How to Reproduce this Book Exactly with LaTeX

A Self-contained Tutorial on Writing Mathematical Notes

v1.0.0

C. L. Loi

A student from
CUHK-EESC/NTU-AS

Preface

The idea of this book was born soon after I completed my first book on Linear Algebra, of course, written using LaTeX. I want to happily share my experience about working with LaTeX during this long period and help other fellow Mathematics lovers to make their own books. In this process, I have gotten to be more familiar with LaTeX and appreciate both the technical and aesthetic aspects of writing a book. This book aims to provide a comprehensive tutorial for beginners that covers all the key parts of creating a good-looking book using LaTeX. The necessary code to exactly produce the outputs is displayed (and hosted on the GitHub repository as well), and there are also a few practice exercises. As the name of this book suggests, readers are encouraged to reproduce the book (no need to really be an exact replica, though; maybe instead with your own variations!) as they progress. This book can also serve as a quick cookbook, and readers can reference any part of it as they see fit. However, notice that this introductory book is by no means exhaustive and cannot substitute for full user manuals. Again, I will be very glad if my work can inspire you to undertake the writing journey of Mathematics.

Sincerely,
Benjamin C. L. Loi

(Github Repository: https://github.com/BenjaminGor/Latex_Notes_Tutorial)

Contents

Chapter 0

Installing or Preparing LaTeX

Introduction The foremost thing we have to do is obviously preparing a LaTeX environment. Here we will introduce two most common approaches that people use: an online editor versus a local installation.

0.1 Online Editor

Overleaf **Overleaf** (https://www.overleaf.com) is a popular online LaTeX editor that is quite simple for beginners to pick up. It also provides a very complete documentation, so we will not go into the details. There are mainly three advantages of using Overleaf. First, it comes with most, if not all, of the common packages, hence there is no need to do extra work. Second, it allows multiple collaborators to view and edit the same project. Finally, the project can be easily linked to a **GitHub** repository for open-access and version control (which this book has been using!).

0.2 Local Installation

MiKTeX/MacTeX On the other hand, local LaTeX distribution is usually provided by **MiKTeX** on Windows (https://miktex.org) and **MacTeX** on Mac (https://www.tug.org/mactex). Particularly for MiKTeX, packages can be easily installed using its console, and there is also an option to download the missing ones when they are needed.

Choice of Editor The traditional local editor for LaTeX is **TeXstudio** (https://www.texstudio.org) that is also not difficult to use. Some users may be more comfortable with **Visual Studio Code**, which is possible with the LaTeX Workshop extension.

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 structure of the content 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*, everything before `\begin{document}`) of the main TeX file. Although not explored in this book, some other notable classes that may be of use include `beamer`, `moderncv`, `article` (or `scrartcl`), and `scrreprt`.

Commands and Options The `scrbook` class provides several *options* to customize the format of the book. We can either supply them in the argument when declaring the class, or use the command `\KOMAoptions` in the preamble. Meanwhile, 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 before their names. In this book, we have used

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

The argument is filled inside the curly brackets `{}` following the command. Clearly, here 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 later. The first package we usually need is the `fontenc` package with the `T1` option, set inside a pair of square brackets. This syntax of square brackets will be the same for configuring options in other commands.

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 achieve the same class setting through the `\documentclass[<options>]{scrbook}` declaration instead of the `\KOMAoptions` command.

1.2 Structure Hierarchy

1.2.1 Chapters and (Sub-)Sections

Chapters, Sections In most of the books, the entire content is divided into *chapters*, which in turn usually consist of several *sections*. To mark the beginning of a chapter or section in LaTeX, we place the commands `\chapter{<chapter_name>}` or `\section{<section_name>}` within the `document` scope, which contains the main content and is marked by a pair of `\begin{document}` and `\end{document}` declarations. As mentioned in the beginning, the preamble has to be inserted before such a `document` scope. 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 `%` symbol indicates a trailing *comment* (highlighted in green) whose purpose is to leave some note about the code. Comments are neither interpreted nor displayed. The `\textit{<text>}` command presents the text in italic shape.

Chapter Numbering The LaTeX system records and updates the numbering for chapters/sections internally every time they are initialized. However, we can manually override that by invoking the command `\setcounter{chapter}{<number`

`>}`. To achieve the effect of Chapter 0 at the beginning, we use `\setcounter{chapter}{-1}` so that the counter increases by 1 to 0 as intended when Chapter 0 is generated. The similar can be done for sections.

Subsections, Paragraphs Attentive readers may have already figured out that it is possible to stack an extra level/depth (a *subsection*) in the content 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 increment the numbering/counters.

Referencing Chapters/Sections We can create references to chapters/sections by appending the `\label{<label_name>}` after their declarations, and then use them via `\ref{<label_name>}`. For example, for Chapter 0, we write

```
\chapter{Installation of \LaTeX{} Distribution}
\label{chap:install}
...
```

and we can call it with `\ref{chap:install}`. The `chap:` prefix is customary for convenience.

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` environment. 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 also 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 sorted 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 again>
\begin{document}
...
```

```
\include{ch0_install}
\include{ch1_basic_structure}
\include{ch2_text_format}
...
\end{document}
```

1.3 Testing the Book Layout by Lipsum

Dummy Text Sometimes we may need to insert some placeholder text into the document 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 locations. 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 describes how to adjust the various aspects of text, such as fonts, size/shapes/styles, and positioning.

2.1 About Fonts

2.1.1 The Three Font Families

(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 by default, 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

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

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 what they look like.*

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.

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 font, we can import the corresponding font package. The **LaTeX Font Catalogue** https://tug.org/FontCatalogue/ provides a comprehensive list of available fonts and the way to load 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
\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 commands within their scope.

Table 2.2: *The various commands for controlling font size in text.*[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?

Fixing Font Size 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

```
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 for update the
   parameters
```

leads to

[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.*

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 shapes/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 \par % 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` (stands for emphasis). You may want to try them out.

2.2.3 Text Color

The Package for More Colors While there are built-in colors in the LaTeX system, we can load a variety of additional colors from the `xcolor` package, often with the `svgnames` and `dvipsnames` 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.} % again, remember to
    limit the scope by 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
...
\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/line spacing* for `\par` (`\\`). There is also `\newline` which is similar and sometimes used.

A blank line in a TEX file has the same effect as `\par`. They, in fact, end the so-called *horizontal mode* and distribute the text into lines held in 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 similar and
    sometimes used.
                % here is a blank line with this comment only
A blank line in a \TeX{} file ... held in 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 % see Section 2.4
... % this code block
```

2.3.2 Justification and Indents

Ragged and Centering The `\raggedright` and `\raggedleft` commands produce *left/right-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 raggedright} and \texttt{\
   textbackslash raggedleft} commands produce ... 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}
```

Flushing (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 itself. 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}
```

`flushleft` (`flushright`) corresponds to (`\raggedright`) `\raggedleft`. However, if you test this new code, notice the increased separation[2] around the

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

environments.

Indents, Paragraph Skip 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` key 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.

Improved Typesetting Finally, for a better typesetting behavior (e.g. hyphenation), it is recommended always to import the `microtype` package, which provides helpful patches on this.

Exercise(s)

2.3) Try experimenting 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 learning how to adjust the size 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.

[3] It will not work if `parskip` is `half` or `full` as they take the priority.

Unit	Description
pt	The usual "point" unit adopted in computer documents.
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 math symbols. (usually used in math mode)

Table 2.4: *The various length units in LaTeX.*

Length Values, Setting Lengths Subsequently, the *lengths* of different markers are stored as parameters, listed in Table 2.5. By using `\setlength{<length_param>}{<length_value>}`, we can modify them and adjust distances on the page.

2.3.4 Horizontal and Vertical Spaces

Horizontal/Vertical Spaces To control 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 some fixed length. For example, the code

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

gives

Hello World !!!

Ouch...

Parameter	Description
`\baselineskip`	The vertical distance between adjacent lines within a paragraph.
`\columnsep`	The 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 (or in the current environment).
`\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.*

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 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 possible 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*

Starred Spaces There also exist starred versions of `\hspace*{<length>}` and `\vspace*{<length>}`. The difference is that 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!

Fill and 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 %
   the asterisks * 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!)

Skips Finally, there are also shorthands for generating vertical line skips: `\smallskip`, `\medskip`, and `\bigskip`. Note that they are just `\vspace` with `\smallskipamount`, `\medskipamount`, and `\bigskipamount` under the hood.

2.3.5 Boxes and Rules

Basic Boxes 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.

Advanced Boxes 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.

Paragraph Boxes Meanwhile, *vertical boxes* where the text inside can break just like normally 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.} ...
```

Empty your mind,
be formless, shape-
that produces 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`.

Raising Boxes Sometimes we may want to raise or lower a text while pretending it still occupies some space with a fixed extent. 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.

(Phantom)

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

Verbatim 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.[4] 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 `}`.

[4]Interested readers can search fragile commands.

Code Snippets 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[5] is

```
\begin{lstlisting} % can pass the option [style=<style_name>] to
   override
\textit{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 understand, but you may want to fiddle with the last four of them.

Like `\verb`, this package also comes up with `\lstinline` for writing inline code.

[5] It is a bit involved to make this one work, the option `escapeinside` (as well as its variants) is intentionally left out below, but you should look it up.

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 regarding how to typeset and align different mathematical expressions and formulas in LaTeX.

3.1 The Two Math Modes

3.1.1 Inline Math Mode and Basic Math Syntax

Inline Math 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 line. We can enter inline mode by enclosing an expression with the dollar signs like `$<expression>$`. For example, typing `$3x+4y-z = 5$` here readily outputs $3x + 4y - z = 5$.

Basic Operators The plus, minus, divide, and equal signs $+$, $-$, $/$, $=$ are just the usual ones on the keyboard 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. 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^{2n+1}_n` is output as C_n^{2n+1}.

Fractions, Smash Fractions can be typeset easily as `\frac{<numerator>}{<denominator>}`, e.g. `\frac{2x^2}{3x+1}` produces $\frac{2x^2}{3x+1}$. However, notice that this is shrunk in inline mode. 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 Environment The second type of math mode is the *display* math mode, which involves putting the expressions inside a separate environment on their own. The most frequently used one is the `equation` environment, 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 Environment 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

$$\frac{d}{dx}(2x+3)^5 = [5(2x+3)^4][\frac{d}{dx}(2x+3)] \qquad \text{(Chain Rule)} \tag{3.2}$$

$$= [5(2x+3)^4](2) = 10(2x+3)^4 \tag{3.3}$$

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

There are some points worth mentioning. First, the `align` environment will create an equation 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 more pieces to the right. In fact, odd-numbered `&` control the exact alignment positions and even-numbered `&` decide how the pieces are partitioned. Finally, the `{}` after `=` are needed for appropriate spacing (try removing them!).

Split Scope 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` scope then comes in handy. It works like `align` but can be embedded in another larger `align` environment. 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 Scope On the contrary, we have the related `aligned`, and the readers can try the following (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 how the `&` are placed). 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 number(s) 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.

Suppressing Individual Equation Number We can also use `\nonumber` to prevent numbering for any line manually. 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 or formulas during the derivation of a new one. This is straightforward if the equations are numbered, where we can explicitly attach a tag to the specific lines by `\label{<label_name>}`. Subsequently, we can call the equation numbers by `\ref{<label_name>}`. This is very much the same as the previous chapter/section referencing. 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

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

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

Display Breaking When we are using the `align` environment (or other similar ones), 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 allowed 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 below with numbered equations. 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 ...

Display Style in Inline Mode Symbols like the summation sign `\sum_{i}^{n}` will appear in inline mode as $\sum_i^n$ where the lower/upper limits are squashed. To make them look normally as they are in math mode, we can attach the `\displaystyle` command inside. For example,

```
$\displaystyle \sum_i^n$
```

produces $\displaystyle \sum_i^n$.

3.2 Advanced Mathematical Expressions and Notations

Helper Packages for Mathematics 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 effects.

3.2.1 Calculus

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

Function/Symbol(s)	Command(s)	Description
∂, ∂_x	`\partial`, `\partial_x`	Partial derivatives (with respect to x).
∇, 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.*

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`, `\coloneq`	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`	(Not) In a set.
$\subset$, $\subseteq$	`\subset`, `\subseteq`	Being a subset of (or equal to) another set.
$\cup_{i=1}^{n}$, $\cap_{i=1}^{n}$	`\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.*

[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.

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.
$\Leftarrow$, $\Rightarrow$ ($\implies$), $\Leftrightarrow$	`\Leftarrow`, `\Rightarrow` (`\implies`), `\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 its 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) $\mu\ll\nu\Leftrightarrow\mu(A)=0,\forall A|\nu(A)=0$;

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

Matrices and Determinants 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 & \sqrt{5} & c
\end{bmatrix}
\end{align*}
```

outputs

$$\begin{bmatrix} 1 & 2 & 3 \\ -4 & \sqrt{5} & c \end{bmatrix}$$

where `&` separates the entries into columns and `\\` marks the end of a row. By replacing `bmatrix` by `matrix`, `pmatrix`, or `Bmatrix`, the delimiters become none, round, and curly brackets correspondingly. Particularly, we have the `vmatrix` (vertical lines) group to represent determinants. For instance, writing

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

leads to

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

On top of that, we can manipulate the ellipses symbols to denote a matrix of an arbitrarily large shape. The following

```
\begin{equation*}
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{equation*}
```

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 then be represented by a matrix consisting of a single column.

Arrays 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 places each entry just like a `matrix`. However, notice the input string `{@{}wc{10pt}wc{10pt}wc{10pt}|r}` before the main content. `@{}` empties the default left spacing. `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 there is `l`) in the last column with a flexible width, and we surround the `array` environment with tall delimiters (see last subsection) manually.

3.2.4 Other Formatting Trivia

(cancel?)

Absolute Value and Norm The `physics` package provides many symbols well-known in the area of physics. Particularly, it defines the `\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 $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$.

Special Math Characters 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`).

SI Units For other science applications, the physical quantities involved are often accompanied by units. The `siunitx` package 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{equation*}
\left\{\begin{aligned}
3x + 4y + 5z &= 6 \\
x - 2y + 3z &= -4 \\
x^2 + y^2 &= 1
\end{aligned}\right.
\end{equation*}
```

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

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 denote the values a variable may take in different cases, e.g. we can write

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

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 math font (= 18 mu).	$a \quad b$
`\qquad`	Double of `\quad` (= 36 mu).	$a \qquad b$
`\enskip`	Half of `\quad` (= 9 mu).	$a \enskip b$
`\,`	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.	$a \! b$
`\(space)`	Space as in normal text.	a b
`~`	Non-breaking space.	a~b

Table 3.5: *Spacing commands in math mode.*

Dashes Related, we may also want to type hyphens or dashes in some situations. The `-` can be used without problems to output -, but we may also want to produce longer dashes. This can be easily done by typing two or three consecutive `-`, leading to an en-dash (–) and em-dash (—) respectively.

Sizes We can control the font size in either math mode with the usual size commands in Table 2.2. For 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 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

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

Coloring for Math 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

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

Exercise(s)

3.4) Reproduce the following output.

$$\begin{cases} x + 2y = 3 \\ x - 3y = -2 \\ -x + y = 1 \end{cases} \quad \Leftrightarrow \quad \begin{bmatrix} 1 & 2 \\ 1 & -3 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix} \tag{3.11}$$

Chapter 4

Various Special Structures in LaTeX

Introduction This chapter presents different special environments in LaTeX apart from the display math mode and verbatim form previously, such as lists, figures, tables, and minipages.

4.1 Lists

4.1.1 Unordered Lists

Unordered Lists The ability to organize things into a *list* is essential in any documenting system. In LaTeX, we can achieve this by using the `itemize` environment with the `\item` command. For example, if we write

```
\begin{itemize}
\item Canada
\item Japan
    \begin{itemize}
    \item Tokyo
    \item Kyoto
    \end{itemize}
```

```
\item Korea
    \begin{itemize}
    \item Seoul
    \item Pusan
    \end{itemize}
\end{itemize}
```

then it will show up as

- Canada
- Japan
  - Tokyo
  - Kyoto
- Korea
  - Seoul
  - Pusan

Notice that the items are *unordered/bulleted* and the list can be nested.

4.1.2 Ordered Lists

Numbered Lists Similarly, we can have an *ordered/numbered* list by using the `enumerate` environment. For example, by typing

```
\begin{enumerate}
    \item A robot may not injure a human being ...
    \item A robot must obey the orders given it by human ...
    \item A robot must protect its own existence ...
\end{enumerate}
```

we acquire the following output:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

It is also possible to have nested `enumerate` groups.

Customizing Labels for Lists The `enumitem` package enhances the typesetting of lists. One of its prime utilities is to change the starting labels or bullets for every item by providing the `label` option. For example,

```
\begin{enumerate}[label=\alph*)]
    \item Apple
    \item Banana
    \item Grape
\end{enumerate}
```

generates

a) Apple
b) Banana
c) Grape

There are other possible choices for `label`, such as `\arabic*`, `\roman*`, `\Roman*`, `\Alph*`. Another usage of the `enumitem` package is to make a continued list. For example, by ticking the `resume*` option:

```
\begin{enumerate}[resume*]
    \item Watermelon
    \item Orange
\end{enumerate}
```

we have (notice their alphabetical labels)

d) Watermelon

e) Orange

4.2 Figures and Tables

4.2.1 Figures

Figures and Including Images To import images into the document, we need to load the `graphics` package and then use the `\includegraphics{<file_name>}` command. The image should be placed under the project directory, and we hereby go through an example which is displayed as Figure 4.1 on the next page. The code to produce that figure is

```
\begin{figure}[ht!]
\centering
\includegraphics[width=0.4\linewidth]{graphics/Ada_Lovelace_portrait.
    jpg} % replace the path with your own file
\caption{The portrait of Ada Lovelace.}
\label{fig:ada}
\end{figure}
```

First, we need to enclose the `\includegraphics` command within a `figure` environment. In fact, `figure` is a type of *floats* which can be freely moved around. The `ht!` option indicates that priority is given to put the figure exactly in the place where the code is inserted (`h`: here), or at the top of a page (`t`). The `width` option enforces the width of the image to some input value (and similarly, there are `height` and `scale`). The `\caption` command unsurprisingly generates the caption, while the `label` command works as it is in math mode and allows us to reference it by writing `\ref{fig:ada}`.

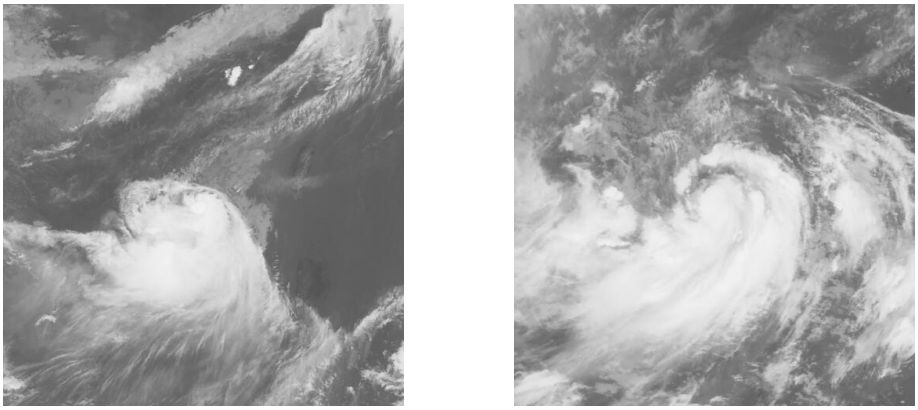
Figure 4.1: *The portrait of Ada Lovelace.*

Subfigures We can create a set of smaller subfigures within a larger figure by utilizing the `subcaption` package and `subfigure` scopes. To illustrate, the following code is deployed to generate Figure 4.2:

```
\begin{figure}[ht!]
\centering
    \begin{subfigure}[b]{0.45\textwidth}
    \centering
    \includegraphics[width=0.8\linewidth]{graphics/MTS
       108082203.200812.jpg}
    \caption{Typhoon Nuri (2008).}
    \end{subfigure}
    \begin{subfigure}[b]{0.45\textwidth}
    \centering
    \includegraphics[width=0.8\linewidth]{graphics/MTS
       212072303.201208.jpg}
    \caption{Typhoon Vicente (2012).}
    \end{subfigure}
```

```
\caption{The infrared satellite images of various Tropical Cyclones
   affecting Hong Kong.}
\label{fig:TC1}
\end{figure}
```

The `b` option fixes the vertical alignment of `subfigure` at the bottom.



(a) *Typhoon Nuri (2008).* (b) *Typhoon Vicente (2012).*

Figure 4.2: *The infrared satellite images of various Tropical Cyclones affecting Hong Kong. (Source: Digital Typhoon)*

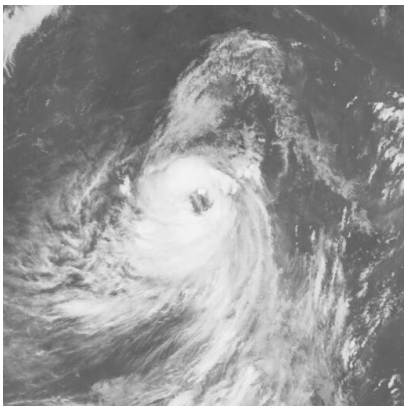
Continued Floats To make an extended figure of subfigures that spans multiple pages, we can simply arrange them into separate `figure` environments and call the `\ContinuedFloat` command in all the subsequent `figure` groups. Continuing from the last example, we may have added

```
\begin{figure}[hb!]
\ContinuedFloat % here!
\caption{(Cont.) The infrared satellite images of various Tropical
   Cyclones affecting Hong Kong.}
\centering
    \begin{subfigure}[b]{0.45\textwidth}
    ...
    \caption{Typhoon Haima (2016).}
```

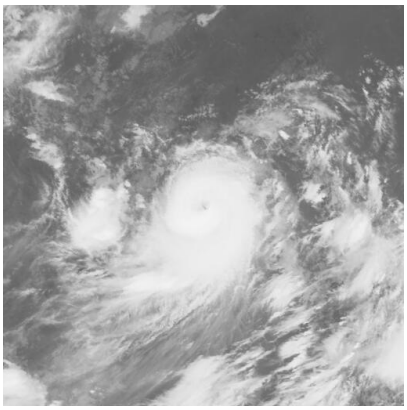
```
    \end{subfigure}
...
    \begin{subfigure}[b]{0.45\textwidth}
    ...
    \caption{Typhoon Saola (2023).}
    \end{subfigure}
\end{figure}
```

producing

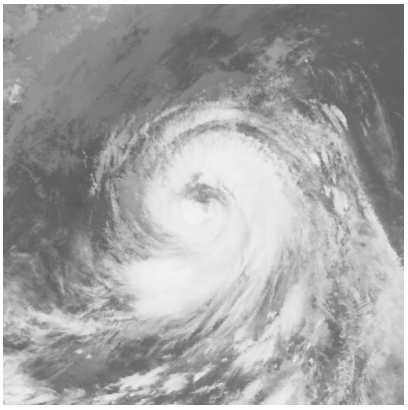
Figure 4.2: *(Cont.) The infrared satellite images of various Tropical Cyclones affecting Hong Kong.*



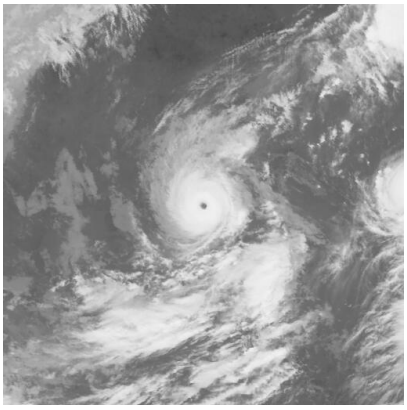
(c) *Typhoon Haima (2016).*



(d) *Typhoon Hato (2017).*



(e) *Typhoon Mangkhut (2018).*



(f) *Typhoon Saola (2023).*

4.2.2 Tables

Tabular Structures Just like embedding figures, building a table requires us to place the content inside the corresponding `table` float environment. While it is possible to use the native `tabular` class for the actual table itself, a more powerful version is provided by the `tabularx` package and its class that bears the same name.[1] This is demonstrated via Table 4.1 thereafter, which is generated by

```
\begin{table}[ht!]
\centering
    \begin{tabularx}{\textwidth}{|l|p{0.55\textwidth}|>{\raggedleft}X
       |>{\raggedleft\arraybackslash}X|}
    \hline
    Unit & Description & Attack & Defense \\
    \hline
    Infantry & The most basic unit and backbone of any army, all-
       around abilities with a cheap cost. & 20 & 25 \\
    \hline
    Cavalry & The shock unit in an army with very strong power. & 40
       & 30 \\
    \hline
    Artillery & The support unit that provides bombardment support
       from far away. & 30 & 5 \\
    \hline
    \end{tabularx}
\caption{The unit statistics table for a hypothetical game.}
\label{tab:armyunits}
\end{table}
```

The `ht!` option, `caption`, and `label` work exactly like the figure counterparts. For the `tabularx` environment, the first argument indicates the width of the entire table, set to `\textwidth` here. The second argument `{|l|p{0.55\textwidth}|`

[1] Sometimes, it can be more advantageous to use `tabular` than `tabularx`, but their syntax are similar.

Unit	Description	Attack	Defense
Infantry	The most basic unit and backbone of any army, all-around abilities with a cheap cost.	20	25
Cavalry	The shock unit in an army with very strong power.	40	30
Artillery	The support unit that provides bombardment support from far away.	30	5

Table 4.1: *The unit statistics table for a hypothetical game.*

`>{\raggedleft}X|>{\raggedleft\arraybackslash}X|}` indicates the justification of the columns: the first column is left-aligned (`l`, similarly we have `c` and `r`) and its size will accommodate the text; the second column (`p`) forces a width of 0.55 times `\textwidth`; the remaining width is distributed evenly to last two `X` columns. The part of `>{\raggedleft}` is applied to the `X` columns, making them right-aligned.[2] Finally, `|` and `\hline` produce vertical/horiztonal separating lines; `&` slices between the columns and `\\` marks the end of a row.

Also, note that `\ContinuedFloat` can also be applied to `table` floats.

Caption by the Side It is also possible to arrange the table so that the caption appears to the side of it. This is done by stacking the `captionbeside` environment provided by KOMA-script. For example, the code

```
\begin{table}[ht]
    \begin{captionbeside}{This caption appears to the left of the
       Fibonacci numbers table.}[l][\textwidth]{
        \adjustbox{valign=t}{
        \begin{tabularx}{0.4\textwidth}{|X|X|}
        \hline
        $n$ & $F_n$ \\
        \hline
```

[2] `\arraybackslash` is needed in the last column, see TEX StackExchange 372464.

```
        $1$ & $1$ \\
        \hline
        $2$ & $1$ \\
        \hline
        $3$ & $2$ \\
        \hline
        $4$ & $3$ \\
        \hline
        $5$ & $5$ \\
        \hline
        $6$ & $8$ \\
        \hline
        \end{tabularx}}
    }
    \end{captionbeside}
\label{tab:fib}
\end{table}
```

produces Table 4.2 below.

Table 4.2: *This caption appears to the left of the Fibonacci numbers table.*

n	F_n
1	1
2	1
3	2
4	3
5	5
6	8

We supply the caption in the first argument, followed by two options: the relative position of the caption and the full width of the structure, finally with the actual `tabularx` object. We also have to load the `adjustbox` package and use the corresponding command to make the table aligned at the top (`valign=t`). This further requires us to first set the `\KOMAoptions` to take `captions=besidetop` (likewise we have `captions=besidebottom` and more).

Shared Numbering between Figures and Tables Sometimes we may want to share the numbering between floats (including figures and tables). This is done by the following patch that can be inserted into the preamble:

```
\makeatletter
\let\c@table\c@figure
\let\ftype@table\ftype@figure
\makeatother
```

This involves the primitive TEX functions, so we will not discuss them there. For more information, read StackOverflow 3865036, and TEX StackExchange 8351 for what the `\makeatletter` and `\makeatother` commands do.

Exercise(s)

4.1) Try to import and load your favorite image into the document.

4.2) Recreate any one of the tables in Chapter 3.

4.3 Minipages and Multiple Columns

Minipages Sometimes we may want to split the content into smaller blocks that are embedded within the current page, and can be placed or ordered (e.g. parallel) in the way we want. The `minipage` environment basically acts like a more versatile version of a `parbox` command and serves this purpose. For example, something like

```
\begin{center}
\begin{minipage}[b]{0.48\textwidth}
\lipsum[6]
\end{minipage}
\hfill
\begin{minipage}[b]{0.48\textwidth}
\lipsum[7]
```

```
\end{minipage}
\end{center}
```

yields

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

The `b` option indicates that the two blocks will be bottom-aligned, provided that their width is fixed to 0.48 times `\textwidth` and thus they fit in the main text area.

Parallel Columns The `parcolumns` package can also achieve an effect similar to `minipage` and is more specialized for typesetting different pieces in two or more parallel columns. It also supports page breaks. Using the same example, we can write

```
\begin{parcolumns}{2}
\colchunk[1]{\lipsum[6]}
\colchunk[2]{\lipsum[7]}
\colplacechunks
\colchunk[1]{\lipsum[8]}
\colchunk[2]{\lipsum[9]}
\end{parcolumns}
```

to get

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. In-

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse

teger vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

where the first argument of `parcolumns` clearly indicates the number of columns and the `\colplacechunks` command releases the loaded `\colchunk[<col_index>]` and goes to the next paragraph.

Multiple Columns A task closely related to what `parcolumns` does above is to typeset a single, continuous stream of text along multiple columns, like in many academic papers. The `multicol` package is designed for this and will carry out the automatic splitting. For example, by encapsulating the text inside the `multicols` environment as

```
\begin{multicols}{2}
Zhuge Liang (born 181, Yangdu [now Yinan, Shandong province], China--
   died August 234, Wuzhangyuan [now in Shaanxi province], China) was
```

```
    a celebrated adviser to Liu Bei, founder of the Shu-Han dynasty
   (221--263/264).
...
A mechanical and mathematical genius, Zhuge is credited with
   inventing a bow for shooting several arrows at once and with
   perfecting the Eight Dispositions, a series of military tactics.
   In the Sanguozhi yanyi (Romance of the Three Kingdoms), the great
   14th-century historical novel, Zhuge is one of the main characters
   ; he is portrayed as being able to control the wind and foretell
   the future.
\end{multicols}
```

(again with the number of columns indicated in the first argument) we may acquire the following layout:

Zhuge Liang (born 181, Yangdu [now Yinan, Shandong province], China–died August 234, Wuzhangyuan [now in Shaanxi province], China) was a celebrated adviser to Liu Bei, founder of the Shu-Han dynasty (221–263/264).

Quick Facts:
Wade-Giles romanization: Chu-ko Liang
Courtesy name: Kongming
Born: 181, Yangdu [now Yinan, Shandong province], China
Died: August 234, Wuzhangyuan [now in Shaanxi province], China (aged 53)

Zhuge, to whom supernatural powers often are ascribed, has been a favoured character of many Chinese plays and stories. Legend states that Liu Bei, then a minor military figure, heard of Zhuge Liang's great wisdom and came three times to the wilderness retreat to which Zhuge had retired to seek him out as an adviser. It is known that Zhuge helped Liu organize a large army and found a dynasty. Liu was so impressed with Zhuge's wisdom that on his deathbed Liu urged his son to depend on Zhuge's advice and urged Zhuge to ascend the throne himself if the prince were unable to rule. Some historical accounts indicate that Zhuge died from illness while leading a military campaign in 234.

A mechanical and mathematical genius, Zhuge is credited with inventing a bow for shooting several arrows at once and with perfecting the Eight Dispositions, a

series of military tactics. In the Sanguozhi yanyi (Romance of the Three Kingdoms), the great 14th-century historical novel, Zhuge is one of the main characters; he is portrayed as being able to control the wind and foretell the future. (Source: Encyclopaedia Britannica)

Two Columns Set-up We can also pass the `twocolumn=true` option to `\KOMAoptions` to demand the entire book to be formatted in two columns globally. However, note that it will greatly mess up the layout of this book. (The decision to adopt such a format should be made at an early time!)

In the Same Page Last but not least, text will normally be distributed across pages, and if we want to force a specific part to stay together on the same page, we can surround it within the `samepage` scope.

Chapter 5

Self-defined Commands and Environments

Introduction This chapter concerns the possibility for users to define new commands and environments in LaTeX, by leveraging flow control just like any other programming language.

5.1 Self-defined Commands

Defining New Commands The main way to define our own commands (*macros*) is to invoke the `\newcommand*{<command_name>}[<no._arg>]{<code>}` statement.[1] The necessity arises mainly when we want to repeatedly apply the same action, in addition to ensuring code readability and maintenance. Now, let's see a simple example of highlighting keywords in a particular style:

```
\newcommand*{\mykeyword}[1]{\textcolor{Red}{\textbf{#1}}}
```

[1]There is an unstarred version, but just sticking to the starred version will be adequate.

This definition can be put either in the preamble (preferable) or any suitable location in the main document. Writing `\mykeyword{Attention!}` then gives **Attention!** The `#1` part indicates where the first argument will go during the code execution, and the logic is similar when there are multiple arguments. Remember that when calling any command, each argument requires exactly one pair of curly brackets to receive it.

Optional Arguments As in many other programming languages, there can also be optional arguments that may come with a default value when defining a command. The syntax will become `\newcommand*{<command_name>}[<no._arg>][<default_value>]{<code>}`. For example, if we define

```
\newcommand*{\homoeqn}[2][0]{$f(#2x,#2y) = #2^{#1}f(x,y)$}
```

then `\homoeqn{t}` gives $f(tx, ty) = t^0 f(x, y)$ while `\homoeqn[1]{s}` gives $f(sx, sy) = s^1 f(x, y)$. When interpreting argument indices in the code statement, optional argument(s) will take precedence over the compulsory argument(s). In this case, `#1` represents the optional argument with the default value of 0, which has been overridden by the new value 1 within the square brackets during the second run.

Renewing Commands If we want to edit a command that is already defined by us or another package, we will need to use the `\renewcommand*` statement to properly update and overwrite the original command. It has the same format as `\newcommand*`. Using the same example, we can write

```
\renewcommand*{\mykeyword}[1]{\textcolor{Green}{\textit{\textbf
    {#1}}}}
```

Writing `\mykeyword{Okay!}` now then gives ***Okay!***

Exercise(s)

5.1) Create a command that takes two arguments and outputs a sentence in the form of: There are `<no._of_population>` (comma-separated) people in `<city>`. The `\num[group-separator={,}]` command by the `siunitx` package will be useful for processing the first argument. Try to execute it multiple times with different inputs.

5.2 Flow Control

5.2.1 If-then-else Structures

If-then-else Statements Commands/functions are rather boring if there is no constraint or checking imposed. As you probably know, *if-then-else* statements are one of the major flow control constructs in all programming languages, and LaTeX is no exception. With these, we can produce more complex outcomes with commands. While there are primitive TeX syntax such as `\if` and `\else` for that, it is easier and more natural if we use the verbose `\ifthenelse` construct provided by the `ifthen` package. The format is

```
\ifthenelse{<boolean_test>}{<then_clause>}{<else_clause>}
```

The first argument contains a test that evaluates to some *boolean* value (true or false). If the test returns true, then the "then clause" in the second argument is executed. Otherwise, if it is false, then the "else clause" in the third argument is executed instead. The most basic test is to compare two quantities, and here is a very simple example: if we type

```
\ifthenelse{1 > 2}{Preposterous!}{Of course not...}
```

we should see "Of course not...".

Parsing and Evaluating Math When designing a boolean test for the `\ifthenelse` statement above, we often need a way to compute the results of math expressions for comparison. The native TEX does offer some commands such as `\numexpr` or `\dimexpr` for that, but here we will utilize the `pgfmath` package to parse math expressions. The usage mainly takes the form of `\pgfmathparse{<expression>}`. For instance, `\pgfmathparse{2+2}\pgfmathresult` returns 4.0 where the `\pgfmathresults` command stores the last value processed and prints it out. Alternatively, we can save it to a macro by `\pgfmathsetmacro{<macro>}{<expression>}`. Using the same example, we can write something like

```
\pgfmathsetmacro{\myans}{2+2}
```

typing `\myans` then gives 4.0. To learn more about how a `pgfmath` expression should be formatted, see https://tikz.dev/math-parsing.

Equality Test for Decimals Subsequently, we can design a command that checks equality and looks like

```
\newcommand*{\myequal}[2]{\pgfmathsetmacro{\Lhs}{#1}\pgfmathsetmacro
   {\Rhs}{#2}% <- this % is needed to consume the spacing
\ifthenelse{\lengthtest{\Lhs pt = \Rhs pt}}{#1 is equal to #2}{#1 is
   not equal to #2.}}
```

Typing `\myequal{2*3}{6}` then outputs "2*3 is equal to 6" as expected. Notice that in the `ifthenelse` boolean test, we have not directly done the naive comparison as `\Lhs = \Rhs`. This is because the original method only handles integers, but the `pgfmath` calculation produces decimal/float numbers. To circumvent this, we must use the `\lengthtest` command, which is designed to compare decimal dimensions, and we will only need to add the same length unit to both sides for it to work. (TEX StackExchange 84625)

Logical Operators In an `ifthenelse` test, we may need to compose different booleans using logical operators. There are the self-explanatory `\AND`, `\OR`, and

`\NOT` for that. As a demonstration,

```
\ifthenelse{\NOT \(1 = 2\) \AND \(0 = 0\)}{These make sense!}{What?}
```

gives "These make sense!". We enclose each smaller test with `\( \)`. Note that there is no precedence rule in `ifthen` and the evaluation goes from left to right straightly, except when using `\( \)`.

Checking Strings and Definitions There are two other boolean tests that can be helpful: `\equal{<string>}{<string>}` and `\isundefined{<command_name>}`, which check whether two strings are equal and if a command exists, respectively. A quick use is to determine if a string is empty by `\equal{<string>}{}`.

5.2.2 For Loops

For Loops Another essential type of flow control is *for loops*, which repeatedly execute a code block over some range of values. The `pgffor` offers this functionality with the `\foreach` construct. The format goes like

```
\foreach \<variable> in {<range>} {
    % do something
}
```

A toy example will be

```
\foreach \x in {5,...,1} {
    \x! \\}
Time is up! \\
```

that outputs

5!

4!

3!

2!
1!
Time is up!
The `{5, ..., 1}` part is a shorthand for `{5, 4, 3, 2, 1}` and it also works for ascending order or other patterns.[2]

Parallel For Loops We can also simultaneously loop over multiple variables by separating them with `/`, for example:

```
\foreach \y/\z in {1/2, 2/3, 3/5, 4/7} {
    \\ Prime $\y \rightarrow \z$}
```

produces
Prime $1 \to 2$
Prime $2 \to 3$
Prime $3 \to 5$
Prime $4 \to 7$

Nested Loops Moreover, we can produce *nested loops* as in other programming languages. For instance, the block

```
\foreach \ii in {0,...,2} {\\%
    \foreach \jj in {1,...,4} {%
    (\ii, \jj)%
    }
}
```

will generate the following pattern:
(0, 1)(0, 2)(0, 3)(0, 4)
(1, 1)(1, 2)(1, 3)(1, 4)
(2, 1)(2, 2)(2, 3)(2, 4)
The `%` are needed to absorb the extra spaces produced by the indents in the code.

[2]For a general discussion, see TEX StackExchange 142188.

Counting When iterating over a range of values, the corresponding index can be saved into a variable as `count`. For example, the previous example of printing out the first four primes can be replaced by the following equivalent snippet:

```
\foreach \z [count=\y] in {2, 3, 5, 7} {
    \\ Prime $\y \rightarrow \z$}
```

Evaluating by the Side Also, in a `\foreach` loop, calculations can be applied over the iterated variable by the `evaluate` option. As an illustrative example, this loop

```
\foreach \x [evaluate=\x as \y using \x^2] in {1,...,10} {
    \\ $\x ^2 = \y$}
```

readily outputs

$1^2 = 1.0$

$2^2 = 4.0$

$3^2 = 9.0$

$4^2 = 16.0$

$5^2 = 25.0$

$6^2 = 36.0$

$7^2 = 49.0$

$8^2 = 64.0$

$9^2 = 81.0$

$10^2 = 100.0$

Remembering the Last Value Another feature of a `\foreach` loop is the `remember` option, which stores the current variable and recalls it in the next iteration. Stealing the example from the PGF User Manual:

```
\foreach \x [remember=\x as \lastx (initially A)] in {B,...,H}{\lastx
   $\to$\x, }
```

produces A→B, B→C, C→D, D→E, E→F, F→G, G→H, in the expected order.

Loop Breaking An important component of any for loop is the ability to *break* it given some certain condition. This is done by the `\breakforeach` command. This will be used in combination with an `\ifthenelse` statement. For example, the code snippet

```
\foreach \x in {1,...,100} {%
\ifthenelse{\NOT \(\x = 13\)}{\x, }{13 is an unlucky number! Stop! \
    breakforeach}
}
```

outputs 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, Wait, 13 is an unlucky number! Stop!

Exercise(s)

5.2) Define a command that converts Celsius temperature to Fahrenheit and prints a warning if the Fahrenheit temperature is higher than 100 degrees.

5.3) By manipulating all the tools introduced in this chapter, efficiently imitate the following outputs. The `int` and `Mod` operators for `pgfmath` will be useful.

1 is not divisible by 3. 1 is not divisible by 5.
2 is not divisible by 3. 2 is not divisible by 5.
3 is divisible by 3. 3 is not divisible by 5.
4 is not divisible by 3. 4 is not divisible by 5.
5 is not divisible by 3. 5 is divisible by 5.
6 is divisible by 3. 6 is not divisible by 5.
7 is not divisible by 3. 7 is not divisible by 5.
8 is not divisible by 3. 8 is not divisible by 5.
9 is divisible by 3. 9 is not divisible by 5.
10 is not divisible by 3. 10 is divisible by 5.
11 is not divisible by 3. 11 is not divisible by 5.
12 is divisible by 3. 12 is not divisible by 5.

13 is not divisible by 3. 13 is not divisible by 5.
14 is not divisible by 3. 14 is not divisible by 5.
15 is divisible by both 3 and 5.
16 is not divisible by 3. 16 is not divisible by 5.
17 is not divisible by 3. 17 is not divisible by 5.
18 is divisible by 3. 18 is not divisible by 5.
19 is not divisible by 3. 19 is not divisible by 5.
20 is not divisible by 3. 20 is divisible by 5.

5.3 Self-defined Environments

New Environments Similarly, we can also define our own environments by the `\newenvironment*{env_name}{begin}{end}` method. `begin`/`end` stores the code to be run before the start/after the end of the new environment. For example, by defining

```
\newenvironment*{mylargeblueeqn}{\color{blue}\LARGE\begin{equation
    }}{\end{equation}}
```

[3] Then writing

```
\begin{mylargeblueeqn}
1+1 = 2
\end{mylargeblueeqn}
```

displays the expected blue, magnified equation:

$$1 + 1 = 2 \tag{5.1}$$

[3] Note that replacing `equation` with `align` instead will throw an error, read TEX StackExchange 236664.

Renewing Environments There is also the concurrent `\renewenvironment*` for renewing the definition of an environment. However, we seldom need to (re)define an environment on our own: In a math document, the interface provided by the `tcolorbox` package (see Chapter 7) will fulfill most of the usages. So we will keep this section short.

Chapter 6

More on Book Layout Design

Introduction This chapter will go into the details about designing and refining the layout of a LaTeX book, including how to edit the page style with running headers/footers, polishing chapter/section headings, making a good-looking title page, and more.

6.1 Page Configuration

6.1.1 Some Universal Settings for the Pages

Two-sided Book In a printed book, there will be a distinction between odd (right) and even (left) pages, i.e. *two-sided*, whereas an electronic PDF document is usually *one-sided* and has no such issue. To choose one of these configurations in a `scrbook`, we can change the `twoside` parameter in `\KOMAoptions` to `false`, `semi`, or `true`. Unsurprisingly, `false` indicates one-sided and `true` means two-sided. Being two-sided means that there will be a difference between the inner/outer margins, with the outer margin (left/right on even/odd pages)

occupying two times the space as the inner one. The running headers on odd and even pages will also show the current section/chapter differently.

In this book, the `twoside` option is set to `semi` by adding `\KOMAoptions {twoside=semi}`. This retains equal margins as if the document is one-sided, but the headers will switch alternately just like two-sided.

Division Factor The extent of the type area in pages is controlled by the `DIV` factor passed to `\KOMAoptions`. The higher the value of `DIV`, the larger the fraction of the main text area and the smaller the margins. The reference value of `DIV` usually ranges from 9 to 12. However, we can delegate the calculation of an optimal `DIV` by setting `DIV` to either `calc` or `classic`. If a new font is loaded, it is also desirable to recalculate an appropriate type area by calling `\KOMAoptions{DIV=last}` that reuses the same setting.

Line Spread Finally, to control the *line spread* in the main text, just add `\linespread{<value>}`. Here it is set to 1.25.

6.1.2 Page Style

Headers and Footers A very important part of a page is its *header* and *footer*. Editing the content within a header/footer requires us to load the `scrlayer-scrpage` package. The default page style for the main content in the book is invoked by `\pagestyle{scrheadings}`. We can refer to the inner/center/outer header/footer via `ihead`, `chead`, `ohead`, `ifoot`, `cfoot`, and `ofoot` correspondingly. For example, the default page number is put at the outer footer, and this book has moved it to the center footer by declaring:

```
\pagestyle{scrheadings}
\ofoot*{}
\cfoot*{\pagemark}
```

where the variable `\pagemark` stores the page number in `scrbook`.

There is also a finer division between even/odd pages for headers/footers: `lehead`, `cehead`, `rehead`, `lohead`, `cohead`, `rohead`, where `l`, `c`, `r` stand for left/center/right, and `e`, `o` represent even and odd respectively. If we want to swap the chapter/section in the even/odd-paged headers, we can write something like

```
\lehead*[]{\rightmark}
\rohead*[]{\leftmark}
```

The content in the optional argument will be applied to the chapter page (more accurately, the *plain* page style), and is left empty as in the default setting. The `\leftmark` and `\rightmark` hold the original left/right headers, the depth of which is controlled by

```
\automark[section]{chapter} % [right]{left}
```

Setting Font Style for Elements Now we will further customize the font style and color of our header by the command `\setkomafont{<element>}{<commands>}`. As its name suggests, it applies commands to set up a certain element in a page. For headers, the corresponding alias is `pagehead`, and in this book, we have

```
\setkomafont{pagehead}{\color{RoyalBlue}\slshape\bfseries}
```

Many other elements can be changed as well, including `chapter`, `section`, `footnote`, `caption`, `pagefoot`, and so on. Another change adopted in this book is

```
\setkomafont{caption}{\itshape}
```

Separating Line for Headers The separating line under the header is added via passing the switch `headsepline=on` to `\KOMAoptions`. We similarly have `footsepline`.

Format for the Chapter Mark Sometimes we may want to change the chapter label in the header too. This is done by applying `\renewcommand*` to `\chaptermarkformat`. In this book, we have

```
\renewcommand*{\chaptermarkformat}{\chapapp~\thechapter\autodot~--~}
   % ~ occupies a space, -- is a long dash
```

`\chapapp` stores the word "Chapter" that can be modified later, `\thechapter` contains the current chapter counter, and `\autodot` is empty, reserved for an extra dot after any chapter/section numbering. As you may have guessed, there is also `\sectionmarkformat`.

Exercise(s)

6.1) Edit the page style to make your own header and footer.

6.2 Appearance of Chapters and Sections

Prefix and Format of Chapter Headings We can single out the chapter number as a prefix in the chapter title by setting `chapterprefix=true` in `\KOMAoptions`. Furthermore, we can customize it by `\addtokomafont`, which behaves similarly to `\setkomafont` and adds new commands on top of the original ones:

```
\addtokomafont{chapterprefix}{\itshape\color{white}}
```

In addition, we can again call `\renewcommand*` to manipulate `\chapterformat`. In this book, the following code is adopted to produce a blue box to hold the chapter number in position:

```
\renewcommand*{\chapterformat}{\raggedleft\colorbox{RoyalBlue}{\
   parbox[b][2.8em]{2.8em}{\vfill\centering{\large\chapapp}\\[-0.4em
   ]\thechapter\vfill}}}
```

Format for Section Numbers In the same way, we have `\sectionformat` that can be edited for section headings. The code to produce the white section number in a black box is

```
\renewcommand*{\sectionformat}{\colorbox{black}{\textcolor{white}{\
    thesection}}\enskip}
```

Format for Section Line Meanwhile, the black line under section headings is governed by `\sectionlinesformat`. To modify that, notice that it has to accept 4 arguments, where only the third (section number) and the last (section name) will be relevant here:

```
\renewcommand*{\sectionlinesformat}[4]{\makebox[0pt][l]{\rule[-\
    fboxsep-\fboxrule]{\textwidth}{\fboxrule}}#3\parbox[b]{0.85\
    textwidth}{\linespread{1}\selectfont#4}}
```

(Reference: TEX StackExchange 581566) The `makebox` command creates an artificial empty box that does not occupy any width (0 pt) and contains the desired separating line (`\rule` with a length of `\textwidth`). It is vertically offset downwards by `\fboxsep` (plus `\fboxrule`) to compensate for the padding around the black numbered box (`#3`) that follows, defined via `\sectionformat` above. The section name (`#4`) is subsequently wrapped by a `\parbox` that is bottom-aligned and can account for any title longer than one line.

End of Chapter Headings While there is also the `\chapterlinesformat` command for making a line below the chapter title, we can achieve more by tackling `\chapterheadendvskip` instead. It controls the stuff (usually some vertical skip) that occurs after a chapter heading. For aesthetics, we will load the `pgfornament` package that supplies many beautiful visual patterns. Then, we can write

```
\renewcommand*{\chapterheadendvskip}{\pgfornament[width=\textwidth
    ]{88}\par} % the 88th ornament
```

to achieve the chapter line layout present in the book.

Start of Chapter Headings There exists the `\chapterheadstartvskip` counterpart for anything before the chapter title as well. The default vertical space above the chapter heading may be too much, and we can shorten it via

```
\renewcommand*{\chapterheadstartvskip}{\addvspace{2em}}
```

`\addvspace` is a variant of `\vspace`, which is a kind of *rubber length*: it adds just enough vertical space so that the total space is as large as the input length if the existing space is shorter than that, and does nothing when it is already long enough.

Default Sans-serif As you may notice, the chapter/section headings are written in sans-serif. To change this, we can set `sfdefaults` to `no` in `\KOMAoptions`. We can also mark up text with `\textmaybesf{<text>}` or the `\maybesf family` switch, which are toggled by `sfdefaults` too.

6.3 Title Page and Front/Back Matter

(interleaf?)

Title Page The easiest way to generate a *title page* is to simply use the `\make title` command. Just enter the book (sub)title, author name, and the like in the preamble using the corresponding commands, for example:

```
\title{How to Reproduce this Book Exactly with \LaTeX}
\subtitle{A Self-contained Tutorial on Writing Mathematical Notes}
\author{C.~L.~Loi}
```

and maybe `\date` and `\publishers`, etc. Then calling `\maketitle` will automatically build a title page for you. In addition, the `titlepage` switch in `\KOMAoptions` can decide if it is embedded in-page. However, it is more flexible to design our own title page using the `titlepage` environment. In this book, we have adopted:

```
\begin{titlepage}
\parbox{0.7\textwidth}{\Huge\raggedright\textbf{\textmaybesf{How to
   Reproduce this Book Exactly with \LaTeX}}}\par
\vspace{2mm}
\parbox[b]{0.9\textwidth}{\large\raggedright\textit{A Self-contained
   Tutorial on Writing Mathematical Notes}}
\hfill\textcolor{RoyalBlue}{\rule{3mm}{3mm}}\par
\vspace{4mm}\hrule\par
{\Large\raggedleft\textmaybesf{v1.0.0}\hfill C.~L.~Loi\par}
\vfill
{\large\raggedleft A student from \\
CUHK-EESC/NTU-AS\par}
\end{titlepage}
```

where we have utilized a lot of boxes and positioning/spacing commands.

Front/Main/Back Matter Usually in a book, there will be *front matter* (title page, preface, table of contents) and *back matter* (bibliography, index). To mark them, just write `\frontmatter` and `\backmatter` in front of the corresponding parts. We also have `\mainmatter` for transitioning to the main content. So it should look like

```
\frontmatter
...
\tableofcontents
...
\mainmatter
\include{ch0_install}
\include{ch1_basic_structure}
...
```

`\frontmatter` will use Roman page numbering, and `\mainmatter` will switch it back to the normal Arabic numbering. We can also manually do this by

```
\pagenumbering{roman}
...
\cleardoubleoddpage
\pagenumbering{arabic}
```

The `\cleardoubleoddpage` command is needed to properly flush the page.

Back of Title Sometimes we may want to put a copyright statement or other information at the back of the title page. If we choose to create the title page by the `\maketitle` method, then we can simply supply the `\lowertitleback` or `\uppertitleback` command, e.g. for this book:

```
\lowertitleback{"How to Reproduce this Book Exactly with \LaTeX"\\
Copyright, C.~L.~Loi, 2025. All rights reserved.}
```

However, if the title page is designed manually, then we may instead write, after it:

```
\thispagestyle{empty}
\vspace*{\fill}
"How to Reproduce this Book ... All rights reserved.
```

`\thispagestyle{empty}` selects the `empty` page style for just this one page.

Exercise(s)

6.2) Design your own title page.

6.4 Footnotes and Markings

6.4.1 Footnotes and Line Numbers

Footnotes The basic way to add a footnote like this[1] is to use the `\footnote{<text>}` command, as

```
The basic way to add a footnote like this\footnote{This is a simple
    footnote that is directly inserted after the desired location.} is
     ...
```

Another way is to use the `\footnotemark \footnotetext` pair like this[2]

```
Another way is to use the \texttt{\textbackslash footnotemark} \
    texttt{\textbackslash footnotetext} pair like this\footnotemark{}
...
where we can put the text anywhere after the mark.\footnotetext{This
    footnote by \texttt{\textbackslash footnotetext} will
    automatically be traced to the latest texttt{\textbackslash
    footnotemark}.}
```

where we can put the text anywhere after the mark.

Multiple Footnote Marks If there is more than one `\footnotemark` before a `\footnotetext`, the index of the `\footnotetext` will be set according to the newest `\footnotemark`. This can be problematic if the previous `\footnotetext` are delayed. Here we demonstrate the fix[3] for such a scenario[4].

[1] This is a simple footnote that is directly inserted after the desired location.

[2] This footnote by `\footnotetext` will automatically be traced to the latest `\footnotemark`.

[3] We can manually decrease the footnote counter by 1 here with `\footnotetext[\numexpr \value{footnote}-1]`.

[4] Try removing the patch above, and the numbering will clash.

Separating Line for Footnotes To customize the appearance of the separating line above the footnotes, we can call `\renewcommand*{\footnoterule}{<code>}`. However, a shortcut is to do `\setfootnoterule<length>`, where it is set to 0.8 times `\textwidth` in this book.

Referencing Footnote A footnote can be labeled and referenced as well. Just put `\label{<label_name>}` inside the footnote and use `\ref` as for other elements.

Line Numbers Meanwhile, to produce line numbers for text (which is common for thesis or publications), we can load the `lineno` package. Then we can initiate line numbers by using `\linenumbers` and stop with `\nolinenumbers`. For example, this paragraph is affected by

```
\linenumbers
Meanwhile, to produce line numbers for text ... For example, this
   paragraph is affected by \par
\nolinenumbers
```

(marginpar?)

6.4.2 Hyperlinks and Bookmarks

Hyperlinks To enable inserting hyperlinks (e.g. websites, or referencing in the book) in the document, we need to import the `hyperref` package. Then we can simply use the `\href{<link>}{<text>}` command, for example

```
\href{https://www.google.com/}{Google}
```

yields the link to Google. Internal referencing links will be automatically formed. As a side note, we can also use `\autoref` that determines the prefix for objects automatically in place of `\ref`.

Highlighting Options for Links The default highlighting effects for hyperlinks by `hyperref` can be controlled by `\hypersetup`. In this book, we have used

```
\hypersetup{
    colorlinks,
    linkcolor = black,
    urlcolor = blue!90!Green,
    pdfauthor = Benjamin Loi,
    pdftitle = How to Reproduce this Book Exactly with LATEX,
    pdfsubject = v1.0.0,
    pdfkeywords = {Mathematics, LATEX}
}
```

The `colorlinks` keyword replaces the default colored boxes by colored text, while `linkcolor` and `urlcolor` indicate the color for internal and external links respectively. The subsequent options are a by-product to set up the metadata for the PDF file.

PDF Bookmarks To further facilitate the PDF file, we can load the `bookmark` package with the following options:

```
\usepackage[open,openlevel=1,atend,numbered]{bookmark}
```

The `open` and `openlevel` options tell to which depth the bookmarks are expanded when the PDF file is open, while the `numbered` option supplies the chapter/section numbering at the start of each bookmark.

6.4.3 Index Page

Indices Many books will provide an index page for important words or concepts at the end of the book. To do this in LaTeX, we can import the `imakeidx` package, immediately followed by the `\makeindex[intoc]` command. Then, to add an index entry, we add the `\index{<name>}` command at the corresponding position. For example, in this paragraph:

```
... To do this in \LaTeX{}, we can import the \texttt{imakeidx}\index
   {imakeidx@\texttt{imakeidx}} package ...
```

Here, inside the `\index` command, we put the internal name used for sorting and the actual shown index, separated by `@`. Meanwhile, a sub-index can be set with `!`. Finally, the full index list will be printed with `\printindex`. To produce alphabetical headers for the indices, we can paste the following patch in the preamble (TEX StackExchange 712863):

```
\begin{filecontents}[overwrite]{myindexstyle.ist}
headings_flag 1
heading_prefix
 "\\bigskip\\par\\penalty-50\\textbf{"
heading_suffix
 "}\\\\[\\medskipamount]"
symhead_positive "Symbols"
symhead_negative "symbols"
numhead_positive "Numbers"
numhead_negative "numbers"
delim_0 ",\~"
\end{filecontents}
```

and now use `\makeindex[intoc, options=-s myindexstyle]`.

Chapter 7

Framed Theorems and Exercises

Introduction This chapter touches on how to make beautiful, colored frames around theorems, examples, and so on. The typesetting and management of exercises and answers will also be discussed.

7.1 Colored Boxes for Theorems and Proofs

Making Colored Boxes We will start by generating a simple colored box first, which is most easily done by importing the `tcolorbox` package. Then we can define the design of the box by the `\newtcolorbox` command. An illustrative template is

```
\newtcolorbox{mybox}[1][]{
  colback=Green!20,
  colframe=Gray,
  coltitle=Yellow,
  title=This is my box,
  boxrule=1pt,
  leftrule=1ex,
  boxsep=1ex,
  left=1ex,
```

```
  right=1ex,
  sharp corners,
  breakable,
  before skip=\topsep,
  after skip=\topsep, #1}
```

that creates a box environment named `mybox`. Then typing

```
\begin{mybox}
The content goes here.
\end{mybox}
```

produces the following box:

This is my box

The content goes here.

`colback`, `colframe`, and `coltitle` denote the color of the background, frame, and title correspondingly. `boxrule` (`leftrule`) is the width of bounding lines (at the left), and `boxsep` indicates the overall padding around the title and content. `left`/`right` further refines the padding to the left/right. The meanings of the `sharp corners` and `breakable` keywords are not hard to guess: the box will have sharp corners instead of rounded ones, and it can break across pages. Finally, `before skip` and `after skip` indicate the vertical spacing to other objects before and after the entire box. The `[1][]` part is added so that the box can receive an optional argument, which can override the given setting of the box via putting `#1` at the end of the `\newtcolorbox` option list.

Colored Numbered Theorems/Examples The readers are probably concerned more about how to construct a colored, numbered box for theorems, examples, definitions, and the like. This requires us to pass

```
\tcbuselibrary{theorems}
```

and then we can use the `\newtcbtheorem[<init>]{<name>}{<display_name>}{<options>}{<prefix>}` construct. The `init` part sets up the way of numbering, the `name` is referred to when an instance of the box environment has to be made, while the `options` part is just like the previous input lists for `\newtcolorbox`. For example, we can define

```
\newtcbtheorem[number within=chapter]{thm}{Theorem}{
  colback=Green!20,
  colframe=Green!50,
  fonttitle=\bfseries,
  boxrule=1pt,
  boxsep=1ex,
  left=1ex,
  right=1ex,
  pad after break=1.5ex,
  sharp corners,
  breakable,
  before skip=\topsep,
  after skip=\topsep}{thm}
```

where we have added some new parameters: `fonttitle` here indicates the title to be in boldface, and `pad after break` adds some padding after the box breaks across the page. The `number within=chapter` option tells the numbering to be based on chapters, and it can be changed to, e.g. `section`. Subsequently, writing

```
\begin{thm}{Mean Value Theorem}{mvt}
If ...
\begin{equation}
f'(c) = \frac{f(b)-f(a)}{b-a}
\end{equation}
\end{thm}
```

produces

Theorem 7.1: Mean Value Theorem

If $f(x)$ is continuous on $[a, b]$ and differentiable on (a, b), then there exists $c \in (a, b)$ such that

$$f'(c) = \frac{f(b) - f(a)}{b - a} \tag{7.1}$$

We can refer to this theorem as Theorem 7.1 by `\ref{thm:mvt}` (`prefix: alias`). It is also possible to supply additional options to override the base setting of the colored box.

Shared Numbering for Definitions and the Others Another feature that may be useful is to enable shared numbering for definitions, lemmas, corollaries, properties, and so on. To do so, we can invoke the `\newtcbtheorem` command again and pass the keyword `use counter from=` for the `init` option:

```
\tcbset{mycommon/.style={
  colback=Green!20,
  colframe=Green!50,
  fonttitle=\bfseries,
  coltitle=black,
  theorem style=plain,
  ...}
}
\newtcbtheorem[use counter from=thm]{defn}{Definition}{mycommon}{defn
    }
```

Here we use `\tcbset` to save the style for repeated use. Then

```
\begin{defn}{Taylor Expansion}{taylor}
For a function $f(x)$ infinitely differentiable at point $x = a$, we
    have its Taylor series as
\begin{equation}
f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \
    frac{f'''(a)}{3!}(x-a)^3 + \cdots
\end{equation}
\end{defn}
```

is rendered as

Definition 7.2 (Taylor Expansion): For a function $f(x)$ infinitely differentiable at point $x = a$, we have its Taylor series as

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots \quad (7.2)$$

Notice that we have set `theorem style=plain` (there are more possible values, like `break`), see if you can figure out where the difference is.

Proofs For proofs or worked steps, their setting will often be slightly different. We can load the `amsthm` package, which comes along with the `proof` environment, and apply `\tcolorboxenvironment` on it. The template will be

```
\tcolorboxenvironment{proof}{
  blank,
  breakable,
  borderline west={0.5ex}{2pt}{black},
  left=2ex,
  before skip=\topsep,
  after skip=\topsep}
```

The `blank` keyword initializes just the frame and the `borderline west` option draws a long black line to the left. Then, writing

```
\begin{proof}
Consider $\vec{w} = \vec{u} + t\vec{v}$, ...
\begin{align*}
\Delta = b^2 - 4ac &\leq 0 \\
(2(\vec{u} \cdot \vec{v}))^2 - 4\norm{\vec{u}}^2\norm{\vec{v}}^2 &\
   leq 0 \\
(\vec{u} \cdot \vec{v})^2 - \norm{\vec{u}}^2\norm{\vec{v}}^2 &\leq 0
   \\
(\vec{u} \cdot \vec{v})^2 &\leq \norm{\vec{u}}^2\norm{\vec{v}}^2 \\
|\vec{u} \cdot \vec{v}| &\leq \norm{\vec{u}}\norm{\vec{v}} \qedhere %
    putting \qedhere to eliminate the spurious gap
```

```
\end{align*}
\end{proof}
```

results in

Proof. Consider $\vec{w} = \vec{u} + t\vec{v}$, where t is any scalar, then $\|\vec{w}\|^2 = \vec{w} \cdot \vec{w} \geq 0$ by positivity. Also, $\vec{w} \cdot \vec{w}$ can be written as a quadratic polynomial in t:

$$\vec{w} \cdot \vec{w} = (\vec{u} + t\vec{v}) \cdot (\vec{u} + t\vec{v}) = \|\vec{u}\|^2 + 2t(\vec{u} \cdot \vec{v}) + t^2\|\vec{v}\|^2$$

Since this quantity is always greater than or equal to zero, i.e. the quadratic polynomial has no root or a repeated root, it means that the discriminant must be negative or zero. So,

$$\begin{aligned} \Delta = b^2 - 4ac &\leq 0 \\ (2(\vec{u} \cdot \vec{v}))^2 - 4\|\vec{u}\|^2\|\vec{v}\|^2 &\leq 0 \\ (\vec{u} \cdot \vec{v})^2 - \|\vec{u}\|^2\|\vec{v}\|^2 &\leq 0 \\ (\vec{u} \cdot \vec{v})^2 &\leq \|\vec{u}\|^2\|\vec{v}\|^2 \\ |\vec{u} \cdot \vec{v}| &\leq \|\vec{u}\|\|\vec{v}\| \end{aligned}$$

□

Moreover, we can copy it with "Proof" replaced by "Solution" as

```
\newenvironment{solution}{\begin{proof}[Solution]}{\end{proof}}
```

Exercise(s)

7.1) Design your own color box to display any example problem, followed by a worked solution.

7.2 Typesetting Exercises and Answers

Exercises The essence of any math book is its exercises. To deliver exercises and their solutions, we can import the `exercise` package with the following options (to be explained soon):

```
\usepackage[lastexercise,answerdelayed]{exercise}
```

Then we can typeset any exercise within the `Exercise` environment, which we have been doing for a long time. As an example, the last exercise was created by

```
\begin{exercisebox}
\begin{Exercise}
\phantomsection%
\label{exer:colorbox}%
Design your own color box to display any example problem, followed by
    a worked solution.
\end{Exercise}
\end{exercisebox}
```

where `exercisebox` here is a self-defined[1] colored box environment generated by `newtcolorbox` as introduced in the last section. We can refer to this exercise as Exercise 7.1 via its label `\ref{exer:colorbox}`. As an extra note, to ensure the internal referencing link to the exercise is correct, we need a patch by inserting `\phantomsection` at its start before `label`.

Answers To typeset the answer for an exercise, we can use the `Answer` environment supplied with the corresponding label of that exercise to the `ref` option. Here we will take Exercise 5.3 as a demonstration, where we can write

```
\begin{Exercise}
\phantomsection%
\label{exer:modulo}%
... % the exercise
\end{Exercise}
\begin{Answer}[ref=exer:modulo]
... % the answer goes here
\end{Answer}
```

[1]The actual implementation can be checked from my raw source code.

Alternatively, one can omit the `ref` part if the `lastexercise` option has been ticked when importing the package. It assumes that the answer is for the latest exercise, and hence we can type it immediately after that exercise.

The `answerdelayed` option saves all the answers until the end, and we can output all of them for once by `\shipoutAnswer`. You should be able to see the answers for Exercise 5.3 and others at the end of the book. The detailed code for the answer section of this book is

```
\cleardoubleoddpage
\chapter*{Answers to Exercises}
\addcontentsline{toc}{chapter}{Answers to Exercises} % add the answer
    section to the table of content
\ohead{Answer to Exercises}
\shipoutAnswer
```

Headers for Answers The default headers for answers may be too plain. To customize them, we can use the solution proposed in TEX StackExchange 369265 which involves declaring a boolean variable `firstanswerofthechapter` by `\newboolean` in the `ifthen` package. A minimal version is

```
\newboolean{firstanswerofthechapter}
\renewcommand*{\AnswerHeader}{\ifthenelse{\boolean{
   firstanswerofthechapter}}
    {\textbf{Answers for Chapter \thechapter}\par\vspace{1ex}%
     \theExercise)}
    {\theExercise)}
}
```

This updates the `\AnswerHeader` command with an if-then-else statement. Then, we can call `\setboolean{firstanswerofthechapter}{true}` whenever we are at the first answer in a chapter, then `\AnswerHeader` will first print the heading "Answers for Chapter `\thechapter`" where `\thechapter` is the chapter counter, and proceed to print the exercise number stored by `\theExercise` with a round bracket to the right. Afterwards, reset `\setboolean{first`

`answerofthechapter}{false}` and the `\AnswerHeader` will just consist of the exercise number.

Chapter 8

Plotting with TikZ (Part I)

Introduction This chapter introduces the usage of the **TikZ** engine to draw various mathematical plots and diagrams in LaTeX. This only means to be a brief guide, and the readers should consult the **PGF Manual** https://tikz.dev/pgfplots/ for full details.

8.1 Basic Drawing Syntax

8.1.1 Coordinates and Nodes

Cartesian Coordinates To create a **TikZ** plot, we first have to import the `pgfplots` package and initialize a `tikzpicture` environment. We will start by specifying *coordinates* and labeling that point on the plot as a *node*. A simple example is given in Figure 8.1 below, and the corresponding code is

```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (4,3);
\coordinate (A) at (2,1);
\coordinate (B) at (3,3);
\node at (A) {\Large $(2,1)$};
```

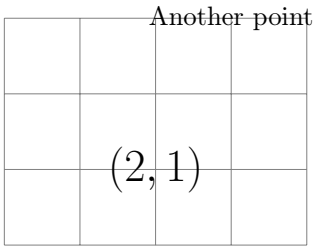


Figure 8.1: *Simple Cartesian coordinates as nodes in TikZ.*

```
\node at (B) {\footnotesize Another point}; % equivalently, directly
   use \node at (3,3) {\footnotesize Another point};
\end{tikzpicture}
```

The `\draw[help lines]` sketches helper grids for refining the positioning. The `\coordinate (<name>) at (<coordinates>)` syntax marks the coordinates of a point internally for later use. Here we use the simplest Cartesian xy-coordinates. The `\node at (<coordinates>) {<text>}` then puts a node, possibly with some text, at the corresponding position.

Polar Coordinates Another common type of coordinates is the polar coordinates, whose expression is `(angle:radius)` where `angle` is relative to the positive x-axis. This is illustrated in the following Figure 8.2:

```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (4,3);
\coordinate (O) at (0,0);
\coordinate[label=above:$A$] (A) at (30:4);
\node at (O) [below left] {$O$}; % below left can be replaced by
   anchor=north east
\node at (A) [circle,fill,inner sep=2pt] {};
\end{tikzpicture}
```

Point A is then positioned at $(4\cos 30°, 4\sin 30°) = (2\sqrt{3}, 2)$.

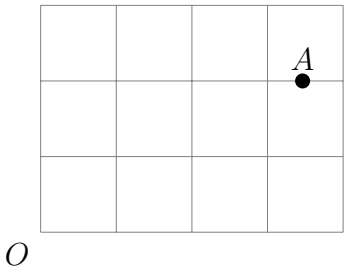


Figure 8.2: *Defining a point in TikZ using polar form instead.*

There are also some other new things. We can place the node text O below and to the left of the origin coordinates by adding `[below left]` (as you may have guessed, there are also `above`, `right`, and their combinations) before it. However, notice that it will also displace the node. Another labeling method is to provide the `[label=<position>:<text>]` option when calling `\coordinate`, which has been applied to point A. Then, we can make a dot to denote the point by using `\node` with the set of options `[circle,fill,inner sep=2pt]` so it fills a small circle with size 2 pt.

8.1.2 Drawing Paths

Straight Lines Given some coordinates, a natural next step is to connect them with curves. We will deal with the simplest case of straight lines first. The basic *path* syntax is `(coordinates) -- (coordinates)`, and can be stacked as we like. This is demonstrated in Figure 8.3 on the next page.

```
\begin{tikzpicture}
\draw[help lines] (-3,-3) grid (3,3);
\coordinate[label=$A$] (A) at (2,1);
\coordinate[label=$B$] (B) at (-2,0);
\coordinate[label=below:$C$] (C) at (1,-1);
\draw[blue,dashed] (-1,-3) -- node[midway,sloped]{Cut} (3,3);
\path[red,draw] (A) -- (B) -- (C) -- cycle;
\end{tikzpicture}
```

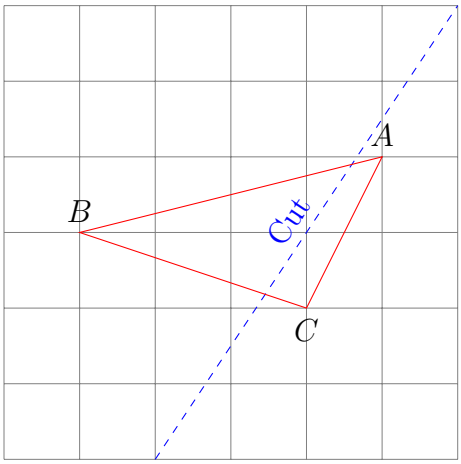


Figure 8.3: *Drawing a line and a closed path as a triangle.*

We have two possible methods to draw a line. The first one is to just use the `\draw` command, whereas the second one is more verbose and uses the `\path` command combined with the `draw` option. For the triangle, the `cycle` alias tells the path to travel back to the initial point. Other takeaways are that we can supply color (`red`, `blue`) and line style (`dashed`, `dotted`) when drawing the path, and we can put a label over the line by adding the `node` syntax after the `--` part with the options `midway` (or `pos=0.5`, to put it in the middle) and `sloped` (sloped with respect to the line).

Relative Coordinates Sometimes it is more convenient to specify coordinates relative to the previous one when constructing a path. This is done by adding the incremental `++` after `--`. Figure 8.4 below is an illustrative example.

```
\begin{tikzpicture}
\draw[help lines] (-3,-3) grid (3,3);
\coordinate[label=below right:$O$] (O) at (0,0);
\draw[Green, line width=1.5] (O) -- (2,1) coordinate[at end](A) --++
   (-1.5,1) coordinate[at end](B) --++ (-3,-3) coordinate[at end](C)
   --++ (-30:2) coordinate[at end](D) -- cycle;
```

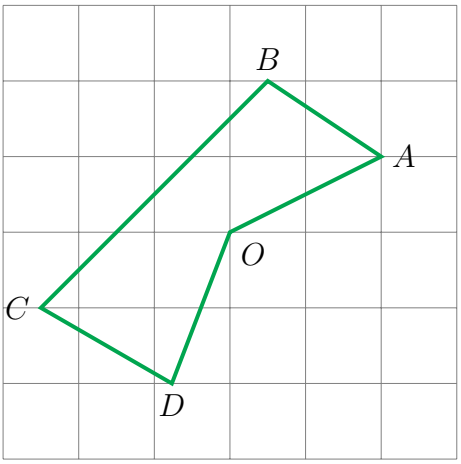


Figure 8.4: *Connecting a path with relative coordinates.*

```
\node[right] at (A) {$A$}; \node[above] at (B) {$B$}; \node[left] at
   (C) {$C$};\node[below] at (D) {$D$};
\end{tikzpicture}
```

where we have used relative coordinates: Cartesian for the second and third segments, and polar for the fourth segment. We may append the `coordinate[at end]` constructs (can be omitted) at each step to remember the last coordinates for subsequent labeling. In addition, we can supply the `line width` parameter (may be substituted by short keywords like `thin`, `thick`, etc.), which is self-explanatory.

Fill Apart from drawing lines, we may also want to fill the area bounded by them. This is done by either the `fill` (or `filldraw`) command or appending the `fill=<color>` option to the `draw` command. This is demonstrated by Figure 8.5 on the next page.

```
\begin{tikzpicture}
\draw[help lines] (-4,-4) grid (4,4);
\coordinate[label={[xshift=7]$A$}] (A) at (3,-1);
\coordinate[label=$B$] (B) at (2,2);
```

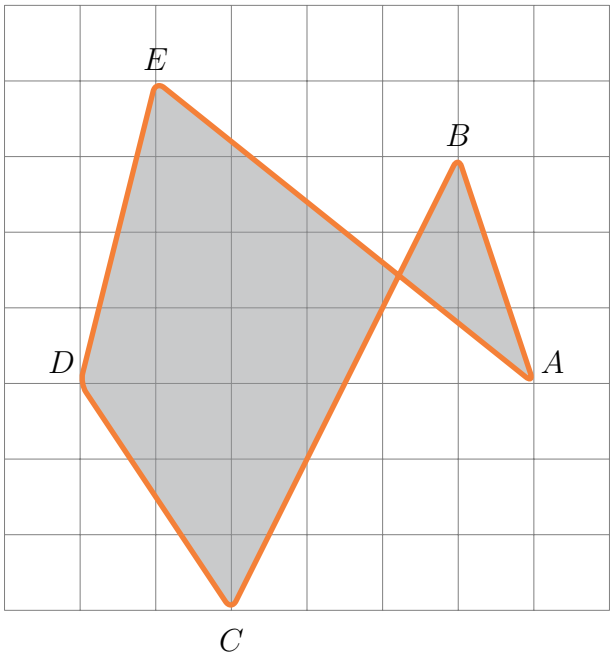


Figure 8.5: *Filling the area enclosed by a path with color.*

```
\coordinate[label={[yshift=-3]below:$C$}] (C) at (-1,-4);
\coordinate[label={[xshift=-7]$D$}] (D) at (-3,-1);
\coordinate[label=$E$] (E) at (-2,3);
\draw[Orange, line width=2, rounded corners, fill=Gray, fill opacity
   =0.5] (A) \foreach \P in {B,...,E} { -- (\P)} -- cycle; %
   equivalent to \draw (A) -- (B) -- (C) -- (D) -- (E) -- cycle;
\end{tikzpicture}
```

We can specify the fill color opacity with the `fill opacity` option. Notice that we have utilized the PGF for-loop functionality to simplify chaining the path. Finally, we have added the `xshift` and `yshift` parameters to fine-tune the positioning of labels, and the effect of `rounded corners` should not be hard to see.

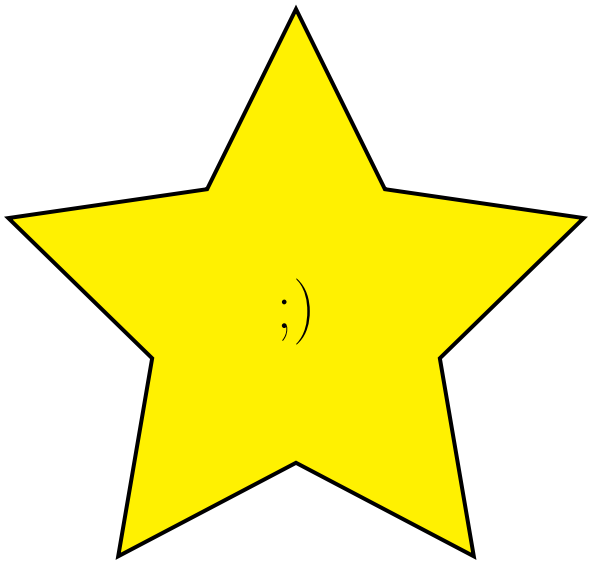


Figure 8.6: *The example star for Exercise 8.1.*

Exercise(s)

8.1) Try to draw and fill a star shape using TikZ. An example is given below as Figure 8.6.

8.1.3 Shapes

Rectangles, Circles, Ellipses Often, we have to draw some simple shapes like rectangles, circles, and ellipses. In TikZ, it is easily done by writing exactly `rectangle`, `circle`, and `ellipse` with the appropriate dimensions after them. This is illustrated in Figure 8.7 below.

```
\begin{tikzpicture}
\draw[help lines] (-2,-3) grid (5,5);
\coordinate[label=$O$] (O) at (0,0);
\draw[Blue] (O) circle (1.5); % at origin with radius = 1.5
\coordinate (A) at (3,2);
```

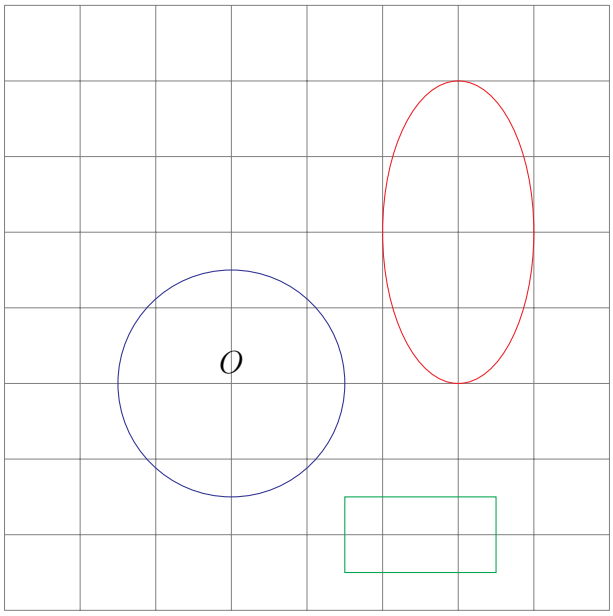


Figure 8.7: *Drawing various simple shapes in Tikz.*

```
\draw[Red] (A) ellipse (1 and 2); % with x/y-axis = 1 and 2
\draw[Green] (1.5,-1.5) rectangle (3.5,-2.5); % two opposite vertices
\end{tikzpicture}
```

There are also other shapes like `parabola`.

Rotation Another useful functionality is to rotate lines and shapes. We can use either `rotate=<degree>` or the more advanced `rotate around=<degree>:<about_coordinates>` to achieve that (the former is a special case of the latter with the reference coordinates determined implicitly, usually the origin). Their difference is demonstrated in Figure 8.8.

```
\begin{tikzpicture}
\draw[help lines] (-3,-3) grid (5,5);
\coordinate[label=$O$] (O) at (0,0);
\coordinate[label=$A$] (A) at (3,0);
\draw[Gray] (A) ellipse (2 and 1);
```

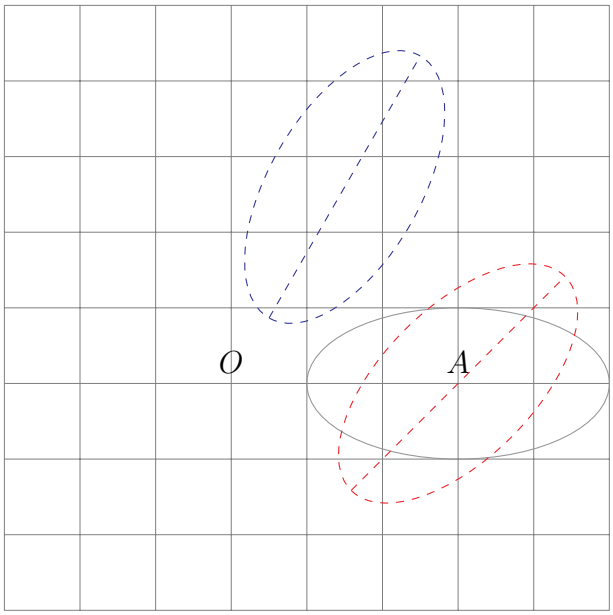


Figure 8.8: *Two different kinds of coordinate rotation in Tikz.*

```
\draw[Red, dashed, rotate=45] (A) ellipse (2 and 1);
\draw[Blue, dashed, rotate around={60:(O)}] ([rotate around={60:(O)}]
   A) ellipse (2 and 1); % an extra rotate around is needed in front
   of A
\draw[Blue, dashed, rotate=60] (1,0) -- (5,0);
\draw[Red, dashed, rotate around={45:(A)}] (1,0) -- (5,0);
\end{tikzpicture}
```

Clipping Sometimes we may want to fill a limited area within a shape clipped by some other shape. This can be done by the `clip` construct. Here we draw a Venn diagram as an illustrative example in Figure 8.9.

```
\begin{tikzpicture}
\draw[Red] (0,0) circle (2) node{A};
\begin{scope}
\clip (0,0) circle (2);
\fill[Green!25] (2.5,0) circle (2);
\end{scope}
```

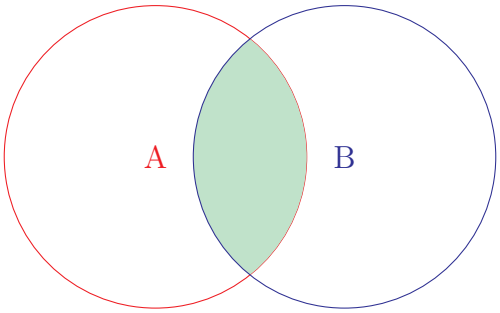


Figure 8.9: *A Venn diagram created by clipping.*

```
\draw[Blue] (2.5,0) circle (2) node{B};
\end{tikzpicture}
```

Be aware that clipping is cumulative, and we will have to limit its effect within a local `scope` so that the blue circle to the right can be drawn without being clipped wrongly.

Perpendicular Lines A convenient feature in TikZ is to draw a line perpendicular to another line without the need to do the manual calculation by loading the extra TikZ library `calc` with

```
\usetikzlibrary{calc}
```

The intersection point for that perpendicular line will then be automatically computed by it along the lines of `$(P)!(Q)!(R)$`. This is showcased in Figure 8.10 below.

```
\begin{tikzpicture}
\coordinate[label={below:$O$}] (O) at (0,0);
\node at (O) [circle,fill,inner sep=1pt] {};
\coordinate[label=$A$] (A) at (-1,2);
\coordinate[label=$B$] (B) at (4,1);
\coordinate[label=$C$] (C) at ($(A)!(O)!(B)$); % here!
\draw (A) -- (B);
\draw[dashed] (O) -- (C);
```

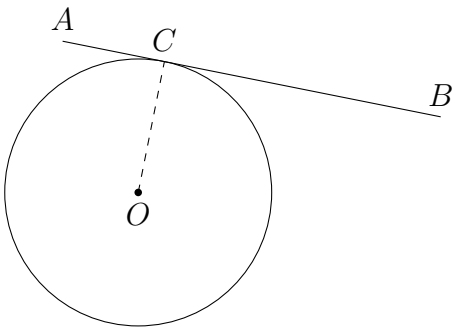


Figure 8.10: *Demonstration of drawing perpendicular lines, in addition to calculating the distance between two coordinates.*

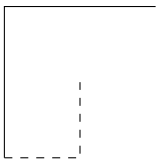


Figure 8.11: *Quick shortcuts for making perpendicular lines.*

```
\draw let \p1 = ($(C)-(O)$) in (O) circle ({veclen(\x1,\y1)});
\end{tikzpicture}
```

We further use the `calc` library with its `let ...in` syntax (using `\p1` to denote the displacement vector resulting from the `$$` calculation and `\x1,\y1` for its x/y-component), and the `veclen` function to compute its length, a.k.a. the radius of the circle tangent to the line.

Alternatively, the special cases of vertical/horizontal perpendicular lines can be done by the `|-` and `-|` syntax. They are demonstrated in Figure 8.11 above.

```
\begin{tikzpicture}
\draw (0,0) |- (2,2);
\draw[dashed] (0,0) -| (1,1);
\end{tikzpicture}
```

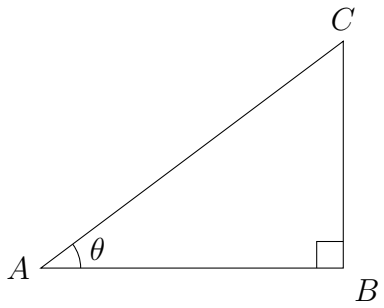


Figure 8.12: *Drawing a right-angled triangle with the angles labeled.*

Angles For geometry purposes, we often need to label angles, like in a triangle or polygon. The `angles` TikZ library is exactly made for this. Similar to above, we import it via writing

```
\usetikzlibrary{angles, quotes}
```

and then we can draw angles as some `pic` (refer to Section 8.3 later) with the construct in the form of `{angle = A--B--C}`, demonstrated in the following code for Figure 8.12:

```
\begin{tikzpicture}
\coordinate[label={left:$A$}] (A) at (-1,0);
\coordinate[label={below right:$B$}] (B) at (3,0);
\coordinate[label=$C$] (C) at (3,3);
\draw (A) -- (B) -- (C) -- cycle;
\pic [draw,angle radius=10] {right angle = A--B--C};
\pic [draw,"$\theta$",angle radius=15,angle eccentricity=1.5] {angle
   = B--A--C};
\end{tikzpicture}
```

The `angle radius` option controls the extent of the angle marking, and `angle eccentricity` determines the distance of the angle and its label (θ in this example).

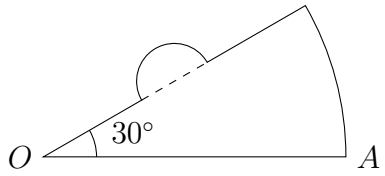


Figure 8.13: *Drawing multiple arcs in one diagram involving relative coordinates.*

Arcs Although drawing circles is not a rare task, sometimes we will need to draw just an arc. It is not hard to do so in TikZ with the `arc` shape, the syntax of which is

```
\draw (x,y) arc (start_angle:stop_angle:radius);
```

The arc will start from point `(x,y)` (it is also possible to use polar coordinates) as a part of the arc with a starting angle, stopping angle, and radius as indicated by the subsequent input values. An example is shown in Figure 8.13 above.

```
\begin{tikzpicture}
\coordinate[label={left:$O$}] (O) at (0,0);
\coordinate[label={right:$A$}] (A) at (4,0);
\draw (O) -- (A) arc (0:30:4) --++ (-150:1.5) arc (30:210:0.5)
   coordinate[at end] (B) -- cycle;
\draw[dashed] (B) --++ (30:1);
\pic[draw,"$30^\circ$",angle radius=20,angle eccentricity=1.75] {
   angle = A--O--B};
\end{tikzpicture}
```

8.2 Advanced Controls on Paths

8.2.1 Curves

Bézier Control Curves Up until now, we have been drawing only straight lines or segments. A reasonable expectation is to go one step further and construct

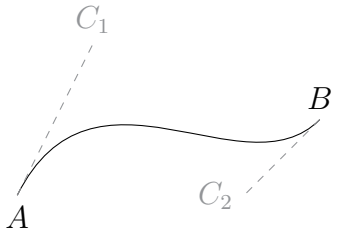


Figure 8.14: *The anatomy of a Bézier control curve.*

curved paths. In TikZ, it is implemented as *Bézier control curves* that take one or two control points, with either one of the following syntaxes:

```
\draw <starting_coords> .. controls <control_coords> .. <end_coords>;
\draw <starting_coords> .. controls <control_coords_1> and <control_
   coords_2> .. <end_coords>;
```

A schematic diagram is given as Figure 8.14 above, and the code to produce that example is

```
\begin{tikzpicture}
\coordinate[label={below:$A$}] (A) at (-1,0);
\coordinate[label=$B$] (B) at (3,1);
\coordinate[label={[Gray]$C_1$}] (C1) at (0,2);
\coordinate[label={[Gray]left:$C_2$}] (C2) at (2,0);
\draw (A) .. controls (C1) and (C2) .. (B);
\draw[dashed, Gray] (A) -- (C1);
\draw[dashed, Gray] (B) -- (C2);
\end{tikzpicture}
```

In and Out Angles An alternative way to draw a curve is to use the `to` operation plus the `in=<degree>, out=<degree>` construct. It is not difficult to guess that the `in` and `out` options represent the direction of the incoming/outgoing ray as angles (relative to the x-axis). This is demonstrated in Figure 8.15 below.

```
\begin{tikzpicture}
\coordinate[label={below:$A$}] (A) at (-1,-1);
\coordinate[label={below right:$B$}] (B) at (2,1);
```

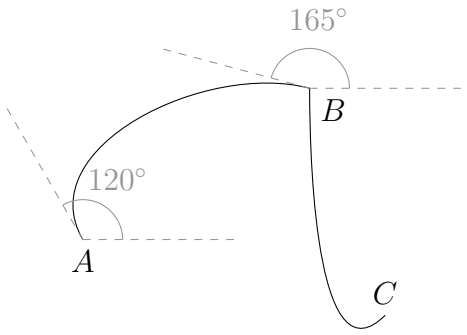


Figure 8.15: *Another way to draw control curves specified by angles.*

```
\coordinate[label=$C$] (C) at (3,-2);
\draw (A) to[in=165, out=120] (B) to [in=225, out=-90] (C);
\draw[dashed, Gray] (A) --++ (120:2) coordinate[at end] (Aa);
\draw[dashed, Gray] (A) --++ (0:2) coordinate[at end] (X1);
\pic[draw,"$120^\circ$",Gray,angle radius=15,angle eccentricity=1.75]
    {angle = X1--A--Aa};
\draw[dashed, Gray] (B) --++ (165:2) coordinate[at end] (Ba);
\draw[dashed, Gray] (B) --++ (0:2) coordinate[at end] (X2);
\pic[draw,"$165^\circ$",Gray,angle radius=15,angle eccentricity=1.75]
    {angle = X2--B--Ba};
\end{tikzpicture}
```

Intersection It is handy if we can mark the intersection point(s) of two different curves. This can be delegated to the TikZ library `intersections`. To use it, we need to give a name to those curves with the `name path` option, and then we can invoke the `name intersections` option that refers to the intersection points as `(intersection-<number>)`. For instance, Figure 8.16 below can be produced by

```
\begin{tikzpicture}
\draw[name path=myellipse, rotate=30] (0,0) ellipse (2 and 1);
\draw[name path=mycurve] (-2,1) to[in=120, out=-45] (2,0) to[in=-90,
   out=-60] (0.5,-0.5);
```

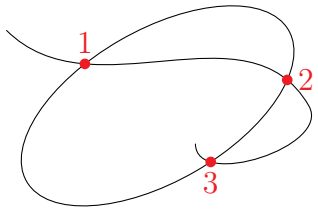


Figure 8.16: *Labeling intersection points between an ellipse and an arbitrary curve.*

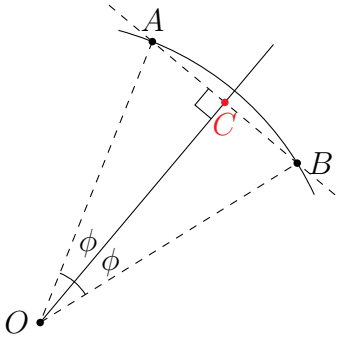


Figure 8.17: *The "Bow" diagram for Exercise 8.2.*

```
\fill[Red, name intersections={of=myellipse and mycurve}]
    (intersection-1) circle (2pt) node[above]{1}
    (intersection-2) circle (2pt) node[right]{2}
    (intersection-3) circle (2pt) node[below]{3};
\end{tikzpicture}
```

Exercise(s)

8.2) Try to reproduce the (essence of) geometry in Figure 8.17. The `shorten >=<length>` (the space is needed!) option may be useful.

8.2.2 Decorations

Decorations/Morphing An interesting effect that can be applied to curves is decorations (or morphing), generating variations along them. This is done by

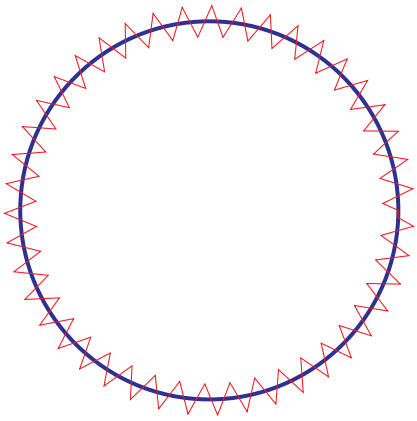


Figure 8.18: *A circle decorated by the zigzag effect.*

loading the TikZ library `decorations.pathmorphing`. The simplest usage is via `decorate, decoration=<shape>` that applies the morphing to the entire path, illustrated by Figure 8.18.

```
\begin{tikzpicture}
\coordinate (O) at (0,0);
\draw[Blue,line width=1.5] (O) circle (2.5);
\draw[Red,decorate,decoration={zigzag,segment length=2ex,amplitude
   =0.5em}] (O) circle (2.5);
\end{tikzpicture}
```

Notice that we have also passed some other options to adjust the shape of the zigzagging line.

Decorating Subpaths The previous syntax will decorate the entire path. If we want to apply the effect only on some parts of it, then we can put the `decorate` statement to enclose each of them correspondingly. Figure 8.19 is shown below as an example.

```
\begin{tikzpicture}
\draw decorate[decoration=saw] {(0,0) -- (2,1)} -- (4,-1) decorate[
   decoration={coil,aspect=1.5}] {-- (6,0)};
\end{tikzpicture}
```

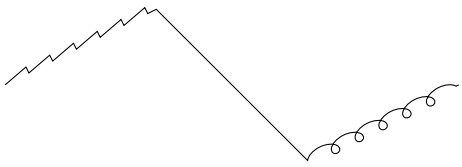


Figure 8.19: *Different decorations on individual segments.*

Figure 8.20: *Fine-tuning a decoration of crosses.*

Positioning and Extent of Decorations Furthermore, we can fine-tune the positioning, as well as the starting/ending points of a decoration. This is achieved by supplying the `raise` (displacement across the path), `pre length` (starts after), and `post length` (ends before) options, demonstrated by Figure 8.20 above.

```
\begin{tikzpicture}
\draw[decoration={pre length=9mm,post length=12mm,raise=-3mm,crosses
   }] decorate{(0,0) -- (5,1)}; % requires the extra library
   decorations.shapes too for the cross symbols
\end{tikzpicture}
```

There are many more possible choices for decorations; Unfortunately, we don't have the scope to include all of them.

8.2.3 Arrows

Arrow Tips To draw arrows, we need to load the `arrows.meta` TikZ library. Then we can specify the type of arrow tip(s) during a `\draw` command. The syntax is easier to understand by directly looking at the examples in Figure 8.21.

```
\begin{tikzpicture}
\draw[->] (0,0) --++ (3,0);
```

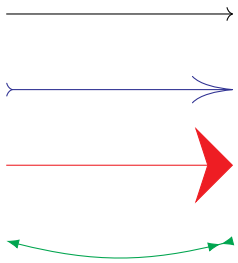


Figure 8.21: *Different types of arrow tips and related options.*

```
\draw[Blue, >-{>[length=3ex, width=2ex]}] (0,-1) --++ (3,0);
\draw[Red, -{Stealth[scale=3, angle'=90]}] (0,-2) --++ (3,0);
\draw[Green, {Latex}-{Latex[]Latex[reversed]}] (0,-3) to[out=-15,in
   =-165]++ (3,0);
\end{tikzpicture}
```

The two most frequently used named arrow tips are `Stealth` and `Latex`. Aside from `length`, `width`, `scale`, and `angle'` (remember the `'`!), there are many more keys like `inset`, `slant`, `left`, and `right`, etc.

It is also possible to set the global arrow style using `\tikzset`, like `\tikzset{>={Stealth}}`, which changes the type for all arrow tips to `Stealth`. Or, we can do it for an individual path by moving `>={<arrow_type>}` inside the corresponding `\draw` option.

Arrow in the Middle More often than not, we would like to put the arrow in the middle of a line. We can utilize the `decorations.markings` TikZ library for that. An example is given by Figure 8.22 below.

```
\begin{tikzpicture}
\draw[postaction={decorate}, decoration={markings,
      mark=at position 0.35 with {\arrow{Latex[Red,scale=2]}},
      mark=at position 0.5 with {\arrow{Stealth[Blue,scale=2.5]}},
      mark=at position 0.8 with {\arrow{Latex[Green,scale=3]}}}]
      (0,1) .. controls (1,-3) and (3,1) .. (5,-2);
\end{tikzpicture}
```

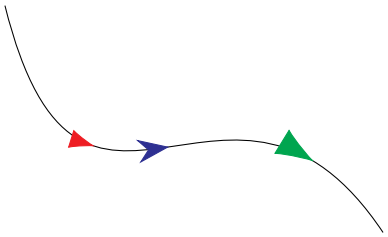


Figure 8.22: *Marking multiple arrow tips along a curve.*

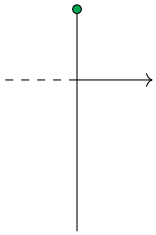


Figure 8.23: *Branching arrows from an intermediate point.*

We need to first supply the `markings` keyword to the decoration option, then we can add the arrow marks as `\arrow{<arrow_type>}` at the corresponding relative positions along the curve. A new thing is that the `decorate` keyword is now placed in the `postaction` option, which indicates that the decorations are laid on the original curve that will be kept.

Edges As an extra note, a handy functionality is to draw branching paths using `edge` while staying on the main path afterwards. This is demonstrated by Figure 8.23 above.

Exercise(s)

8.3) Draw the pendulum diagram in Figure 8.24.

8.4) Try to reproduce the closed integration path in Figure 8.25.

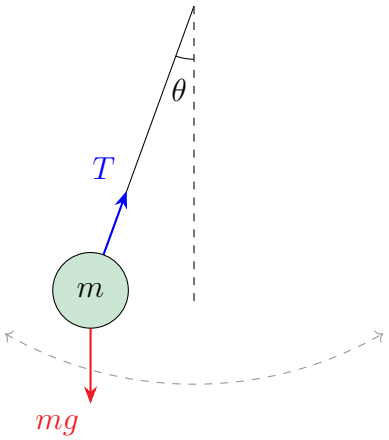


Figure 8.24: *The pendulum schematic for Exercise 8.3.*

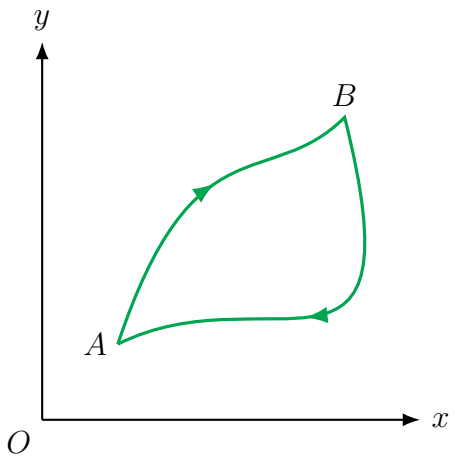


Figure 8.25: *The integration path for Exercise 8.4.*

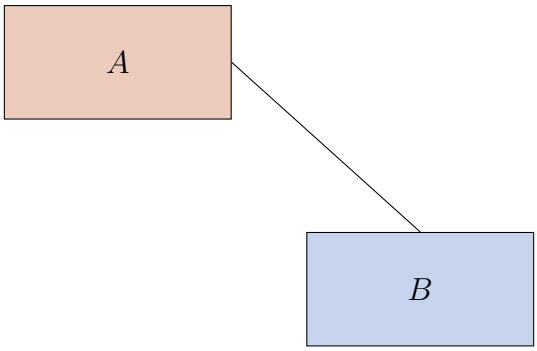


Figure 8.26: *Two styled rectangle nodes.*

8.3 Styles and Pics

(mix color)

Styles and Nodes It is convenient if we can repeatedly apply some style to similar objects in TikZ. This can be done by providing the name and attributes of that style at the beginning of the `tikzpicture` using the `/.` syntax. Figure 8.26 here demonstrates the usage.

```
\begin{tikzpicture}[myrectangle/.style={rectangle, minimum width=3cm,
    minimum height=1.5cm, draw=black, fill=Mahogany!20}]
\node[myrectangle] (myA) at (0,0) {$A$}; % using style defined above
\node[rectangle, minimum width=3cm, minimum height=1.5cm, draw=black,
fill=RoyalBlue!20] (myB) at (4,-3) {$B$}; % equivalent syntax except
   the fill color
\draw (myA.east) -- (myB.north);
\end{tikzpicture}
```

Here we have created two nodes that are in the shape of a rectangle. The `minimum width` and `minimum height` keys work exactly as their name suggest and maintain the extent of the rectangles beyond the node text. We additionally draw a line connecting the two nodes with the anchors (at where the two ends of the line are fixed) stated as directions (optional).

Pics - Small Pictures Similarly, it will be handy if we can insert and reuse the same piece of drawing that is needed many times. This is known as a `pic` (small picture) in TikZ, and we have been using this feature for labeling angles. To define a `pic`, we do it like it is a style by `<pic_name>/.pic={<drawing_code>}`. Then we can append the `pic` after a path. This is illustrated by the damper defined for the mechanical system (Reference: TEX StackExchange 476076) in the subsequent Figure 8.27.

```
\begin{tikzpicture}
    % Styles
    [dampic/.pic={
    \fill[pgftransparent!0] (-0.1,-0.3) rectangle (0.3,0.3);
    \draw (-0.3,0.3) -| (0.3,-0.3) -- (-0.3,-0.3);
    \draw[line width=1mm] (-0.1,-0.3) -- (-0.1,0.3);},
    spring/.style={decorate,decoration={zigzag,pre length=0.4cm,post
       length=0.4cm,segment length=2mm,amplitude=3mm}},
    mass/.style={rectangle,minimum height=1.6cm, minimum width=2.4cm,
        draw=black, fill=brown!50},
    ground/.style={fill,pattern=north east lines,draw=none}]
% Drawing
\node[mass] (mass1) at (0,0) {$m$};
\node[mass] (mass2) at (4,0) {$m$};
\draw ($(mass1.east)+(0,0.5cm)$) -- ($(mass2.west)+(0,0.5cm)$) pic[
   midway,sloped]{dampic};
\draw[spring] ($(mass1.east)-(0,0.5cm)$) -- ($(mass2.west)-(0,0.5cm)
   $);
\node[ground,minimum width=3mm,minimum height=2.5cm] (g1) at (-3,0)
   {};
\draw (g1.north east) -- (g1.south east);
\draw ($(mass1.west)+(0,0.5cm)$) -- ($(g1.east)+(0,0.5cm)$) pic[
   midway,sloped]{dampic};
\draw[spring] ($(mass1.west)-(0,0.5cm)$) -- ($(g1.east)-(0,0.5cm)$);
\end{tikzpicture}
```

There are some other points worth mentioning. The special `pgftransparent!0` color code is an alias equivalent to the white color, and covers the original path

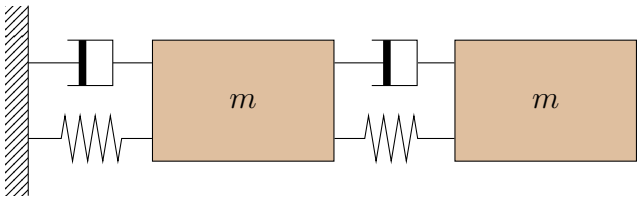


Figure 8.27: *A schematic for a mechanical mass-spring-damper system.*

under it. Then, we sketch the outline of the damper. We also need to load the `patterns` TikZ library to produce the hatching lines (`pattern=north east lines`) for the ground style. Also, we have used the `$$` syntax to carry out coordinate calculations in deriving the starting/ending points of the connecting lines.

Styles for Every Node/Path An even more convenient shortcut is to assign the same style for every node/path (of the same kind) at once. This is unsurprisingly done by providing the `every node`/`every path` name, and is readily showcased in Figure 8.28 below.

```
\begin{tikzpicture}[every node/.style={circle,black,solid,draw=black,
   fill=Red!20,minimum size=30pt},
    every rectangle node/.style={black,solid,draw=black,fill=Blue!20,
       minimum height=15pt, minimum width=30pt},
    every path/.style={Green,dashed}]
\node (A) at (0,0) {$A$};
\node (B) at (2,-1) {$B$};
\node[rectangle] (C) at (-0.5,-2) {$C$};
\node (D) at (3,-3) {$D$};
\draw (A) -- (B) -- (C) -- (D);
```

Notice that `every path` also affects node texts and lines, and we have to override them in `every node` for it to work as intended.

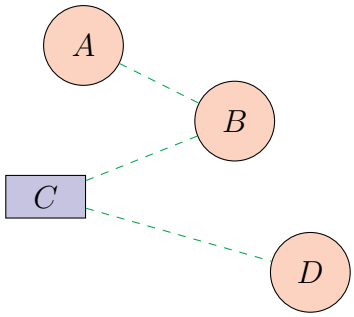


Figure 8.28: *Reusing styles for every node and path.*

Path Building We can further break down a path into the corresponding components (lines, curves, etc.) and execute certain code for each of them every time they occur. This is accomplished by the `show path construction` keyword for `decoration`, demonstrated by the contour integration plot in Figure 8.29 as follows.

```
\begin{tikzpicture}
% Defining parameters
\newcommand*{\gap}{0.3}
\newcommand*{\bigradius}{3}
\newcommand*{\littleradius}{0.7}
% Drawing
\draw[very thick,-Latex] (-1.15*\bigradius, 0) -- (1.15*\bigradius,
    0);
\draw[very thick, decorate, decoration={zigzag, segment length=0.5cm,
     amplitude=0.2cm}] (0, 0) -- (1.15*\bigradius, 0);
\draw[very thick,-Latex] (0, -1.15*\bigradius) -- (0, 1.15*\bigradius
    );
% The red contour
\draw[red, thick, postaction=decorate, decoration={show path
    construction, curveto code={
     \draw[decorate, decoration={markings, mark=at position 0.75 with
        {\arrow{Latex[Red,scale=1.25]}}}]
     (\tikzinputsegmentfirst) .. controls (\tikzinputsegmentsupporta)
        and (\tikzinputsegmentsupportb) .. (\tikzinputsegmentlast);},
        % no need to change this unless you know what you are doing
```

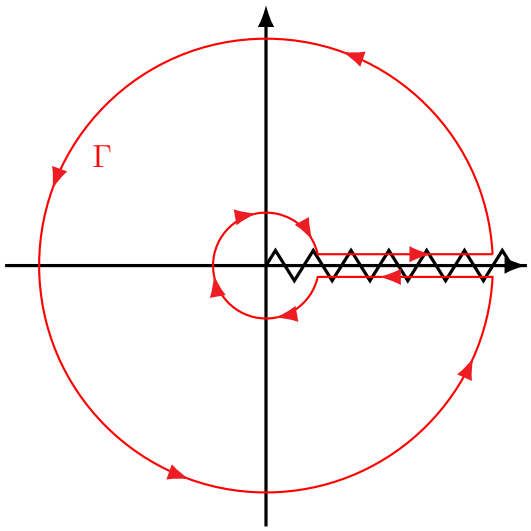


Figure 8.29: *A standard complex contour integration path with a branch cut over the positive x-axis.*

```
    lineto code={
    \draw[decorate, decoration={markings, mark=at position 0.65 with
       {\arrow{Latex[Red,scale=1.25]}}}]
    (\tikzinputsegmentfirst) -- (\tikzinputsegmentlast);} % don't
       change this too
    }]
let
\n1 = {asin(\gap/2/\bigradius)},
\n2 = {asin(\gap/2/\littleradius)}
in (\n1:\bigradius) arc (\n1:360-\n1:\bigradius) node[pos=0.4,below
   right]{$\Gamma$} -- (-\n2:\littleradius) arc (-\n2:-360+\n2:\
   littleradius) -- (\n1:\bigradius);
\end{tikzpicture}
```

The code is a bit complex, and we will have it explained step by step. As their names suggest, the `curveto code`/`lineto code` part will be applied to every curve/line. The main purpose of involving `\tikzinputsegmentfirst`, `\tikzinputsegmentlast`, `\tikzinputsegmentsupporta`, in addition to `\tikzinputsegmentsupportb`, is to replicate the curve/line internally, which

can be left untouched as a template. Then the `decorate` and `markings` parts are the same as previously to put an arrow mark along each replicated path segment. Finally, to draw the actual contour, we use `\newcommand*` and the `let ...in` syntax to store lengths as variables and compute the coordinates for the turning points (copying TEX StackExchange 103176).

8.4 Plotting Functions

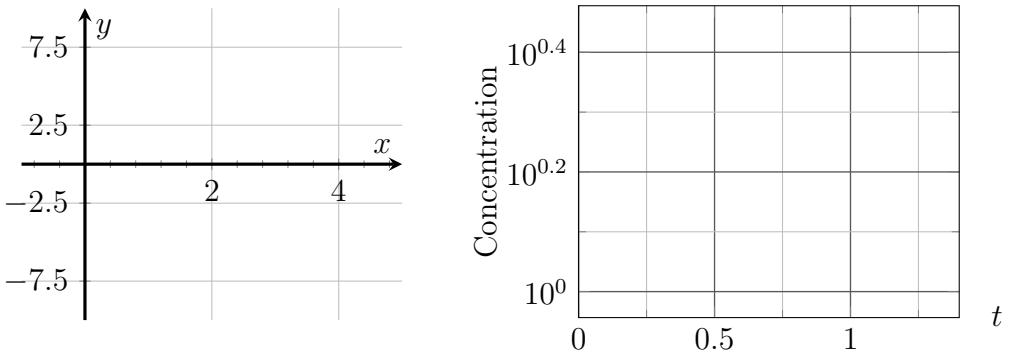
Axis Settings To plot a function or graph in TikZ, we have to configure an `axis` scope first. There are many options to customize an axis, some of which are showcased in the two examples in Figure 8.30 as follows.

```
\begin{tikzpicture}
\begin{axis}[xlabel=$x$, ylabel=$y$, xmin=-1, xmax=5, ymin=-10, ymax
   =10, ytick={-7.5,-2.5,2.5,7.5}, minor x tick num=4, grid=major,
   axis lines=center, axis line style={line width=1.2pt}, width=\
   textwidth]

\end{axis}
\end{tikzpicture}
\begin{axis}[ymode=log, xlabel=$t$, ylabel=Concentration, x label
   style={at={(axis description cs:1.1,0.2)}}, minor ytick={10^(0.1),
    10^(0.3)}, minor x tick num=1, major grid style={line width=.5pt,
   draw=black!66}, grid=both, enlarge x limits={0.4,upper}, width=\
   textwidth]

\end{axis}
```

Most of the options are not hard to comprehend, except `axis description cs:`, which refers to the position relative to the axis frame so that we can adjust the label position, and `enlarge limits`, which extends the axis limits by the amount indicated. The biggest difference between the two example axes is perhaps



(a) *A typical axis.* (b) *A semi-logarithmic axis.*

Figure 8.30: *Two example TikZ axes.*

the appearance of the axis lines, produced by the option `axis lines=center` in the first one.

Plotting Simple Functions Of course, an axis is nothing if there are no data or functions plotted on it. To add points or graphs on it, we can use the `\addplot` command. For points, we may add a list of `coordinates` after that, and use the `only marks` keyword to draw only the dots; while for functions, we can simply enter the expression in PGF format. We can control the `domain` of `x` and the number of points (`samples`) used in drawing. Both of these are demonstrated in Figure 8.31 on the next page.

```
\begin{tikzpicture}
\begin{axis}[xlabel=$T$, ylabel=$N$, title=Half-life Experiment, xmin
   =0, xmax=3, ymin=0, ymax=1, enlargelimits=0.1, legend pos=south
   west, width=0.9\textwidth, height=0.6\textwidth]
\addplot[Red, only marks]
coordinates {
(0,1) (0.2,0.86) (0.6,0.67) (0.9,0.53)
(1.4,0.37) (1.7,0.29) (2.5,0.17)};
\addplot[blue, domain=0:4, samples=100]{(1/2)^(x)} node[pos=0.6,above
   ,sloped] {$N = (1/2)^T$};
```

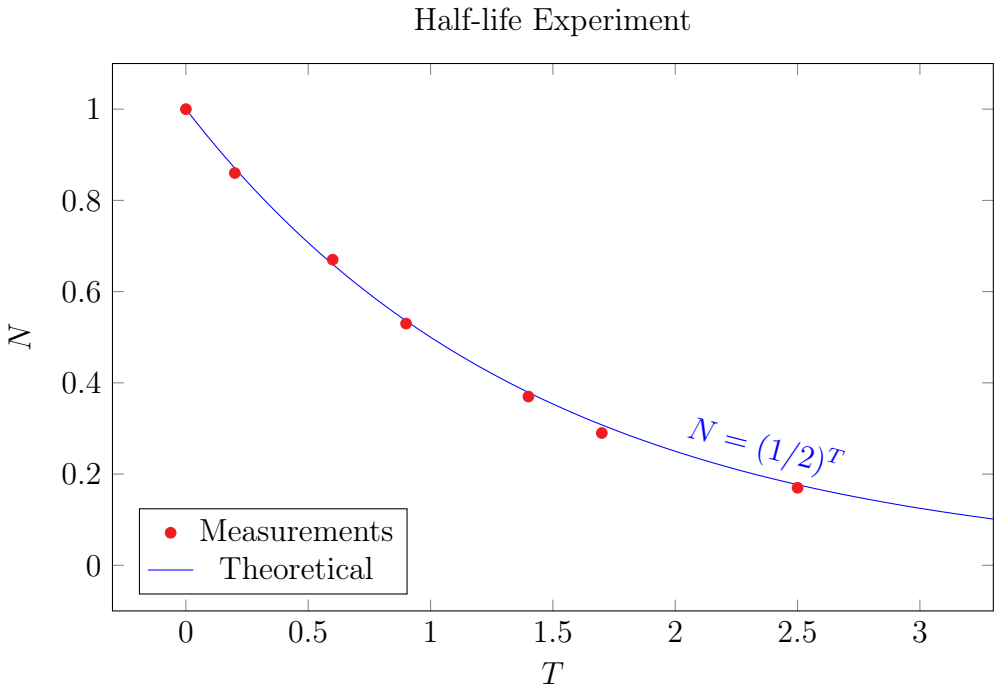


Figure 8.31: *The half-life process as an example plot.*

```
\legend{Measurements, Theoretical}
\end{axis}
```

We have also adjusted the size of the figure and made a legend.

Exercise(s)

8.5) Try to reproduce the following plot in Figure 8.32. Notice that to draw with usual TikZ commands but now inside an axis, we can call the axis coordinate system with the syntax `axis cs:<coords>`.

Parametric Equations A natural generalization of `\addplot` is to draw parametric curves where the command now treats `x` as the parameter and accepts two

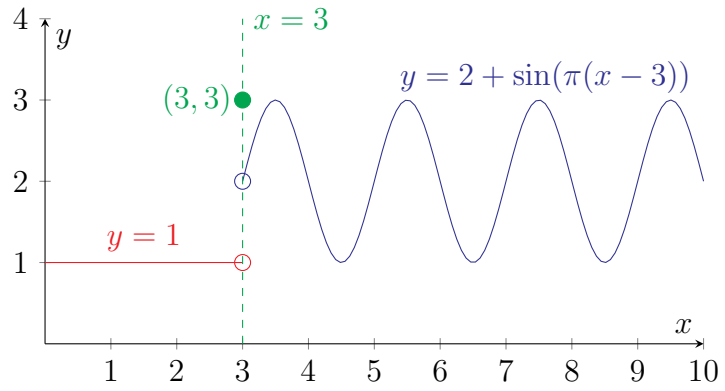


Figure 8.32: *The plot for Exercise 8.5.*

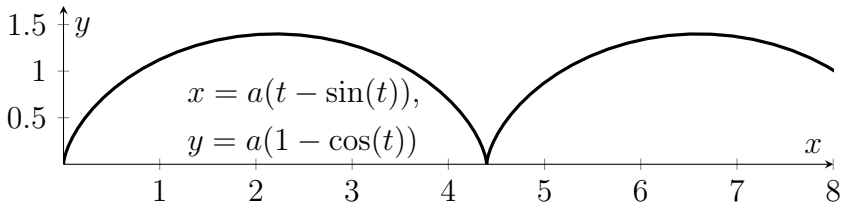


Figure 8.33: *A parametric cycloid graph.*

equations. We will borrow the famous cycloid to illustrate the usage in Figure 8.33.

```
\begin{tikzpicture}
\newcommand*{\ap}{0.7}
\begin{axis}[xlabel=$x$, ylabel=$y$, axis lines=center, xmin=0, xmax
   =8, ymin=0, ymax=1.7, width=0.8\textwidth, height=0.25\textwidth]
\addplot[very thick, domain=0:5*pi, samples=100] ({\ap*(x - sin(deg(x
   ))},{\ap*(1 - cos(deg(x)))});
\node[align=left] at (axis cs:2.5,0.5) {$x = a(t-\sin(t))$,\\ $y = a
   (1-\cos(t))$};
\end{axis}
\end{tikzpicture}
```

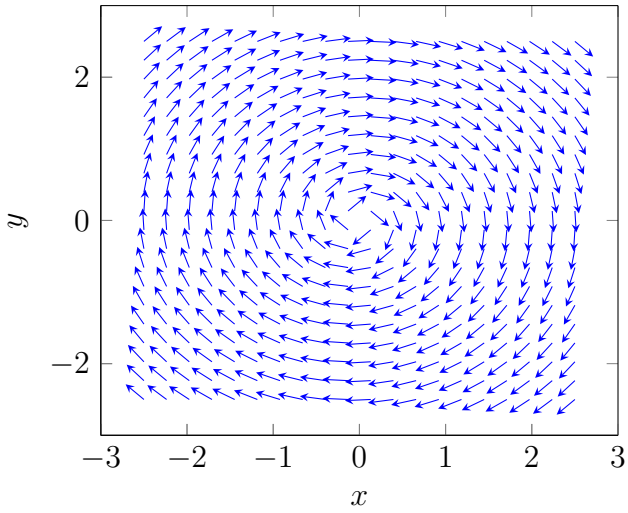


Figure 8.34: *A clockwise rotational vector field.*

Vector Fields The final topic to introduce in this chapter is about drawing a vector field using the `\addplot3` function (used for 3D plotting, more in the next chapter) with the `quiver` option. This is illustrated in Figure 8.34 here, with the vector field defined by $(y/\sqrt{x^2+y^2}, -x/\sqrt{x^2+y^2})$.

```
\begin{tikzpicture}
\begin{axis}[xlabel=$x$, ylabel=$y$, xmin=-3, xmax=3, ymin=-3, ymax
   =3, view={0}{90}]
\addplot3[blue, domain=-2.5:2.5, quiver={u={y/(x^2+y^2)^0.5}, v={-x/(
   x^2+y^2)^0.5}, scale arrows=0.3}, -stealth, samples=20] {0};
\end{axis}
\end{tikzpicture}
```

The `u` and `v` keys represent the velocities along the x/y-axes, and we have set the scale, type, and density for the arrows. Finally, `view={0}{90}` is needed for the `axis` since we have invoked the 3D plotting method and need to reset the camera to look downward overhead.

Chapter 9

Plotting with TikZ (Part II)

Introduction This chapter continues to discuss the finer details and broad applications of TikZ plotting.

9.1 Advanced Axis Options

9.1.1 Axis Scales

Axis Units We can adjust the units of the coordinate axes in a TikZ plot by specifying them at the beginning. This is readily demonstrated in the small example of Figure 9.1 below.

```
\begin{tikzpicture}[x=1.2cm, y=0.9cm]
\draw[help lines, step = {(1,1)}] (0,0) grid (4,5);
\node at (0,0) [below left] {$0$};
\foreach \ii in {1,...,4} {\node at (\ii,0) [below]{$\ii$};}
\foreach \jj in {1,...,5} {\node at (0,\jj) [left]{$\jj$};}
\draw[decorate, decoration=brace, very thick, Green] (3,5) -- (4,5)
   node [midway, above] {$1.2$ cm};
```

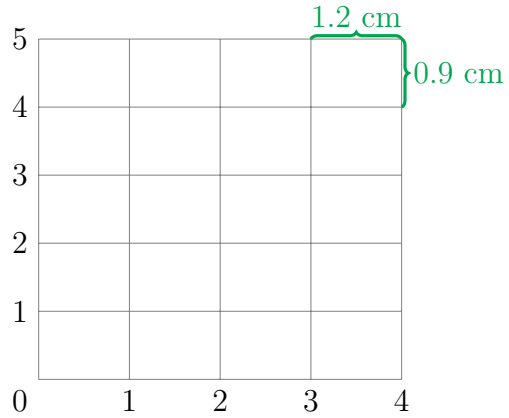


Figure 9.1: *Cartesian coordinate system transformation.*

```
\draw[decorate, decoration={brace, mirror}, very thick, Green] (4,4)
    -- (4,5) node [midway, right] {$0.9$ cm};
\end{tikzpicture}
```

Be reminded that after the scale transformation, we have to supply `step = (1,1)` so that the grid lines are drawn in the new units. We have also used the `brace` and `mirror` decorations, which are not hard to understand.

Coordinate Rotation Another simple way to transform the coordinate axes is via rotation. It is extremely straightforward and immediately done in Figure 9.2.

```
\begin{tikzpicture}[x=1.2cm, y=0.9cm, rotate=20]
\draw[help lines, step = {(1,1)}] (0,0) grid (4,5);
\end{tikzpicture}
```

9.1.2 3D Plotting

3D Coordinate Axes To initialize a 3D TikZ plot, the easiest automatic way is to write the drawing code just as usual, but now with the coordinates being 3D. We can also control the axis scales, like in the 2D scenario, where we can specify

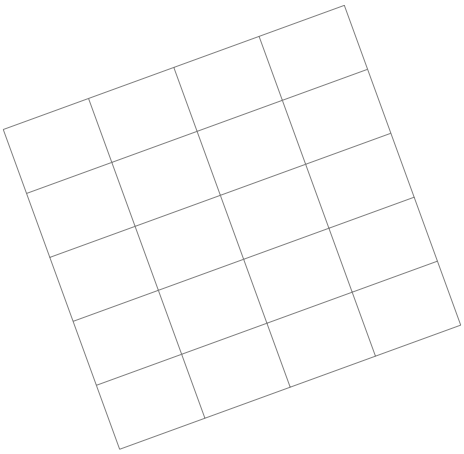


Figure 9.2: *Same as Figure 9.1 but with an anti-clockwise rotation applied to the frame.*

the direction of each axis displayed on the page. This is demonstrated in Figure 9.3.

```
\begin{tikzpicture}[x={(-1cm, -1.5cm)}, y={(1.5cm, -0.75cm)}, z={(0cm
    , 1.8cm)}, every path/.append style={>=Latex}] % Axis settings
\draw [->] (0,0,0) -- (3,0,0) node [below left] {$x$};
\draw [->] (0,0,0) -- (0,3,0) node [below right] {$y$};
\draw [->] (0,0,0) -- (0,0,3) node [above] {$z$};
% Unit vectors
\draw [->, very thick, Red] (0,0,0) -- (1,0,0) node [left] {$\hat{\
    imath} = (1,0,0)^T$};
\draw [->, very thick, Red] (0,0,0) -- (0,1,0) node [above right,
    midway] {$\hat{\jmath} = (0,1,0)^T$} ;
\draw [->, very thick, Red] (0,0,0) -- (0,0,1) node [left] {$\hat{k}
    = (0,0,1)^T$};
% The blue vector
\draw [Gray, dashed] (1,2,0) -- (1,0,0) node[below, midway, sloped]{$
    y=2$};
\draw [Gray, dashed] (1,2,0) -- (0,2,0) node[below, midway, sloped]{$
    x=1$};
\draw [Gray, dashed] (1,2,0) -- (1,2,2.5) node[midway, right]{$z
    =2.5$};
```

```
\draw [->, blue, line width=1.2] (0,0,0) -- (1,2,2.5) node [right]
    {$\vec{v} = (1,2,2.5)^T$};
\end{tikzpicture}
```

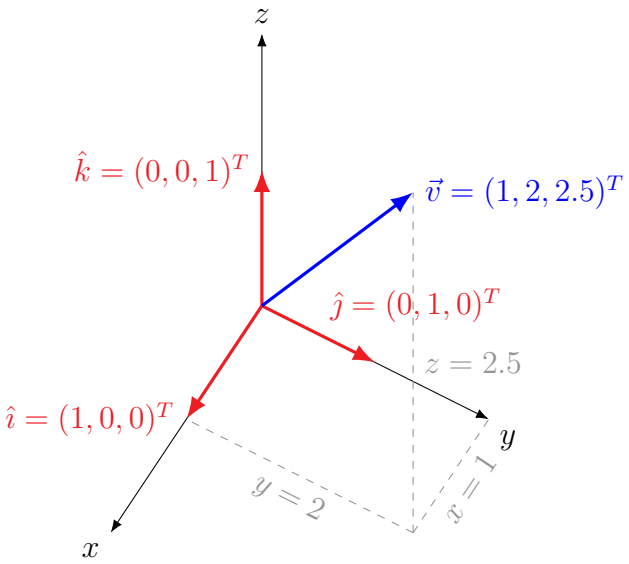


Figure 9.3: *A vector in three-dimensional space.*

Notice that we have utilized the `append style` function to add the requirement that all arrow tips will use the `LaTeX` type on top of the default behavior.

Surface Plots Moving from 2D to 3D, we may now want to draw surface plots instead of just curves. This can be done by declaring a normal axis scope and using the `\addplot3` command with the `surf` keyword on the level equation. Figure 9.4 below serves as an example.

```
\begin{tikzpicture}
\begin{axis}[grid=major,colormap/viridis,zmin=-0.25]
\addplot3[surf,samples=20,domain=-2:2,y domain=-1:1] {exp(-(x^2+y^2))
    };
\end{axis}
\end{tikzpicture}
```

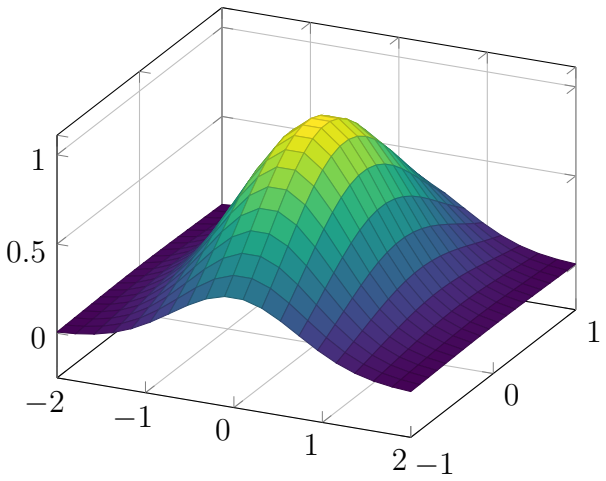


Figure 9.4: *A 2D Gaussian surface plot.*

We can add a mesh grid and choose the color map (viridis here) at the start of the axis environment. It is also possible to provide a `colorbar`. To draw a contour plot instead, we simply replace `surf` by `contour gnuplot`[1] and use the `view={0}{90}` trick introduced at the end of the last chapter.

3D Spheres and Arcs To construct a 3D sphere and draw arcs on it, we need to import the `tikz-3dplot` package (requires `\usepackage{tikz-3dplot}` [2] this time). Drawing only the sphere is not hard (plus the `ball color` option). Meanwhile, drawing the arcs requires us to first locate the center as well as the two ends of each of them via `\tdplotdefinepoints`[3], and then actually execute that with `\tdplotdrawpolytopearc`. The whole procedure is illustrated by Figure 9.5.

```
\tdplotsetmaincoords{70}{110}
\begin{tikzpicture}[scale=2.5,tdplot_main_coords,rotate=15]
```

[1] Or `contour lua` if you are using LuaLaTeX.
[2] A positive side effect is that it also loads the smaller `3d` TikZ library, which allows the user to write in cylindrical/spherical coordinates.
[3] However, this command only supports Cartesian coordinates.

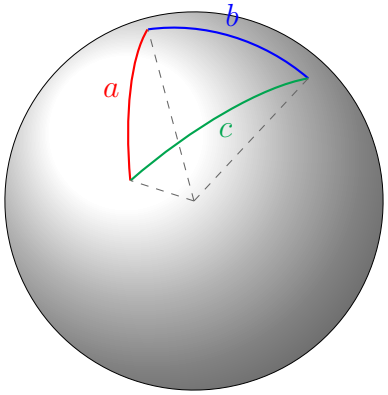


Figure 9.5: *Drawing a sphere with a spherical triangle on it.*

```
\coordinate (O) at (0,0,0);
\draw [ball color=white,very thin] (O) circle (1cm);
\tdplotdefinepoints(0,0,0)(0,0,1)(3^0.5/2,0,0.5)
\tdplotdrawpolytopearc[thick, red]{1}{left, red}{$a$}
\tdplotdefinepoints(0,0,0)(0,0,1)(0,0.8,0.6)
\tdplotdrawpolytopearc[thick, blue]{1}{above, blue}{$b$}
\tdplotdefinepoints(0,0,0)(0,0.8,0.6)(3^0.5/2,0,0.5)
\tdplotdrawpolytopearc[thick, Green]{1}{below, Green}{$c$}
\draw[dashed, color=black!60] (O) -- (0,0,1) node(C){};
\draw[dashed, color=black!60] (O) -- (3^0.5/2,0,0.5) node(B){};
\draw[dashed, color=black!60] (O) -- (0,0.8,0.6) node(A){};
\end{tikzpicture}
```

Note that we also call `\tdplotsetmaincoords` and deploy `tdplot_main_coords` to adjust the viewing angle. Furthermore, we have used the `rotate` option as a workaround to tune the orientation.

9.2 Data Visualization

Line Charts There are many other types of plots that we can make with TikZ. The most basic one will probably be line charts, and we will use the Mauna Loa CO_2 concentration data (downloaded from https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv) as an example. As before, the `\addplot` command is called, and it can accept and read a `.csv` (comma-separated values) file. We will supply the one downloaded from the link above, while providing the respective header names for the `x` and `y` axes. The result is shown in Figure 9.6 above.

```
\begin{tikzpicture}
\begin{axis}[width=0.95\textwidth, height=0.65\textwidth, x tick
    label style={rotate=45,/pgf/number format/.cd,set thousands
    separator={}}, enlarge x limits={abs=2}, xlabel=year, ylabel=$\
    text{CO}_2$ concentration (ppm), title=Mauna Loa Observatory
    Measurement, xlabel style={yshift=-15pt}]
\addplot[Green, mark=*] table [x=year, y=mean, col sep=comma] {co2_
    annmean_mlo.csv};
\addplot[Green, dashed, domain=2000:2025] {400} node[below left
    ]{$400$ ppm};
\end{axis}
\end{tikzpicture}
```

Some noticeable points include that if we change the graph color, we have to explicitly append `mark=*` to get back the dots along the curve. Removing it will produce only the curve. A reference horizontal dashed line is drawn too by simply providing the constant. We enforce `enlarge x limits` to take `abs=2` so that the x-axis extends by 2 years. We also rotate the x-axis tick labels by 45° and use some options to format out the `,` originally in the year numbers. (Try removing them to see what happens!) Accounting for that, we also shift the x-axis label downward by 15 pt.

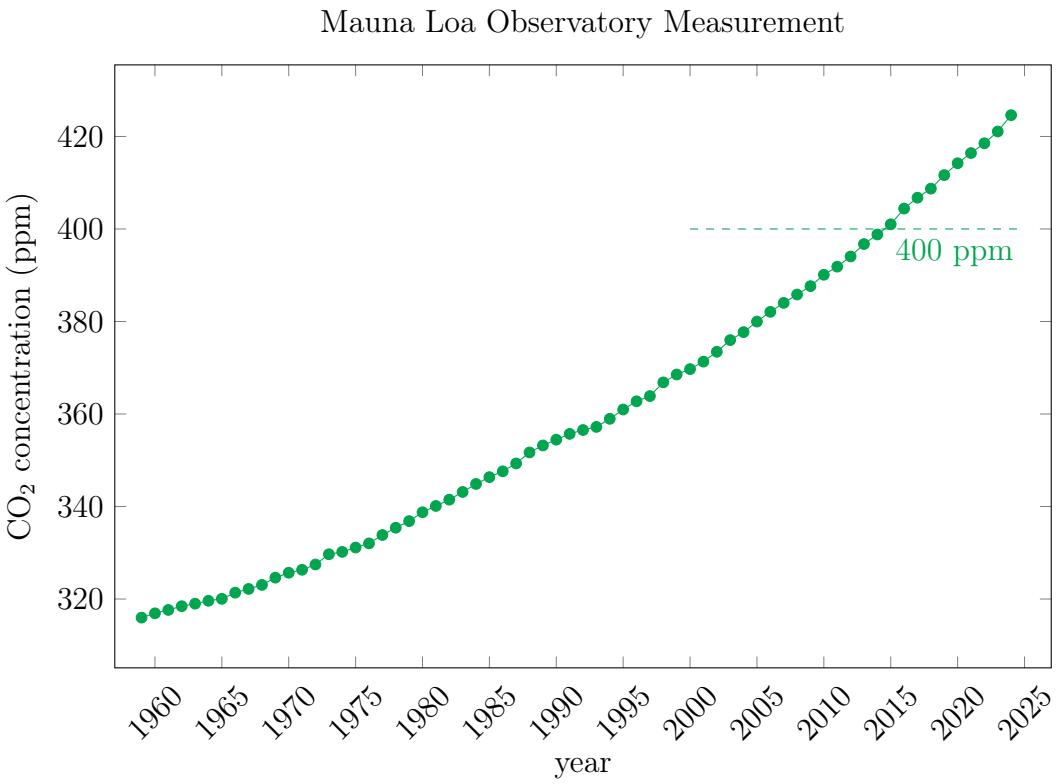


Figure 9.6: *The time-series of CO_2 concentration recorded at Mauna Loa Observatory from 1959 to 2024.*

Bar Plots Bar plots are another commonly seen plot type. To construct a bar plot, we either declare `xbar` or `ybar` in the `axis` option at the start and call the `\addplot` command for each set of bars, or put the `xbar`/`ybar` keyword directly after `\addplot`. A small example is given as Figure 9.7.

```
\begin{tikzpicture}
\begin{axis}[title=Class Subject Performance,
    ybar,
    enlargelimits=0.15,
    legend style={at={(0.5,-0.15)},anchor=north,legend columns=-1},
    ylabel=Score,
    symbolic x coords={Class A,Class B,Class C},
    xtick=data,
    nodes near coords,
    nodes near coords align=vertical]
\addplot coordinates {(Class A,75) (Class B,81) (Class C,73)};
\addplot coordinates {(Class A,88) (Class B,80) (Class C,65)};
\addplot coordinates {(Class A,86) (Class B,82) (Class C,70)};
\legend{Language,Mathematics,Science}
\end{axis}
\end{tikzpicture}
```

Here we use the `symbolic x coords` option to tell the `axis` that the x-coordinates are now strings, and `xtick=data` guarantees that the tick labels will be those appearing in the input data. `nodes near coords` produces the numbers above the bars. Last but not least, we have tweaked the style for the legend, particularly `legend columns=-1` to arrange the entries horizontally.

9.3 Referencing between TikZ Pictures

Remembering Names and Overlay Sometimes we need to combine multiple TikZ pictures, and share name labels between them so that they can refer to objects in each other. We can enable this functionality by setting the `remember`

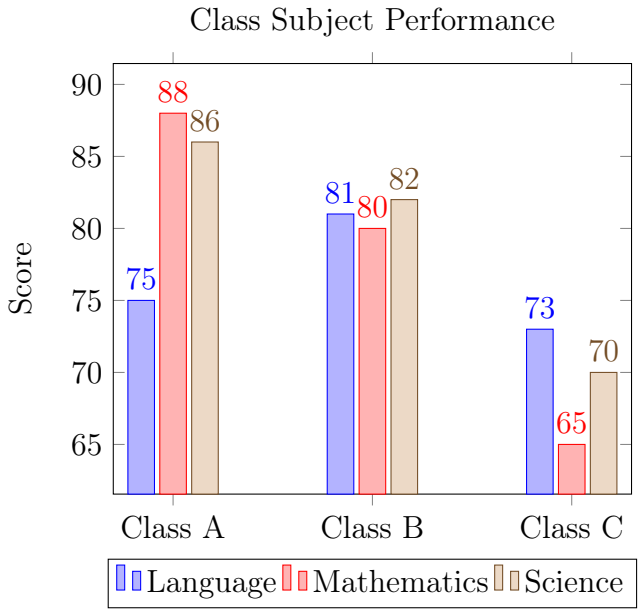


Figure 9.7: *A toy dataset about the scores of three classes in three different subjects.*

`picture` option for them. This is demonstrated by the rather lengthy example of different phase portraits in Figure 9.8.

```
\tikzset{decorated arrows/.style={postaction=decorate,
         decoration={markings,mark={at position 0.5 with {\arrow{
            stealth}}}}}}
% Main diagram
\begin{tikzpicture}[remember picture]
\draw[thick, ->] (-6,0) -- (6,0) node[right]{$\tr(A)$};
\draw[thick, ->] (0,-6) -- (0,6) node[above]{$\det(A)$};
\node[below left] (O) at (0,0) {$O$};
\draw[gray, thick] plot[domain=-4.5:4.5,samples=100] (\x, {(\x)^2/4})
   ;
\node[gray, rotate=-60] at (-3.5,2.5) {$\Delta = \tr(A)^2 - 4\det(A)
   = 0$};
\node[gray, rotate=60] at (3,3) {Complex};
\node[gray, rotate=60] at (3.5,2.5) {Real};
\node[anchor=center, Green, rotate=90] at (-6, 5) {\large Stable};
\node[anchor=center, Red, rotate=-90] at (6, 5) {\large Unstable};
```

```
% Defining the named coordinates
\coordinate (saddle) at (0,-3);
\node[anchor=center, blue] at (0,-5) {Saddle Point};
\coordinate (center) at (0,3);
\node[anchor=center, blue] at (0, 1) {Center};
... % omitted other cases
\end{tikzpicture}
% Saddle point, notice the "at" key
\begin{tikzpicture}[remember picture, overlay]
\begin{axis}[at=(saddle), anchor=center, scale=0.5, xmin=-2, xmax=2,
    ymin=-2, ymax=2, axis lines=center, hide axis]
\addplot[domain=-1:1.5,samples=50,blue,decorated arrows] ({0.5*e^(x)
    + 0.5*e^(-x)},{0.5*e^(x) - 1*e^(-x)});
\addplot[domain=-1.5:2,samples=50,blue,decorated arrows] ({0.3*e^(x)
    + 0.3*e^(-x)},{0.3*e^(x) - 0.6*e^(-x)});
\addplot[domain=-1:1.5,samples=50,blue,decorated arrows] ({0.5*e^(x)
    - 0.5*e^(-x)},{0.5*e^(x) + 1*e^(-x)});
\addplot[domain=-1.5:2,samples=50,blue,decorated arrows] ({0.3*e^(x)
    - 0.3*e^(-x)},{0.3*e^(x) + 0.6*e^(-x)});
...
\end{axis}
\end{tikzpicture}
% Center
\begin{tikzpicture}[remember picture, overlay]
\begin{axis}[at=(center), anchor=center, scale=0.5, xmin=-2, xmax=2,
    ymin=-2, ymax=2, axis lines=center, hide axis]
\addplot[domain=0:360,samples=50,blue,decorated arrows] ({1*(cos(x))
    + 0.8*(sin(x))},{-1*(cos(x)) + 1*(sin(x))});
\addplot[domain=0:360,samples=50,blue,decorated arrows] ({2/3*(cos(x)
    ) + 1.6/3*(sin(x))},{-2/3*(cos(x)) + 2/3*(sin(x))});
\addplot[domain=0:360,samples=50,blue,decorated arrows] ({1/3*(cos(x)
    ) + 0.8/3*(sin(x))},{-1/3*(cos(x)) + 1/3*(sin(x))});
\end{axis}
\end{tikzpicture}
...
```

(The full code can be checked from the source file.) In the main procedure, the

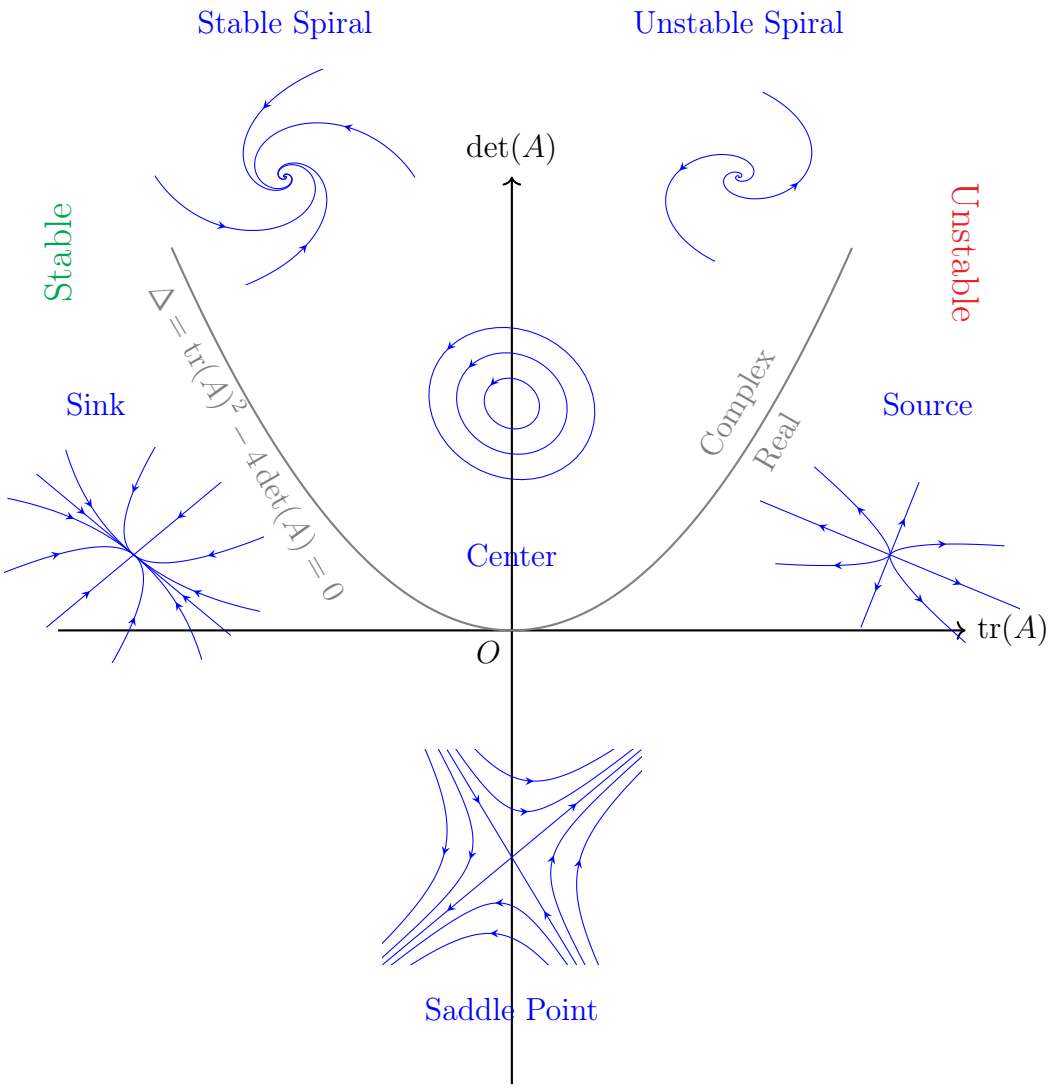


Figure 9.8: *Different types of equilibrium points and their representative phase portraits in two-dimensional dynamical systems.*

underlying axes and parabola are drawn, and the coordinates at which each phase portrait will be placed are marked and named. Then, a smaller TikZ picture is initialized for each flow type over the main diagram with the `overlay` option. The axis containing the phase lines is then centered at the marked coordinates by filling the `at` option with the shared name.

9.4 Matrices in TikZ

Matrices of Nodes To generate a matrix in TikZ that can be labeled, we can use the `\matrix` command with the `matrix of math nodes` option. This is illustrated by Figure 9.9 below.

```
\begin{tikzpicture}[remember picture]
\node (A) {$A=$};
\matrix(mymatrix)[matrix of math nodes, left delimiter={[}, right
   delimiter={]}, right=1.5ex of A, row sep=6pt, column sep=6pt,
   inner sep=1.5pt, nodes={text width=16pt, align=center}]
{2 & 1 & 7 & \frac{8}{9} \\
\mathcolor{red}{5} & -\frac{1}{3} & 5 & 0 \\
-3 & \frac{4}{11} & 6 & -\frac{1}{6} \\};
\begin{scope}[on background layer] % the blue shadings
\draw [Green, dashed, fill=blue!12, fill opacity=0.5] ($(mymatrix
   -1-1.north west)+(-0.5ex,1ex)$) rectangle ($(mymatrix-3-1.south
   east)+(0.5ex,-1ex)$);
\draw [Green, dashed, fill=blue!12, fill opacity=0.5] ($(mymatrix
   -2-1.north west)+(-0.5ex,1ex)$) rectangle ($(mymatrix-2-4.south
   east)+(0.5ex,-1ex)$);
\end{scope}
\node at (mymatrix-1-1) [above, font=\footnotesize, yshift=10, Green]
    {Col 1};
\node at (mymatrix-2-4) [right, font=\footnotesize, xshift=15, Green]
    {Row 2};
\draw[Red, -Stealth] (pic cs:Ap) to[in=-45, out=-180] (mymatrix-2-1);
    % to the tikzmark
```

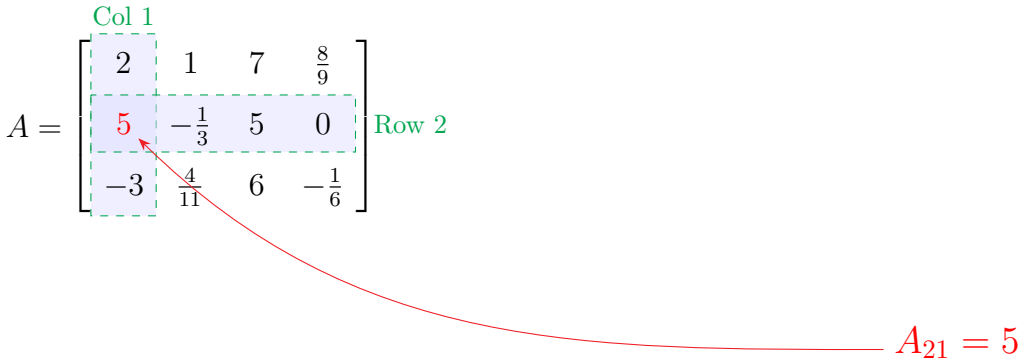


Figure 9.9: *Typesetting and labeling a TikZ matrix.*

```
\end{tikzpicture}
\large$\mathcolor{red}{\tikzmark{Ap}A_{21} = 5}$
```

Particularly, the `left/right delimiter` settings enclose the nodes with the square brackets, and the `right of` syntax puts the matrix to the right of the `A=` part. The positioning of the nodes can be controlled by changing `row sep`, `column sep`, and `inner sep`.

We have also used two additional TikZ libraries. The first one is `backgrounds` to draw blue shadings along the targeted row/column in the background via a `on background layer` scope. The second one is `tikzmark` so that we can place a corresponding `tikzmark` tag at some other location outside the TikZ picture, which can then refer to this tag as `pic cs:<name>` with the help of `remember picture` just introduced before.

9.5 Flow Charts

Flow Charts with Shapes A general TikZ application is to produce flow charts for various processes. This is demonstrated by the bisection algorithm example

in Figure 9.10. We will need to load the `shapes.geometric` TikZ library for defining the shapes representing different components.

```
\begin{tikzpicture}[startend/.style={rectangle, rounded corners,
    minimum width=3cm, minimum height=1cm, draw=black},
io/.style={trapezium, trapezium left angle=70, trapezium right angle
    =110, minimum width=4cm, minimum height=1cm, draw=black},
process/.style={rectangle, minimum width=3cm, minimum height=1cm,
    draw=black},
decision/.style={diamond, minimum width=5cm, minimum height=1cm, draw
    =black}]
\node[startend] (start) {Start};
\node[io,below=1.2cm of start,align=center] (input) {$f(x)$, $a < b$,
     \\ $f(a)<0$, $f(b)>0$};
\draw[->] (start) -- (input);
\node[process,below=1.2cm of input] (bisect) {$c = (a+b)/2$};
\draw[->] (input) -- (bisect) node[midway] (back) {};
\node[decision,below=1.2cm of bisect] (check1) {$f(c)>0$};
\draw[->] (bisect) -- (check1);
\node[process,below=1.2cm of check1] (moveb) {$b = c$};
\draw[->] (check1) -- (moveb) node[midway,right]{Yes};
\node[process,left=1 of moveb] (movea) {$a = c$};
\draw[->] (check1) -- (check1 -| movea) -- (movea) node[midway,left]{
    No};
\node[decision,below=1.2cm of moveb] (check2) {$\abs{f(c)-0} < \
    varepsilon$};
\draw[->] (moveb) -- (check2);
\draw[->] (movea) -- (check2 -| movea) -- (check2);
\draw[->] (check2.east) --++ (1.4cm,0) |- node[midway,right]{No} (
    back);
\node[startend,below=1.2cm of check2] (end) {End};
\draw[->] (check2) -- (end) node[midway,right]{Yes};
\end{tikzpicture}
```

Again, we have used `<direction>= of` syntax throughout the code to place the nodes at relative positions. A node named `back` is also pinpointed in the middle of the second arrow for the return arrow later.

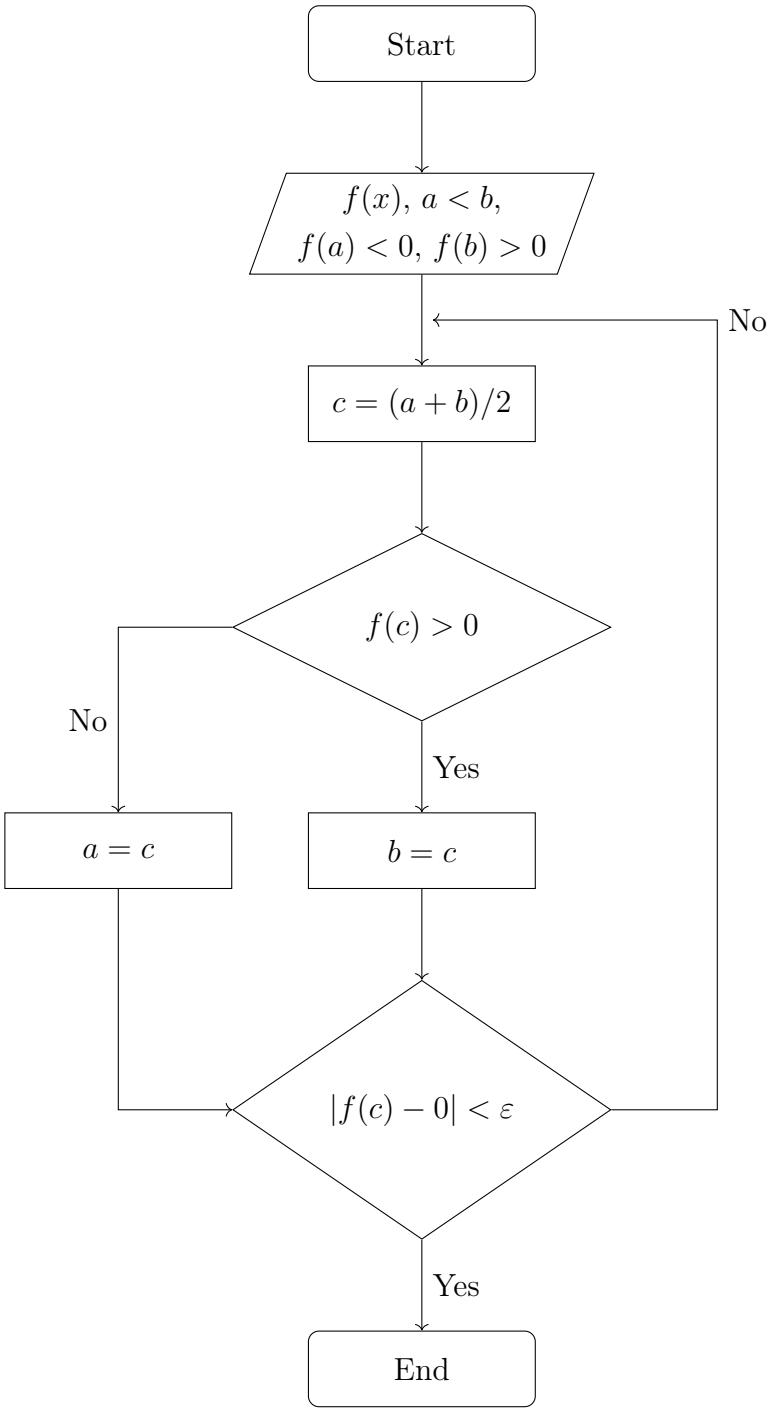


Figure 9.10: *The flow chart of the bisection algorithm.*

9.6 Electrical Circuit Diagrams

Electrical Circuits with Components Finally, we can draw electrical circuit diagrams by importing the `circuitikz` package. A small example is given as Figure 9.11.

```
\begin{circuitikz}
\draw (-3,0) to[vsource, l=$E$] (3,0);
\draw (3,0) to[nos,n=S1] (3,-3)
      node[ocirc] at (S1.w) {}
      node[ocirc] at (S1.e) {};
\draw (-3,0) to[R=$R$] (-3,-3);
\draw (-3,-3) to[C=$1/C$] (3,-3);
\draw (-1.5,0) to[L=$L$, *-*] (-1.5,-3);
\end{circuitikz}
```

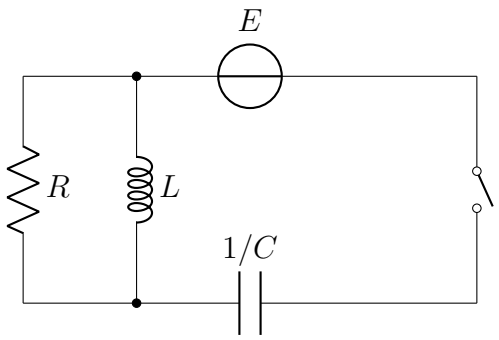
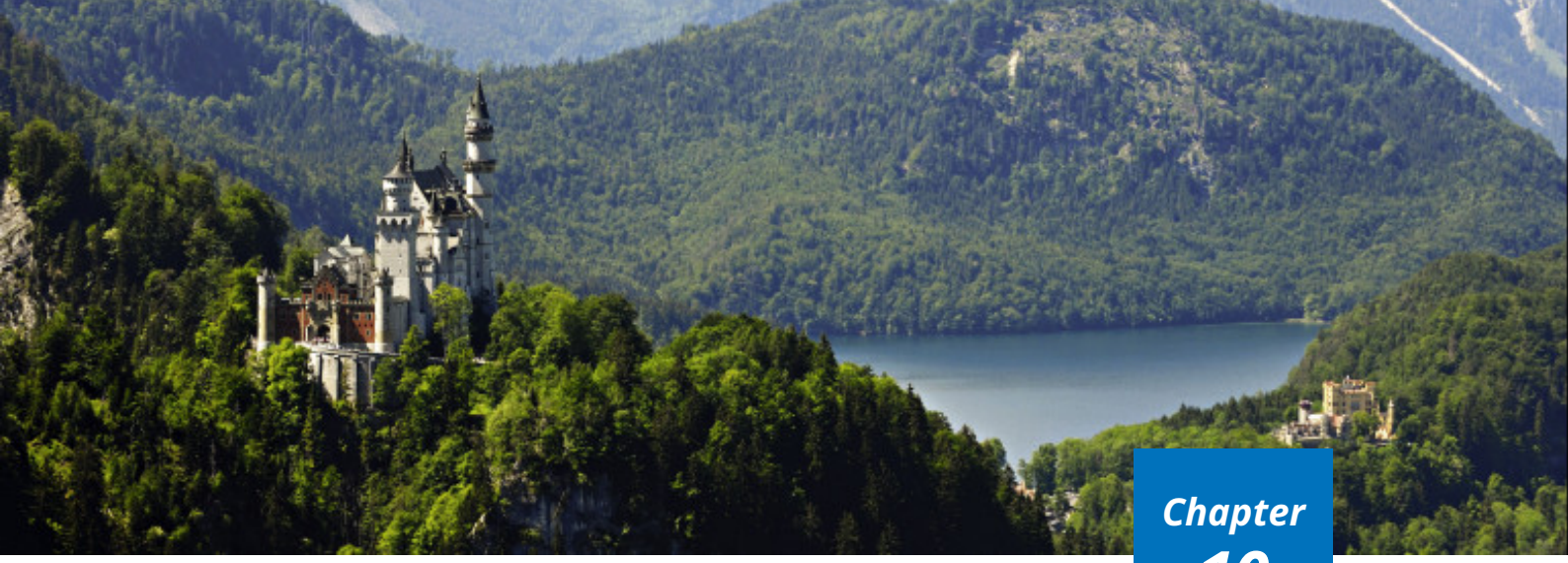


Figure 9.11: *A toy circuit with a voltage source, resistor, inductor, and capacitor.*

Specifically, the `nos,n=S1` part draws a switch with the internal name `S1`, and the `*-*` syntax produces the junctions around the inductor.



Chapter 10

Miscellaneous

> It is not even the beginning of the end. But it is perhaps the end of the beginning.
>
> *(Winston Churchill)*

Introduction This chapter touches on some minor details about how to enhance a LaTeX book.

10.1 Custom Page Style and Design

Inserting Pictures to (Chapter) Pages Many books will have a banner placed on the first page of a chapter. To achieve the same effect, we need to declare our own page style, and before that, we also need a way to store and update the banner picture for different chapters. This may be done by defining two commands:

```
\newcommand*{\titlepic}{}
\newcommand*{\puttitlepic}[2][\paperwidth]{\renewcommand*{\titlepic
   }{\includegraphics[width=#1]{#2}}}
```

The `\titlepic` command stores and displays the picture (empty by default), which can be modified by the `\puttitlepic` command at any time. Then, we can define the new page style, in addition to the background layer, as follows.

```
\puttitlepic{graphics/sight_hohenschwangau-castle.jpg} % replace it
   with any image path
\DeclareNewLayer[align=lt,voffset=0.25\paperheight,background,mode=
   picture,contents={\putUL{\titlepic}}]{titlepiclayer}
\newpairofpagestyles[scrheadings]{Styledpages}{\AddLayersToPageStyle{
   plain.Styledpages}{titlepiclayer}}
\pagestyle{Styledpages} % don't forget to actually use the new page
   style
```

The `\DeclareNewLayer` command prepares the layer that contains the picture indicated by `\titlepic`. We need to set the `background` flag and change the `mode` to `picture`. The `\titlepic` is subsequently provided in the `\putUL` command to put it at the upper-left corner. Furthermore, the `align` and `voffset` options fine-tune the positioning of the block. Eventually, we declare the desired new page style via the `\newpairofpagestyles[<inherit>]{<name>}{<content>}` statement. We inherit the pre-existing `scrheadings` style and add the layer with the `\AddLayersToPageStyle{<pagestyle>}{<layer>}` construct to the derived `plain` style, which is used in a chapter page. By the same essence, we can also decorate different parts of any page.

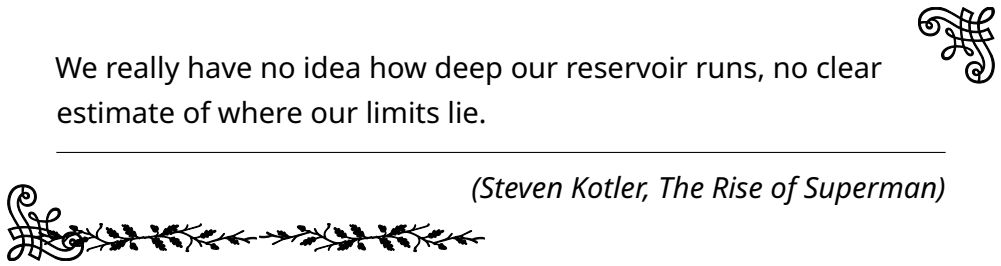
10.2 Quotation Boxes

Dictum at the Start of Chapter To insert some famous quote with KOMA-Script, we simply need to use the `\dictum[<author>]{<quote>}` syntax. And to place it just below the chapter heading, we can supply the dictum with the `\setchapterpreamble[<position>]` command right before the `\chapter` declaration. In this chapter, the following code is used:

```
\setchapterpreamble[u]{\medskip\KOMAoptions{sfdefaults=false}\dictum[
   Winston Churchill]{It is not even the beginning of the end. But it
    is perhaps the end of the beginning.}}
\chapter{Miscellaneous}
```

to display Churchill's speech at the very beginning. Notice that we have locally activated `\KOMAoptions{sfdefaults=false}` to preserve the usual serif font since a dictum by default uses sans-serif via `\maybesffamily` (see Section 6.2). Moreover, a `\medskip` is added to properly adjust the vertical spacing from the header line.

Beautiful Quotes It is also straightforward to write any quote as a dictum in the main text. However, we will go one step further to customize and decorate the quote box. We will put it inside a `tikzpicture` and make use of `pgfornament` for that. Let's take a look at the nice-looking example below.



In producing this, we have written

```
{\centering\begin{tikzpicture}
\renewcommand*{\raggeddictum}{}
\renewcommand*{\dictumwidth}{0.8\textwidth}
\node (quote) {\dictum[Steven Kotler, The Rise of Superman]{We really
    have no idea how deep our reservoir runs, no clear estimate of
   where our limits lie.}};
\node at (quote.north east) {\pgfornament[width=1cm,symmetry=v]{39}};
\node at (quote.south west) {\pgfornament[width=1cm,symmetry=h]{39}};
\coordinate (QSA) at ($(quote.south)-(0.2cm,0.2cm)$);
\coordinate (QSB) at ($(quote.south west)-(0.2cm,0.2cm)$);
```

```
\pgfornamentline{QSA}{QSB}{2}{87}
\end{tikzpicture}\par}
```

The first two statements reset the alignment and width of the dictum. Then we wrap the main quote within a named node. Finally, we add two mirrored ornaments at its top-right and bottom-left while specifying `symmetry` for the `\pgfornament` commands, in addition to a horizontal double branch under the quote via the `\pgfornamentline` macro.

10.3 Asian Characters Support

Chinese/Japanese/Korean Layer To be able to render Asian characters in LaTeX, we can use the package `CJKutf8` and enclose the content within a `CJK*` environment. The syntax goes like this:

```
\begin{CJK*}{UTF8}{bkai} % gkai for simplified Chinese, min for
    Japanese, ...
... % can insert the desired Asian characters here
\end{CJK*}
```

Two example outputs are shown right below. (Read the source code for the original input words.)

天長地久有時盡，此恨綿綿無絕期。
白居易　《長恨歌》

もう一回　もう一回
僕はこの手を伸ばしたい (Extra cookie if you know which song it comes from.)

XeLaTeX and LuaLaTeX As a remark, if the document will be mainly in Chinese/Japanese/Korean or any other non-Latin language in general, then it may be more beneficial to just switch to the XeLaTeX or LuaLaTeX compiler since

they directly support UTF-8 encoded text. However, we will keep ourselves to the default pdfLaTeX compiler to avoid complications.

10.4 Organizing References by BibTeX

Importing and Citing References Most publications or books will come with the *BibTeX* metadata. To cite them, we can first download and (optionally) combine them in a single `.bib` file. Subsequently, we need to import the `biblatex` package and let it read the `.bib` file(s). Finally, we may call `\printbibliography` to print them out. The commands to do so look like:

```
\usepackage[style=ieee]{biblatex} % can use other style like apa
\addbibresource{references.bib} % change to the name of your .bib
    file
\nocite{*} % will still show the entries if they are not cited in the
     main text
...
\printbibliography[heading=bibintoc]
```

And to refer to any entry in the references, we simply add `\cite{<name_in_bib>}`. For example, here we shall write `\cite{kottwitz2024latex}` (can check my `.bib` file) to produce this: [1]. We may also want to change the color of the citation link by setting the `citecolor` option for `\hypersetup`.

10.5 Fine-tuning Colored Boxes

In Section 7.1, we have learnt how to create simple colored boxes with the `tcolorbox` package. Here, we provide a simple recipe to design a stylish box further:

```
\begin{tcolorbox}[enhanced, title=The title, attach boxed title to
    top text left={yshift=-0.5mm}, interior style tile={width=0.5\
    textwidth}{graphics/grid_paper.jpg},
boxed title style={frame code={\path[tcb fill frame] (frame.north
    west) -- (frame.north east) -- ([xshift=3mm]frame.south east) -- (
    frame.south west) -- cycle;},
interior code={\path[tcb fill interior] (interior.north west) -- (
    interior.north east) -- ([xshift=3mm]interior.south east) -- (
    interior.south west) -- cycle;}}]
For demonstration.
\end{tcolorbox}
```

This produces:

The title

For demonstration.

We need the `enhanced` keyword to enable the extra features. The `attach boxed title to <position>` option controls the position of the title. We have also specified the `interior style tile` option to use a stock grid paper picture as tiles that fill the interior background. Finally, in the `boxed title style` option, we enter some drawing code, just like it is in TikZ, to the `frame code` and `interior code` parts to customize the appearance of the title box.

Epilogue

Thanks for reading this book to the end. I hope that my delicate efforts do help you in learning LaTeX. However, as Napoleon said, "In war, the moral is to the physical as three is to one." Similarly, for writing a book, I believe the mentality/spirit of the author is much more important than merely the (LaTeX) techniques or domain knowledge. And so, I want to chip in my personal list of useful resources for equipping authors to all of you as a gift:

- The Creative Penn (In particular, her book "The Successful Author Mindset")
- Writers Helping Writer ®
- Jane Friedman
- Robert J. Sawyer

as well as the masterpiece "Bird by Bird" by Anne Lamott [2]. This is clearly not an exhaustive list and is limited by my exposure, but I think that you will likely find motivations in them. Let me finish with an excerpt taken from "Bird by Bird":

"I still encourage anyone who feels at all compelled to write to do so. I just try to warn people who hope to get published that publication is not all it is cracked up to be.

But writing is.

Writing has so much to give, so much to teach, so many surprises. That thing you had to force yourself to do – the actual act of writing – turns out to be the best part. It's like discovering that while you thought you needed the tea ceremony

for the caffeine, what you really needed was the tea ceremony. The act of writing turns out to be its own reward."

Enjoy!

Index

Answers to Exercises

Answers for Chapter 5

5.3)

```
\foreach \x in {1,...,20} {%
\pgfmathsetmacro{\diva}{int(Mod(\x, 3))}%
\pgfmathsetmacro{\divb}{int(Mod(\x, 5))}%
\\ \ifthenelse{\(\diva = 0\) \AND \(\divb = 0\)}{\x{} is
   divisible by both 3 and 5.}{
\ifthenelse{\diva = 0}{\x{} is divisible by 3.}{\x{} is not
   divisible by 3.}
\ifthenelse{\divb = 0}{\x{} is divisible by 5.}{\x{} is not
   divisible by 5.}}}
```

Bibliography

[1] S. Kottwitz, *LaTeX Cookbook: Over 100 Practical, Ready-to-use LaTeX Recipes for Instant Solutions*. Packt Publishing, 2024, ISBN: 9781835080320. [Online]. Available: https://books.google.com.tw/books?id=N_Cb0AEACAAJ.

[2] A. Lamott, *Bird by Bird: Some Instructions on Writing and Life*. Knopf Doubleday Publishing Group, 1995, ISBN: 9780385480017. [Online]. Available: https://books.google.com.tw/books?id=t9cuMLk15PYC.