How to Reproduce this Book Exactly with LaTeX

A Self-contained Tutorial on Writing Mathematical Notes

C. L. Loi

September 18, 2025

"How to Reproduce this Book Exactly with LaTeX"

Contents

Chapter 1

The Basic Set-up and Structure of a LaTeX Book

Introduction The first chapter discusses how to properly configure LaTeX files and organize the content's structure so that we can generate our first readable LaTeX book PDF.

1.1 Class, Commands, Options, and Packages

Class For each LaTeX document, we need to specify its *class*. Throughout this book, we will use the `scrbook` class provided by the **KOMA-Script**. To do so, we write `\documentclass{scrbook}` at the very beginning (*preamble*) of the main TeX file. Although not explored in this book, some other notable classes that may be of use include `beamer`, `moderncv`, and `article` (or `scrartcl`).

Commands and Options The `scrbook` class provides several *options* to customize the format of the book. We can either supply the arguments when declaring the class, or use the command `\KOMAoptions` in the preamble. A *command* works like a function in common programming languages and performs some specific action. Commands in LaTeX are denoted by the backslash `\` as the first character. In this book, we have used

```
\KOMAoptions{paper=a4, fontsize=12pt, chapterprefix=true, twoside=
    semi, DIV=classic, parskip=half}
```

The arguments are typed inside the curly brackets `{}` following the name of the command. Clearly, the `paper` option requires the pages to be in A4 size while `fontsize` indicates that the font is 12 pt large. The remaining options will be explained as we go through the later chapters.

Packages To enable extra functionalities, we need to import *packages*. We can write along the lines of `\usepackage[<options>]{<package_name>}` in the preamble to do so. We will not list all the required packages now at once, but only when they are needed. The first package we usually need is the `fontenc` package with the `T1` option, flagged inside a pair of square brackets.

Exercise(s)

1.1) Try to import the `fontenc` package with the `T1` option as suggested above. There may not be any noticeable difference, but at least you should not be receiving errors.

1.2) Also, try to use `\documentclass[<options>]{scrbook}` instead of the `\KOMAoptions` command to achieve the same class setting.

1.2 Structure Hierarchy

1.2.1 Chapters and (Sub-)Sections

Chapters, Sections As in any other book, the entire content is divided into *chapters*, which in turn usually consist of several *sections*. To mark the beginning of a chapter or section, we place the commands `\chapter{<chapter_name>}` or `\section{<section_name>}` within the `document` environment, which contains the main content and is marked by a pair of `begin` and `end` declarations. The preamble has to be inserted before `document`. So, to typeset the very first section at the start, we write

```
<preamble before the main document>
\begin{document}
...
\chapter{The Basic Set-up and Structure of a \LaTeX{} Book}
...
\section{Class, Options, and Packages}
\paragraph{Class}
For each \LaTeX{} document, we need to specify its \textit{class}.
   Throughout this book, ...
...
\end{document}
```

The LaTeX system updates the chapter/section's numbering internally. The `\textit{<text>}` command presents the text in italic shape.

Subsections, Paragraphs An attentive reader may have already figured out that it is possible to stack an extra layer (a *subsection*) in the hierarchy. This is aptly done not long ago by the `\subsection{<section_name>}` command:

```
\section{Structure Hierarchy}

\subsection{Chapters and (Sub-)Sections}
```

```
\paragraph{Chapters, Sections} As in any other book, the entire
   content is divided into \textit{chapters}, ...
```

He/she may also notice that we have used the `\paragraph` command a few times to attach an unnumbered heading for each *paragraph*. There are also starred versions like `\chapter*{<chapter_name>}`, `\section*{<section_name>}`, `\subsection*{<section_name>}`, and so on, which neither display nor increase the numbering/counters.

1.2.2 Generating Table of Contents

Table of Contents After establishing the structure of the book, it is convenient to generate a *table of contents (TOC)* as well. In the `scrbook` class, it is easily done by adding the command `\tableofcontents` within the main `document` group. To control the depth of layers shown, we can call `\setcountertocdepth {<integer>}` in the preamble, where the `integer` usually ranges from -1 to 3 (0: chapters, 1: sections, 2: subsections).

Exercise(s)

1.3) Try to add some (numbered or unnumbered) chapters, sections, subsections, or even subsubsections (which are, not surprisingly, produced by `\subsubsection`) to see how they are displayed in the book. You may want to check out `\part`.

1.4) As a follow-up to the last exercise, turn on the table of contents and confirm how the new entries are linked to it. Also, try to adjust the value for `\setcountertocdepth` as proposed above to see the effect.

1.2.3 Organizing the TEX Files behind the Scenes

include As the size of the project scales up, it is often helpful to keep the files arranged in a clean order for maintenance. We can put the content of each chapter into separate TEX files, and then use the `\include{<tex_file_name>}` command to import them into the main script. For example, this chapter is stored as `ch1_basic_structure.tex` in my project space, and in the main TEX file, we shall write something like

```
<preamble>
\begin{document}

\tableofcontents
\include{ch1_basic_structure}
...
\end{document}
```

1.3 Testing the Book Layout by Lipsum

Dummy Text Sometimes we may need to insert some placeholder text into the code to test how well the book will look in a specific layout. In this case, we can borrow the standard dummy text *Lorem Ipsum* (or in short *Lipsum*) widely used by the community. Just import the `lipsum` generator package, and add `\lipsum[<paragraph_no.>]` to the desired positions. For example, the code segment

```
...
produces the following text exactly: \par
\lipsum[1-2]
```

produces the following text exactly:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu

libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

The `\par` command signals the end of a paragraph and appends a vertical line spacing afterwards.

Chapter 2

Formatting of Text and Paragraphs

Introduction This chapter explains how to adjust the various aspects of text, such as fonts, shape/size/style, and positioning.

2.1 About Fonts

2.1.1 The Three Font Family Types

(Sans) Serif, Typewriter In any LaTeX document, the text can be typed in three different *font families*: *serif*, *sans serif*, and *typewriter*. In this book, headings (of chapters, sections, etc.) are in the sans serif family, while the remaining main text is in serif. Table 2.1 below demonstrates how to select a specific font family for a piece of text. For instance, both

Font Family	Command	Switch	Output
Serif	`\textrm{Hello World!}`	`\rmfamily`	Hello World!
Sans Serif	`\textsf{Hello World!}`	`\sffamily`	Hello World!
Typewriter	`\texttt{Hello World!}`	`\ttfamily`	`Hello World!`

Table 2.1: The commands for switching between the three font families and how they appear.

```
...
produces the following output: \par
\textsf{\lipsum[3]} % or {\sffamily \lipsum[3]}, remember the curly
   brackets {} to limit the scope of the \sffamily command.
```

produces the following output:

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

The % symbol indicates a trailing *comment* (highlighted in green) that is neither interpreted nor displayed.

2.1.2 Changing the Actual Font for a Font Family

Font Libraries Each of the previous font families is internally assigned a specific *font*. To change the actual fonts, we can call the corresponding font package(s). The **LaTeX Font Catalogue** https://tug.org/FontCatalogue/ provides a comprehensive list of available fonts and the way to import them. This book has substituted the **Noto Sans** font for the sans serif family, via the preamble

```
\usepackage[T1]{fontenc}
\usepackage[sf]{noto}
```

Exercise(s)

2.1) Change the font family just for the dummy Lipsum paragraph above to typewriter.

2.2) Choose a font of your liking from the Font Catalogue to replace the original one in the book.

2.2 Text Attributes

2.2.1 Font Size

Size Commands In Section 1.1 we talked about setting the base global font size by `\KOMAoptions`. However, to control the *local* font size for some places, we can use the *size commands*, listed in Table 2.2 below. For example, writing

```
... produces \par
{\small Though she be but little} {\LARGE she is fierce} \\ % scope
\scriptsize % switch
taken from Shakespeare's A Midsummer Night's Dream
```

Table 2.2: The various commands for text size.[1]

Command	Output
`\tiny`	Who am I?
`\scriptsize`	Who am I?
`\footnotesize`	Who am I?
`\small`	Who am I?
`\normalsize`	Who am I?
`\large`	Who am I?
`\Large`	Who am I?
`\LARGE`	Who am I?
`\huge`	Who am I?
`\Huge`	Who am I?

```
\normalsize % back to default ...
```

produces

Though she be but little she is fierce

taken from Shakespeare's A Midsummer Night's Dream

The `\\` sign breaks the current line and starts a new line right below. And again, the curly brackets `{}` limit the effect of command(s) within the scope.

selectfont It is also possible to fix a numerical value for the font size using `\fontsize{<font_size>}{<line_spacing>}` and `\selectfont`. As an illustration, the code

[1] `\huge` and `\Huge` have the same size when the font size is 12 pt (but different for 10 or 11 pt).

Font Style	Command	Switch	Output
Bold	`\textbf{"10 Downing"}`	`\bfseries`	**"10 Downing"**
Medium	`\textmd{"10 Downing"}`	`\mdseries`	"10 Downing"
Italic	`\textit{"10 Downing"}`	`\itshape`	*"10 Downing"*
Slanted	`\textsl{"10 Downing"}`	`\slshape`	*"10 Downing"*
Small Caps	`\textsc{"10 Downing"}`	`\scshape`	"10 Downing"
Upright	`\textup{"10 Downing"}`	`\upshape`	"10 Downing"

Table 2.3: The commands for different font styles. The medium/upright style is effectively the default normal.

```
... leads to \par
{\fontsize{15pt}{21pt}\selectfont May those who accept their fate be
   granted happiness. May those who defy their fate be granted glory.
    \\
-- Princess Tutu \par} % the \par is needed to renew the line spacing
```

leads to

May those who accept their fate be granted happiness. May those who defy their fate be granted glory.

– Princess Tutu

2.2.2 Font Shapes

Italic, Bold, and More Similar to font families, there are different *font shape/styles* such as the commonly seen italic or bold. Table 2.3 above shows the relevant commands to invoke them. Adding to the previous example, we can write

```
... which produces \par
\textit{\small Though she be but little} {\LARGE \bfseries \scshape
   she is fierce} \\ % scope
```

```
\scriptsize % switch
taken from \slshape \underline{Shakespeare's A Midsummer Night's
    Dream}
\normalsize \upshape % back to default ...
```

which produces

Though she be but little SHE IS FIERCE

taken from *Shakespeare's A Midsummer Night's Dream*

We also have `\underline` and `\emph`. You may want to try them out.

2.2.3 Text Color

xcolor While there are default colors in the LaTeX system, we can load a variety of additional colors from the `xcolor` package, often with flags as

```
\usepackage[svgnames, dvipsnames]{xcolor}
```

The reference color list can be found in https://www.overleaf.com/learn/latex/Using_colors_in_LaTeX. To set the color for a piece of text, we can enclose it with the `\textcolor{<color_name>}{<text>}` command. It is also possible to change the color within a group by `\color{<color_name>}`. For instance,

```
... outputs \par
\textcolor{Red}{Roses are red,} \\
\textcolor{Blue}{violets are blue,} \\
{\color{Purple} Sugar is sweet and so are you.} % remember to limit
    the scope by the curly brackets!
```

outputs

Roses are red,
violets are blue,
Sugar is sweet and so are you.

Self-defined colors It is also possible to design a custom color by the command `\definecolor{<color_name>}{<color_model>}{<values>}`. There are 4 possible color models: `rgb`, `RGB`, `cmyk`, and `gray`. For example,

```
...
\definecolor{mint}{rgb}{0.24, 0.71, 0.54} % in the preamble
... gives
\textcolor{mint}{Mint Tears}
```

gives Mint Tears. Color codes can be checked via https://latexcolor.com/.

In addition, we can mix colors by the expression `<color_1>!<mix_ratio>!<color_2>`. For instance,

```
\textcolor{Blue!40!Green}{Copper (II)} \textcolor{Orange!50}{Sulphate
    }
```

is displayed as Copper (II) Sulphate.

2.3 Paragraphs and Positioning

2.3.1 Paragraphs and Line Breaks

New Lines As explained before, the `\\` symbol issues a *line break*, and the `\par` command ends a paragraph and starts a new one.
Both of them initiate a *new line*, but with (without) an extra *line skip/spacing* for `\par` (`\\`). There is also `\newline` which is seldom used.

A blank line in the TEX file has the same effect as `\par`. They, in fact, end the so-called *horizontal mode* and distribute the text into lines placed on the current vertical list (see TEX StackExchange 82664).

The effects of `\\`, `\par`, and blank lines can be observed right in this subsection, which is typed as

```
\paragraph{New Lines} As explained before, ... ends a paragraph and
   starts a new one. \\
Both of them initiate a \textit{new line}, ... which is seldom used.
                % Here is a blank line plus this comment only
A blank line in the \TeX{} file ... placed on the current vertical
   list (see ...). \par
The effects of \texttt{\textbackslash\textbackslash}, \texttt{\
   textbackslash par}, and blank lines can be observed right in this
   subsection, which is typed as
... % this code block
```

2.3.2 Justification and Indents

raggedleft/right, centering The `\raggedleft` and `\raggedright` commands produce *right/left-justified* text respectively. As you may have figured out, this paragraph is "*ragged* right" (although not very obvious, notice →) so that the text sticks to the left boundary, but the right side is now uneven.

Meanwhile, this lipsum text is "ragged left": Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

The default setting is *fully-justified* so that the text extends to both edges like this one. `\raggedleft` and `\raggedright` act like a switch, changing all paragraphs beyond, and we may want to put them within a group enclosed by curly brackets.

We also have `\centering` which is quite self-explanatory and is demonstrated here. For these three commands to work properly, we require `\par` to finish,

similar to before. The code to generate the above paragraphs is

```
\paragraph{Raggedleft/right, Centering}
{\raggedright The \texttt{\textbackslash raggedleft} and \texttt{\
   textbackslash raggedright} ... but the right side is now uneven. \
   par}
{\raggedleft Meanwhile, this lipsum text is "ragged left": \lipsum
   [4]\par}
The default setting is \textit{fully-justified} ... we may want to
   put them within a group enclosed by curly brackets. \par
{\centering We also have \texttt{\textbackslash centering} ... The
   code to generate the above paragraphs is ... \par}
```

flushleft/right, center (Environments) The alternative to the above is to put the text into a `flushleft`/`flushright`/`center` environment. An *environment* contains content that is to be processed and displayed according to the specific design indicated by the environment. Environments always start with the `\begin{<env_name>}` and end with the `\end{<env_name>}` statements. For example, the previous part can also be reproduced by

```
...
\begin{flushright}
Meanwhile, this lipsum text is "ragged left": \lipsum[4]
\end{flushright}
...
\begin{center}
We also have \texttt{\textbackslash centering} ... The code to
   generate the above paragraphs is
\end{center}
```

`flushright` corresponds to `\raggedleft` and so is the opposite direction. If you test this new code, notice the increased separation[2] around the environments.

[2]This is dictated by `\topsep`, see the next subsection.

Indents, parskip Attentive readers may have figured out that there is no *indent* for paragraphs in the book, and they are only separated by a slight vertical spacing. This is controlled by the `parskip=half` value inside `\KOMAoptions` in the preamble, which means that paragraphs are identified with a vertical spacing of half a line. The two other options `parskip=no` and `parskip=full` use indents (without vertical spacing) and one full line instead.

Also, we can control indents manually by adding `\indent`[3] or `\noindent` to the start of paragraphs.

microtype Finally, for a better typesetting behavior, it is recommended to always import the `microtype` package, which provides helpful patches on this.

Exercise(s)

2.3) Test with different `parskip` options (there are additional modifiers like `half-`, `half+`, `half*`, similar for `full`) for the KOMA-script class, as well as the on-and-off of indents.

2.3.3 Lengths and Sizes

Length Units Before adjusting the extent of objects and spacings, we need to be able to express and measure lengths in LaTeX. There are various *length units* for this, summarized in Table 2.4 below.

Length Values, setlength Subsequently, the *lengths* of different markers are stored as parameters, listed in Table 2.5 below. By using `\setlength`

[3] It will not work if `parskip` is `half` or `full`.

Unit	Description
pt	The usual "point" unit adopted in other documenting software.
mm/cm/in	A millimeter/A centimeter/An inch.
ex	The height of a lowercase "x" character in the current font. (usually used for vertical distance)
em	The width of an uppercase "M" in the current font. (usually used for horizontal distance)
mu	1/18 of an em with respect to the maths symbols. (usually used in math mode)

Table 2.4: The various length units in LaTeX.

`{<length_param>}{<length_value>}`, we can modify them and adjust distances on the page.

2.3.4 Horizontal and Vertical Spaces

hspace, vspace To adjust the position of different objects or blocks, the primary way is via the `\hspace{<length>}` and `\vspace{<length>}` commands. As their names hint, they add a horizontal/vertical space of fixed lengths. For example, the code

```
\hspace{3ex} Hello \hspace{5ex} World \vspace{1.5em} !!! \\
Ouch...
```

gives

Hello World !!!

Ouch...

The first two `hspace` commands should work as you have expected, but notice that on the other hand, `vspace` in the middle of a line will only take effect after

Parameter	Description
`\baselineskip`	Vertical distance between adjacent lines within a paragraph.
`\columnsep`	Distance between columns.
`\columnwidth`	The width of a column.
`\fboxsep` and `\fboxrule`	The padding and line width around boxes.
`\linewidth`	The width of a line.
`\paperheight` and `\paperwidth`	The height and width of the page.
`\parindent`	The length of the indent before a paragraph.
`\parskip`	The vertical spacing between paragraphs.
`\textheight` and `\textwidth`	The height and width of the text area in a page.
`\topmargin`	The length of the top margin.
`\topsep` and `\itemsep`	The vertical space added above and below an environment, as well as around the items within it.

Table 2.5: Commonly involved length parameters in LaTeX.

it, and so the exclamation marks above are not moved down (but "Ouch..." is). Finally, they accept negative lengths, and you may want to play with that.

It is also to achieve the same effect after a line break by writing something along the lines of `\\[<length>]`, e.g.

```
Don't come any closer!!!\\[-1em]
Nope *Taking out the axe*
```

Don't come any closer!!!
Nope *Taking out the axe*

hspace*, vspace* There also exist starred versions of `\hspace*{<length>}` and `\vspace*{<length>}`. The original ones will be "gobbled up" (see TEX StackExchange 89082) and disappear at line breaks, but the new ones will not. To see this clearly, let's try

```
x\hspace{3ex}y\\
\hspace{4ex}y?\\
\hspace*{4ex}y!
```

which gives

x y
y?
  y!

hfill, vfill, fill, stretch In the case where a fixed distance is only needed in a certain place, while other remaining empty spaces can extend automatically, we can make use of the `\hfill`, `\vfill` commands, or more generally `\fill`, plus `\stretch{<factor>}`. `\hfill` and `\vfill` will take up all the possible spaces after other `hspace` or `vspace` commands are calculated.

If there are multiple `\hfill` or `\vfill`, then the length will be partitioned equally. To assign different weightings to the partition, we can go back and write `\hspace{\stretch{<factor>}}` (similarly for `\vspace`). For example,

```
\hfill Hope \hspace{4cm} Faith \hspace*{\stretch{2}} \\
\hspace*{\stretch{2}} Love \hspace{4cm} Luck \hspace*{\fill} \par % *
    are needed!
```

yields

Hope Faith
Love Luck

Notice how we have to use the starred forms to circumvent the gobbling. (Try not using them and see how it fails!)

2.3.5 Boxes and Rules

mbox, fbox By calling `\mbox{<text>}`, a piece of text may be placed and contained inside a *horizontal box*. This also means that the text will not be disrupted by automatic line breaks or stretched (see TEX StackExchange 475056), and can spill out of the main area into the margin. There is also `\fbox{<text>}` as a wrapped version of `\mbox` with a frame around it, and we will use it for a visualized comparison: The code

```
Preparation is the key to success, but a good plan today is better
   than a perfect plan tomorrow.
\fbox{Preparation is the key to success, but a good plan today is
   better than a perfect plan tomorrow.}
```

produces: Preparation is the key to success, but a good plan today is better than a perfect plan tomorrow. Preparation is the key to success, but a good plan today is better than a pe From this, we can clearly see how the horizontal box extends all the way outside.

makebox, framebox An improved version for the box commands above consists of `\makebox[<width>][<alignment>]{<text>}` and also similarly `\framebox[<width>][<alignment>]{<text>}`, where we can specify the width of the box and how the text inside is justified (`l, c, r, s`: left, center, right, spread) inside the box. For example,

```
\framebox[100pt][c]{I fit inside!} and \\
\framebox[130pt][l]{Unfortunately, this one is too small for me...}
```

generates I fit inside! and
Unfortunately, this one is too small for me...
These box commands can be manipulated to control the distribution of text.

parbox Meanwhile, *vertical boxes* where the text inside can break just like normal can be constructed by the `\parbox[<alignment>]{<width>}{<text>}` command. The effect is not hard to inspect from the input

```
that produces \parbox[b]{100pt}{Empty your mind, be formless,
   shapeless, like water.} ...
```

that produces Empty your mind, be formless, shape- less, like water. ... This time, the alignment option (`t, c, b`: top, center, bottom) decides how the `\parbox` will be positioned relative to the current line. To add a frame around it, simply enclose it with an extra `\fbox`.

raisebox Sometimes we may want to raise or lower a text while pretending it still occupies some space with a fixed size. Then the `\raisebox{<vertical_distance>}[<extend_above>][<extend_below>]{<text>}` command will do the job. This is demonstrated by including a `\fbox` to visualize the effect:

```
\fbox{\raisebox{15pt}[10pt][10pt]{I am a rising star!}} and this is
   my stage!
```

I am a rising star! and this is my stage!

This command can be very useful in achieving several invisible spacing tricks.

Rules Another useful ingredient is the possibility to draw *rules* as lines. The basic command is `\rule{<horizontal_extent>}{vertical_extent}`. For example, `\rule{5ex}{1ex}` generates this: ▬. We also have more primitive versions of `\hrule` and `\vrule`. The code below will yield

```
\vrule \hspace{6pt} If you remove me, the vertical rule to the left
   will disappear! \hrule
```

| If you remove me, the vertical rule to the left will disappear!

Exercise(s)

2.4) Use the `\setlength` command to change different lengths and test what the result would look like, e.g. `\setlength{\parindent}{5cm}`.

2.5) Copy your favorite quote or paragraph to the document, and use the commands/techniques introduced in these two sections to make it beautiful and stylish.

2.4 Verbatim Mode

verb To type short inline code pieces, we can use the *verbatim* mode through the `\verb|<content>|` command. This preserves the input exactly as it is typed, without invoking any would-be LaTeX command or special character. For example, entering `\verb|func|` will output `func` here. However, a major pitfall is that `\verb` can fail when it is used inside the argument of a command. Since we may use the `\include` command to import each chapter separately as suggested by Section 1.2.3, this will be problematic. An alternative is to use `\texttt{<content>}`, with `\textbackslash` as the replacement for `\`, and writing `\_` for `_`, `\{` and `\}` for `{` and `}`.

lstlisting When we need to display larger blocks of code, we can use the `listings` package and its `lstlisting` environment. Actually, it has already been used (shown as yellow areas) in this book many times. A self-explanatory example[4] is

[4]It is a bit involved to make this one work, the option `escapeinside` is intentionally left out below, but you should look it up.

```
\begin{lstlisting} % can pass the overriding option [style=<style_
    name>]
I guess this counts as a recursion...
\end{lstlisting}
```

To design the appearance of the code blocks, we can define our own `lstlisting` style. The one adopted in the book is given by

```
\lstdefinestyle{lstTeXstyle}{ % Give a name for the lstlisting style
    language=[latex]TeX,
    basicstyle=\footnotesize\ttfamily, % The font style
    backgroundcolor=\color{Goldenrod!20},
    keywordstyle=\color{blue!80}\bfseries, % For highlighting
       functions
    commentstyle=\color{Green},
    breaklines=true,
    numbers=none, % none, left, or right
    showstringspaces=false,
    belowskip=0pt}
\lstset{style=lstTeXstyle} % Set the style
```

Most of the options above are not hard to get, but you may want to fiddle with the last four of them.

Exercise(s)

2.6) Take any of the code blocks in this book and reproduce it using the `lstlisting` environment.

Chapter 3

The Fundamentals of Writing Mathematics in LaTeX

Introduction This chapter covers the basic methods about how to typeset and align different mathematical expressions and formulae in LaTeX.

3.1 The Two Math Modes

3.1.1 Inline Math Mode and Basic Math Syntax

Inline Math by $$ To be able to write mathematical expressions in LaTeX, we need to first enter the so-called *math mode*. There are two types of math mode in LaTeX, and the simpler one will be the *inline* math mode. As its name suggests, it renders the mathematical expressions as a usual part of a paragraph. We can enter the inline mode by enclosing an expression with the dollar signs like `$<expression>$`. For example, typing `$3x+4y-z = 5$` here reaily outputs $3x + 4y - z = 5$.

Basic Operators The plus, minus, divide, and equal signs $+$, $-$, $/$, $=$ are just the usual ones and can be typed directly in math mode. Meanwhile, the multiplication sign ($\times$) has to be typed explicitly as `\times`, and we may also use the dot sign ($\cdot$) through `\cdot` instead. By the same logic, round and square brackets in math mode are also simply given by `()`, `[]`.

Superscripts and Subscripts Superscripts (e.g. raising to a power) and subscripts can be added via `^{<superscript>}` and `_{<subscript>}`. For example, `C^n_r` is rendered as C^n_r.

Fractions, smash Fractions can be typed as `\frac{<numerator>}{<denominator>}`, e.g. `\frac{2x^2}{3x+1}` produces $\frac{2x^2}{3x+1}$. However, notice that this `\frac` in the inline mode is shrunk. One workaround is to simply use the slash / instead, but we can also replace `\frac` by `\dfrac`, which gives $\dfrac{2x^2}{3x+1}$. Unfortunately, this leads to another issue where the full-size fraction interferes with the line spacing (the lines directly above and below the `\dfrac` are slightly pushed away if you look closely). A quick fix is to enclose it with the `\smash{}` command to tell LaTeX to ignore its extent.

Common Mathematical Functions, Symbols The commands for some notable, frequently used mathematical functions and symbols are summarized in Table 3.1 below.

Function/Symbol(s)	Command(s)	Description
sin, cos, tan, csc, sec, cot	`\sin()`, `\cos()`, `\tan()`, `\csc()`, `\sec()`, `\cot()`	Trigonometric Functions.
exp, log, ln	`\exp()`, `\log()`, `\ln()`	Exponential and (Natural) Logarithm.
$\sqrt{x}$, $\sqrt[3]{x}$	`\sqrt{x}`, `\sqrt[3]{x}`	Square (Cubic) Root of x.
i, e, π	`i`, `e`, `\pi`	Important constants: The imaginary number, e, and pi.
α, β, γ, ...	`\alpha`, `\beta`, `\gamma`	Greek letters. (see the full list at http://www.phys.uri.edu/~nigh/TeX/sym1.html)
$\pm$, ∞	`\pm`, `\infty`	The plus/minus sign and infinity symbol.
$\sum_i^n$, $\int_a^b$	`\sum_{i}^{n}`, `\int_{a}^{b}`	Summation and integral signs with lower and upper limits.

Table 3.1: Commonly used mathematical commands in LaTeX.

Exercise(s)

3.1) Try to reproduce the following mathematical expressions.

a) $ax^2 + by^2 + c(z-4)^2 = R^2$;

b) $g(x) = \frac{1}{e^{-qx}+1}$;

c) $A_{ij}^2 = A_{ik}A_{kj}$;

d) $\int_0^\infty \frac{\sin(\pi x)}{x} dx = ?$;[1]

e) $\beta \pm \ln\left(\sqrt{\frac{\alpha}{10}}\right)i$.

3.1.2 Display Math Mode

equation The second type of math mode is the *display* math mode, which involves putting the expressions inside an environment on their own. The most frequently used one is the `equation` group, which processes a single line of equation or formula. For instance,

```
\begin{equation}
f(t) = 1 - e^{-at}
\end{equation}
```

results in

$$f(t) = 1 - e^{-at} \tag{3.1}$$

Notice that the `equation` is automatically numbered.

align More often than not, we want to show the detailed steps involved in a calculation. The `align` environment enables us to write them in multiple lines, in addition to providing the `%` character as the anchor for aligning these lines. The `\\` symbol is again used as a line break just like in any ordinary text. As an example,

```
\begin{align}
\frac{d}{dx}(2x+3)^5 ={}& [5(2x+3)^4][\frac{d}{dx}(2x+3)] & & \text{(
   Chain Rule)} \\
={}& [5(2x+3)^4](2) = 10(2x+3)^4
\end{align}
```

will give

$$\begin{aligned} \frac{d}{dx}(2x+3)^5 &= [5(2x+3)^4][\frac{d}{dx}(2x+3)] && \text{(Chain Rule)} && (3.2)\\ &= [5(2x+3)^4](2) = 10(2x+3)^4 && && (3.3) \end{aligned}$$

[1]To the curious readers, the result is $\pi/2$.

There are some points worth mentioning. First, the `align` environment will create a line number for each individual line by default. Second, the two lines above are aligned via the first % character in them, as expected. Third, by adding some extra %, we can append any comment to the right. In fact, odd-numbered % control the exact alignment position and even-numbered % dictate the partition of pieces. Finally, the `{}` after = are needed for appropriate spacing (try removing them!).

split Sometimes, an entire expression is too long to be captured in a single line and may require us to break it into multiple lines, while still treating it as a whole entity. The `split` sub-environment then comes in handy. It works like `align` but can be embedded in another larger `align` group. For example,

```
\begin{align}
(2x+3)^5 ={}& \sum_{k=0}^{5} C^{5}_k (2x)^k(3)^{5-k}\\
\begin{split}
={}& 32x^5 + 240x^4 + 720x^3 \\
& + 1080x^2 + 810x + 243
\end{split}
\end{align}
```

produces

$$(2x+3)^5 = \sum_{k=0}^{5} C_k^5 (2x)^k (3)^{5-k} \tag{3.4}$$

$$\begin{aligned} &= 32x^5 + 240x^4 + 720x^3 \\ &\quad + 1080x^2 + 810x + 243 \end{aligned} \tag{3.5}$$

As you can see, `split` only occupies a single equation number (in the middle) and the % inside it can "communicate" with those outside `split`.

aligned On the contrary, we have the related `aligned`, and the readers can try (strongly recommended as an exercise)

```
\begin{align}
(2x+3)^5 ={}& \sum_{k=0}^{5} C^{5}_k (2x)^k(3)^{5-k}\\
={}&
\begin{aligned}
& 32x^5 + 240x^4 + 720x^3 \\
& + 1080x^2 + 810x + 243
\end{aligned}
\end{align}
```

to see the difference (particularly the %). There is also `multline`, however, most of the usages are already covered by `split` and `aligned`, so we will not discuss it.

Starred Equations Sometimes, the equations may not be worthy of assigning an equation number. By using the starred versions of these environments (`equation*`, `align*`, etc.), the equation numbers will be suppressed. For example,

```
\begin{equation*}
1 + 1 = 2
\end{equation*}
```

yields

$$1 + 1 = 2$$

A quick alternative is to use the `\[<math>\]` shorthand.

nonumber We can also use `\nonumber` to manually prevent numbering for specific lines. For instance,

```
\begin{align}
\int xe^{-x} dx &= -\int x d(e^{-x}) \nonumber \\
&= -[xe^{-x}] + \int e^{-x} dx & & (\text{Integration by Parts}) \
   nonumber \\
&= -xe^{-x} - e^{-x} + C
\end{align}
```

will give

$$\begin{aligned}\int xe^{-x}dx &= -\int xd(e^{-x}) \\ &= -[xe^{-x}] + \int e^{-x}dx \qquad \text{(Integration by Parts)} \\ &= -xe^{-x} - e^{-x} + C \end{aligned} \tag{3.6}$$

Equation Numbers Referencing From time to time, we may need to refer to previous equations during the derivation of a new one. This is straightforward if the equations are numbered, where we can explicitly attach a *label* to the specific lines by `\label{<name>}`. Subsequently, we can call the equation numbers by `\ref{<name>}`. To demonstrate, we may update the integration by parts example in the above paragraph:

```
...
&= -xe^{-x} - e^{-x} + C \label{eqn:IBP1}
```

then `(\ref{eqn:IBP1})` will properly return (3.6).

It is also possible to achieve letter numbering in the `subequations` mode, e.g.

```
\begin{subequations}
\begin{align}
\cos (2x) &= \cos^2 x - \sin^2 x \\
\sin (2x) &= 2 \sin x \cos x
\end{align}
\end{subequations}
```

will generate

$$\cos(2x) = \cos^2 x - \sin^2 x \tag{3.7a}$$

$$\sin(2x) = 2\sin x \cos x \tag{3.7b}$$

allowdisplaybreaks When we are using the `align` environment (or other similar), the blocks may become too lengthy to be included in a single page. By appending the switch `\allowdisplaybreaks` in the preamble, the LaTeX system will then be appointed to break them across multiple pages. This may or may not be desirable and will depend on the situation. As a side note, if an inline expression in a paragraph is too long and hangs outside the main text area, we may add the command `\allowbreak` so that a line break may be inserted there.

Exercise(s)

3.2) Try to reproduce the paragraphs with numbered equations below. Notice that the enlarged brackets can be obtained by `\left(<math>\right)`.

Solving

$$x^2\frac{d^2y}{dx^2} - 3x\frac{dy}{dx} + 3y = 0 \tag{3.8}$$

Let $z = \ln x$, then

$$\frac{dy}{dx} = \frac{dy}{dz}\frac{dz}{dx} = \frac{1}{x}\frac{dy}{dz} \tag{3.9}$$

$$\begin{aligned}\frac{d^2y}{dx^2} &= \frac{d}{dx}\left(\frac{dy}{dx}\right) = \frac{d}{dx}\left(\frac{1}{x}\frac{dy}{dz}\right) \qquad \text{(continuing from (3.9))}\\ &= \frac{1}{x}\frac{d}{dx}\left(\frac{dy}{dz}\right) - \frac{1}{x^2}\frac{dy}{dz}\\ &= \frac{1}{x}\frac{dz}{dx}\frac{d}{dz}\left(\frac{dy}{dz}\right) - \frac{1}{x^2}\frac{dy}{dz}\\ &= \frac{1}{x^2}\frac{d^2y}{dz^2} - \frac{1}{x^2}\frac{dy}{dz}\end{aligned} \tag{3.10}$$

Substituting (3.9) and (3.10) into (3.8), we have ...

3.2 Advanced Mathematical Expressions and Notations

amsmath, amssymb, mathtools Before getting into the main section, it is necessary to load the prerequisite `amsmath`, `amssymb`, and `mathtools` packages for the symbols, as well as enhancing the mathematical typesetting.

3.2.1 Calculus

Differentiation and Integral Symbols Table 3.2 below is a list of notable symbols used to denote derivatives and integrals for calculus, aside from Table 3.1.

Function/Symbol(s)	Command(s)	Description
∂, ∂_y	`\partial`, `\partial_y`	Partial derivatives (with respect to y).
∇, Δ	`\nabla`, `\Delta`	The del and Laplacian operators.
$\lim_{x\to 0}$, $\liminf$, $\limsup$	`\lim_{x\to 0}`, `\liminf`, `\limsup`	Limit (inferior and superior).
$\iint_S$, $\iiint$, $\oint$	`\iint_S`, `\iiint`, `\oint`	Double, triple[2], and contour integrals.

Table 3.2: Commonly used differentiation and integral symbols.

[2]If the limits of the multiple integral have to be spelled out explicitly, then just resort to using the original `\int_{}^{}` for multiple times.

3.2.2 Logic and Description

Logic and Set Symbols Meanwhile, Table 3.3 below contains a number of commonly used logical operators and set symbols.

Function/Symbol(s)	Command(s)	Description
$<$, $>$, $\leq$, $\geq$, $\ll$, $\gg$	`<`, `>`, `\leq`, `\geq`, `\ll`, `\gg`	(Much) Smaller and greater than (or equal to).
$\neq$	`\neq`	Not equal to.
$\equiv$, $\coloneqq$	`\equiv`, `\coloneqq`	Equivalence, Definition of a quantity.
$\approx$, $\sim$	`\approx`, `\sim`	Approximately equal to, similar to.
min, max	`\min`, `\max`	Minimum and Maximum.
$\forall$, $\exists$, $\nexists$	`\forall`, `\exists`, `\nexists`	For all, (not) exists.
$\in$, $\notin$	`\in`, `\notin`	In/not in a set.
$\subset$, $\subseteq$	`\subset`, `\subseteq`	Being a subset of (or equal to) another set.
$\bigcup_{i=1}^{n}$, $\bigcap_{i=1}^{n}$	`\cup^{n}_{i=1}`, `\cap^{n}_{i=1}`	Union and Intersection.
$\emptyset$	`\emptyset`	The empty set.
$\perp$	`\perp`	Perpendicular/orthogonal to.
$\binom{n}{k}$	`\binom{n}{k}`	The binomial coefficient.
$\ldots$, $\cdots$, $\ddots$, $\vdots$	`\ldots`, `\cdots`, `\ddots`, `\vdots`	Various ellipses.

Table 3.3: Some important logical and set symbols.

Arrows and Braces The subsequent Table 3.4 shows different methods of making *arrows* and *braces*, possibly with text above/below them.

Function/Symbol(s)	Command(s)	Description
$\leftarrow$, $\rightarrow$, $\leftrightarrow$	`\leftarrow`, `\rightarrow`, `\leftrightarrow`	Single arrows.
$\Leftarrow$, $\Rightarrow$, $\Leftrightarrow$	`\Leftarrow`, `\Rightarrow`, `\Leftrightarrow`	Double arrows.
$\xleftarrow[l]{u}$, $\xrightarrow[l]{u}$, $\xleftrightarrow[l]{u}$	`\xleftarrow[l]{u}`, `\xrightarrow[l]{u}`, `\xleftrightarrow[l]{u}`	Single arrows with labels.
$\xLeftarrow[l]{u}$, $\xRightarrow[l]{u}$, $\xLeftrightarrow[l]{u}$	`\xLeftarrow[l]{u}`, `\xRightarrow[l]{u}`, `\xLeftrightarrow[l]{u}`	Double arrows with labels.
$\overleftarrow{xyz}$, $\overrightarrow{xyz}$	`\overleftarrow{xyz}`, `\overrightarrow{xyz}`	Over-arrows.
$\overline{xyz}$, $\underline{xyz}$	`\overline{xyz}`, `\underline{xyz}`	Overline and Underline.
$\overbrace{xyz}^{abc}$, $\underbrace{xyz}_{abc}$	`\overbrace{xyz}^{abc}`, `\underbrace{xyz}_{abc}`	Overbrace and underbrace with labels.

Table 3.4: Arrows and braces in LaTeX.

Delimiters Simple *delimiters* can be typed directly in math mode (except the curly brackets {} that require `\{\}`), like

```
\begin{align*}
&\frac{1}{N}(1+\frac{n}{N}) & & [\ln|x|]_a^b & & \{x|f(x) \neq 0\}
\end{align*}
```

outputs

$$\frac{1}{N}(1+\frac{n}{N}) \qquad [\ln|x|]_a^b \qquad \{x|f(x) \neq 0\}$$

However, if the content inside the delimiters is too tall, then we can append `\left`

and `\right` before the delimiters on both sides to match the height. Note that they must be balanced. For example,

```
\begin{equation*}
\left[p + a\left(\frac{n}{V}\right)^2\right](V-nb) = nRT
\end{equation*}
```

is rendered as

$$\left[p + a\left(\frac{n}{V}\right)^2\right](V - nb) = nRT$$

Exercise(s)

3.3) Try to type the following statements.

a) $\oint Mdx + Ndy = \iint (\frac{\partial N}{\partial x} - \frac{\partial M}{\partial y})dxdy$;

b) $\mu \ll \nu \implies \mu(A) = 0, \forall A | \nu(A) = 0$;

c) $A \subseteq B \cup (A \cap B^c)$.

3.2.3 Vectors and Matrices

Denoting Vectors The most essential object in linear algebra is undoubtedly *vectors*, and we need a standard way to denote and distinguish them. One possible solution is to use an overhead arrow: the command `\vec{v}` will output $\vec{v}$. For longer objects, we can instead use `\overrightarrow` introduced in the last subsection. Another approach is to use boldface, which can be applied to general mathematical symbols if we load the `bm` package: `\bm{v}` will then produce $\boldsymbol{v}$.

Unit Vectors For unit vectors, we often use the hat symbol to denote them, e.g. `\hat{v}` gives $\hat{v}$. Particularly, for the three-dimensional standard unit vectors $\hat{\imath}$,

$\hat{\jmath}$, $\hat{k}$, we use `\hat{\imath}`, `\hat{\jmath}`, and `\hat{k}` where we use the versions `\imath`: $\imath$, `\jmath`: $\jmath$ for the first two without the usual dot at the top for placing the hat.

Matrices and Determinants: bmatrix, vmatrix Another class of objects closely related to vectors is *matrices*. To typeset a matrix in LaTeX, we use the `bmatrix` environment provided by the `amsmath` package. For example,

```
\begin{align*}
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6
\end{bmatrix}
\end{align*}
```

outputs

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

where `%` separates the entries into columns and `\\` marks the end of a row. By replacing `bmatrix` by `matrix`, `pmatrix`, or `Bmatrix`, the delimiters become nil, round, and curly brackets correspondingly. Particularly, we have the `vmatrix` group to represent determinants. For instance, writing

```
\begin{align*}
\det(A) =
\begin{vmatrix}
a & b \\
c & d
\end{vmatrix} = ad-bc
\end{align*}
```

leads to

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

As a supplement, we can manipulate the ellipses symbols to denote a matrix of an arbitrary shape. The following

```
\begin{align*}
A_{m \times n} =
\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\
a_{21} & a_{22} & a_{23} &        & a_{2n} \\
a_{31} & a_{32} & a_{33} &        & a_{3n} \\
\vdots &        &        & \ddots & \vdots \\
a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn}
\end{bmatrix}
\end{align*}
```

produces

$$A_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & & a_{2n} \\ a_{31} & a_{32} & a_{33} & & a_{3n} \\ \vdots & & & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

A column vector can be represented by a matrix consisting of a single column.

array For an advanced control of matrices, we can use the `array` environment instead. Let's first see how the code will look in the scenario of Gaussian Elimination, and then break down the details. Given

```
\begin{align*}
\left[\begin{array}{@{}wc{10pt}wc{10pt}wc{10pt}|r}
1 & 2 & 1 & -1 \\
2 & 5 & 3 & 2 \\
0 & 1 & 1 & 0
\end{array}\right]
& \to
\left[\begin{array}{@{}wc{10pt}wc{10pt}wc{10pt}|r}
```

```
1 & 2 & 1 & 1 \\
0 & 1 & 1 & 4 \\
0 & 1 & 1 & 0
\end{array}\right]
& & R_2 - 2R_1 \to R_2
\end{align*}
```

the output will be

$$\left[\begin{array}{ccc|c} 1 & 2 & 1 & -1 \\ 2 & 5 & 3 & 2 \\ 0 & 1 & 1 & 0 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & 1 & 4 \\ 0 & 1 & 1 & 0 \end{array}\right] \qquad R_2 - 2R_1 \to R_2$$

The `array` group typesets each entry just like the `matrix` one. However, notice the input string `{@{}wc{10pt}wc{10pt}wc{10pt}|r}` before the main content. `@{}` replaces the default left padding with an empty space. `wc` indicates the entries along that column to take a fixed width (`w`) of 10 pt and are centered (`c`). This is repeated for the first three columns to the left. A bar `|` then generates a vertical separating line at the desired location. (For a horizontal line, put `\hline` between the rows inside.) Finally, `r` makes the entries right-aligned (similarly with `l`) in the last column with a varying width, and we surround the `array` environment with tall delimiters (see last subsection) manually.

3.2.4 Other Formatting Trivia

abs, norm from physics The `physics` package provides many symbols well-known in the area of physics. Particularly, it defines `\abs{<expression>}` and `\norm{<expression>}` commands for absolute value and norm (length/magnitude), which are quite convenient even for other usages. For example,

```
$\norm{\bm{x}}_1 = \sum_{i=1}^n\abs{x_i}$
```

gives $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$.

mathbb, mathcal Two other types of symbols that may be of interest come from `\mathbb{<character>}` and `\mathcal{<character>}` for sets and classes. For example, the set of all real numbers is commonly denoted by $\mathbb{R}$ (`\mathbb{R}`), while the class of continuously differentiable functions is denoted by $\mathcal{C}^1$ (`\mathcal{C}^1`).

siunitx For other science applications, the physical quantities involved are often accompanied by units. The `siunitx` helps facilitate the typesetting of units and expressing the exponents. For example, `\si{\N} = \si{\kg \per \m \per \square \s}` is interpreted as $\mathrm{N} = \mathrm{kg\,m^{-1}\,s^{-2}}$, while `\SI{4.184e3}{\J \per \kg \per \K}` generates $4.184 \times 10^3\,\mathrm{J\,kg^{-1}\,K^{-1}}$.

System of Equations To typeset a system of equations, we can use either `aligned` with a large curly bracket to the left, or the `cases` environment. There will be slight differences between these two methods. For instance,

```
\begin{align*}
\left\{\begin{aligned}
3x + 4y + 5z &= 6 \\
x - 2y + 3z &= -4 \\
x^2 + y^2 &= 1
\end{aligned}\right.
\end{align*}
```

will produce (notice that we need `\right.` at the end to make a placeholder delimiter to the right for balance)

$$\begin{cases} 3x + 4y + 5z = 6 \\ x - 2y + 3z = -4 \\ x^2 + y^2 = 1 \end{cases}$$

Alternatively, we can write

```
\begin{align*}
\begin{cases}
3x + 4y + 5z &= 6 \\
x - 2y + 3z &= -4 \\
x^2 + y^2 &= 1
\end{cases}
\end{align*}
```

to achieve

$$\begin{cases} 3x + 4y + 5z &= 6 \\ x - 2y + 3z &= -4 \\ x^2 + y^2 &= 1 \end{cases}$$

As its name suggests, `cases` is actually designed to represent the values of a variable in different cases, e.g. we may write

```
\begin{align*}
\begin{aligned}
y(x) =
\begin{cases}
1 & x \in \mathbb{Q} \\
0 & x \notin \mathbb{Q}
\end{cases}
\end{aligned}
\end{align*}
```

to get

$$y(x) = \begin{cases} 1 & x \in \mathbb{Q} \\ 0 & x \notin \mathbb{Q} \end{cases}$$

Spacing in Math Mode In math mode, we often employ pre-defined commands instead of `\hspace` or `\vspace` to adjust the spacing. They are shown in Table 3.5 below.

Command	Description	Effect
`\quad`	Space of 1 em in the current maths font size (= 18 mu)	a b
`\qquad`	Double of `\quad` (= 36 mu)	a b
`\,`	3/18 of `\quad`/3 mu	a b
`\:`	4/18 of `\quad`/4 mu	a b
`\;`	5/18 of `\quad`/5 mu	a b
`\!`	−3/18 of `\quad`/−3 mu	ab
`\(space)`	Space as in normal text	a b

Table 3.5: Spacing commands in math mode.

Sizes We can control the font size in either math mode with the usual size commands in Table 2.2. For the inline mode, we can write something like

```
{\Large$N(0,1) \sim e^{-x^2/2}$}
```

to get $N(0,1) \sim e^{-x^2/2}$. On the other hand, for the display mode, we may put the size command before the math environment, e.g.

```
\footnotesize
\begin{align*}
\mathcal{L}[y^{(n)}](s) &= s^n Y(s) - s^{n-1}y(0) - s^{n-2}y'(0) - \
   cdots - y^{(n-1)}(0) \\
&= s^n Y(s) - \sum_{k=0}^{n-1} s^{(n-1)-k}y^{(k)}(0)
\end{align*}
\normalsize % Back to default font size
```

will yield

$$\begin{aligned}\mathcal{L}[y^{(n)}](s) &= s^nY(s) - s^{n-1}y(0) - s^{n-2}y'(0) - \cdots - y^{(n-1)}(0) \\ &= s^nY(s) - \sum_{k=0}^{n-1} s^{(n-1)-k}y^{(k)}(0)\end{aligned}$$

mathcolor To apply colors in math mode, we can replace the `\textcolor` command with `\mathcolor`. For example,

```
\begin{align*}
\mathcolor{Blue}{\frac{\partial \vec{u}}{\partial t}} + \mathcolor{
   Green}{\vec{u}\cdot\nabla\vec{u}} = \mathcolor{Red}{\vec{F}}
\end{align*}
```

is displayed as

$$\mathcolor{Blue}{\frac{\partial \vec{u}}{\partial t}} + \mathcolor{Green}{\vec{u}\cdot\nabla\vec{u}} = \mathcolor{Red}{\vec{F}}$$

Exercise(s)

3.4) Reproduce the following output.

$$\begin{cases} x + 2y = 3 \\ x - 3y = -2 \\ -x + y = 1 \end{cases} \quad \Leftrightarrow \quad \begin{bmatrix} 1 & 2 \\ 1 & -3 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix}$$

Chapter
4

Various Special Structures in LaTeX

Chapter
5

Self-defined Commands and Environments

Chapter 6

More on Book Layout Design

Chapter
7

Framed Theorems and Exercises

Chapter
8

Plotting with Tikz

Chapter
9

Miscellaneous