for book projects. These will have their own set of*options* and additional commands, which should be described in separatedocumentation or instructions. However, since they are normally modificationsof one of the standard classes, most of the following optionsapply to them too.

2.2.1 Standard class options:

The *options* available allow various modifications to be made to the formatting. They can be grouped as follows.

Selecting font size:

The basic font size is selected with one of the options

10pt 11pt 12pt

This is the size of the font in which the normal text in the document willbe set. The default is 10pt, which means that this is the value assumed ifno size option is specified. All other font size declarations are relative tothis standard size, so that the section titles, footnotes, and so on will allchange size automatically if a different basic font size is selected.

Specifying paper size:

LATEX calculates the text line width and lines per page according to theselected font size and paper mode. It also sets the margins so that thetext is centered both horizontally and vertically. In order to do this, itneeds to know which paper format is being used. This is specified by oneof the following options:

letterpaper (11 × 8.5 in)
legalpaper (14 × 8.5 in)
executivepaper (10.5 × 7.25 in)
a4paper (29.7 × 21 cm)
a5paper (21 × 14.8 cm)
b5paper (25 × 17.6 cm)

The default is letterpaper, American letter size paper, 11 × 8.5 in.

Normally, the paper format is such that the longer dimension is thevertical one, the so-called *portrait* mode. With the option

landscape

the shorter dimension becomes the vertical one, the *landscape* mode.(One still has to ensure that the output is printed as landscape.)

Page formats:

The text on the page may be formatted into one or two columns with the options

onecolumn twocolumn

The default is onecolumn. In the case of the twocolumn option, theseparation between the columns as well as the width of any rule betweenthem may be specified by \columnsep and \columnseprule, describedbelow.

The even- and odd-numbered pages may be printed differently accordingto the options

oneside twoside

With oneside, all pages are printed the same; however, with twoside, therunning heads are such that the page number appears on the right on oddpages and on the left on even pages. *It does not force the printer to outputdouble-sided.* The idea is that when these are later printed back-to-back,the page numbers are always on the outside where they are more easilynoticed. This is the default for the book class. For article and report,the default is oneside.

With the book class, chapters normally start on a right-hand, odd-numbered page. The options

openright openany

control this feature: with openany a chapter always starts on the nextpage, but with openright, the default, a blank page may be inserted ifnecessary.

Normally the title of a book or report will go on a separate page, whilefor an article, it is placed on the same page as the first text. With theoptions

notitlepage titlepage

this standard behavior may be overruled.

Further options:

The remaining standard options are:

Leqno:

Equation numbers in displayed formulas will appear on the leftinstead of the normal right side.

Fleqn:

Displayed formulas will be set flush left instead of centered. The amount of indentation may be set with the parameter \mathindent described below.

openbib:

The format of bibliographies may be changed so that segmentsare set on new lines. By default, the texts for each entry are runtogether.

draft:

If the LATEX line-breaking mechanism does not function properlyand text must stick out into the right margin, then this is markedwith a thick black bar, to make it noticeable.

final:

The opposite of draft, and the default. Lines of text that are toowide are not marked in any way.

If multiple options are to be given, they are separated by commas, asfor example, \documentclass[11pt,twoside,fleqn]{article}. Theorder of the options is unimportant. If two conflicting options are specified,say oneside and twoside, it is not obvious which one will beeffective. That depends entirely on the definitions in the class file itself,so it would be best to avoid such situations.

Parameters associated with some options:

Some options make use of parameters that have been given certain default values:

\mathindent

specifies the indentation from the left margin for the equationnumbers when fleqnis selected;

\columnsep

specifies the space between the two columns for the twocolumnoption;

\columnseprule

determines the width of the vertical line between the two columnsfor the twocolumn option. The default is zero width, that is, no vertical rule.

The standard values of these parameters may be changed with theLATEX command \setlength. For example, to change \mathindent to2.5 cm, give

\setlength{\mathindent}{2.5cm}

These parameters may be assigned values either in the preamble orat any place in the document. Parameters in the preamble apply to theentire document, whereas those within the text are in effect until the nextchange or until the end of the environment in which they were made. In the latter case, the previous values become effective oncemore.

2.2.2 Loading packages:

A package is nothing more than a set of LATEX (or TEX) commandsstored in a file with the extension.sty, although there are some specialcommands that may only appear within them. To invoke a package, simplycall

\usepackage{*package*}

in the preamble, where *package* is the root name of the file. More than onepackage may be loaded with one call to \usepackage. For example, twopackages provided with standard LATEX are stored in files makeidx.sty and ifthen.sty. They may be read intogether with

\usepackage{makeidx,ifthen}

A package may have options associated with it, which may be selectedin the same way as for document classes: by including the option nameswithin square braces. The general syntax is thus:

\usepackage[*opt1,opt2. . .*]{*package1,package2,. . .* }

where all the listed options will be applied to all the selected packages.If any of the packages does not understand one of the options, a warningmessage is output to the monitor.

2.2.3 Global and local options:

One interesting feature about options specified with the \documentclasscommand is that they also apply to any packages that follow. This means thatif several packages all take the same option, it is only necessary to declare itonce in \documentclass. For example, one might design a package to modifyarticle for generating a local house style that might do different things forsingle or double column text; this package could make use of the class optionsonecolumn and twocolumn to achieve this. Or it could elaborate on the draftoption to produce double line spacing, as for a manuscript. Alternatively, severalpackages might have language-dependent features that could be activatedwith options like french or german; it is sufficient to list such options only in\documentclass to apply them to all packages. Such options are called *global*,for they are passed on to all subsequent packages automatically.

Global options need not be limited to the standard class options. A warning message is printed only if neither the class nor anyof the packages understand one or more of them. By contrast, any optionsspecified with \usepackage will be applied only to those packages listed in thatone command; and it is applied to all of them. A warning is printed if one ormore of those packages does not recognize any one of these *local* options.

2.2.4 Class and package versions:

Class and package files normally have an internal version specification in theform of their release date, as *yyyy*/*mm*/*dd*. If we wish to make use of somefeature that we know was added on a certain date, we include that date insquare brackets after the class or package name.

The version date may also be added to the \documentclass command toensure that the right version of the class file is being employed. The reason fordoing this is to ensure that the source files are processed properly, say on othersystems.

2.3 Page style:

The basic page format is determined by the *page style*. With one exception,this command is normally given in the preamble. Its form is:

\pagestyle{*style*}

The mandatory argument *style* takes on one of the following values:

plain:

The page head is empty, the foot contains the centered pagenumber. This is the default for the article and report classeswhen no \pagestyle is given in the preamble.

empty:

Both head and footlines are empty; no page numbers are printed.

headings:

The head contains the page number as well as title information(chapter and section headings); the foot is empty. This is thedefault for book class.

myheadings:

The same as headings except that the page titles in the head arenot chosen automatically but rather are given explicitly by thecommands \markright or \markboth (see below).

The command

\thispagestyle{*style*}

functions exactly as \pagestyle except that it affects only the currentpage. For example, the page numbering may be suppressed for just thecurrent page with the command \thispagestyle{empty}. It is only the*printing* of the page number that is suppressed; the next page will benumbered just as though the command had never been given.

2.3.1 Heading declarations:

For the page styles headings and myheadings, the information appearingin the headline may be given with the declarations

\markright{*right head*}

\markboth{*left head*}{*right head*}

The declaration \markboth is used with the document class optiontwoside, with even-numbered pages considered to be on the *left* andodd-numbered pages on the *right*. Furthermore, the page number isprinted on the left side of the head for a left page and on the right sidefor a right page.

For one-sided output, all pages are considered to be right-handed. Inthis case, the declaration \markright is appropriate. It may also be usedwith two-sided output to overwrite the *right head* given in \markboth.

With the page style headings, the standard titles in the page headlineare the chapter, section, or subsection headings, depending on thedocument and page style, according to the following scheme:

Style		Left Page	Right Page
book, report	one-sided	—	*Chapter*
	two-sided	*Chapter*	*Section*
article	one-sided	—	*Section*
	two-sided	*Section*	*Subsection*

If there are more than one \section or \subsection on a page, it isthe heading of the last one that appears in the page head.

2.3.2 Customized head and footlines:

Package: ***fancyhdr***

The standard page styles described how the head andfootlines are to appear, and what information they contain. This is a verylimited choice, and the fancyhdr package by Piet van Oostrum offers theuser considerable more flexibility.

This package makes available an additional page style named fancywhich the user can easily redefine. Head and footlines consist each ofthree parts, left, center, right, each of which can be individually definedwith

\lhead{*Left head*} \chead{*Center head*} \rhead{*Right head*}

\lfoot{*Left foot*} \cfoot{*Center foot*} \rfoot{*Right foot*}

where the various texts may be explicit, or a command like \thepage toprint the current page number. Both head and footlines may be decoratedwith a rule, the widths of which are set by commands \headrulewidthand \footrulewidth. By default, the fancy head and footlines are muchthe same as for the headings page style, but the head rule is set to 0.4 ptand the foot rule set to 0 (no rule). The rules may be redefined with, forexample,

\renewcommand{\footrulewidth}{0.4pt}

to turn the foot rule on.

The above defining commands are in fact specific examples of themore general commands \fancyhead and \fancyfoot, where

\lhead{..} is \fancyhead[L]{..}

\cfoot{..} is \fancyfoot[C]{..}

and so on, with L C R standing for ‘left’, ‘center’, ‘right’.

For two-sided output with the twoside option, one normally wantsthe left and right parts to alternate with page number. The easiest way todo this is with

\fancyhead[LE,RO]{*Text 1*} \fancyhead[LO,RE]{*Text 2*}

to put the same *Text 1* in the left part of even pages, and right part ofodd pages, and *Text 2* for the other way round. With \fancyhead{}, allhead line parts are set to blanks, something that should be done beforeresetting them explicitly. Similarly \fancyfoot{} sets all foot entries toblank.

The default (two-sided) definitions for the fancy page style are

\fancyhead[EL,OR]{\textsl{\rightmark}}

\fancyhead[ER,OL]{\textsl{\leftmark}}

where \rightmark and \leftmark contain the automatic texts for theheadings page style generated by the \chapter, \section, \subsectioncommands, while \textsl sets its argumentin a slanted typeface. The user may also make use of these to redefinethe head line with automatic texts.

There is also the most general \fancyhf command taking optionalarguments [H] and [F] to apply to head or footlines. Thus \fancyhf[HL]{..} is the same as \fancyhead[L]{..}. Clearly, \fancyhf{} resetseverything.

In many classes, the first page of a chapter, or the very first pageof the document, is switched to plain automatically. If the user wantsto change this, he or she must redefine that page style. The fancyhdrpackage simplifies this task with

\fancypagestyle{plain}{*definitions*}

where *definitions* consist of \fancyhead, \fancyfoot, and/or rule redefinitionsthat are to apply to the revised plain style. In fact, any existingpage style can be redefined in this way.

2.3.3 Page numbering:

The declaration that specifies the style of the page numbering has theform

\pagenumbering{*num style*}

The allowed values of *num style* are:

arabic for normal (Arabic) numerals,
roman for lower case Roman numerals,
Roman for upper case Roman numerals,
alph for lower case letters,
Alph for upper case letters.

The standard value is arabic. This declaration resets the page counterto 1. In order to paginate the foreword of a document with Roman numeralsand the rest with Arabic numbers beginning with page 1 for chapter 1,one must declare \pagenumbering{roman} at the start of the forewordand then reset the page numbering with \pagenumbering{arabic} immediatelyafter the first \chapter command.

Pages may be numbered starting with a value different from 1 by givingthe command

\setcounter{page}{*page num*}

where *page num* is the number to appear on the current page.

Exercise 2.1:*Expand your exercise text file so that it fills more than one page ofoutput and include the following preamble:*

\documentclass{article}
\pagestyle{myheadings} \markright{Exercises}
\pagenumbering{Roman}
\begin{document}

2.3.4 Paragraph formatting:

The following parameters affect the appearance of a paragraph and maybe given new values with \setlength.

\parskip

The distance between paragraphs, expressed in units of ex sothat it will automatically change with character font size. Thisshould be a *rubber* length.

\parindent

The amount of indentation for the first line of a paragraph.

\baselinestretch

This is a number that magnifies the normal distance between*baselines*, the line on which the letters sit. This number is initially1, for standard line spacing. It may bechanged to anothernumber with

\renewcommand{\baselinestretch}{*factor*}

where *factor* is any decimal number, such as 1.5 for a 50% increase.This then applies to all font sizes. If this command isgiven outside the preamble, it does not come into effect untilanother font size has been selected.

These parameters may be set either in the preamble or anywhere in thetext of the document. In the latter case, the changes remain in effect untilthe next change or until the end of the environment in which they weremade.

To suppress indentation for one paragraph, or to force it where itwould otherwise not occur, place

\noindent or \indent

at the beginning of the paragraph to be affected.

Package: ***indentfirst***

Normally, the first paragraph of a section is not indented, not even with\indent. However, by including the package indentfirst one ensuresthat all paragraphs are indented.

Package: ***parskip***

By default, LATEX indicates paragraphs by indenting the first line.An alternative is without indentation but with extra spacing betweenparagraphs. One could redefine \parindent and \parskip accordingly,or one could employ one of the oldest and simplest packages datingback to LATEX 2.09 days: parskip, written by H. Partl. For consistency, thispackage also makes some changes in the parameters for lists.One loads this package with

\usepackage{parskip}

There is a recent update to the parskip package by Robin Fairbairnsthat includes the option parfill, given as

\usepackage[parfill]{parskip}

that avoids ugly-looking rectangular paragraphs by ensuring that thereis always space at the end of the last line. This update is dated April 9,2001, so to be sure that this package version is loaded, add this date as

\usepackage[parfill]{parskip}[2001/04/09].

A warning will be issued on processing if theactual version of parskip is earlier than this. There will also be warningabout the unknown option parfill.

Exercise 2.2:*Add the following to the preamble of your exercise file:*

\usepackage{parskip}

\renewcommand{\baselinestretch}{1.2}

After processing this exercise, repeat it with another value for the parameter \baselinestretch, say 1.5, in order to get a feeling for how it works. Remove these lines from the exercise file afterwards.

2.3.5 Page format:

Each page consists of a *head*, the *body* containing the actual text, and a*foot*. The selection of the page style determines what information is to befound in the head and footlines.

LATEX uses default values for the distances between the head, body, andfoot, for the upper and left margins, and for the text line width and heightsof the head, body, and foot. These formatting lengths are illustrated inFigure 2.1. They may be changed by declaring new valuesfor them, preferably in the preamble, with the command \setlength. For example, give

\setlength{\textwidth}{12.5cm}

to make the text line width to be 12.5 cm.

There is also a parameter \linewidth equal to the text line width inwhatever environment one is currently in. This must never be changed,but is used when one needs to know this width.

Package: ***layout***

We can examine our own page layout with the layout package fromthe tools collection. Simply issue the command \layout and a diagramsimilar to that in Figure 2.1 will be drawn at that point, together with alist of the current values of the layout parameters. Naturally, we wouldnot do this in the middle of the final version of a document, but only asdiagnostic check.

\oddsidemargin
left margin for odd pages,
\evensidemargin
left margin for even pages,
\topmargin
upper margin to top of head,
\headheight
height of head,
\headsep
distance from the bottom of headline to top of body,
\topskip
distance from top of body to baseline of first line of text,
\textheight, \textwidth
height and width of main text,
\footskip
distance from bottom of body to bottom of foot,
\paperwidth, \paperheight
total width and height of paper as given by paper size option, including all margins.

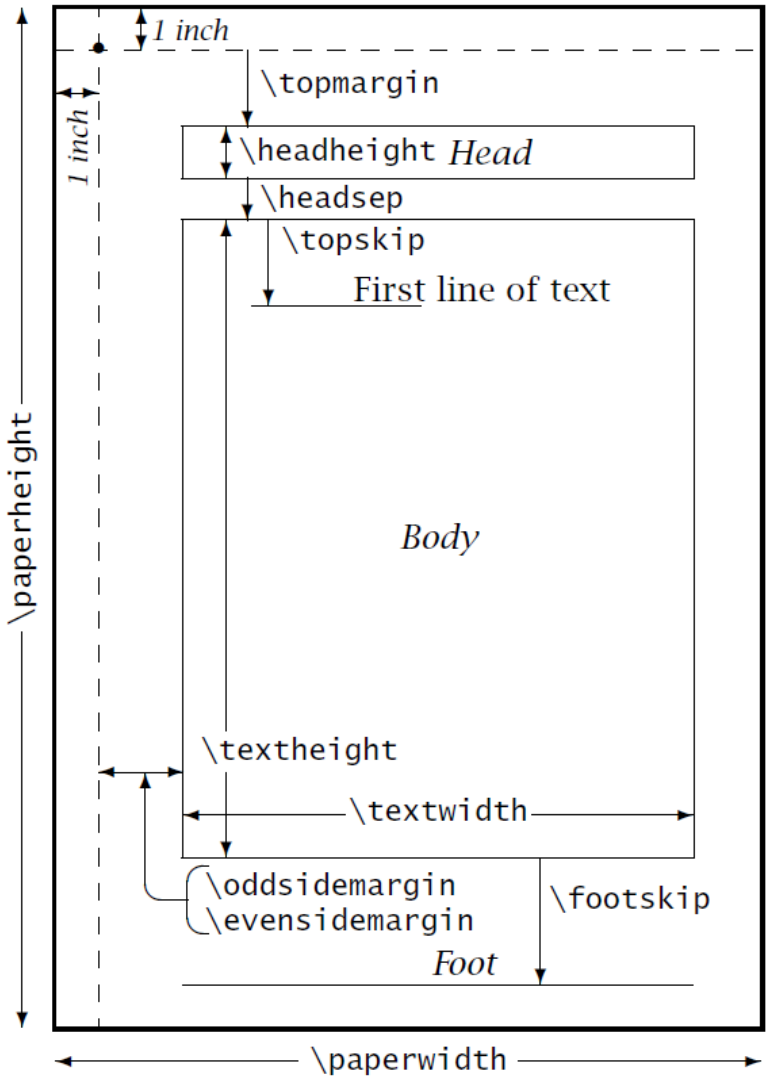


Figure 2.1: Page layout parameters

Exercise 2.3: *You can change the page format of your text by altering the aboveparameters. Add the following to the preamble of your text:*

\setlength{\textwidth}{13cm} \setlength{\textheight}{20.5cm}

The upper and left margins of your output will now seem too small. Selectnew values for \oddsidemargin and \topmargin to correct this. Note: do notforget the 1 inch margin at the left and top from which additional margins aremeasured. You must take this into account when we select \oddsidemarginand \topmargin.

Exercise 2.4: *Expand your text so that the output requires more than two fullpages with the reduced page format. Add \flushbottom to the preamble andcheck that the last line of all pages is at exactly the same location.*

Exercise 2.5: *Remove the command \flushbottom and select the documentclass \documentclass[twoside]{article}. Now the last lines are at the samelocation without the \flushbottom command. On the other hand, the left marginof the odd pages probably does not agree with the right margin of the even pages.Adjust the value of \evensidemargin to correct this.*

2.3.6 Simplified page formatting:

Package: ***geometry***

Getting the page layout to be exactly the way you want it can be very tedious. Just centering the text on the page involves a complex set of settings that are not at all intuitive. The geometry package by HideoUmeki offers considerable assistance.

With this package, one can easily give values for some of the layout parameters, and the rest will be set automatically, taking into account the total paper size. For example, to set \textwidth to 15 cm and\textheight to 25 cm on A4 paper, one gives

```
\usepackage{geometry}
\geometry{a4paper,textwidth=15cm,textheight=25cm}
```

which will also automatically set \oddsidemargin and \topmargin so that the text is centered horizontally and vertically, including the head and footlines. Or, one can set all the margins to be 1 inch on US letterpaper with

```
\geometry{letterpaper,margin=1in}
```

Rather than using the \geometry command, one may also place the parameters as options to \usepackage, for example as

```
\usepackage[a4paper,left=3cm,right=2cm]{geometry}
```

to set the left and right margins to definite values, and \textwidth to what is left over.

In general, all the parameters in Figure 2.1 may be specified by geometry by giving their names (without the backslash character). However, the package is far more powerful than that. Here we describe the essential features of version 2.3 from 2000/06/28.

• The paper size is either inherited from the \documentclass option, given as a predefined option like a4paper, given explicitly aspaperwidth=*pwidth* and paperheight=*pheight*, or as papersize={*pwidth*,*pheight*}.

• By default, the other layout parameters are set so that \textwidth is 80% of \paperwidth and \textheight + \headheight + \headsep + \footskip is 90% of \paperheight, centered horizontally and vertically.

• The text width and height may be set explicitly with width=*width* and height=*height*, or with body={*width*,*height*}. The margins are then set to center. Here *height* means total text height including head and footlines.

• Margins may be explicitly given with left=*lmarg* and right=*rmarg*, or with hmargin={*lmarg*,*rmarg*}. Similarly with top=*tmarg* and bottom=*bmarg*, or vmargin={*tmarg*,*bmarg*}. If only one value of a pair is given, the other is set to that same value, unless the corresponding text height or width has been given explicitly. All margins may be set to a common value with margin=*marg*.

Unlike LATEX standard \oddsidemargin and \topmargin, geometry's margins are measured from the edge of the paper, and not from a point 1 inch removed.

• With nohead, nofoot, noheadfoot, one tells geometry not to include the corresponding head or footline in the height calculation. Thus with noheadfoot, the total height is identical to \textheight.

• With includemp, the marginal note parameters \marginparwidth and \marginparsep are included in the total width calculation, which is then less than \textwidth. With reversemp, the marginal notes appear in the left margin.

• The text width and height may be set to a fraction of the paper size with hscale=*h* and vscale=*v*, or scale={*h*,*v*}. With scale=*s*, both *h* and *v* are set to *s*. For example, \geometry{scale=0.8} sets width and height to 80% of \paperwidth and \paperheight

respectively.

• For two-sided output, add the option twoside. In this case, the values of the left and right margins will switch for even page numbers. In addition, a quantity of 20 pt is added to and subtracted from the left margin of odd and even pages, respectively. This value may be changed with twosideshift=*shift*, which also sets the twoside option automatically.

• With the verbose option, the calculated values of all the layout parameters are printed to the monitor and to the transcript file.

As we see, the geometry package is an enormous aid to setting the page layout, working fairly intuitively, automatically setting values for the unspecified parameters in a way that is normally what one would want anyway. There are many more aspects explained in the accompanying documentation, but the above synopsis should cover most requirements.

2.3.7 Single and double column pages:

The document class option twocolumn sets the entire document in two columns per page. The default is one column per page. Individual pages may be output in one or two columns with the declarations:

\twocolumn[*header text*]

Terminates the current page, starting a new one with *two columns* per page. The optional *header text* is written at the top of the page in one column with the width of the whole page.

\onecolumn

Terminates the current two-column page and continues with one column per page.

The option twocolumn automatically changes certain page style parameters, such as indentation, compared with the one-column format. This does not occur with

the command \twocolumn. These additional changes must be made with the corresponding \setlength declarations if they are desired. If the bulk of the document is in two-column format, the class option is to be preferred.

An additional page style parameter is \columnwidth, the width of one column of text. For single column text, this is the same as \textwidth, but when twocolumn has been selected, LATEX calculates it from the values of \textwidth and \columnsep. The author should never change this parameter, but he or she may make use of it, for example to draw a rule the width of a column of text.

The length \linewidth is even more general, always containing the text line width in the current environment, minipage, parbox. Within a single column, it is the same as \columnwidth. It too may never be changed.

2.3.8 Multicolumn text:

Package: ***multicol***

The commands \twocolumn and \onecolumn always start a new page, and when two-column text is terminated, the two columns are of unequal length. These problems are solved with the multicol package in the tools collection, written by Frank Mittelbach, which also allows up to 10 columns of text. Once this package has been loaded, one can switch the number of columns in the middle of a page with

\begin{multicols}{*num cols*}[*header text*][*pre space*]

Text set in num cols *columns*

\end{multicols}

where the optional *header text* is written across all the columns before switching to multicolumns.

Some automatic control for page breaking before and after switching to multicolumns is offered by the two lengths \premulticols and \postmulticols: if the remaining space on the current page is less than \premulticols, a new page is started

before switching to multicolumns. Similarly, if at the end of the environment, there is less than \postmulticols on the page, a page break is inserted before continuing. The standard values of these lengths may be altered by the user with \setlength, or in the case of \premulticols may be overridden by the second optional argument *pre space*.

The lengths \columnsep and \columnseprule that apply to twocolumn texts are also in effect for the multicols environment, to set the widths of the gap between columns and a possible separating rule, respectively.

There is also a starred version \begin{multicols*}{*num cols*}. . . \end{multicols*} for which the columns on the last page are not balanced, with all the remaining space put into the final column.

2.4 Parts of the document:

Every document is subdivided into chapters, sections, subsections, and so on. There can be an appendix at the end and at the beginning a title page, table of contents, an abstract, etc. LATEX has a number of markup commands available to indicate these structures. In addition, sequential numbering and sub-numbering of headings take place automatically. Even a table of contents may be produced with a single command. The effects of some sectioning commands depend on the selected document class and not all commands are available in every class.

2.4.1 Title page:

A title page can be produced either unformatted with the environment

\begin{titlepage} Title page text \end{titlepage}

or with the commands

\title{*Title text*}

\author{*Author names and addresses*}

\date{*Date text*}

\maketitle

in the preprogrammed LATEX style.

In the standard LATEX layout for the title page, all entries are centered on the lines in which they appear. If the title is too long, it will be broken up automatically. The author may select the break points himself with the \\ command, that is, by giving \title{...\\...\\...}.

```
\title{%
   How to Write DVI Drivers}

\author{%
   Helmut Kopka\thanks{Tel.
     [+49] 5556--401--451}\\
   Max--Planck--Institut\\
   f\"ur Aeronomie
 \and
   Phillip G. Hardy
     \thanks{Tel.
     [+1] 319--824--7134}\\
   University\\of Iowa}

\maketitle
```

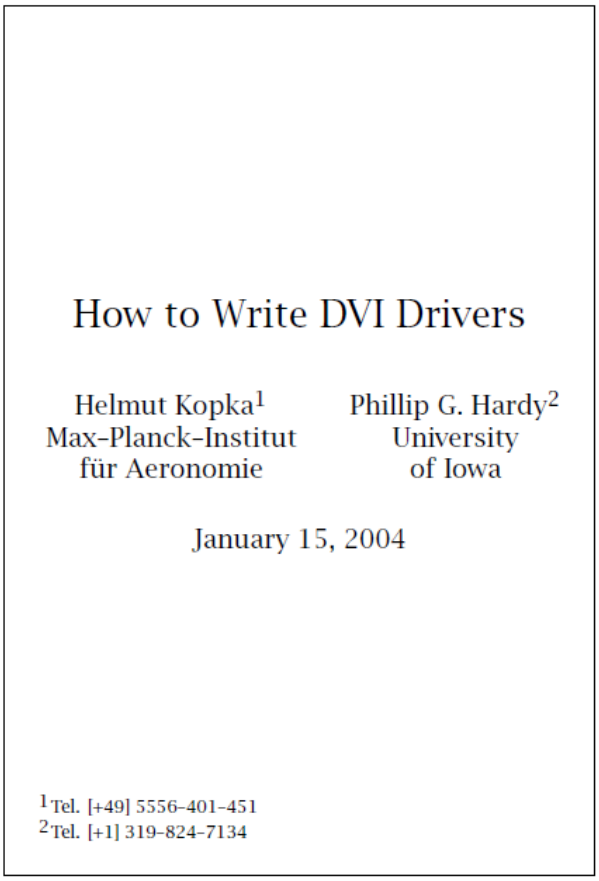


Figure 2.2: Sample title page and the text that produced it

If there are several authors, their names may be separated with \and from one another, such as \author{G. Smith \and J. Jones}. These names will be printed next to each other in one line. The sequence

\author{*Author1**Institute1**Address1*
\and *Author2**Institute2**Address2*}

separately centers the entries, one per line, in each of the sets *Author1*, *Institute1*, *Address1* and *Author2*, *Institute2*, *Address2* and places the two blocks of centered entries beside each other on the title page.

Instead of printing the author names next to each other, one may position them on top of one another by replacing \and with the \\ command. In this case, the vertical spacing may be adjusted with an optional length specification [*space*] following the \\.

If the command \date is omitted, the current date is printed automatically below the author entries on the title page. On the other hand, the command \date{*Date text*} puts the text *Date text* in place of the current date. Any desired text may be inserted here, including line break commands \\ for more than one line of centered text.

The command

\thanks{*Footnote text*}

may be given at any point in the *title*, *author*, or *date text*. This puts a marker at that point where the command appears and writes *footnote text* as a footnote on the title page.

The title page is created using the entries in \title, \author, \date, and \thanks when the command

\maketitle

is issued. The title page itself does not possess a page number and the first page of the following document is number 1. A separate title page is only produced for document classes book and report. For article, the command \maketitle creates a title heading on the first page using the centered entries from the \title, \author, and, if present, \date and \thanks declarations. If the document class option titlepage has been given, the title appears on a separate page even for the article class.

An example of a title page in the standard LATEX format is shown in Figure 2.2. Note that the current date appears automatically since the command \date is missing in

the definition of the title page. This command may be used to put any desired text in place of the date.

For the unformatted title page produced with the titlepage environment, the commands \title and \author are left out and the entire title page is designed according to the author's specifications within the environment. In this case the printing of the title page is implemented at the end of the titlepage environment, so the command \maketitle is also left out.

Exercise 2.6: *Remove the declarations for changing the page format in Exercises 2.3–2.5. Add to your exercise text a title heading with the title 'Exercises', your name as author, and your address, together with a date entry in the form 'place, date'. To do this, write the following commands after \begin{document}:*

\title{Exercises} \author{Your name\\Your address}

\date{Your town, \today} \maketitle

Make sure that you have selected document class article. After printing the document, change the document class command to

\documentclass[titlepage]{article}

to put the title information on to a title page instead of a title heading. Deactivate these commands by putting the comment character % at the beginning of each of the lines. In this way you avoid getting a title page in the following exercises but you can easily reactivate the commands simply by removing the % characters.

2.4.2 Abstract:

The abstract is produced with the command

\begin{abstract} *Text for the abstract* \end{abstract}

In document class report, the abstract appears on a separate page without a page number; in article, it comes after the title heading on the first page, unless the document class option titlepage has been selected, in which case it is also printed on a separate page. An abstract is not possible in document class book.

2.4.3 Sections:

The following commands are available for producing automatic, sequential sectioning:

\part \chapter \subsection \paragraph
\section \subsubsection \subparagraph

With the exception of \part, these commands form a sectioning hierarchy. In document classes book and report, the highest sectioning level is \chapter. The chapters are divided into sections using the \section command, which are further subdivided by means of \subsection, and so on. In document class article, the hierarchy begins with \section since \chapter is not available.

The syntax of all these commands is

sec command[*short title*]{*title*} *or*

*sec command**{*title*}

In the first case, the section is given the next number in the sequence, which is then printed together with a heading using the text *title*. The text *short title* becomes the entry in the table of contents and the page head (provided that page style headings has been selected). If the optional *short title* is omitted, it is set equal to *title*; this is the normal situation unless *title* is too long to serve for the other entries.

In the second (*-form) case, no section number is printed and no entry in the table of contents is made.

The size of the title heading and the depth of the numbering depend on the position of the sectioning command within the hierarchy. For document class article, the \section command generates a single number (say 7), the \subsection command a double number with a period between the two parts (say 7.3), and so on.

In document classes book and report, the chapter headings are given a single number with the \chapter command, the \section command creates the double number, and so on. Furthermore, the command \chapter always starts a new page and prints Chapter n over the chapter title, where n is the current chapter number. At this point in the present book, we are in *Chapter 3, Section 3.3, Subsection 3.3.3*.

For each sectioning command there is an internal counter that is incremented by one every time that command is called, and reset to zero on every call to the next higher sectioning command. These counters are not altered by the *-forms, a fact that can lead to difficulties if standard and *-forms of the commands are mixed such that the *-forms are higher in the hierarchy than the standard forms. There are no problems, however, if the *-forms are always lower than the standard forms. The sequence

\section ... \subsection ... \subsubsection* ...

numbers the headings for \section and \subsection while leaving the headings for \subsubsection without any numbering.

The sectioning command \part is a special case and does not affect the numbering of the other commands.

The automatic numbering of sections means that the numbers might not necessarily be known at the time of writing. If he or she wants to refer to a section number in the text, some mechanism other than typing the number explicitly will be needed. The LATEX cross-reference system, accomplishes this task with the two basic commands

\label{*name*} \ref{*name*}

the first of which assigns a keyword *name* to the section number, while the second may be used as reference in the text for printing that number. The keyword *name* may be any combination of letters, numbers, or symbols. For example, in this book the command \label{sec:xref} has been typed in at the start of Section 9.2.1, so that this

sentence contains the input text at the start of Section \ref{sec:xref}. A second referencing command is \pageref for printing the page number where the corresponding \label is defined. The referencing commands may be used in many other situations for labeling items that are numbered automatically, such as figures, tables, equations.

2.4.4 Appendix:

An appendix is introduced with the declaration

\appendix

It has the effect of resetting the section counter for article and the chapter counter for book and report and changing the form of the numbering for these sectioning commands from numerals to capital letters A, B, ... Furthermore, the word ‘Chapter’ is replaced by ‘Appendix’ so that subsequent chapter headings are preceded by ‘Appendix A’, ‘Appendix B’, etc. The numbering of lower sectioning commands contains the letter in place of the chapter number, for example A.2.1.

2.4.5 Book structure:

To simplify the structuring of a book, the commands

\frontmatter

preface, table of contents

\mainmatter

main body of text

\backmatter

bibliography, index, colophon

are provided in the book class. The \frontmatter command switches page numbering to Roman numerals and suppresses the numbering of chapters; \mainmatter resets the page numbering to 1 with Arabic numbers and reactivates the chapter numbering; this is once again turned off with \backmatter.

Exercise 2.7: *Change the chapter command to*

\chapter[Short form]{Chapter title}

by putting an abbreviated version of Chapter title for Short form. Now the page head contains the shortened title where the full chapter title previously appeared.

2.5 Table of contents:

2.5.1 Automatic entries:

LATEX can prepare and print a table of contents automatically for the whole document. It will contain the section numbers and corresponding headings as given in the standard form of the sectioning commands, together with the page numbers on which they begin. The sectioning depth to which entries are made in the table of contents can be set in the preamble with the command

\setcounter{tocdepth}{*num*}

The value *num* has exactly the same meaning and effect as it does for the counter secnumdepth described above, by which the maximum level of automatic subsectioning is fixed. By default, the depth to which entries are included in the table of contents is the same as the standard level to which automatic sectioning is done: that is, to level \subsection for book and report and to level \subsubsection for article.

2.5.2 Printing the table of contents:

The table of contents is generated and printed with the command

\tableofcontents

given at the location where the table of contents is to appear, which is normally after the title page and abstract.

This leads to a paradox, for the information in the table of contents is to be printed near the beginning of the document, information that cannot be known until the end. LATEX solves this problem as follows: the first time the document is

processed, no table of contents can be included but instead LATEX opens a new file with the same name as the source file but with the extension.toc; the entries for the table of contents are written to this file during the rest of the processing.

The next time LATEX is run on this document, the \tableofcontents command causes the .toc file to be read and the table of contents is printed. As the processing continues, the .toc file is updated in case there have been major changes since the previous run. This means that the table of contents that is printed is always the one corresponding to the previous version of the document. For this reason, it may be necessary to run LATEX more than once on the final version.

2.5.3 Additional entries:

The *-form sectioning commands are not entered automatically in the table of contents. To insert them, or any other additional entry, the commands

\addcontentsline{toc}{sec name}{entry text}

\addtocontents{toc}{entry text}

may be used.

With the first command, the entries will conform to the format of the table of contents, whereby section headings are indented more than those for chapter but less than those for subsection. This is determined by the value of the argument *sec name*, which is the same as one of the sectioning commands without the \ character (for example, section). The *entry text* is inserted in the table of contents along with the page number. This command is most useful to enter unnumbered section headings into the table of contents. For example,

\section{Author addresses}*

\addcontentsline{toc}{section}{Author addresses}

The \addtocontents command puts any desired command or text into the .toc file. This could be a formatting command, say \newpage, which takes effect when the table of contents is printed.

2.5.4 Other lists:

In addition to the table of contents, lists of figures and tables can also be generated and printed automatically by LATEX. The commands to produce these lists are

listoffigures reads and/or produces file.lof

listoftables reads and/or produces file.lot

The entries in these lists are made automatically by the \caption command in the figure and table environments. Additional entries are made with the same commands as for the table of contents, the general form of which is

\addcontentsline{*file*}{*format*}{*entry*}

\addtocontents{*file*}{*entry*}

where *file* stands for one of the three types toc (*table of contents*), lof (*list of figures*), or lot (*list of tables*). The argument *format* is one of the sectioning commands for the table of contents, as described above, or figure for the list of figures, or table for the list of tables. The argument *entry* stands for the text that is to be inserted into the appropriate file.

Exercise 2.8: *In your exercise file, insert after the deactivated title page commands*

\pagenumbering{roman}

\tableofcontents \newpage

\pagenumbering{arabic}

Process your exercise file twice with LATEX and print out the second results.

Deactivate the above commands with % before doing the next run.


KNOWLEDGE IS POWER

Unit-III:

Displayed Text: Changing font style – Centering and indenting – Lists – Generalized lists Theorem like-declarations.

3.1 Displayed Text:

There are a variety of ways to display or emphasize the text: changing font style or font size, centering, indentation, marking the paragraphs, and so on. LATEX supplies us with commands for the most common forms of display.

Many parts of this unit violate the concept of logical markup, especially those dealing with selection of fonts. The author should not attempt to decorate the document with arbitrary switches of font size and style, but should pack his or her source text into a structure that indicates its purpose. The exercises in this book are an example of this. Rather than starting each one with the word ‘Exercise’ in bold face followed by an explicit number, and then shifting to a slanted font, we defined an exercise environment to do all that automatically. This not only ensures consistency, it also allows a change of style to be easily implemented, simply by redefining the environment. This is where the typographical font commands come into play. They should not appear in the main text at all, but rather in the preamble as part of the definitions of environments and commands.

On the other hand, many of the topics in this unit really do involve logical markup, such as the verse, quote, and quotation environments, lists, bibliographies, theorems, and tables.

3.2 Changing font style:

In typography, a set of letters, numbers, and characters of a certain size and appearance is called a *font*. The standard font in LATEX for the main body of text is an *upright*, *Roman* one of *medium* weight, in the size specified in the \documentclass statement at the start. The three possible basic sizes are 10, 11, and 12 pt, depending

on the size options 10pt (default), 11pt, and 12pt. (Recall, there are 72.27 points per inch or about 28.45 pt per cm.) The parenthesis characters ( ) extend the full height and depth of the font size.

The differences in the visual appearance of the three standard sizes are greater than would be expected from the ratios of the numbers:

This is an example of the 10 pt font. ( )

And this is the 11 pt font for comparison. ( )

And finally this is a sample of 12 pt font. ( )

3.2.1 Emphasis:

The usual way to emphasize text in a typewritten manuscript is by underlining. The typesetter will transform underlined text into *italics* for the printed version. Switching from standard to *emphasized* text is carried out in LATEX with the command \emph or the declaration \em.

The \em declaration functions just as the other font declarations described below: the change of font remains in effect until negated by another appropriate declaration (which can be \em itself), or until the end of the current *environment*. An environment may also be created with a pair of curly braces {...}. The command \emph, on the other hand, operates only on the text in the following argument. This is easiest way to *emphasize* short pieces of text, as for example:

This is the easiest way to \emph{emphasize} short ...

The \em declaration is more appropriate for longer text that is enclosed in an environment, *named or nameless.*

...enclosed in an environment, {\em named or nameless.}

Note carefully the difference between the *declaration* that remains in effect until the local environment is ended with the closing curly brace, and the command

that operates on an argument enclosed in curly braces. Another more subtle difference is that the command \emph automatically inserts extra spacing at the end if necessary, the so-called *italic correction*, to improve the appearance at the interface between sloping and upright fonts.

Both the declaration and the command switch to an emphasizing font. That means, if the current font is upright it switches to *italics, whereas if the text is already slanted, an* upright *font is selected.*

Nested emphasis is possible and is simple to understand:

The \emph{first}, second, and \emph{third font switch}

The {\em first, {\em second, and {\em third font switch}}}

both produce 'The *first*, second, and *third font switch*'.

3.2.2 Choice of font size:

The following declarations are available in LATEX for changing the font size:

\tiny	smallest	\Large	larger
\scriptsize	very small	\LARGE	even larger
\footnotesize	smaller	\huge	still larger
\small	small	\Huge	largest
\normalsize	normal		
\large	large		

all of which are relative to the standard size selected in the document class option. In this book, the standard size is 10 pt, which is then the size selected with \normalsize.

The font size declarations behave as all other declarations: they make an immediate change that remains in effect until counteracted by another size declaration, or until the current environment comes to an end. If issued within curly braces {..}, the effect of the declaration extends only to the closing brace, as in a nameless environment:

```
normal {\large large \Large larger} normal again
```

normal large larger normal again

Changing the font size with one of the above commands also automatically changes the interline spacing. For every font size, there is a corresponding *natural* line spacing \baselineskip. This may be altered at any time. If the natural line spacing is 12 pt, the command \setlength{\baselineskip}{15pt} will increase it to 15 pt.

The value of \baselineskip that is effective at the end of the paragraph is used to make up the whole paragraph. This means that if there are several changes to \baselineskip within a paragraph, only the last value given will be taken into account.

With every change in font size, \baselineskip is reset to its natural value for that size. Any previous setting with \setlength will be nullified.

In order to create a change in the line spacing that is valid for all font sizes, one must make use of the factor \baselinestretch, which has a normal value of 1. The true interline spacing is really

\baselinestretch×*\baselineskip*

which maintains the same relative spacing for all font sizes. The user may change this spacing at any time with:

\renewcommand{\baselinestretch}{factor}

where *factor* is any decimal number. A value of 1.5 increases the interline spacing (baseline to baseline) by 50% over its natural size for all font sizes.

The new value of \baselinestretch does not take effect until the next change in font size. In order to implement a new value in the current font size, it is necessary to switch to another size and back again immediately. If the present font size is \normalsize, the sequence

\small\normalsize

will do the trick. Any size command may be used in place of \small.

3.2.3 Font attributes:

The size of a font is only one of several *attributes* that may be used to describe it. With the New Font Selection Scheme (NFSS), which was introduced as part of LATEX2″, it is possible to select fonts strictly by these attributes, as described in Appendix A. However, for normal usage, there are some declarations and corresponding commands to simplify this procedure.

For the Computer Modern fonts provided with TEX and LATEX, the following attributes and values exist:

Family:

for the general overall style. Traditional typographical families have names like *Baskerville*, *Bodoni*, *Times Roman*, *Helvetica*, and so on. The standard LATEX installation provides three families with declarations

\rmfamily to switch (back) to a Roman font;

\ttfamily to switch to a typewriter font;

\sffamily to select a sans serif font.

Shape:

for the form of the font. The shape declarations available with the standard installation are

\upshape to switch (back) to an upright font;

\itshape to select an italic shape;

\slshape to choose a font that is slanted;

\scshape to switch to Caps and Small Caps.

Series:

for the width and/or weight (boldness) of the font. The declarations possible are

\mdseries to switch (back) to medium weight;

\bfseries to select a bold face font.

These do not exhaust all the possible attribute settings, but they do cover the most standard ones, especially for the Computer Modern fonts. For other fonts, especially PostScript ones, additional attribute values exist.

These declarations are used just like any others, normally enclosed in a pair of curly braces {...}, such as {\scshape Romeo and Juliet} producing Romeo and Juliet. For longer sections of text, an environment is preferable:

\begin{*font style*} . . . *text in new font* . . . \end{*font style*}

This keeps better track of the beginning and end of the switch-over. For *font style*, any of the above font commands may be used, leaving off the initial \ character.

Since changing any one attribute leaves the others as they were, all possible combinations may be obtained. (However, this does not mean that a font exists for each possible combination; if not, a substitution will be made.) If we select first a bold series with \bfseries, and then a slanted shape with \slshape, we obtain a bold, slanted font.

normal and {\bfseries bold and

{\slshape slanted} and back} again.

produces: normal and **bold and *slanted* and back** again.

Finally, the declaration \normalfont resets all the attributes (except size) back to their defaults: Roman, upright, medium weight. It is often useful to issue this command just to be sure of the font in effect.

3.2.4 Font commands:

For each of the font declarations listed above, there is a corresponding *font command* that sets its argument in a font with the specified attribute.

Family:	\textrm{*text*}	\texttt{*text*}	\textsf{*text*}
Shape:	\textup{*text*}	\textit{*text*}	\textsl{*text*}
	\textsc{*text*}		
Series:	\textmd{*text*}	\textbf{*text*}	
Default:	\textnormal{*text*}		
Emphasis:	\emph{*text*}		

Note that the \emph command is included here, corresponding to the declaration \em. The argument of \textnormal is set in the standard font selected with \normalfont.

The use of such commands to change the font for short pieces of text, or single words, is much more logical than placing a declaration inside an implied environment. The previous example now becomes

normal and \textbf{bold and \textsl{slanted}

and back} again.

to make: normal and **bold and *slanted* and back** again.

As for the \emph command, these font commands automatically add any necessary *italic correction* between upright and slanted/italic fonts.

The old two-letter TEX declarations such as \bf and \tt, which were part of LATEX 2.09, are still available but are now considered obsolete and should be avoided.

3.2.5 Additional fonts:

It is likely that our computing center or our TEX installation has even more fonts and sizes than those listed above. If so, they may be made available for use within a LATEX document either by referring to them by name, or by their attributes, if they have been set up for NFSS.

To load a new font explicitly by name, the command

\newfont{*fnt*}{*name* scaled *factor*} *or*

\newfont{*fnt*}{*name* at *size*}

is given, which assigns the font to the new command named *fnt*. In the first case, *factor* is a number 1000 times the scaling factor that is to be used to magnify or reduce the font from its basic or design size. In the second case, the font is scaled to be of the *size* specified. To install a slanted, sans serif font of size 20.74 pt, as \sss, we load cmssi17 at 20.74pt with

\newfont{\sss}{cmssi17 at 20.74pt}

Now the declaration \sss switches directly to this font but without altering the baseline separation.

Alternatively, the new font declaration can be made by attributes with

\DeclareFixedFont{\sss}{OT1}{cmss}{m}{sl}{20.74}

Indeed, if one wants to use the current encoding and \sffamily without knowing what they are, or without worrying so precisely what size must be stated, it is also possible to give

\DeclareFixedFont{\sss}{\encodingdefault}{\sfdefault}
{m}{sl}{20}

3.2.6 Character sets and symbols:

The individual character sets are each stored in their own files. The names of the 75 standard TEX fonts are listed.

Each symbol within a character set is addressed by means of a number between 0 and 127 (or 255). The command

\symbol{*num*}

will produce that symbol with the internal identification number *num* in the current font. The symbol ¿ in the present font has the internal number 62 and can be printed with the command \symbol{62}. The identification number may also be given as an *octal* (prefix ’) or *hexadecimal* (prefix ") number. Thus the symbol commands \symbol{28}, \symbol{’34}, and \symbol{"1C} are all identical, producing ‘ø’.

The \symbol command may also be used to generate symbols for which no other command has been defined: for example, {\ttfamily\symbol{'40} \symbol{'42} \symbol{'134}} produces " \ in typewriter font.

3.3 Centering and indenting:

3.3.1 Centered text:

The environment

\begin{center} line 1 \\ line 2 \\ . . . line n \end{center}

centers the sections of text that are separated by the \\ command. (An optional additional line spacing may be inserted with \\[*len*].) If the text is too long for one line, it is split over several lines using uniform word spacing, filling the whole line width as best it can, except for the last line. Word division does not occur.

Within an environment, the command \centering may be used to center the following text, again with \\ as the line divider. The effect of this declaration lasts until the end of that environment.

A single line may be centered by typing its text as the argument of the TEX command \centerline{*text*}.

3.3.2 One-sided justification:

The environments

\begin{flushleft} line 1 \\ line 2 \\ . . . line 2 \end{flushleft}

\begin{flushright} line 1 \\ line 2 \\ . . . line 2 \end{flushright}

produce text that is left (flushleft) or right (flushright) justified. If a section of text does not fit on to one line, it is spread over several with fixed word spacing, the same as for the center environment. Again, word division does not occur.

The same results may be produced within an environment with the declarations

\raggedright replacing the flushleft environment, and

\raggedleft replacing the flushright environment.

3.3.3 Two-sided indentation:

A section of text may be displayed by indenting it by an equal amount on both sides, with the environments

\begin{quote} text \end{quote}

\begin{quotation} text \end{quotation}

Additional vertical spacing is inserted above and below the displayed text to separate it visually from the normal text.

The text to be displayed may be of any length; it can be part of a sentence, a whole paragraph, or several paragraphs.

Paragraphs are separated as usual with an empty line, although no empty lines are needed at the beginning and end of the displayed text since additional vertical spacing is inserted here anyway.

The difference between the above two forms is thus:

In the quotation environment, paragraphs are marked by extra indentation of the first line, whereas in the quote environment, they are indicated with more vertical spacing between them.

The present text is produced within the quotation environment, while the sample above was done with the quote environment.

The quotation environment is only really meaningful when the regular text makes use of first-line indentation to show off new paragraphs.

3.3.4 Verse indentations:

For indenting rhymes, poetry, verses, etc. on both sides, the environment

\begin{verse} *poem* \end{verse}

is more appropriate.

Stanzas are separated by blank lines

while the individual lines of the stanza are divided by the \\ command.

If a line is too long for the reduced text width, it will be left and right justified and continued on the next line, which is indented even further.

The above indenting schemes may be nested inside one another. Within a quote environment there may be another quote, quotation, or verse environment. Each time, additional indentations are created on both sides of the text and vertical spacing is added above and below; these quantities however decrease as the depth of nesting increases. A maximum of six such nestings is allowed.

Exercise 3.1:

Put some appropriate sections of text in your exercise file into the quote and quotation environments, that is, enclose these sections within \begin{quote} . . . . . \end{quote} or \begin{quotation} . . . . . \end{quotation} commands.

Exercise 3.2:

Make up a new file with the name poem.tex and type your favorite poem in the verse environment. Select 12pt as the standard font size and italic as the typeface. Put the title of the poem before the verse environment in a larger bold typeface, such as \Large\bfseries. Include the name of the poet right justified.

Note: remember that you may include declarations to change the font style or size within an environment and that these remain in effect only until the end of that environment.

Exercise 3.3:

Make up another file with the name title.tex. Do you recall the titlepage environment for producing a free-form title page? Create a title page with this environment using font sizes and styles of your choice, centering all the entries.

Note: within the titlepage environment you may of course make use of the center environment, but it is also sufficient to give the \centering declaration instead, since this will remain in effect only until the end of the titlepage environment.

*Choose the individual line spacings with the command \\[len] using an appropriate value for the spacing len. Remember that vertical spacing before the first line of text must be entered with the *-form of the command \vspace*[len].*

Experiment with different font sizes and styles for the various parts of the title page, such as title, author's name, address, until you are satisfied with the results.

Compare your own title page with that of Exercise 2.6. If your creation appeals to you more, include it in your standard exercise file by replacing the commands \title, \author, \date, and \maketitle with the titlepage environment and your own entries.

3.4 Lists:

There are three environments available for producing formatted lists:

\begin{itemize} list text \end{itemize}
\begin{enumerate} list text \end{enumerate}
\begin{description} list text \end{description}

In each of these environments, the *list text* is indented from the left margin and a label, or marker, is included. What type of label is used depends on the selected list environment. The command to produce the label is \item.

3.4.1 Sample *itemize*:

• The individual entries are indicated with a black dot, a so-called *bullet*, as the label.

• The text in the entries may be of any length. The label appears at the beginning of the first line of text.

• Successive entries are separated from one another by additional vertical spacing.

The above text was produced as follows:

```
\begin{itemize}
\item The individual entries are indicated with a black dot, a
      so-called \emph{bullet}, as the label.
\item The text in the entries may be of any length. The label
      appears at the beginning of the first line of text.
\item Successive entries are separated from one another by
      additional vertical spacing.
\end{itemize}
```

3.4.2 Sample *enumerate*:

1. The labels consist of sequential numbers.
2. The numbering starts at 1 with every call to the enumerate environment.

The above example was generated with the following text:

```
\begin{enumerate}
\item The labels consist of sequential numbers.
\item The numbering starts at 1 with every call to the
      \texttt{enumerate} environment.
\end{enumerate}
```

3.4.3 Sample *description*:

Purpose

This environment is appropriate when a number of words or expressions are to be defined.

example

A keyword is used as the label and the entry contains a clarification or explanation.

other uses

It may also be used as an author list in a bibliography.

The above sample was created using the following:

```
\begin{description}
\item[purpose] This environment is appropriate when a number of
    words or expressions are to be defined.
\item[example] A keyword is used as the label and the entry
    contains a clarification or explanation.
\item[other uses] It may also be used as an author list in a
    bibliography.
\end{description}
```

The \item[*option*] command contains an optional argument that appears in bold face as the label.

3.4.4 Nested lists:

The above lists may be included within one another, either mixed or of one type, to a depth of four levels. The type of label used depends on the depth of the nesting. The indentation is always relative to the left margin of the enclosing list. A fourfold nesting of the itemize environment appears as follows:

- The label for the first level is a black dot, a *bullet.*
  - That of the second level is a long dash.
    - That of the third level is an asterisk.
      - And the label for the fourth level is a simple dot.
      - At the same time, the vertical spacing is decreased with increasing depth.
    - Back to the third level.
  - Back to the second level.
- And here we are at the first level of `itemize` once again.

Similarly for the enumerate environment, where the style of the numbering changes with the nesting level:

1. The numbering at the first level is with Arabic numerals followed by a period.
   (a) At the second level, it is with lower case letters in parentheses.
      i. The third level is numbered with lower case Roman numerals with a period.
         A. At the fourth level, capital letters are used.
         B. The label style can be changed, as described in the next section.
      ii. Back to the third level.
   (b) Back to the second level.
2. And the first level of `enumerate` again.

An example of a nested list with mixed types:

- The `itemize` label at the first level is a bullet.
  1. The numbering is with Arabic numerals since this is the first level of the `enumerate` environment.

- This is the third level of the nesting, but the second `itemize` level.
    (a) And this is the fourth level of the overall nesting, but only the second of the `enumerate` environment.
    (b) Thus the numbering is with lower case letters in parentheses.
  - The label at this level is a long dash.

2. Every list should contain at least two points.

- Blank lines ahead of an `\item` command have no effect.

The above mixed list was produced with the following text:

```
\begin{itemize}
  \item The \texttt{itemize} label at the first level is a ...
  \begin{enumerate}
    \item The numbering is with Arabic numerals since this ...
    \begin{itemize}
      \item This is the third level of the nesting, but the ...
      \begin{enumerate}
        \item And this is the fourth level of the overall ...
        \item Thus the numbering is with lower case letters ...
      \end{enumerate}
      \item The label at this level is a long dash.
    \end{itemize}
    \item Every list should contain at least two points.
  \end{enumerate}

  \item Blank lines ahead of an \verb+\item+ command ...
\end{itemize}
```

Exercise 3.4:

Produce a nested list using the itemize and enumerate environments as in the above example, but with a different sequence of these commands.

Exercise 3.5:

Prepare a list of conference participants with their place of residence using the description environment, where the name of the participant appears as the argument in the \item command.

Note: for all three types of lists, any text before the first \item command will yield an error message on processing.

3.5 Generalized lists:

Lists such as those in the three environments itemize, enumerate, and description can be formed in a quite general way. The type of label and its width, the depth of indentation, spacings for paragraphs and labels, and so on, may be wholly or partially set by the user by means of the list environment:

\begin{list}{*stnd lbl*}{*list decl*} *item list* \end{list}

Here *item list* consists of the text for the listed entries, each of which begins with an \item command that generates the corresponding label.

The *stnd lbl* contains the definition of the label to be produced by the \item command when the optional argument is missing.

3.5.1 Standard label:

The first argument in the list environment defines the *stnd lbl*, that is, the label that is produced by the \item command when it appears without an argument. In the case of an unchanging label, such as for the itemize environment, this is simply the desired symbol. If this is to be a mathematical symbol, it must be given as $*symbol name*$, enclosed in $ signs. For example, to select $\Rightarrow$ as the label, *stnd lbl* must be defined to be $\Rightarrow$.

3.5.2 List style parameters:

There are a number of style parameters used for formatting lists that are set by LATEX to certain standard values. These values may be altered by the user in the *list decl* for that particular list. The assignment is made in the usual way with the \setlength command. However, if the assignment is made outside the list environment, in most cases it will simply be ignored. This is because there are preset default values for each parameter at each level that can only be overridden by *list decl*.

The style parameters are listed below and are also illustrated in Figure 3.1.

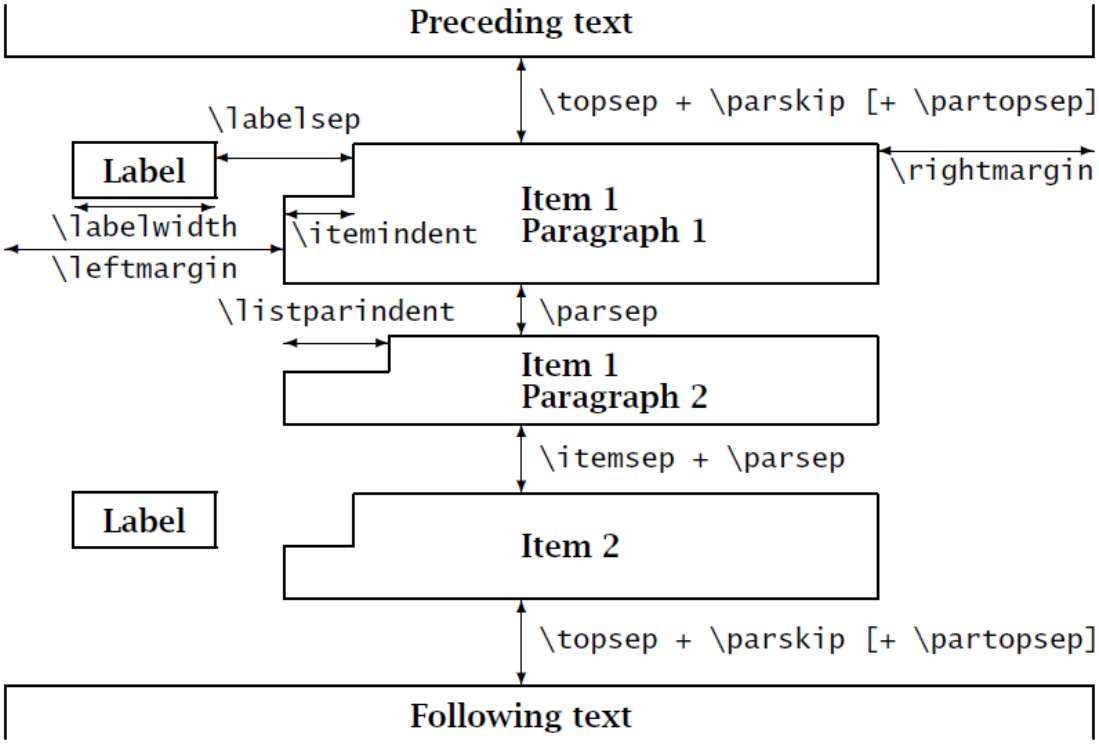


Figure 3.1: The list parameters

\topsep
: is the vertical spacing in addition to \parskip that is inserted between the list and the enclosing text above and below. Its default value is set at each list level and cannot be globally redefined outside the *list_decl*.

\partopsep
: is the vertical spacing in addition to \topsep + \parskip that is inserted above and below the list when a blank line precedes the first or follows the last \item entry. It may be redefined globally, but only for the first and second levels.

\parsep
is the vertical spacing between paragraphs of a single \item. Its default value is reset at each level, as for \topsep.

\itemsep
is the vertical spacing in addition to \parsep that is inserted between two \item entries. As for \topsep and \parsep, its default value is reset at each level and cannot be globally changed.

\leftmargin
is the distance from the left edge of the current environment to the left margin of the list text. There are default values for it at each level that may be globally redefined, as described in Section 4.4.6.

\rightmargin
is the distance from the right edge of the current environment to the right margin of the list text. Its standard value is 0 pt, which can only be altered in *list_decl*.

\listparindent
is the indentation depth of the first line of a paragraph within an \item with respect to the left margin of the list text. It is normally set to 0 pt so that no indentation occurs. This can only be changed in *list_decl*.

\labelwidth
is the width of the box reserved for the label. The text of the label is printed right justified within this space. A new default value may be set globally which then applies to all list levels.

\labelsep
is the spacing between the label box and the list text. A new value may be assigned globally, but it is only effective at the first level.

\itemindent
is the distance by which the label and the first line of text in an \item are indented to the right. It is normally set to 0 pt and so has no effect. This value can only be redefined in *list_decl*.

When changing the vertical spacings from their standard values, it is recommended that a rubber length be used.

The label created by the \item command normally appears right justified within a box of width \labelwidth. It is possible to make it left justified, as in the following list of parameters, by putting \hfill at the end of the definition of the standard label or in the \makelabel command.

3.5.3 Example of a user's list:

List of Figures:

Figure 1: *Page format with head, body, and foot, showing the meaning of the various elements involved.*
Figure 2: *Format of a general list showing its elements.*
Figure 3: *A demonstration of some of the possibilities for drawing pictures with LaTeX.*

This list was produced with the following input:

```
\newcounter{fig}
\begin{list}{\bfseries\upshape Figure \arabic{fig}:}
  {\usecounter{fig}
  \setlength{\labelwidth}{2cm}\setlength{\leftmargin}{2.6cm}
  \setlength{\labelsep}{0.5cm}\setlength{\rightmargin}{1cm}
  \setlength{\parsep}{0.5ex plus0.2ex minus0.1ex}
  \setlength{\itemsep}{0ex plus0.2ex} \slshape}
  \item Page format with head, body, and foot, showing the
        meaning of the various elements involved.
  \item Format of a general list showing its elements.
  \item A demonstration of some of the possibilities for
        drawing pictures with \LaTeX.
\end{list}
```

The command \newcounter{fig} sets up the counter fig. The standard label is defined to be the word **Figure** in upright, bold face, followed by the running Arabic number, terminated by **:**. This label is printed for each \item command.

The list declaration contains \usecounter{fig} as its first command, which makes the counter fig operational within the list. The width of the label box (\labelwidth) is set to 2.0 cm, the left margin of the list text (\leftmargin) to 2.6 cm, the distance between the label and the text (\labelsep) to 0.5 cm, and the right edge of the list (\rightmargin) is set to be 1 cm from that of the enclosing text.

The vertical spacing between paragraphs within an item (\parsep) is 0.5 ex but can be stretched an extra 0.2 ex or shrunk by 0.1 ex. The additional spacing between items (\itemsep) is 0 ex, stretchable to 0.2 ex.

Standard values are used for all the other list parameters. The last command in the list declaration is \slshape, which sets the list text in a *slanted* typeface.

Note: If \upshape were not given in the label definition, the text of each \item would also be slanted, as **Figure 1:**.

3.6 Theorem like-declarations:

In scientific literature one often has text structures like

Theorem 1 (Balzano–Weierstrass) *Every infinite set of bounded points possesses at least one maximum point.*

or

Axiom 3.1 *The natural numbers form a set S of distinct elements. For any two elements a, b, they are either identical, a = b, or different from one another, a ≠ b.*

Similar structures frequently appear with names such as *Definition*, *Corollary*, *Declaration*, *Lemma*, instead of *Theorem* or *Axiom*. What they have in common is that a keyword and a running number are printed in **bold face** and the corresponding text in *italic*.

Of course, these could be generated by the user by explicitly giving the type styles and appropriate number, but if a new structure of that type is later inserted in the middle of the text, the user would have the tedious job of renumbering all the following occurrences. With the command

\newtheorem{*struct type*}{*struct title*}[*in counter*]

LATEX will keep track of the numbering automatically. Here *struct type* is the user's arbitrary designation for the structure, while *struct title* is the word that is printed in bold face followed by the running number (for example, **Theorem**). If the optional argument *in counter* is missing, the numbering is carried out sequentially throughout the entire document. However, if the name of an existing counter, such as chapter, is given for *in counter*, the numbering is reset every time that counter is augmented, and both are printed together, as in **Axiom 3.1** above.

The predefined structures are called with the command

\begin{*struct type*}[*extra title*] *text* \end{*struct type*}

which also increments the necessary counter and generates the proper number. The above examples were produced with

```
\newtheorem{theorem}{Theorem} \newtheorem{axiom}{Axiom}[chapter]
. . . . . .
\begin{theorem}[Balzano--Weierstrass] Every .... \end{theorem}
\begin{axiom} The natural numbers form ..... \end{axiom}
```

The optional *extra title* also appears in bold face within parentheses ( ) following the running number.

Occasionally a structure is not numbered on its own but together with another structure. This can be included in the definition with another optional argument

\newtheorem{*struct type*}[*num like*]{*struct name*}

where *num like* is the name of an existing theorem structure that shares the same counter. Thus by defining \newtheorem{subthrm}[theorem] {Sub-Theorem}, the two structures theorem and subthrm will be numbered as a single series: **Theorem 1, Sub-Theorem 2, Sub-Theorem 3, Theorem 4**, and so on.

Unit-IV:

Text in Boxes: Boxes - Footnotes and marginal notes. Tables: Tabular stops – Tables.

4.1 Text in Boxes:

4.2 Boxes:

Sometimes it can be useful to typeset text in an imaginary box, and treat that box as a single large character. A single-line box can be created with the \text or \makebox commands and a multiline box of a prescribed width can be created with the \parbox command or minipage environment.

4.2.1 LR boxes:

To create LR boxes containing single line text the commands

\mbox{*text*} and \makebox[*width*][*pos*]{*text*}

\fbox{*text*} and \framebox[*width*][*pos*]{*text*}

are available. The two commands at the left produce an LR box with a width exactly equal to that of the *text* given between the braces { }. The \fbox command is the same as \mbox except that the *text* is also framed.

With the two commands at the right, the width is predetermined by the optional length argument *width*. The other optional argument *pos* specifies how the text is positioned within the box. With no value given, the *text* is centered. Otherwise *pos* may be

`l` to left justify the *text*,
`r` to right justify it,
`s` to stretch it to fill up the full width.

Thus `\makebox[3.5cm]{centered text}` creates a box of width 3.5 cm in which the text is centered, as centered text , filled with white space, while with `\framebox[3.5cm][r]{right justified}` the text is right justified inside a framed box of width 3.5 cm: right justified . One may also give

```
\framebox[3.5cm][s]{stretched\dotfill text}
```

to fill up the box, as stretched ..... text, in which case some rubber length or other filler must be added where the stretching is to occur.

If the *text* has a natural width that is larger than that specified in *width*, it will stick out of the box on the left, right, or both sides, depending on the choice of *pos*.

For example,

```
\framebox[2mm]{centered} produces
```

centered

The above application may appear rather silly for \framebox, but it can indeed be very useful for \makebox. A width specification of 0 pt for \makebox can generate a centered, left, or right justified positioning of text in diagrams made with the picture environment. It may also be used to cause two pieces of text to overlap, as \makebox[0pt][l]{/}S prints a slash through an S, as $\/$S.

Note: Length specifications must always contain a dimensional unit, even when they are zero. Thus `0pt` must be given for the width, not `0`.

It is also possible to specify the *width* of an LR box relative to its natural dimensions (those produced by the simple `\mbox` command):

`\width` is the natural width of the box,
`\height` is the distance from baseline to top,
`\depth` is the distance from baseline to bottom,
`\totalheight` is `\height` plus `\depth`.

To make a framed box such that the width is six times the total height, containing centered text,

```
\framebox[6\totalheight]{Text}
```

Text

4.2.2 Line boxes:

The \text command provides a *line box* that typesets its argument without line breaks. As a result, we may find the argument extending into the margin. The resulting box is handled by LATEX as if it were a single large character. For instance,

\text{database}

causes LATEX to treat the eight characters of the word database as if they were one. This technique has a number of uses.

The argument of \text is typeset in a size appropriate for its use, for example, as a subscript or superscript.

Line boxes—a refinement:

The \mbox command is the short form of the \makebox command. Both \mbox and \text prevent breaking the argument, but \mbox does not change size in subscripts and superscripts.

The full form of the \makebox command is

\makebox[width][alignment]{text }

where the arguments are

- *width*, the (optional) width of the box. If [*width*] is omitted, the box is as wide as necessary to enclose its contents.
- *alignment*, (optionally) one of c (the default), l, r, or s. The text is centered by default, l sets the argument flush left, r right, and s stretches the text the full length of the box if there is blank space in the argument.
- *text*, the text in the box.

A *width* argument can be specified in inches (in), centimeters (cm), points (pt), em, or ex.

The following examples,

\makebox{Short title.}End
\makebox[2in][l]{Short title.}End
\makebox[2in]{Short title.}End
\makebox[2in][r]{Short title.}End
\makebox[2in][s]{Short title.}End

typeset as

Short title.End

Short title. *End*

Short title. *End*

Short title.End

Short *title.End*

The optional width argument, *width*, can use four length commands:

\height \depth \totalheight \width

These are the dimensions of the box that would be produced without the optional width argument.

Here is a simple example. The command

\makebox{hello}

makes a box of width \width. To typeset hello in a box three times the width, that is, in a box of width 3\width, use the command

\makebox[3\width]{hello}

So

start\makebox[3\width]{hello}end

typesets as

start hello end

The formal definition of these four length commands is the following:

- \height is the height of the box above the baseline
- \depth is the depth of the box below the baseline
- \totalheight is the sum of \height and \depth
- \width is the width of the box

4.2.3 Frame boxes:

Boxed text is very emphatic. For example, Do not touch! is typed as

\fbox{Do not touch!}

This is a *frame box*, hence the command \fbox or \framebox.

Boxed text cannot be broken, so if we want a frame around more than one line of text, we should put the text as the argument of a \parbox command or within a minipage environment, and then put that into the argument of an \fbox command. For instance,

\fbox{\parbox{3in}{Boxed text cannot be broken,
so if we want to frame more than one line
of text, place it in the argument of a
\bsl\texttt{parbox}
command or within a
\texttt{minipage} environment.}}

produces

Boxed text cannot be broken, so if we want to frame more than one line of text, place it in the argument of a \parbox command or within a minipage environment.

The \framebox command works exactly like \makebox, except that it draws a frame around the box.

\framebox[2in][l]{Short title}

produces

Short title

We can use this command to typeset the number 1 in a square box, as required by the title of Michael Doob's [12]:

TEX *Starting from* 1

\framebox{\makebox[\totalheight]{1}}

which typesets as

1

Note that

\framebox[\totalheight]{1}

typesets as

1

which is not a square box. Indeed, \totalheight is the height of 1, which becomes the width of the box. The total height of the box, however, is the height of the character 1 to which we have to add twice the \fboxsep, the separation between the contents of the box and the frame, defined as 3 points, and twice the \fboxrule, the width of the line, or rule, defined as 0.4 points. These lengths are in general also added to the width of the box, but not in this case, because we forced the width to equal the height of the character.

We can use the \fbox command to frame the name of an author:

\author{\fbox{author's name}}

4.2.4 Paragraph boxes:

A paragraph box works like a paragraph. The text it contains is wrapped around into lines. The width of these lines is set by the user.

The \parbox command typesets the contents of its second argument as a paragraph with a line width supplied as the first argument. The resulting box is handled by LATEX as a single large character. For example, to create a 3-inch wide column,

Fred Wehrung's new result shows the limitation of E. T. Schmidt's construction, especially for large lattices.

type

\parbox{3in}{Fred Wehrung's new result shows the limitation of E.\,T. Schmidt's construction, especially for large lattices.}

Paragraph boxes are especially useful when working within a tabular environment.

The width of the paragraph box can be specified in inches (in), centimeters (cm), points (pt), or the relative measurements em and ex, among others.

4.2.5 Marginal comments:

A variant of the paragraph box, the \marginpar command, allows us to add marginal comments. So

\marginpar{Do not use this often}

produces the comment displayed in the margin.

Rule for Marginal comments and math environments:

Do not use marginal comments in equations or multiline math environments.

Avoid using too many marginal comments on any given page—LATEX may have to place some of them on the next page.

If the document is typeset two-sided, then the marginal comments are set in the outside margin. The form

\marginpar[left-comment]{right-comment }

uses the required argument *right-comment* when the marginal comment is set in the right margin and the optional argument *left-comment* when the marginal comment is set in the left margin.

The width of the paragraph box for marginal comments is stored in the length command \marginparwidth. If we want to change it, use

\setlength{\marginparwidth}{*new_width* }

as in

\setlength{\marginparwidth}{90pt}

The default value of this width is set by the document class.

4.2.6 Solid boxes:

A solid filled box is created with a \rule command. The first argument is the width and the second is the height. For instance, to obtain

end of proof symbol:■

type

end of proof symbol: \rule{1.6ex}{1.6ex}

In fact, this symbol is usually slightly lowered:

end of proof symbol: ■

This positioning is done with an optional first argument:

end of proof symbol: \rule[-.23ex]{1.6ex}{1.6ex}

Here is an example combining \rule with \makebox and \hrulefill:

1 inch:\quad\makebox[1in]{\rule{.4pt}{4pt}%
\hrulefill\rule{.4pt}{4pt}}

which produces

1 inch: ⌊————————⌋

Struts:

Solid boxes of zero width are called *struts*. Struts are invisible, but they force LATEX to make room for them, changing the vertical alignment of lines. Standard struts can also be added with the \strut or \mathstrut command. To see how struts work, compare

$\boxed{\text{ab}}$ and $\boxed{\text{ab}}$ and $\boxed{\text{ab}}$

typed as

\fbox{ab} and \fbox{\strut ab} and \fbox{$\mathstrut$ab}

Struts are especially useful for fine tuning tables and formulas.

Zero distance:

0pt, 0in, 0cm, 0em all stand for zero width. 0 by itself is not acceptable. For example, *\rule{0}{1.6ex}* gives the message

! Illegal unit of measure (pt inserted).
<to be read again>
h
l.251 \rule{0}{1.6ex}

If the \rule command has no argument or only one, LATEX generates a message.

4.2.7 Fine tuning boxes:

The command

\raisebox{*displacement* }{*text* }

typesets *text* in a box with a vertical *displacement*. If *displacement* is positive, the box is raised; if it is negative, the box is lowered.
The \raisebox command allows us to play games:

fine-\raisebox{.5ex}{tun}\raisebox{-.5ex}{ing}

produces fine-$^{\text{tun}}_{\text{ing}}$.

The \raisebox command has two optional arguments:

\raisebox{0ex}[1.5ex][0.75ex]{text }

forces LATEX to typeset *text* as if it extended 1.5 ex above and 0.75 ex below the line, resulting in a change in the interline space above and below the line.

4.3 Footnotes and marginal notes:

4.3.1 Standard footnotes:

Footnotes are generated with the command

\footnote{*footnote text*}

which comes immediately after the word requiring an explanation in a footnote. The text *footnote text* appears as a footnote in a smaller typeface at the bottom of the page. The first line of the footnote is indented and is given the same footnote marker as that inserted in the main text. The first footnote on a page is separated from the rest of the page text by means of a short horizontal line.

The standard footnote marker is a small, raised number[1], which is sequentially numbered. This footnote is produced with:

... raised number\footnote{The usual method of marking footnotes in a typewritten ... same page.}, which is ...

The footnote numbering is incremented throughout the whole document for the article class, whereas it is reset to 1 for each new chapter in the report and book classes.

The \footnote command may only be given within the normal paragraph mode, and not within math or LR modes. In practice, this means it may not appear within an LR box or a parbox. However, it may be used within a minipage environment, in which case the footnote text is printed beneath the minipage and not at the bottom of the actual page.[2]

The \footnote command must immediately follow the word that is to receive the note, without any intervening blanks or spacing. A footnote at the end of a sentence can be given after the period, as in the last example above:

... of the actual page.\footnote{With nested ... wrong place.}

4.3.2 Non-standard footnotes:

If the user wishes the footnote numbering to be reset to 1 for each \section command with the article class, this may be achieved with

\setcounter{footnote}{0}

just before or after a \section command.

The internal footnote counter has the name footnote. Each call to \footnote increments this counter by one and prints the new value in Arabic numbering as the footnote marker. A different style of marker can be implemented with the command

\renewcommand{\thefootnote}{*number style*{footnote}}

where *number style* is one of the counter print commands; \arabic, \roman, \Roman, \alph, or \Alph. However, for the counter footnote, there is an additional counter print command available, \fnsymbol, which prints the counter values 1–9 as one of nine symbols:

* † ‡ § ¶ k ** †† ‡‡

It is up to the user to see that the footnote counter is reset to zero sometime before the tenth \footnote call.

An optional argument may be added to the \footnote command

\footnote[*num*]{*footnote text*}

where *num* is a positive integer that is used instead of the value of the footnote counter for the marker. In this case, the footnote counter is not incremented. For example**,

\renewcommand{\thefootnote}{\fnsymbol{footnote}}

For example\footnote[7]{The 7th symbol ... marker.},

\renewcommand{\thefootnote}{\arabic{footnote}}

where the last line is necessary to restore the footnote marker style to its standard form. Otherwise, all future footnotes would be marked with symbols and not with numbers.

4.3.3 Footnotes in forbidden modes:

A footnote marker can be inserted in the text with the command

\footnotemark[*num*]

even where the \footnote command is normally not allowed, that is, in LR boxes, tables, and math mode. The marker is either the optional argument *num* or, if it is omitted, the incremented value of the footnote counter. The footnote itself is not generated. This must be done external to the forbidden mode with the command

\footnotetext[*num*]{*footnote text*}

If the optional argument has been used for the footnote marker, the same *num* must be given as the option for the text command. Similarly, if no option was used for the marker, none may appear with the text. The footnote will be generated with the value of *num* or with that of the footnote counter.

This counter is incremented by a call to \footnotemark without an optional argument. The corresponding \footnotetext command, on the other hand, does not alter the counter.

If there are a number of \footnotemark commands without optional arguments appearing before the next \footnotetext command, it is necessary to adjust the counter with the command

\addtocounter{footnote}{*dif* }

where *dif* is a negative number saying how many times the counter must be set back. Then before every \footnotetext command, the counter must be incremented by one. This can be done either with the command \addtocounter, with *dif* =1, or with the command

\stepcounter{footnote}

which adds 1 to the given counter.

For example: mosquitoes[3] and elephants[4]

```
For example: \fbox{mosquitoes\footnotemark\ and
   elephants\footnotemark}
```

generates the footnote markers [3] and [4]. Now the counter has the value 4. In order for the first \footnotetext outside the framed box to operate with the correct counter value, it must first be decremented by one. The two footnote texts are made with

\addtocounter{footnote}{-1}\footnotetext{Small insects}

\stepcounter{footnote}\footnotetext{Large mammals}

immediately following the \fbox{} command. The footnote counter now has the same value as it did on leaving the \fbox.

4.3.4 Footnotes in minipages:

As footnote commands are allowed inside the minipage environment. However, the footnote appears underneath the minipage, not below the main page.

Footnote commands within a mini-page[a] have a different marker style. The footnote comes after the next \end{minipage} command.[b] Minipage footnotes have a counter separate from that of the main page, called mpfootnote, counting independently of footnote.

```
\begin{minipage}{6cm}
Footnote commands within
a minipage\footnote{The
marker is a raised
lower-case letter.} have
a different...
\end{minipage}
```

Footnotes within a tabular environment can normally only be generated with the commands described above: \footnotemark within the table and \footnotetext outside the environment. However, if the tabular environment is inside a minipage,

normal \footnote commands may also be used inside the table. The footnote appears below the table where the minipage comes to an end.

Exercise 4.1:

Produce a number of footnotes in your standard exercise file by inserting them where you think fit and by selecting some appropriate footnote text.

Exercise 4.2:

Redefine the command \thefootnote so that the footnote markers become the symbols. Add the redefinition to the preamble of your standard exercise file.

4.3.5 Marginal notes:

Notes in the page margin are produced with the command

\marginpar{*note text*}

which puts the text *note_text* into the margin beginning at the level of the line where the command is given. The marginal note appearing here was generated with

This is a marginal note

```
... The marginal note \marginpar{This\\ is a\\ margin-\\al note}
    appearing here ...
```

The text is normally enclosed in a parbox of width 1.9 cm (0.75 in). Such a narrow box causes great difficulties with line breaking, which is why the lines are manually broken with the \\ command in the above example. Such a box is far more appropriate for marginal notes in the form of a single symbol, such as the arrow shown here. ⟹

Another common use for the marginal note is to draw attention to certain text passages by marking them with a vertical bar in the margin. This is often done to indicate changes in a text, for comparison with earlier versions after updated single sheets have been redistributed. The example marking this paragraph was made by including

```
\marginpar{\rule[-17.5mm]{1mm}{20mm}}
```

in the first line.

4.4 Tabular stops:

4.4.1 Basics:

On a typewriter it is possible to set *tabulator stops* at various positions within a line; then by pressing the tab key the print head or carriage jumps to the next tab location.

A similar possibility exists in LATEX with the tabbing environment:

\begin{tabbing} *lines* \end{tabbing}

One can think of the set tab stops as being numbered from left to right. At the beginning of the tabbing environment, no tabs are set, except for the left border, which is called the *zeroth* tab stop. The stops can be set at any spot within a line with the command \=, and a line is terminated by the \\ command:

```
Here is the \=first tab stop, followed by\= the second\\
```

sets the first tab stop after the blank following the word *the*, and the second immediately after the word *by*.

After the tab stops have been set in this way, one can jump to each of the stops in the subsequent lines, starting from the left margin, with the command \>. A new line is started with the usual \\ command.

Example:

Type	Quality	Color	Price
Paper	med.	white	low
Leather	good	brown	high
Card	bad	gray	med.

```
\begin{tabbing}
Type\qquad\= Quality\quad\=
Color\quad\= Price\\[0.8ex]
Paper    \> med. \> white \> low\\
Leather \> good \> brown \> high\\
Card     \> bad  \> gray  \> med.
\end{tabbing}
```

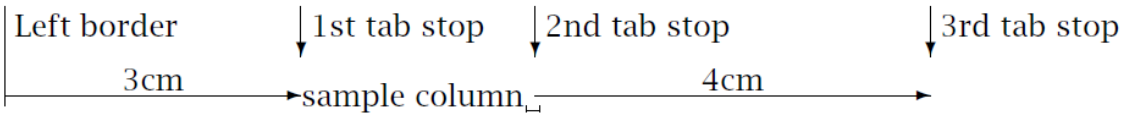
4.4.2 Sample line:

It is often advantageous or even necessary to set the tab stops in a sample line that is not actually printed. It could contain, for example, the widest entries in the various columns that appear later, or the smallest intercolumn spacing between stops. The sample line may also contain \hspace commands to force the distance between stops to be a predetermined amount.

To suppress the printing of the sample line, it is ended with the command \kill instead of the \\ terminator.

```
\hspace*{3cm}\=sample column \=\hspace{4cm}\= \kill
```

In addition to the left border, the above statement sets three tab stops:



An \hspace command at the beginning of a sample line must be of the *-form, otherwise the inserted spacing will be deleted at the line margin.

4.4.3 Tab stops and the left margin:

The left border of each line of the tabbing environment is at first identical with the left margin of the enclosing environment, and is designated the *zeroth* stop. By activating the 'tab key' \> at the start of a line, one sets the following text beginning at the first tab stop. However, the command \+ has the same effect, putting the left border permanently at the first stop, for all subsequent lines. With \+\+ at the beginning or end of a line, all the next lines will start two stops further along. There can be as many \+ commands in all as there are tab stops set on the line.

The command \- has the opposite effect: it shifts the left border for the following lines one stop to the left. It is not possible to set this border to be to the left of the *zeroth* stop.

The effect of the \+ commands may be overridden for a single line by putting \< at the start for each tab to be removed. This line then starts so many tabs to the left of the present border. With the next \\ command, the new line begins at the current left border determined by the total number of \+ and \- commands.

4.4.4 Further tabbing commands:

Tab stops can be reset or added in every line. The command \= will add a stop if there have been sufficient \> commands to have jumped to the last stop, otherwise it will reset the next stop.

For example:

```
Old column 1 Old column 2
Left column   Middle col Extra col
New col 1 New col 2       Old col 3
Column 1 Column 2         Column 3
```

```
Old column 1 \= Old column 2\\
Left column \> Middle col
  \= Extra col\\
New col 1 \= New col 2 \>
  Old col 3\\
Column 1\> Column 2 \> Column 3
\end{tabbing}
```

Occasionally it is desirable to be able to reset the tab stops and then to reuse the original ones later. The command \pushtabs accomplishes this by storing the current tabs and removing them from the active line. All the tab stops can then be set once again. The stored stops can be reactivated with the command \poptabs. The \pushtabs command may be given as many times as needed, but there must be the same number of \poptabs commands within any one tabbing environment.

It is possible to position text on a tab stop with *left text* \' *right text*, where *left text* goes just before the current tab (or left border) with a bit of spacing, while *right*

text starts exactly at the stop. The amount of spacing between the *left text* and the tab stop is determined by the tabbing parameter \tabbingsep. This may be changed by the user with the \setlength command as usual.

Text may be right justified up against the right border of a line with the command \‘ *text*. There must not be any more \> or \= commands in the remainder of the line.

The commands \=, \‘, and \’ function as accent commands outside of the tabbing environment. If these accents are actually needed within tabbing, they must be produced with \a=, \a‘, and \a’ instead. For example, to produce ´o, `o, or ¯o inside a tabbing environment, one must give \a’o, \a‘o, or \a=o. The command \- also has another meaning outside of the tabbing environment (suggested word division) but since lines are not broken automatically within this environment, there is no need for an alternative form.

Here is an example illustrating all the tabbing commands:

```
Apples:      consumed by: people      \begin{tabbing}
                          horses      Grapefruits: \= \kill
                      and sheep       Apples: \> consumed by: \= people\+\+\\
             reasonably juicy         horses \\
Grapefruits: a delicacy               and \’ sheep\-\\
(see also: melons                     reasonably juicy\-\\
           pumpkins)                  Grapefruits: \> a delicacy\\
                                      \pushtabs
Horses       feed on        apples    (see also: \= melons\\
                                       \> pumpkins)\\
                                      \poptabs
                                      Horses \> feed on \> apples
                                      \end{tabbing}
```

4.4.5 Remarks on tabbing:

TEX treats the tabbing environment like a normal paragraph, breaking a page if necessary between two lines within the environment. However, the commands

\newpage and \clearpage are not allowed within it, and the command \pagebreak is simply ignored. If the user wishes to force a page break within the tabbing environment, there is a trick that he or she may employ: specify a very large interline spacing at the end of the line where the break should occur (for example, \\[10cm]). This forces the break and the spacing disappears at the start of the new page.

Each line of text is effectively within a { } pair, so that any size or font declarations remain in force only for that one line. The text need not be put explicitly inside a pair of curly braces.

It is not possible to nest tabbing environments within one another.

Beware: the tab jump command \> always moves to the next logical tab stop. This could actually be a move backwards if the previous text is longer than the space available between the last two stops. This is in contrast to the way the tabulator works on a typewriter.

There is no automatic line breaking within the tabbing environment. Each line continues until terminated by a \\ command.

The commands \hfill, \hrulefill, and \dotfill have no effect inside a tabbing environment, since no *stretching* takes place here.

Exercise 4.3: *Generate the following table with the tabbing environment.*

Project: Total Requirements = $900 000.00
of which 2003 = $450 000.00
2004 = $350 000.00
2005 = $100 000.00

2003 approved:	$350 000.00	Deficiency:	$100 000.00
2004	$300 000.00		$150 000.00
2005	$250 000.00	Surplus:	$150 000.00

4.5 Tables:

With the *box* elements and tabbing environment it would be possible to produce all sorts of framed and unframed tables. However, LATEX offers the user far more convenient ways to build such complicated structures.

4.5.1 Constructing tables:

The environments `tabular`, `tabular*`, and `array` are the basic tools with which tables and matrices can be constructed. The syntax for these environments is

`\begin{array}[`*pos*`]{`*cols*`}`	*rows*	`\end{array}`
`\begin{tabular}[`*pos*`]{`*cols*`}`	*rows*	`\end{tabular}`
`\begin{tabular*}{`*width*`}[`*pos*`]{`*cols*`}`	*rows*	`\end{tabular*}`

The array environment can only be applied in *mathematical mode*. It is described here only because its syntax and the meaning of its arguments are exactly the same as those of the tabular environment. All three environments actually create a minipage. The meaning of the arguments is as follows:

pos Vertical positioning argument. 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 (see below) `@{\extracolsep{\fill}}` somewhere after the first entry. For the other two 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 intercolumn spacings.

The possible *column formatting symbols* are

`l` the column contents are *left* justified;

`r` the column contents are *right* justified;

`c` the column contents are *centered*;

`p{`*wth*`}` the text in this column is set into lines of width *wth*, and the top line is aligned with the other columns. In fact, the text is set in a parbox with the command `\parbox[t]{`*wth*`}{`*column text*`}`;

`*{`*num*`}{`*cols*`}` the *column format* contained in *cols* is reproduced *num* times, so that `*{5}{|c|}` is the same as `|c|c|c|c|c|`.

The available *formatting symbols* for the left and right borders and for the intercolumn spacing are

`|` draws a vertical line;

`||` draws two vertical lines next to each other;

`@{`*text*`}` this entry is referred to as an *@-expression*, and inserts *text* in every line of the table between the two columns where it appears.

An @-expression removes the intercolumn spacing that is automatically put between each pair of columns. If white space is needed between the inserted text and the next column, this must be explicitly included with `\hspace{ }` within the *text* of the @-expression. If the intercolumn spacing between two particular columns is to be something other than the standard, this may be easily achieved by placing `@{\hspace{`*wth*`}}` between the appropriate columns in the formatting argument. This replaces the standard intercolumn spacing with the width *wth*.

An `\extracolsep{`*wth*`}` within an @-expression will put extra spacing of amount *wth* between all the following columns, until countermanded by another `\extracolsep` command. In contrast to the standard spacing, this additional spacing is not removed by later @-expressions. In the `tabular*` environment, there must be a command `@{\extracolsep\fill}` somewhere in the column format so that all the subsequent intercolumn spacings can stretch out to fill the predefined table width.

If the left or right borders of the table do not consist of a vertical line, spacing is added there of an amount equal to half the normal intercolumn spacing. If this spacing is not wanted, it may be suppressed by including an empty @-expression `@{}` at the beginning or end of the column format.

rows contain the actual entries in the table, each horizontal row being terminated with a \\ command. These rows consist of a sequence of column entries separated from each other by the & symbol. Thus each row in the table contains the same number of column entries as in the column definition *cols*. Some entries may be empty. The individual column entries are treated by LaTeX as though they were enclosed in braces { }, so that any changes in type style or size are restricted to that one column.

`\hline` This command may only appear before the first row or immediately after a \\ row termination. It draws a horizontal line the full width of the table below the row that was just ended, or at the top of the table if it comes at the beginning.

Two `\hline` commands together draw two horizontal lines with a little space between them.

`\cline{`$m-n$`}` This command draws a horizontal line from the left side of column m to the right side of column n. Like `\hline`, it may only be given just after a \\ row termination and there may be more than one after another. The command `\cline{1-3} \cline{5-7}` draws two horizontal lines from column 1 to 3 and from column 5 to 7, below the row that was just ended. In each case, the full column widths are underlined.

`\multicolumn{`*num*`}{`*col*`}{`*text*`}` This command combines the following *num* columns into a single column with their total width including intercolumn spacings. The argument *col* contains exactly one of the positioning symbols `l`, `r`, or `c`, with possible @-expressions and vertical lines `|`. A value of 1 may be given for *num* when the positioning argument is to be changed for that column in one particular row.

In this context, a 'column' starts with a positioning symbol `l`, `r`, or `c`, and includes everything up to but excluding the next one. The first column also includes everything before the first positioning symbol. Thus `|c@{}rl|` contains three columns: the first is `|c@{}`, the second `r`, and the third `l|`.

The `\multicolumn` command may only come at the start of a row or right after a column separation symbol &.

`\vline` This command draws a vertical line with the height of the row at the location where it appears. In this way, vertical lines that do not extend the whole height of the table may be inserted within a column.

If a p-type column contains \raggedright or \centering, the \\ forces a new line *within the column entry* and not the end of the whole row. If this occurs in the last column, then \\ cannot be used to terminate the row; instead one must use \tabularnewline to end such a row.

Since a table is a vertical box of the same sort as parbox and minipage, it may be positioned horizontally with other boxes or text. In particular, the table must be enclosed within

\begin{center} *table* \end{center}

in order to center it on the page.

4.5.2 Table style parameters:

There are a number of style parameters used in generating tables which LATEX sets to standard values. These may be altered by the user, either globally within the preamble or locally inside an environment. They should not be changed within the tabular environment itself.

`\tabcolsep` is half the width of the spacing that is inserted between columns in the `tabular` and `tabular*` environments;

`\arraycolsep` is the corresponding half intercolumn spacing for the `array` environment;

`\arrayrulewidth` is the thickness of the vertical and horizontal lines within a table;

`\doublerulesep` is the separation between the lines of a double rule.

Changes in these parameters can be made with the `\setlength` command as usual. For example, to make the line thickness to be 0.5 mm, give `\setlength {\arrayrulewidth}{0.5mm}`. Furthermore, the parameter

`\arraystretch` can be used to change the distance between the rows of a table. This is a multiplying factor, with a standard value of 1. A value of 1.5 means that the inter-row spacing is increased by 50%. A new value is set by redefining the parameter with the command

`\renewcommand{\arraystretch}{`*factor*`}`

4.5.3 Table examples:

Creating tables is much easier in practice than it would seem from the above list of formatting possibilities. This is best illustrated with a few examples.

The simplest table consists of a row of columns in which the text entries are either centered or justified to one side. The column widths, the spacing between the columns, and thus the entire width of the table are automatically calculated.

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
3	Clarkson Chargers	33	17	7	9	70:44	41:25
4	Daysdon Bombers	33	14	10	9	66:50	38:28
5	Edgartown Devils	33	16	6	11	63:53	38:28
6	Freeburg Fighters	33	15	7	11	64:47	37:29
7	Gadsby Tigers	33	15	7	11	52:37	37:29
8	Harrisville Hotshots	33	12	11	10	62:58	35:31
9	Idleton Shovers	33	13	9	11	49:51	35:31
10	Jamestown Hornets	33	11	11	11	48:47	33:33
11	Kingston Cowboys	33	13	6	14	54:45	32:34
12	Lonsdale Stompers	33	12	8	13	50:57	32:34
13	Marsdon Heroes	33	9	13	11	50:42	31:35
14	Norburg Flames	33	10	8	15	50:68	28:38
15	Ollison Champions	33	8	9	16	42:49	25:41
16	Petersville Lancers	33	6	8	19	31:77	20:46
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

The above table is made up of eight columns, the first of which is right justified, the second left justified, the third centered, the next three right justified again, and the last two centered. The column formatting argument in the tabular environment thus appears as

$$\{rlcrrrcc\}$$

The text to produce this table is

```
\begin{tabular}{rlcrrrcc}
Position & Club & Games & W & T & L & Goals & Points\\[0.5ex]
 1  & Amesville Rockets & 33 & 19 & 13 &  1 & 66:31 & 51:15 \\
 2  & Borden Comets     & 33 & 18 &  9 &  6 & 65:37 & 45:21 \\
... & .....             & .. & .. & .. & .. &  ...  &  ...  \\
17  & Quincy Giants     & 33 &  7 &  5 & 21 & 40:89 & 19:47 \\
18  & Ralston Regulars  & 33 &  3 & 11 & 19 & 37:74 & 17:49
\end{tabular}
```

In each row, the individual columns are separated from one another by the symbol & and the row itself is terminated with the \\ command. The [0.5ex] at the end of the first row adds extra vertical spacing between the first two rows. The last row does not need the termination symbol since it is ended automatically by the \end{tabular} command.

The columns may be separated by vertical rules by including the symbol | in the column formatting argument. Changing the first line to

$$\backslash begin\{tabular\}\{r|l||c|rrr|c|c\}$$

results in

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
⋮	⋮						⋮
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

The same symbol | before the first or after the last column format generates a vertical line on the outside edge of the table. Two symbols || produce a double vertical

line. Horizontal lines over the whole width of the table are created with the command \hline. They may only appear right after a row termination \\ or at the very beginning of the table. Two such commands \hline\hline draw a double horizontal line.

```
\begin{tabular}{|r|l||c|rrr|c|c|} \hline
Position & Club & Games & W & T & L & Goals & Points\\
 \hline\hline
 1  & Amesville Rockets & 33 & 19 & 13 &  1 & 66:31 & 51:15 \\
 \hline
 . . . . . .
18  & Ralston Regulars  & 33 &  3 & 11 & 19 & 37:74 & 17:49 \\
 \hline
\end{tabular}
```

The table now appears as

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
⋮	⋮						⋮
17	Quincy Giants	33	7	5	21	40:89	19:47
18	Ralston Regulars	33	3	11	19	37:74	17:49

In this case, the row termination \\ must be given for the last row too because of the presence of \hline at the end of the table.

Exercise 4.4:

Generate the following timetable.

Day	6.15–7.15 pm		7.20–8.20 pm		8.30–9.30 pm	
	Subj.	Teacher Room	Subj.	Teacher Room	Subj.	Teacher Room
Mon.	UNIX	Dr. Smith Comp. Ctr	Fortran	Ms. Clarke Hall A	Math.	Mr. Mills Hall A
Tues.	LaTeX	Miss Baker Conf. Room	Fortran	Ms. Clarke Conf. Room	Math.	Mr. Mills Hall A
Wed.	UNIX	Dr. Smith Comp. Ctr	C	Dr. Jones Hall B	ComSci.	Dr. Jones Hall B
Fri.	LaTeX	Miss Baker Conf. Room	C++	Ms. Clarke Conf. Room	canceled	

Unit-V:

Mathematical Formulas: Mathematical Environment – Main elements of math mode – Mathematical symbols – Additional Elements.

5.1 Mathematical Formulas:

Mathematics is the soul of TEX. It was because the setting of mathematical formulas is so complicated in normal printing, not to mention on a typewriter, that Donald Knuth invented his text formatting system. On the other hand, the soul of LATEX is logical markup. Nevertheless, all the power of TEX's math setting is also available in LATEX, offering an unbeatable combination.

In this unit, we confine ourselves to the elements of mathematical typesetting available in standard LATEX.

Mathematical formulas are produced by typing special descriptive text. This means that LATEX must be informed that the *following text* is to be interpreted as a *mathematical formula*, and it must also be told when the *math text* has come to an end and normal text recommences. The processing of *math text* is carried out by switching to *math mode*. Mathematical environments serve this purpose.

5.2 Mathematical Environment:

Mathematical formulas may occur within a line of text, as $(a + b)^2 = a^2 + 2ab + b^2$, or separated from the main text as

$$\int_0^\infty f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i \mathrm{e}^{x_i} f(x_i)$$

These two types are distinguished by referring to them as *text* and *displayed formulas* respectively.

Text formulas, or equations, are generated with the environment

```
\begin{math}  formula_text  \end{math}
```

Since text formulas are often very short, sometimes consisting of only a single character, a shorthand version is available as `\(` *formula_text* `\)`.

However, most authors prefer the very short form $*formula text* $ which is actually the TEX method. All three are essentially the same, and there is no reason not to use the $ sign.

The contents of the formula, *formula text*, consist of math constructs, which are described in the following sections.

Displayed formulas, or equations, are produced in the environments

\begin{displaymath} formula text \end{displaymath}
\begin{equation} formula text \end{equation}

The difference between these two is that the equation environment automatically adds a sequential equation number. The displaymath environment may be given with the shorthand forms \[. . . \] or $$. . . $$.

By default displayed formulas are centered horizontally with the equation number, if it is present, set flush with the right margin. By selecting the document class option fleqn, the formulas are set left justified with an adjustable indentation. This option remains valid for the entire document whereas the amount of indentation may be changed at will with \setlength{\mathindent}{*indent*}, where *indent* is a length specification. Moreover, the document class option leqno sets the equation numbers flush with the left margin throughout the whole document.

Finally, multiline formulas can be created with the environments

\begin{eqnarray} formula text \end{eqnarray}
\begin{eqnarray} formula text \end{eqnarray*}*

where the standard form adds a sequential equation number for each line and the *-form is without equation numbers.

5.3 Main elements of math mode:

5.3.1 Constants and variables:

Numbers that appear within formulas are called *constants*, whereas *simple variables* are represented by single letters. The universal practice in mathematical typesetting is to put constants in Roman typeface and variables in *italics*. LATEX adheres to this rule automatically in math mode. Blanks are totally ignored and are included in the input text simply to improve the appearance for the writer. Spacing between constants, variables, and operators like $+, -, =$ are set automatically by LATEX.

For example \$$z = 2a + 3y$\$, \$ $z \ = \ 2\ a \ + \ 3\ y$ \$ both produce $z \ = \ 2a \ + \ 3y$.

Mathematical symbols that are available on the keyboard are

$$+ \ - \ = \ < \ > \ / \ : \ ! \ ' \ | \ [\ ] \ (\ )$$

all of which may be used directly in formulas. The curly braces { } serve the purpose of logically combining parts of the formula and therefore cannot act as printable characters. To include braces in a formula, the same commands \{ and \} are used as in normal text.

$M(s) < M(t) < \|M\| = m$	`$M(s)<M(t)<|M| = m$`
$y'' = c\{f[y', y(x)] + g(x)\}$	`$y'' = c\{f[y',y(x)] + g(x)\}$`

Preparation: *Create a new LATEX file with the name math.tex containing at first only the commands \documentclass{article}, \begin{document}, and \end {document}.*

Exercise 5.1: *Produce the following text with your math exercise file: 'The derivative of the indirect function $f[g(x)]$ is $\{f[g(x)]\}' = f'[g(x)]g'(x)$. For the second derivative of the product of $f(x)$ and $g(x)$ one has $[f(x)g(x)]'' = f''(x)g(x) + 2f'(x)g'(x) + f(x)g''(x)$.'*
Note: higher derivatives are made with multiple ' symbols: \$y'''\$ yields y'''.

5.3.2 Exponents and indices:

Mathematical formulas often contain exponents and indices, characters that are either raised or lowered relative to the main line of the formula, and printed in a smaller typeface. Although their mathematical meanings are different, superscripts and subscripts are typographically the same things as exponents and indices, respectively. It is even possible that exponents themselves have exponents or indices, and so on. These are produced by multiple applications of the raising and lowering operations.

LATEX and TEX make it possible to create any combination of exponents and indices with the correct type size in a simple manner: the character command ^ sets the next character as an exponent (raised), while the character command _ sets it as an index (lowered).

x^2 `x^2` a_n `a_n` x_i^n `x^n_i`

When exponents and indices occur together, their order is unimportant. The last example above could also have been given as `x_i^n`.

If the exponent or index contains more than one character, the group of characters must be enclosed in braces { }:

x^{2n} `x^{2n}` x_{2y} `x_{2y}` $A_{i,j,k}^{-n!2}$ `A_{i,j,k}^{-n!2}`

Multiple raisings and lowerings are generated by applying ^ and _ to the exponents and indices:

x^{y^2} `x^{y^2}` x^{y_1} `x^{y_1}`

$A^{x_i^2}_{j^{2n}_{n,m}}$ `A^{x_i^2}_{j^{2n}_{n,m}}`

Note: The raising and lowering commands ^ and _ are only permitted in math mode.

5.3.3 Fractions:

Short fractions, especially within a text formula, are best represented using the slash character /, as in $(n + m)/2$ for $(n + m)/2$. For more complicated fractions, the command

\frac{*numerator*}{*denominator*}

is employed to write the *numerator* on top of the *denominator* with a horizontal fraction line of the right width between them.

$\frac{1}{x+y}$ `\[ \frac{1}{x+y} \]`

$\frac{a^2 - b^2}{a+b} = a - b$ `\[ \frac{a^2 - b^2}{a+b} = a-b \]`

Fractions may be nested to any depth within one another.

$\frac{\frac{a}{x-y} + \frac{b}{x+y}}{1 + \frac{a-b}{a+b}}$

```
\[ \frac{\frac{a}{x-y} + \frac{b}{x+y}}
        {1 + \frac{a-b}{a+b}}  \]
```

LATEX sets fractions within fractions in a smaller typeface.

5.3.4 Roots:

Roots are printed with the command

\sqrt[*n*]{*arg*}

as in the example: `$\sqrt[3]{8} = 2$` produces $\sqrt[3]{8} = 2$. If the optional argument n is omitted, the square root is generated: `$\sqrt{a}` yields $\sqrt{a}$.

The size and length of the root sign are automatically fitted to *arg*:

`$\sqrt{x^2 + y^2 + 2xy} = x+y$` $\sqrt{x^2 + y^2 + 2xy} = x + y$, or

$\sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}}$ `\[ \sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}} \]`

Roots may be nested inside one another to any depth:

$\sqrt[3]{-q + \sqrt{q^2 + p^3}}$ `\[ \sqrt[3]{-q + \sqrt{q^2 + p^3}} \]`

5.3.5 Sums and integrals:

Summation and integral signs are made with the two commands \sum and \int, which may appear in two different sizes depending on whether they occur in a text or displayed formula.

Sums and integrals very often possess upper and lower limits. These are printed with the exponent and index commands ^ and _. The positioning of the limits also depends on whether the formula is in text or displayed.

In a text formula `\sum_{i=1}^n` and `\int_a^b` produce $\sum_{i=1}^n$ and $\int_a^b$, whereas in a displayed formula they appear as at the left below:

$$\sum_{i=1}^{n} \quad \int_{a}^{b}$$

Some authors prefer the limits for the integral to be placed above and below the integral sign, the same as for summation. This is achieved with the command `\limits` immediately following the integral sign: `\int\limits_{x=0}^{x=1}`

$$\int\limits_{x=0}^{x=1}$$

The rest of the formula text coming before and after the sum and integral signs is correctly aligned with them.

$$2\sum_{i=1}^{n} a_i \int_{a}^{b} f_i(x)g_i(x)\,\mathrm{d}x$$

```
\[ 2\sum_{i=1}^n a_i \int^b_a
    f_i(x)g_i(x)\,\mathrm{d}x \]
```

5.3.6 Continuation dots—ellipsis:

Formulas occasionally contain a row of dots . . . , meaning *and so on*. Simply typing a period three times in a row produces an undesirable result: ..., that is, the dots are too close together. Therefore LATEX provides several commands

`\ldots`	$\ldots$	*low dots*	`\cdots`	$\cdots$	*center dots*
`\vdots`	$\vdots$	*vertical dots*	`\ddots`	$\ddots$	*diagonal dots*

to space the dots correctly. The difference between the first two commands is best illustrated by the examples $a_0, a_1, \ldots, a_n$ and $a_0 + a_1 + \cdots + a_n$, which are produced with `$a_0,a_1,\ldots,a_n$` for the first and `$a_0 + a_1 + \cdots + a_n$` for the second.

The command \ldots is also available in normal text mode, whereas the other three are only allowed in math mode. In text mode, the command \dots may be used in place of \ldots with the same effect.

Exercise 5.2: *Generate the following output:*
The reduced cubic equation $y^3 + 3py + 2q = 0$ has one real and two complex solutions when $D = q^2 + p^3 > 0$. These are given by Cardan's formula as

$$y_1 = u + v, \quad y_2 = -\frac{u+v}{2} + \frac{\mathrm{i}}{2}\sqrt{3}(u-v), \quad y_3 = -\frac{u+v}{2} - \frac{\mathrm{i}}{2}\sqrt{3}(u-v)$$

where

$$u = \sqrt[3]{-q + \sqrt{q^2+p^3}}, \qquad v = \sqrt[3]{-q - \sqrt{q^2+p^3}}$$

Note: the spacings between the parts of the displayed equations are made with the spacing commands `\quad` *and* `\qquad`.

Exercise 5.3: *Select the option* `fleqn` *in the document class command and include the specification* `\setlength{\mathindent}{2cm}` *in the preamble. Redo the three* y *equations above, each as a separate displayed formula, using the* `equation` *environment instead of* `displaymath` *or* `\[...\]` *brackets.*

Exercise 5.4: *Create the following text:*
Each of the measurements $x_1 < x_2 < \cdots < x_r$ occurs $p_1, p_2, \ldots, p_r$ times. The mean value and standard deviation are then

$$x = \frac{1}{n}\sum_{i=1}^{r} p_i x_i, \qquad s = \sqrt{\frac{1}{n}\sum_{i=1}^{r} p_i (x_i - x)^2}$$

where $n = p_1 + p_2 + \cdots + p_r$.

Exercise 5.5: *Although this equation looks very complicated, it should not present any great difficulties:*

$$\int \frac{\sqrt{(ax+b)^3}}{x}\,\mathrm{d}x = \frac{2\sqrt{(ax+b)^3}}{3} + 2b\sqrt{ax+b} + b^2 \int \frac{\mathrm{d}x}{x\sqrt{ax+b}}$$

The same applies to $\int_{-1}^{8} (\mathrm{d}x / \sqrt[3]{x}) = \frac{3}{2}(8^{2/3} + 1^{2/3}) = 15/2$.

5.4 Mathematical symbols:

There is a very wide range of symbols used in mathematical text, of which only a few are directly available from the keyboard. LATEX provides many of the mathematical symbols that are commonly used. They are called with the symbol name prefixed with the \ character. The names themselves are derived from their mathematical meanings.

5.4.1 Greek letters:

Lower case letters

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

Upper case letters

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

The Greek letters are made simply by putting the command character \ before the name of the letter. Upper case (capital) letters are distinguished by capitalizing the first letter of the name. Greek letters that do not appear in the above list are identical with some corresponding Latin letter. For example, upper case ρ is the same as Latin P and so needs no special symbol.

5.4.2 Calligraphic letters:

The following 26 *calligraphic* letters may also be used in math formulas:

$\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{H}, \mathcal{I}, \mathcal{J}, \mathcal{K}, \mathcal{L}, \mathcal{M}, \mathcal{N}, \mathcal{O}, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}, \mathcal{V}, \mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}$

These are called with the math alphabet command \mathcal:

```
$\mathcal{A, B, C,...,Z}$
```

5.4.3 Binary operators:

Two mathematical quantities combined with one another to make a new quantity are said to be joined by a *binary operation*. The symbols that are available for use as binary operators are

$\pm$	\pm	$\cap$	\cap	$\circ$	\circ	$\bigcirc$	\bigcirc
$\mp$	\mp	$\cup$	\cup	$\bullet$	\bullet	$\Box$	\Box
$\times$	\times	$\uplus$	\uplus	$\diamond$	\diamond	$\Diamond$	\Diamond
$\div$	\div	$\sqcap$	\sqcap	$\lhd$	\lhd	$\triangle$	\bigtriangleup
$\cdot$	\cdot	$\sqcup$	\sqcup	$\rhd$	\rhd	$\triangledown$	\bigtriangledown
$*$	\ast	$\vee$	\vee	$\unlhd$	\unlhd	$\triangleleft$	\triangleleft
$\star$	\star	$\wedge$	\wedge	$\unrhd$	\unrhd	$\triangleright$	\triangleright
$\dagger$	\dagger	$\oplus$	\oplus	$\oslash$	\oslash	$\setminus$	\setminus
$\ddagger$	\ddagger	$\ominus$	\ominus	$\odot$	\odot	$\wr$	\wr
$\amalg$	\amalg	$\otimes$	\otimes				

The underlined symbol names in the above and following tables are only available if one of the packages latexsym or amsfonts has been loaded.

5.4.4 Relations and their negations:

When two mathematical quantities are compared, they are connected by a *relation*. The different types of relational symbols for the various comparisons are

$\le$ \le \leq	$\ge$ \ge \geq	$\neq$ \neq	$\sim$ \sim
$\ll$ \ll	$\gg$ \gg	$\doteq$ \doteq	$\simeq$ \simeq
$\subset$ \subset	$\supset$ \supset	$\approx$ \approx	$\asymp$ \asymp
$\subseteq$ \subseteq	$\supseteq$ \supseteq	$\cong$ \cong	$\smile$ \smile
$\sqsubset$ \sqsubset	$\sqsupset$ \sqsupset	$\equiv$ \equiv	$\frown$ \frown
$\sqsubseteq$ \sqsubseteq	$\sqsupseteq$ \sqsupseteq	$\propto$ \propto	$\bowtie$ \bowtie
$\in$ \in	$\ni$ \ni	$\prec$ \prec	$\succ$ \succ
$\vdash$ \vdash	$\dashv$ \dashv	$\preceq$ \preceq	$\succeq$ \succeq
$\models$ \models	$\perp$ \perp	$\parallel$ \parallel \\|	$\mid$ \mid |

A number of the above symbols may be called by more than one name. For example, $\leq$ may be produced with either \le or \leq.

The following symbols may be negated in this manner. Note that the last two, \not\in and \notin, are not exactly the same: $\not\in$ and $\notin$. The latter form is the preferred one.

$\not<$ \not<	$\not>$ \not>	$\not=$ \not=
$\not\le$ \not\le	$\not\ge$ \not\ge	$\not\equiv$ \not\equiv
$\not\prec$ \not\prec	$\not\succ$ \not\succ	$\not\sim$ \not\sim
$\not\preceq$ \not\preceq	$\not\succeq$ \not\succeq	$\not\simeq$ \not\simeq
$\not\subset$ \not\subset	$\not\supset$ \not\supset	$\not\approx$ \not\approx
$\not\subseteq$ \not\subseteq	$\not\supseteq$ \not\supseteq	$\not\cong$ \not\cong
$\not\sqsubseteq$ \not\sqsubseteq	$\not\sqsupseteq$ \not\sqsupseteq	$\not\asymp$ \not\asymp
$\not\in$ \not\in	$\notin$ \notin	

5.4.5 Arrows and pointers:

Mathematical manuscripts often contain arrow symbols, also called *pointers*. The following arrow symbols are available:

←	\leftarrow	\gets	⟵	\longleftarrow		↑	\uparrow
⇐	\Leftarrow		⟸	\Longleftarrow		⇑	\Uparrow
→	\rightarrow	\to	⟶	\longrightarrow		↓	\downarrow
⇒	\Rightarrow		⟹	\Longrightarrow		⇓	\Downarrow
↔	\leftrightarrow		⟷	\longleftrightarrow		↕	\updownarrow
⇔	\Leftrightarrow		⟺	\Longleftrightarrow		⇕	\Updownarrow
↦	\mapsto		⟼	\longmapsto		↗	\nearrow
↩	\hookleftarrow		↪	\hookrightarrow		↘	\searrow
↼	\leftharpoonup		⇀	\rightharpoonup		↙	\swarrow
↽	\leftharpoondown		⇁	\rightharpoondown		↖	\nwarrow
⇌	\rightleftharpoons		⇝	\leadsto			

5.4.6 Various other symbols:

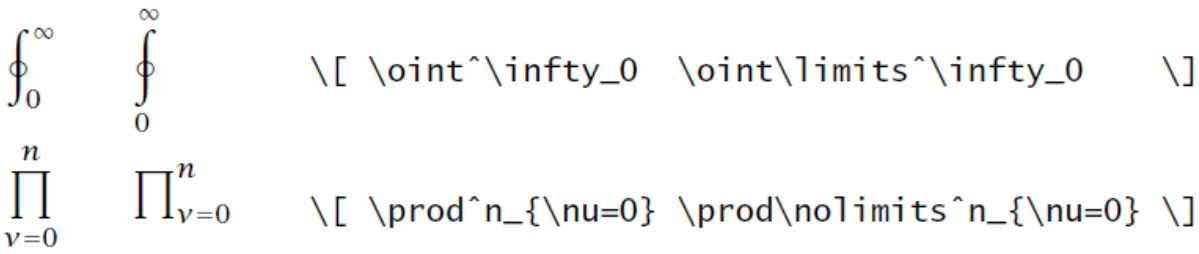
The above lists by no means exhaust the complete repertoire of mathematical symbols. However, the following are additional characters that standard LATEX does make available.

ℵ	\aleph	′	\prime	∀	\forall	□	\Box
ℏ	\hbar	∅	\emptyset	∃	\exists	◇	\Diamond
ı	\imath	∇	\nabla	¬	\neg	△	\triangle
ȷ	\jmath	√	\surd	♭	\flat	♣	\clubsuit
ℓ	\ell	∂	\partial	♮	\natural	◆	\diamondsuit
℘	\wp	⊤	\top	♯	\sharp	♥	\heartsuit
ℜ	\Re	⊥	\bot	∥	\\|	♠	\spadesuit
ℑ	\Im	⊢	\vdash	∠	\angle	⋈	\Join
℧	\mho	⊣	\dashv	\	\backslash	∞	\infty

5.4.7 Symbols with two sizes:

The following symbols are printed in different sizes depending on whether they appear in text or displayed formulas:

Similarly the complementary command \nolimits sets them beside the symbol when the standard positioning is above and below.



5.4.8 Function names:

The universal standard for mathematical formulas is to set variable names in *italics* but the names of functions in Roman. If one were simply to write the function names *sin* or *log* in math mode, LATEX would interpret these as variables s i n and l o g and write them as *sin* and *log*. In order to tell LATEX that a function name is wanted, it is necessary to enter a command consisting of the backslash \ plus the function name. The following names are recognized by LATEX:

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

5.4.9 Mathematical accents:

The following mathematical accents are available within math mode:

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

The letters i and j should be printed without their dots when they are given an accent. To accomplish this, type the symbols `\imath` and `\jmath` instead of the letters, as in

`$\vec{\imath} + \tilde{\jmath}$:` $\vec{\imath} + \tilde{\jmath}$

There are wider versions of `\hat` and `\tilde` available with the names `\widehat` and `\widetilde`. In this way, these accents may be placed over parts of a formula:

$\widehat{1-x} = \widehat{-y}$ `$\widehat{1-x}=\widehat{-y}$`

$\widetilde{xyz}$ `$\widetilde{xyz}$`

Exercise 5.6: *The union of two sets $\mathcal{A}$ and $\mathcal{B}$ is the set of all elements that are in at least one of the two sets, and is designated as $\mathcal{A} \cup \mathcal{B}$. This operation is commutative $\mathcal{A} \cup \mathcal{B} = \mathcal{B} \cup \mathcal{A}$ and associative $(\mathcal{A} \cup \mathcal{B}) \cup C = \mathcal{A} \cup (\mathcal{B} \cup C)$. If $\mathcal{A} \subseteq \mathcal{B}$, then $\mathcal{A} \cup \mathcal{B} = \mathcal{B}$. It then follows that $\mathcal{A} \cup \mathcal{A} = \mathcal{A}$, $\mathcal{A} \cup \{\varnothing\} = \mathcal{A}$ and $\mathcal{J} \cup \mathcal{A} = \mathcal{J}$.*

Exercise 5.7: *Applying l'Hôpital's rule, one has*

$$\lim_{x\to 0} \frac{\ln \sin \pi x}{\ln \sin x} = \lim_{x\to 0} \frac{\pi \frac{\cos \pi x}{\sin \pi x}}{\frac{\cos x}{\sin x}} = \lim_{x\to 0} \frac{\pi \tan x}{\tan \pi x} = \lim_{x\to 0} \frac{\pi / \cos^2 x}{\pi / \cos^2 \pi x} = \lim_{x\to 0} \frac{\cos^2 \pi x}{\cos^2 x} = 1$$

Exercise 5.8: *The gamma function $\Gamma(x)$ is defined as*

$$\Gamma(x) \equiv \lim_{n\to\infty} \prod_{\nu=0}^{n-1} \frac{n!\, n^{x-1}}{x+\nu} = \lim_{n\to\infty} \frac{n!\, n^{x-1}}{x(x+1)(x+2)\cdots(x+n-1)} \equiv \int_0^\infty \mathrm{e}^{-t} t^{x-1}\, \mathrm{d}t$$

The integral definition is valid only for $x > 0$ (2nd Euler integral).

Exercise 5.9: *Remove the option `fleqn` from the document class command in Exercise 5.3 and redo the output.*

Exercise 5.10:

$\alpha\vec{x} = \vec{x}\alpha$, $\alpha\beta\vec{x} = \beta\alpha\vec{x}$, $(\alpha+\beta)\vec{x} = \alpha\vec{x} + \beta\vec{x}$, $\alpha(\vec{x}+\vec{y}) = \alpha\vec{x} + \alpha\vec{y}$.
$\vec{x}\vec{y} = \vec{y}\vec{x}$ *but* $\vec{x}\times\vec{y} = -\vec{y}\times\vec{x}$, $\vec{x}\vec{y} = 0$ *for* $\vec{x} \perp \vec{y}$, $\vec{x}\times\vec{y} = 0$, *for* $\vec{x} \parallel \vec{y}$.

5.5 Additional Elements:

The math elements described in the previous sections already permit the construction of very complex formulas, such as

$$\lim_{x\to 0}\frac{\sqrt{1+x}-1}{x}=\lim_{x\to 0}\frac{(\sqrt{1+x}-1)(\sqrt{1+x}+1)}{x(\sqrt{1+x}+1)}=\lim_{x\to 0}\frac{1}{\sqrt{1+x}+1}=\frac{1}{2} \tag{5.1}$$

$$\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial y^2}=0 \quad\Longrightarrow\quad U_M=\frac{1}{4\pi}\oint\limits_{\Sigma}\frac{1}{r}\frac{\partial U}{\partial n}\,\mathrm{d}s-\frac{1}{4\pi}\oint\limits_{\Sigma}\frac{\partial\frac{1}{r}}{\partial n}U\,\mathrm{d}s \tag{5.2}$$

$$I(z)=\sin(\frac{\pi}{2}z^2)\sum_{n=0}^{\infty}\frac{(-1)^n\pi^{2n}}{1\cdot 3\cdots(4n+1)}z^{4n+1}-\cos(\frac{\pi}{2}z^2)\sum_{n=0}^{\infty}\frac{(-1)^n\pi^{2n+1}}{1\cdot 3\cdots(4n+3)}z^{4n+3} \tag{5.3}$$

By reading the formulas from left to right there should be no difficulty in reconstructing the text that produced them. For example, the last equation is generated with

```
\begin{equation}
I(z) = \sin( \frac{\pi}{2} z^2 ) \sum_{n=0}^\infty
   \frac{ (-1)^n \pi^{2n} }{1 \cdot 3 \cdots (4n+1) } z^{4n+1}
  -\cos( \frac{\pi}{2} z^2 ) \sum_{n=0}^\infty
  \frac{ (-1)^n \pi^{2n+1} }{ 1 \cdot 3 \cdots (4n+3) } z^{4n+3}
\end{equation}
```

The above examples were made using the equation environment instead of the displaymath environment or its abbreviated form \[. . . \], which has the effect of adding the equation numbers automatically. In the document classes book and report, equations are sequentially numbered within the chapter, the number being preceded by the chapter number and set within parentheses ( ), as illustrated above. For document class article, the equations are numbered sequentially throughout the entire document.

5.5.1 Automatic sizing of bracket symbols:

Mathematics often contains bracketing symbols, usually in pairs that enclose part of the formula. When printed, these bracket symbols should be the same size as the included partial formula. LATEX provides a pair of commands

\left *lbrack* *sub form* \right *rbrack*

to accomplish this. The command \left is placed immediately before the opening (left hand) bracket symbol *lbrack* while \right comes just before the closing (right hand) symbol *rbrack*.

$$\left[\int + \int\right]_{x=0}^{x=1}$$

```
\[ \left[ \int + \int \right]_{x=0}^{x=1} \]
```

The pair of brackets [ ] is adjusted to the size of the enclosed formula, as are the raised exponent and lowered index as well.

The commands \left and \right must appear as a pair. For every \left command there must be a corresponding \right command somewhere afterwards. The pairs may be nested. The first \left is paired with the last \right; the following \left with the second last \right, and so on. There must be the same number of \right as \left commands in a nesting.

The corresponding bracket symbols *lbrack* and *rbrack* may be perfectly arbitrary and do not need to be a logical pair.

$$\vec{x} + \vec{y} + \vec{z} = \left(\begin{array}{c} a \\ b \end{array} \right[$$

This set of brackets is admittedly unusual but permissible.

```
\[ \vec{x} + \vec{y} + \vec{z} =
    \left( ... \right[ \]
```

Sometimes a formula contains only a single opening or closing bracket without a corresponding counterpart. However, the \left...\right commands must still be given as a pair, but with a period '.' as an *invisible* bracket symbol.

$$y = \left\{ \begin{array}{rcl} -1 & : & x < 0 \\ 0 & : & x = 0 \\ +1 & : & x > 0 \end{array} \right.$$

```
\[  y = \left\{ \begin{array}
      {r@{\quad:\quad}l}
    -1 & x<0 \\ 0 & x=0 \\ +1 & x>0
      \end{array} \right.   \]
```

The \left...\right commands may be applied to a total of 22 different symbols. These are

((	))	⌊ \lfloor	⌋ \rfloor
[[	]]	⌈ \lceil	⌉ \rceil
{ \{	} \}	⟨ \langle	⟩ \rangle
| |	‖ \\|	↑ \uparrow	⇑ \Uparrow
/ /	\ \backslash	↓ \downarrow	⇓ \Downarrow
		↕ \updownarrow	⇕ \Updownarrow

For example, \left|...\right| produces two vertical bars adjusted in height to contain the enclosed formula text.

5.5.2 Ordinary text within a formula:

It is often necessary to include some *normal* text within a formula, for example single words such as *and*, *or*, *if*, and so on. In this case one must switch to LR mode while staying in math mode. This is carried out with the command \mbox{*normal text*} given inside the formula, together with horizontal spacing commands such as \quad or \hspace. For example:

$$X_n = X_k \quad \text{if and only if} \quad Y_n = Y_k \quad \text{and} \quad Z_n = Z_k$$

```
\[  X_n = X_k \qquad\mbox{if and only if}\qquad
    Y_n = Y_k \quad\mbox{and}\quad Z_n = Z_k          \]
```

5.5.3 Matrices and arrays:

$$\begin{matrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{matrix}$$

Structures like the one at the left are the basis for matrices, determinants, system of equations, and so on. They will all be referred to here as *arrays*.

Arrays are produced by means of the array environment. The array environment generates a table in math mode, that is, the column entries are interpreted as formula text. For example:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{22}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

```
\[  \begin{array}{*{3}{c@{\:+\:}}c@{\;=\;}c}
      a_{11}x_1 & a_{12}x_2 & \cdots & a_{1n}x_n & b_1 \\
      a_{22}x_1 & a_{22}x_2 & \cdots & a_{2n}x_n & b_2 \\
        \multicolumn{5}{c}{\dotfill}                  \\
      a_{n1}x_1 & a_{n2}x_2 & \cdots & a_{nn}x_n & b_n
    \end{array}     \]
```

5.5.4 Lines above and below formulas:

The commands

\overline{*sub_form*} *and* \underline{*sub_form*}

can be used to draw lines over or under a formula or sub-formula. They may be nested to any level:

$$\overline{\overline{a}^2 + \underline{xy} + \overline{\overline{z}}}$$

```
\[  \overline{\overline{a}^2 + \underline{xy}
                + \overline{\overline{z}}}     \]
```

The command \underline may also be employed in normal text mode to underline text, whereas \overline is allowed only in math mode.

Exactly analogous to these are the two commands

\overbrace{*sub_form*} *and* \underbrace{*sub_form*}

which put horizontal curly braces above or below the sub-formula.

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

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

In displayed formulas, these commands may have exponents or indices attached to them. The (raised) exponent is set above the *overbrace* while the (lowered) index is placed below the *underbrace*.

$$\underbrace{a+\overbrace{b+\cdots+y}^{123}+z}_{\alpha\beta\gamma}$$

```
\[  \underbrace{a + \overbrace{b + \cdots +
      y}^{123} + z}_{\alpha\beta\gamma}  \]
```

Exercise 5.11: The total number of permutations of *n* elements taken *m* at a time (symbol P_n^m) is

$$P_n^m = \prod_{i=0}^{m-1}(n-i) = \underbrace{n(n-1)(n-2)\ldots(n-m+1)}_{\text{total of } m \text{ factors}} = \frac{n!}{(n-m)!}$$

5.5.5 Stacked symbols:

The command

```
\stackrel{upper_sym}{lower_sym}
```

places the symbol *upper_sym* centered on top of *lower_sym*, whereby the symbol on top is set in a smaller typeface.

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

$A \xrightarrow{\alpha'} B \xleftarrow{\beta'} C$ `$ A \stackrel{\alpha'}{\longrightarrow} B ... $`

By making use of math font size commands it is possible to construct new symbols with this command. For example, some authors prefer to have the `\le` symbol appear as $\overset{<}{=}$ instead of $\leq$. This is achieved by combining $<$ and $=$ with `$\stackrel{\textstyle<}{=}$`. If the command `\textstyle` were omitted, the symbol would be printed as $\overset{<}{=}$.

5.5.6 Additional TEX commands for math:

The TEX math commands \atop and \choose are useful additions to the set of commands and may be applied within any LATEX document. (In fact, all TEX math commands except \eqalign, \eqalignno, and \leqaligno may be used in a LATEX manuscript.) Their syntax is

{*top* \atop *bottom*}

{*top* \choose *bottom*}

Both commands produce a structure that looks like a fraction without the dividing line. With the \choose command, this structure is also enclosed within round brackets (in mathematics this is called a *binomial coefficient*).

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$$

```
\[ {n+1 \choose k} =
   {n \choose k} + {n \choose k-1} \]
```

$$\prod_{j\ge0}\left(\sum_{k\ge0} a_{jk}z^k\right) = \sum_{n\ge0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\ge0\\ k_0+k_1+\cdots=0}} a_{0k_0}a_{1k_1}\ldots\right)$$

```
\[ \prod_{j\ge0}\left( \sum_{k\ge0} a_{jk}z^k \right) =
   \sum_{n\ge0} z^n \left(\sum_{k_0,k_1\ldots\ge0 \atop
       k_0+k_1+\cdots=0} a_{0k_0} a_{1k_1}\ldots \right)   \]
```

Similar structures can be generated with the LaTeX environments

```
\begin{array}{c}   upper_line \\ lower_line \end{array}        (atop)
\left(\begin{array}{c} upper \\ lower \end{array}\right)  (choose)
```

The difference between these array structures and those of the TEX commands is that the former are always printed in the size and style of normal text formulas, whereas the latter will have varying sizes depending on where they appear within the formula.

For comparison

$$\Delta^{p_1 p_2 \cdots p_{n-k}}_{p_1 p_2 \cdots p_{n-k}}$$ The index array is produced using \atop

$$\Delta \begin{array}{l} p_1 p_2 \cdots p_{n-k} \\ p_1 p_2 \cdots p_{n-k} \end{array}$$ The index array is produced using \array

The above TEX commands may also be employed to produce small matrices within text formulas, such as $\binom{1\,0}{0\,1}$ or $\left(\begin{smallmatrix} a & b & c \\ l & m & n \end{smallmatrix}\right)$. Here the first matrix was typed in with

```
${1\,0\choose0\,1}$
```

and the second with

```
$\left({a\atop l}{b\atop m}{c\atop n}\right)$
```

5.5.7 Multiline equations:

A multiline equation is one that is developed over several lines in which the relation symbols (for example, $=$ or $\leq$) in each line are all vertically aligned with each other. For this purpose, the environments

\begin{eqnarray} *line 1*\\ ... \\ *line n* end{eqnarray}
\begin{eqnarray*} *line 1*\\ ... \\ *line n* end{eqnarray*}

are used to set several lines of formulas or equations in displayed math mode. The individual lines of the equation or formula are separated from one another by \\. Each entry line has the form

left formula & *mid formula* & *right formula* \\

When printed, all the *left formula*s appear right justified in a left column, the *right formula*s left justified in a right column, and the *mid formula*s centered in between. The column separation character & designates the various parts of the

formula. Normally the *mid formula* is a single math character, the relation operator mentioned above. The individual lines thus have the same behavior as they would in a \begin{array} {rcl}. . . \end{array} environment.

Examples:

$$\begin{aligned}(x+y)(x-y) &= x^2 - xy + xy - y^2 \\ &= x^2 - y^2 \qquad (5.4)\\ (x+y)^2 &= x^2 + 2xy + y^2 \qquad (5.5)\end{aligned}$$

```
\begin{eqnarray}
    (x+y)(x-y) & = & x^2-xy+xy-y^2 \nonumber\\
               & = & x^2 - y^2 \\
    (x+y)^2    & = & x^2 + 2xy + y^2
\end{eqnarray}
```

$$\begin{aligned}x_n u_1 + \cdots + x_{n+t-1}u_t &= x_n u_1 + (ax_n + c)u_2 + \cdots \\ &\quad + \left(a^{t-1}x_n + c(a^{t-2} + \cdots + 1)\right)u_t \\ &= (u_1 + au_2 + \cdots + a^{t-1}u_t)x_n + h(u_1, \ldots, u_t)\end{aligned}$$

```
\begin{eqnarray*}
x_nu_1 + \cdots + x_{n+t-1}u_t & = & x_nu_1 + (ax_n + c)u_2 +
                                                    \cdots\\
 &   & + \left(a^{t-1}x_n + c(a^{t-2} + \cdots+1)\right)u_t\\
 & = & (u_1 + au_2 + \cdots + a^{t-1}u_t)x_n + h(u_1,\ldots,u_t)
\end{eqnarray*}
```

5.5.8 Framed or side-by-side formulas:

Displayed formulas or equations may be put into vertical boxes of appropriate width, that is, in a \parbox command or minipage environment. Within the vertical box, the formulas are horizontally centered or left justified with indentation \mathindent according to the selected document class option.

Vertical boxes may be positioned relative to one another just like single characters. In this way the user may place displayed formulas or equations side by side.

$$\alpha = f(z) \quad (5.8)$$
$$\beta = f(z^2) \quad (5.9)$$
$$\gamma = f(z^3) \quad (5.10)$$

$$x = \alpha^2 - \beta^2$$
$$y = 2\alpha\beta$$

The left-hand set of equations is set in a \parbox of width 4 cm, the right-hand set in one of width 2.5 cm, while this text is inside a minipage of width 4.5 cm.

```
\parbox{4cm}{\begin{eqnarray} \alpha &=& f(z)...\end{eqnarray}}
\hfill  \parbox{2.5cm}{\begin{eqnarray*}
  x &=& \alpha^2 - \beta^2\\ y &=& 2\alpha\beta \end{eqnarray*}}
\hfill  \begin{minipage}{4.5cm} The left-hand ... \end{minipage}
```

To add a vertically centered equation number to a set of equations, for example,

$$\begin{aligned} P(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \\ P(-x) &= a_0 - a_1x + a_2x^2 - \cdots + (-1)^n a_nx^n \end{aligned} \quad (5.11)$$

the following text may be given:

```
\parbox{10cm}{\begin{eqnarray*} ... \end{eqnarray*}} \hfill
\parbox{1cm}{\begin{eqnarray}\end{eqnarray}}
```

5.5.9 Chemical formulas and bold face in math formulas:

In mathematics it is sometimes necessary to set individual characters or parts of the formula in bold face. This can be achieved simply with the math alphabet command \mathbf.

```
$\mathbf{S^{-1}TS = dg(\omega_1,\ldots,\omega_n) = \Lambda}$
```

produces $\mathbf{S^{-1}TS = dg(\omega_1,\ldots,\omega_n) = \Lambda}$.

In this example, the entire formula has been set as the argument of \mathbf so that everything should be set in bold face. In fact, only numbers, lower and upper case

Latin letters, and upper case Greek letters are set in **bold Roman** with \mathbf. Lower case Greek letters and other math symbols appear in the normal *math* font.

If only part of the formula is to be set in bold face, that part must be given as the argument of the \mathbf command.

```
$\mathbf{2\sqrt{x}/y} = z$
```

$\mathbf{2\sqrt{x}/y} = z$

The math font style command \boldmath will set all characters in bold face, with the following exceptions:

- raised and lowered symbols (exponents and indices)
- the characters + : ; ! ? ( ) [ ]
- symbols that exist in two sizes

The \boldmath declaration may not appear in math mode. It must be called before switching to math mode or within a parbox or minipage. The countercommand \unboldmath resets the math fonts back to the normal ones.

$$\oint_C V\,\mathrm{d}\boldsymbol{\tau} = \oint_\Sigma \nabla \times V\,\mathrm{d}\boldsymbol{\sigma}$$

```
\boldmath \[ \oint\limits_C V
  \,\mathrm{d}\tau =\oint\limits_\Sigma
  \nabla\times V\,d\sigma \] \unboldmath
```

If \boldmath has been turned on outside the math mode, it may be temporarily turned off inside with \mbox{\unboldmath$...$}.

```
\boldmath\( P = \mbox{\unboldmath$m$}b\)\unboldmath
```

yields: $\boldsymbol{P} = m\boldsymbol{b}$. Similarly, \boldmath can be temporarily turned on within math mode with the structure \mbox{\boldmath$...$}: $W_r = \int \boldsymbol{M}\,\mathrm{d}\boldsymbol{\varphi} = r^2 m\omega^2/2$

```
\( W_r = \int\mbox{\boldmath$M\,\mathrm{d}\varphi$} =..\)
```

An alternative method of printing single symbols in bold face is provided by the bm package.

Chemical formulas are normally set in Roman type, not in italics as for mathematical formulas. This may be brought about by setting the formula as the argument of the font command \mathrm:

```
$\mathrm{Fe_2^{2+}Cr_2O_4}$
```

$\mathrm{Fe_2^{2+}Cr_2O_4}$

Study Learning Material Prepared by

Dr. I. VALLIAMMAL, M.Sc., M.Phil.,Ph.D.,

Assistant Professor,Department of Mathematics,

ManonmaniamSundaranarUniversity,Tirunelveli-12,

Tamil Nadu, India.