

# 第一个 L<sup>A</sup>T<sub>E</sub>X 项目——L<sup>A</sup>T<sub>E</sub>X 使用教程

李奔

2025 年 9 月 29 日

# 目录

1 基础篇	3
1.1 基本命令与环境	3
1.1.1 基本命令	3
1.1.2 基本环境	3
1.2 包管理	4
1.3 章节与段落	5
1.3.1 章节	5
1.3.2 段落	5
1.4 文本	5
1.4.1 字体大小	5
1.4.2 特殊字符	6
1.4.3 空格	6
1.4.4 换页	6
1.5 表格	6
1.5.1 tabular	6
1.5.2 table	7
1.6 目录基础	8
1.7 L <sup>A</sup> T <sub>E</sub> X 代码注释	8
1.8 纸张布局	9
1.8.1 documentclass	9
1.8.2 geometry 宏包	9
1.9 交叉引用	10
1.9.1 基本原理	10
1.9.2 基础使用方法	10
1.10 URL	11
1.11 图片插入基础	11
1.12 数学公式基础	13
1.12.1 常用宏包	13
1.12.2 行内公式和行间公式	13
1.12.3 数学公式排版基础	13
1.12.4 积分、求和和极限	14
1.12.5 矩阵和行列式	14

1.12.6	多行公式	15
1.12.7	公式编号与引用	18
<b>2</b>	<b>进阶篇</b>	<b>19</b>
2.1	字体控制进阶	19
2.1.1	字体颜色	19
2.1.2	fontspec 使用	19
2.2	引用进阶	20
2.2.1	hyperref	20
2.2.2	cleveref	21
2.3	表格插入进阶	22
2.3.1	multirow 宏包	22
2.3.2	array 宏包	23
2.3.3	booktabs 宏包	24
2.4	图片插入进阶	25
2.4.1	并排显示图片	25
2.4.2	图片跨双栏	26
2.5	tcolorbox	27
2.5.1	创建单个盒子	27
2.5.2	创建自定义可重复盒子	28
2.6	算法表示	28
2.7	代码块	30
2.7.1	插入简单代码	30
2.7.2	全局设置	30
2.7.3	从外部文件插入代码	31
2.8	数学公式进阶	31
2.8.1	数理逻辑	31
2.8.2	导数和微分	32
2.8.3	自定义符号	33
2.8.4	公式中的文本	34
2.8.5	符号和字体的精细控制	34
2.8.6	定理类环境与结构化数学写作	35
2.9	文献管理与引用——BibTeX	37
2.9.1	BibTeX 简单介绍	37
2.9.2	编译流程	37
2.9.3	.bib 数据库文件	37
2.9.4	.bst 样式文件	38
2.9.5	.tex 中实现引用	39
2.9.6	natbib 宏包	39

# Chapter 1

## 基础篇

### 1.1 基本命令与环境

#### 1.1.1 基本命令

在  $\text{\LaTeX}$  中，**命令**是用于执行特定操作的指令。它们通常以反斜杠  $\backslash$  开头，后面跟着命令名称。有些命令可以接受参数，这些参数用花括号  $\{\}$  包裹。常见的命令包括：

- $\backslash\text{documentclass}\{\}$ ：这是每个  $\text{\LaTeX}$  文档的第一个命令，用于声明文档的类型，例如 `article`、`report` 或 `book`。
- $\backslash\text{usepackage}\{\}$ ：用于导入宏包，扩展  $\text{\LaTeX}$  的功能。
- $\backslash\text{section}\{\}$  和  $\backslash\text{subsection}\{\}$ ：用于创建文档的标题结构。
- $\backslash\text{title}\{\}$  和  $\backslash\text{author}\{\}$ ：设置文档的标题和作者信息。
- $\backslash\text{maketitle}$ ：用于生成并显示标题部分。
- $\backslash\text{textbf}\{\}$ 、 $\backslash\text{textit}\{\}$ 、 $\backslash\text{texttt}\{\}$ ：用于设置文本样式。
  - $\backslash\text{textbf}\{\}$ ：生成**加粗**的文本。
  - $\backslash\text{textit}\{\}$ ：生成斜体的文本。
  - $\backslash\text{texttt}\{\}$ ：生成等宽的文本，可用于代码和命令。
- $\backslash\backslash$ ：用于手动换行。

#### 1.1.2 基本环境

**环境**用于对文档的某个区域进行特殊的排版，它们由一对  $\backslash\text{begin}\{\}$  和  $\backslash\text{end}\{\}$  命令定义。所有需要特殊处理的内容都放在这对命令之间。常见的环境包括：

- `document`：最重要的环境，所有可见的文档必须放在  $\backslash\text{begin}\{\text{document}\}$  和  $\backslash\text{end}\{\text{document}\}$  之间。
- `itemsize`：用于创建**无序列表**。在  $\backslash\text{begin}\{\text{itemsize}\}$  和  $\backslash\text{end}\{\text{itemsize}\}$  之间，使用  $\backslash\text{item}$  命令来标记每一个列表项。具体使用方法如下：

```

1 \begin{itemize}
2   \item 第一个项目
3   \item 第二个项目
4 \end{itemize}

```

- `enumerate`: 用于创建有序列表。具体使用方法如下:

```

1 \begin{enumerate}
2   \item 第一个项目
3   \item 第二个项目
4 \end{enumerate}

```

- `center`: 用于将文本或内容居中对齐。具体使用方法如下:

```

1 \begin{center}
2   居中对齐的文本。
3 \end{center}

```

- `figure`: 用于管理图片。这是一个“浮动体”环境， $\text{\LaTeX}$  会自动将图片放置在最佳位置，以避免排版不美观。
- `table`: 用于创建表格。这是一个“浮动体”环境

## 1.2 包管理

在  $\text{\LaTeX}$  中，宏包是一个核心概念。可以把它们理解成扩展  $\text{\LaTeX}$  功能的插件或库。宏包可以做很多基础  $\text{\LaTeX}$  无法直接完成的事情，比如：插入图片、创建复杂的数学公式、改变页面布局，或者使用特定字体等。

要使用一个宏包，需要在文档的**导言区**中使用 `\usepackage` 命令。导言区是指 `\documentclass{}` 和 `\begin{document}` 之间的部分。

基本语法是：

Latex 代码示例

```

1 \usepackage{package_name}

```

有些宏包可以接受**选项**，这些选项用方括号 `[]` 包裹，放在宏包名称的前面。选项用于修改宏包的行为。例如 `geometry` 宏包可以用来设置文档的页边距。代码如下所示：

```

1 \usepackage[a4paper, margin=2cm]{geometry}

```

对于初学者来说，以下宏包是必不可少的：

- `amsmath`: 增强数学公式排版功能。
- `graphicx`: 在文档中插入图片。

- `hyperref`: 创建可点击的超链接, 让 PDF 文档中的目录, 引用和网址都变得可交互。
- `xcolor`: 提供丰富的颜色选项, 为文本、背景上色。

## 1.3 章节与段落

在  $\text{\LaTeX}$  中, 需要通过一些规则和命令来告诉  $\text{\LaTeX}$  如何处理文本。

### 1.3.1 章节

在  $\text{\LaTeX}$  中, 可以使用不同的标题命令来创建不同级别的章节和子章节。常用命令为:

- `\section{标题}`: 一级标题
- `\subsection{标题}`: 二级标题
- `\subsubsection{标题}`: 三级标题

### 1.3.2 段落

$\text{\LaTeX}$  将文本组织成段落。默认情况下, 它会自动识别段落。要开始一个新的段落, 只需要在两个段落之间留一个空行即可。

- **自动缩进**:  $\text{\LaTeX}$  会在每个段落的开头自动缩进
- **换行**: 无需手动换行,  $\text{\LaTeX}$  会根据版面的宽度自动处理

示例如下:

```
1 这是第一个段落
2
3 这是第二个段落
```

如果想在不开新段落的情况下**强制换行**, 可以使用命令 `\` 或 `\newline`。

## 1.4 文本

### 1.4.1 字体大小

$\text{\LaTeX}$  定义了一系列用于改变字体大小的命令, 这些命令通常用于文档的特定部分, 比如标题或标注。它们是**开关命令**, 一旦使用就会影响其后面的所有文本, 直到遇到另一个字体大小命令或作用范围结束。

因此设置字体大小时, 更常用的一个办法如下所示:

```
1 这是普通大小的文本, {\Large 大的字体} 在这里。
```

从大到小排列的字体大小命令:

- `\Huge`

- `\huge`
- `\LARGE`
- `\Large`
- `\large`
- `\normalsize`
- `\small`
- `\footnotesize`
- `\scriptsize`
- `\tiny`

### 1.4.2 特殊字符

LaTeX 对一些字符有特殊的用途，比如 #、\$、% 等。如果想在文本中使用这些字符，需要用反斜杠进行转义。

- 对于 # 等普通的特殊字符，写法为：`\#`
- 对于 \ 这种特殊字符，写法为：`\textbackslash`

### 1.4.3 空格

几种空格方法如下：

表 1.1: 常用空格方法

空格方式	源代码	显示	宽度
普通空格	<code>a b</code>	a b	一个英文字符的宽度
quad 空格	<code>a\quad b</code>	a   b	一个中文字符的宽度
qqquad 空格	<code>a\qqquad b</code>	a     b	两个中文字符的宽度
大空格	<code>a\   b</code>	a b	1/3 字符的宽度

### 1.4.4 换页

使用 `\newpage` 进行换页，一般在一页的最后写

## 1.5 表格

### 1.5.1 tabular

创建表格最常用的环境是 `tabular`。而 `tabular` 环境是用来创建表格内容的，它需要一个参数来定义表格的列格式，基本语法如下：

```

1 \begin{tabular}{格式}
2   \hline %顶部横线, 一般来说必须要写
3   表头1 & ... & 表头 n \\ [\hline]
4   表项11 & ... & 表项1n \\ [\hline]
5   ...
6   表项n1 & ... & 表项nn \\ [\hline]
7 \end{tabular}
    
```

- {格式}: 这是一个必要的参数, 用于定义表格中每一列的对齐方式。
  - l: 左对齐
  - c: 居中对齐
  - r: 右对齐
  - |: 绘制一条垂直线
- 对于表头/每一行表项后面的 [\hline] 代表 \hline 是可以省略的, 如果写的话, 那么该表头/表项上就会出现一个横线, 不写的话就不会出现这条横线

对于如下 L<sup>A</sup>T<sub>E</sub>X 代码:

```

1 \begin{tabular}{|l|c|r|}
2 \hline % 顶部横线
3 姓名 & 年龄 & 城市 \\ \hline % 表格头
4 小明 & 25 & 北京 \\ \hline
5 小红 & 28 & 上海 \\ \hline
6 \end{tabular}
    
```

会出现如下效果

姓名	年龄	城市
小明	25	北京
小红	28	上海

### 1.5.2 table

tabular 环境只是创建了表格内容, 它不会自动添加标题或编号。如果想让表格拥有标题、编号并且能够像图片一样“浮动”(让 L<sup>A</sup>T<sub>E</sub>X 自动寻找最佳位置放置它), 你需要将 tabular 环境放在 table 浮动体环境中。基本语法如下:

```

1 \begin{table}[h!]
2   \centering
3   \caption{一个示例表格}
4   \begin{tabular}{|l|c|r|}
5     \hline
6     表头1 & ... & 表头 n \\ [\hline]
    
```



```

7      表项11 & ... & 表项1n \\ [\hline]
8      ...
9      表项n1 & ... & 表项nn \\ [\hline]
10     \end{tabular}
11     \label{tab:sample}
12 \end{table}

```

- `\caption{}`: 为表格添加标题。
- `\label{}`: 为表格添加表填, 之后可以用 `\ref{tab:sample}` 命令来引用这个表格的编号。
- `\centering`: 将整个表格居中。
- `[h!]`: 这是一个位置指示符, 告诉  $\text{\LaTeX}$  应该如何放置这个浮动体, 常用的位置指示符有:
  - `h(here)`: 尽可能放在这里
  - `t(top)`: 放在页面的顶部
  - `b(buttom)`: 放在页面的底部
  - `p(page)`: 放在单独的浮动页

## 1.6 目录基础

在  $\text{\LaTeX}$  中创建目录非常简单, 而且完全自动化。你不需要手动输入章节名称和页码,  $\text{\LaTeX}$  会自动扫描你的文档结构, 生成一个完整的、可点击的目录。

创建目录的核心命令只有一个: `\tableofcontents`。你只需要在文档中你希望目录出现的位置 (通常在 `\maketitle` 命令之后) 插入这个命令即可。

## 1.7 $\text{\LaTeX}$ 代码注释

单行注释:

Latex 代码示例

```
1 % 注释内容
```

多行注释 1:

Latex 代码示例

```

1 \iffalse
2 注释内容
3 \fi

```

多行注释 2(使用 `\usepackage{verbatim}`):

```

1 \begin{comment}
2 注释内容
3 \end{comment}

```

## 1.8 纸张布局

### 1.8.1 documentclass

语法如下：

Latex 代码示例

```

1 \documentclass[options]{class_name}

```

- **class\_name**: 文档类的名称, 例如 `article`、`report`或`book`
- **options**: 可选参数, 用于修改文档类的默认配置, 可以使用多个选项, 不同选项之间用逗号分隔
  - 字体大小: 设置文档的基础字体大小 (例如: `12pt`)
  - 纸张大小: 设置纸张规格 (例如: `a4paper`、`b5paper`)
  - 排版模式: `twocolumn`用于双栏排版, `twoside`用于双面排版
  - 草稿模式: `draft`会在超出边界的地方用黑线标记, 方便检查排版问题

### 1.8.2 geometry 宏包

`geometry` 宏包是 L<sup>A</sup>T<sub>E</sub>X 中最强大和灵活的页面布局工具。它可以让你精确地控制纸张大小、页边距、页眉页脚的位置, 从而取代文档类自带的粗略设置。

基本语法如下:

```

1 \usepackage[选项1,选项2,...]{geometry}

```

- 纸张大小
  - `a4paper`: A4 纸张 (默认)。
  - `letterpaper`: Letter 纸张。
  - `b5paper`: B5 纸张。
  - `a3paper`, `a5paper`, `b4paper` 等更多选项。
- 页边距
  - `margin=2.5cm`: 设置所有四个边距都为 2.5cm。这是最简洁的用法。
  - `left=2cm, right=3cm`: 分别设置左右边距。

- `top=2cm, bottom=2.5cm`: 分别设置上下边距。
- `hmargin=2.5cm`: 设置水平（左右）边距。
- `vmargin=3cm`: 设置垂直（上下）边距。
- 页眉和页脚
  - `headheight=1.2cm`: 设置页眉区域的高度。
  - `headsep=0.8cm`: 设置页眉底部与正文顶部的距离。
  - `footskip=1cm`: 设置正文底部与页脚顶部的距离。
- 正文区域大小
  - `textwidth=15cm`: 设置正文区域的宽度。
  - `textheight=22cm`: 设置正文区域的高度。
  - `width=15cm, height=22cm`: 与 `textwidth` 和 `textheight` 相同。

## 1.9 交叉引用

在  $\text{\LaTeX}$  中，交叉引用是一个非常强大的功能，它能让你在文档中自动引用章节、图表、公式和表格的编号，而无需手动管理这些编号。这不仅省去了很多麻烦，还能确保文档在修改时（例如增删章节）编号依然正确。

$\text{\LaTeX}$  的交叉引用主要依赖于两个命令：`\label` 和 `\ref`

### 1.9.1 基本原理

1. 定义标签 (Label): 你需要在引用的地方使用 `\label{key}` 来设置一个唯一的标签，这里的 `key` 是自己定义的。
2. 引用标签 (Reference): 在文档的任何其他地方，可使用 `\ref{key}` 来引用这个标签。 $\text{\LaTeX}$  编译时会自动查找 `key` 对应的编号，并将其插入到 `\ref` 的位置。

由于编号的生成和引用是一个两步过程，因此你的  $\text{\LaTeX}$  文档通常需要编译两次才能使所有的引用都正确显示。第一次编译时， $\text{\LaTeX}$  会生成一个 `.aux` 文件，其中包含了所有的标签及其对应的编号信息；第二次编译时， $\text{\LaTeX}$  会读取这个 `.aux` 文件，用正确的编号替换 `\ref` 命令。

### 1.9.2 基础使用方法

可以对以下几种主要对象进行交叉引用：

#### 章节

你可以为任何章节标题设置标签，然后引用它。如下所示：

```

1 \section{简介}
2 \label{sec:intro}
3
4 这是一个关于章节的例子。正如我们在 \ref{sec:intro} 节中所讨论的，...
```

## 图表

通常在 `figure` 或 `table` 浮动环境中设置标签，注意 `\label` 应该放在 `\caption` 命令的后面。如果放在前面，`\ref` 可能会引用到错误的编号（例如章节编号）。具体例子如下所示：

```
1 \begin{figure}[htbp]
2   \centering
3   \includegraphics[width=0.7\textwidth]{example-image-a}
4   \caption{一个示例图片}
5   \label{fig:sample} % 放在这里！
6 \end{figure}
7
8 如图 \ref{fig:sample} 所示，这是一个非常重要的示例。
```

## 1.10 URL

最简单的用法为：引用 `url` 宏包，之后用 `\url{指定的 URL}` 设置 URL 链接。

例如可以用 `\url{https://www.bilibili.com/}` 生成 B 站的 URL: <https://www.bilibili.com/>

URL 宏包的特点是：

- 自动处理特殊字符
- 默认使用等宽字体
- 默认不可点击（只显示文本）

为了更好地展示 URL 链接（修改颜色、使其可点击等），我们使用 `hyperref` 宏包，该宏包更加全面，具体内容参考：进阶篇——引用进阶。

## 1.11 图片插入基础

在插入图片之前，需要引入 `graphicx` 宏包。

插入图片的基本命令是：`\includegraphics[参数]{filename}`。这里的 `filename` 是图片路径名（绝对路径和相对路径都可以）。注意：传统的 `LaTeX` 引擎只支持 `.eps` 格式。而 `pdfLaTeX`、`XeLaTeX`、`LuaLaTeX` 在此基础上还支持 `.pdf`、`.jpg`、`.png` 格式的图片。

`\includegraphics[参数]{filename}` 中的参数用于指定图片的大小。有以下指定方式：

- 指定高度和宽度：其中 `width` 和 `height` 一般设置为以 `cm` 为单位

```
1 \includegraphics[width=..., height=...]{filename}
```

- 指定相对尺寸

```
1 % 图片宽度为文本宽度的 a 倍
2 \includegraphics[width=a\textwidth]{filename}
3 % 图片高度为文本高度的 a 倍
```

```
4 \includegraphics[height=a\textheight]{filename}
```

- 保持宽高比：如果只指定了 `width` 或 `height` 中的一个， $\text{\LaTeX}$  会自动保持图片的原始宽高比

上面的代码仅能确保图片以一个合适的大小被插入到文档中，如果需要更好地排版文字和图片，通常会把图片放在一个浮动环境中，最常用的是 `figure` 环境。浮动环境的作用是让  $\text{\LaTeX}$  自动决定图片最佳的放置位置，避免图片将页面分隔得非常突兀。`figure` 的基本语法如下：

```
1 \begin{figure}[options]
2   \centering
3   \includegraphics[图片大小设置选项]{filename}
4   \caption{这是图片的标题}
5   \label{fig:mylabel}
6 \end{figure}
```

- `[options]`：可选参数，用于控制图片在页面上的放置位置。
  - `h`：Here，尽量放在代码所在位置。
  - `t`：Top，尽量放在页面的顶部。
  - `b`：Bottom，尽量放在页面的底部。
  - `p`：Page，单独放在一页上。
  - `!`：强制执行，与上述选项结合使用。
- `centering`：让图片居中。
- `\caption{...}`：为图片添加标题，这个标题会显示在图片下方，并且会出现在图表目录中。
- `label{...}`：为图片设置一个标签，可以方便地在正文中进行交叉引用。

对于如下代码段，可以如下的插入效果：

```
1 \begin{figure}[h]
2   \centering
3   \includegraphics[width=10cm, height=3cm]{江景.jpg}
4   \caption{湘江晚霞图}
5   \label{fig:xiangjiangwanxia}
6 \end{figure}
7
8 湘江晚霞如图 \ref{fig:xiangjiangwanxia} 所示。
```

湘江晚霞如图 1.1 所示。

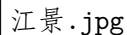
江景.jpg

图 1.1: 湘江晚霞图

## 1.12 数学公式基础

### 1.12.1 常用宏包

为了排版数学公式，最常用的是 `amsmath` 宏包。当然还有以下常用的宏包：

- `amssymb`：提供了一些额外的数学符号。
- `mathtools`：增强了 `amsmath` 的功能，提供了更多有用的命令。
- `bm`：用于在数学模式下让符号加粗。

### 1.12.2 行内公式和行间公式

在  $\text{\LaTeX}$  中，数学公式需要放在特定的“数学环境”中。主要有两种类型：行内公式和行间公式。

- 行内公式：公式与正文在同一行显示。这适用于简单、短小的表达式。
  - 可用 `$...$` 包围公式。
  - 可用 `\(...\)` 包围公式（更推荐）。
- 行间公式：公式独占一行居中显示（如果需要的话会编号）。这适用于重要或复杂的表达式。
  - 可用 `$$...$$` 包围公式（一些旧编译器可能不支持）。
  - 可用 `\[...\]` 包围公式（推荐）。
  - 可用 `equation` 环境。

### 1.12.3 数学公式排版基础

以下内容可以直接通过键盘输入：

- 英文字母
- 数字
- `+`、`-`、`=`、`*` 等符号

#### 上下标

上标：使用 `^`。例如 `x^2` 表示  $x^2$ 。

下标：使用 `_`。例如 `x_i` 表示  $x_i$ 。

## 分数和根式

**分数**: 使用 `\frac{分子}{分母}`。例如 `\frac{1}{2}` 表示  $\frac{1}{2}$

**根式**: 使用 `\sqrt{expression}`。例如 `\sqrt{b-ac}` 表示  $\sqrt{b-ac}$

### 1.12.4 积分、求和和极限

不定积分公式为 `\int f(x) dx`:

$$\int f(x)dx$$

而定积分则表示为: `\int_a^b f(x) dx`, 其中 a 表示下界, b 表示上界:

$$\int_a^b f(x)dx$$

对于求和, 通常形式为: `\sum_{下标}^{上标} 表达式`。例如 `\sum_{i=1}^n i^2` 表示:

$$\sum_{i=1}^n i^2$$

在行内环境中, 求和的上下标会排版在右侧, 如 `\sum_{i=1}^n i` 会表示为:  $\sum_{i=1}^n i$ 。若要强制在上下显示, 可以在 `\sum` 后加上 `\limits`。例如 `\sum\limits_{i=1}^n i` 会表示为:  $\sum_{i=1}^n i$ 。

极限公式可以表示为 `\lim_{变量 \to 某个值} 极限式子`, 例如 `\lim_{x \to 0} \frac{\sin x}{x}` 可以表示为:

$$\lim_{x \rightarrow 0} \frac{\sin x}{x}$$

在行内, `\lim_{x \to 0} f(x)` 表示为:  $\lim_{x \rightarrow 0} f(x)$ , 如果要强制在上下显示, 可以在 `\lim` 后加上 `\limits`。

### 1.12.5 矩阵和行列式

矩阵和行列式的基本格式为:

```
1 \begin{matrix/pmatrix/bmatrix/Bmatrix/vmatrix/Vmatrix}
2 a_{11} & a_{12} & \dots & a_{1n} \\
3 \dots
4 a_{n1} & a_{n2} & \dots & a_{nn}
5 \end{matrix/pmatrix/bmatrix/Bmatrix/vmatrix/Vmatrix}
```

其中:

- `matrix`: 不带任何括号。
- `pmatrix`: 圆括号。
- `bmatrix`: 方括号。
- `Bmatrix`: 大括号。
- `vmatrix`: 单竖线, 通常用于行列式。

- Vmatrix: 双竖线, 通常用于行列式。

例如:

```

1 \[
2 \begin{Vmatrix}
3 1 & 2 & 3 \\
4 4 & 5 & 6 \\
5 7 & 8 & 9
6 \end{Vmatrix}
7 \]
```

表示:

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

### 1.12.6 多行公式

#### align

align 环境可以对齐多行公式, 并为每一行自动编号, 如果不需要对公式进行编号, 可以使用 align\* 环境。此外\align 允许通过 & 符号指定对齐点。通过插入多个 & 符号, 你可以创建多列公式, 每列独立对齐。语法为:

```

1 \begin{align/align*}
2 公式1 \\
3 公式2 \\
4 ...
5 公式n
6 \end{align/align*}
```

例如:

```

1 \begin{align*}
2 2x + 3y = 5 \\
3 4x - y = 3
4 \end{align*}
5
6 \begin{align}
7 x &= 1+2 & y &= \sin(\theta) \\
8 x' &= 3+4 & y' &= \cos(\phi)
9 \end{align}
```

表示:

$$\begin{aligned} 2x + 3y &= 5 \\ 4x - y &= 3 \end{aligned}$$



$$x = 1 + 2 \qquad y = \sin(\theta) \qquad (1.1)$$

$$x' = 3 + 4 \qquad y' = \cos(\phi) \qquad (1.2)$$

## split

当单个公式太长需要换行，但希望你它只获得一个编号时，应使用 `split` 环境。

- `split` 必须嵌套在带编号的公式环境（如 `equation` 或 `gather`）内部。
- 每行公式中最多只有一个对齐点 `&`。
- 编号会居中放在所有行的垂直中心。

例如：

```
1 \begin{equation}
2 \begin{split}
3 V &= \int_D \left( f(x, y) + g(x, y) \right) \mathrm{d}A \\
4 &= \int_a^b \int_c^d f(x, y) \mathrm{d}y \mathrm{d}x + \text{剩} \\
&\quad \text{余项} \\
5 \end{split}
6 \end{equation}
```

效果为：

$$\begin{aligned} V &= \int_D (f(x, y) + g(x, y)) \mathrm{d}A \\ &= \int_a^b \int_c^d f(x, y) \mathrm{d}y \mathrm{d}x + \text{剩余项} \end{aligned} \qquad (1.3)$$

## gather

用于将多行公式居中堆叠，每行公式独立编号。例如：

```
1 \begin{gather}
2 E = mc^2 \\
3 \nabla \cdot \mathbf{D} = \rho \\
4 \text{其他公式} \\
5 \end{gather}
```

效果为：

$$E = mc^2 \qquad (1.4)$$

$$\nabla \cdot \mathbf{D} = \rho \qquad (1.5)$$

$$\text{其他公式} \qquad (1.6)$$

## multiline

专为超长公式设计，它自动将第一行左对齐，最后一行右对齐，中间行居中，且只在最后一行编号。

例如：

```
1 \begin{multiline}
2   \frac{\partial}{\partial t} \int_{\Omega} \mathbf{D} \cdot \mathbf{B}
   \quad , \quad \mathrm{d}V = \iiint_{\partial \Omega} (\mathbf{E} \times \mathbf{H}) \cdot \mathrm{d}\mathbf{S} \quad \backslash\backslash
3   + \int_{\Omega} \left( \mathbf{J} \cdot \mathbf{E} + \mathbf{M} \cdot \mathbf{H} \right) \cdot \mathrm{d}V
4 \end{multiline}
```

效果为：

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{D} \cdot \mathbf{B} \, \mathrm{d}V = \iint_{\partial \Omega} (\mathbf{E} \times \mathbf{H}) \cdot \mathrm{d}\mathbf{S} + \int_{\Omega} (\mathbf{J} \cdot \mathbf{E} + \mathbf{M} \cdot \mathbf{H}) \, \mathrm{d}V \quad (1.7)$$

## cases

此外，还可以用 cases 环境定义分段函数。语法如下：

```
1 \begin{cases}
2   \text{第一段函数的表达式} & \& \text{第一段函数适用的条件} \\
3   \text{第二段函数的表达式} & \& \text{第二段函数适用的条件} \\
4   \dots \\
5   \text{第n段函数的表达式} & \& \text{第n段函数适用的条件}
6 \end{cases}
```

例如：

```
1 \[
2 f(x) =
3 \begin{cases}
4 x^2 & \& \text{if } x \geq 0 \\
5 -x^2 & \& \text{if } x < 0
6 \end{cases}
7 \]
```

表示：

$$f(x) = \begin{cases} x^2 & \text{if } x \geq 0 \\ -x^2 & \text{if } x < 0 \end{cases}$$

### 1.12.7 公式编号与引用

使用 `equation` 环境可以为公式自动编号，并可以为其添加标签以便交叉引用。语法如下

```
1 \begin{equation}
2   公式
3   \label{标签}
4 \end{equation}
```

例如：

```
1 \begin{equation}
2   E=mc^2
3   \label{eq:einstein}
4 \end{equation}
5
6 质能方程如公式 \eqref{eq:einstein} 所示。
```

表示：

$$E = mc^2 \tag{1.8}$$

质能方程如公式 (1.8) 所示。

# Chapter 2

## 进阶篇

### 2.1 字体控制进阶

#### 2.1.1 字体颜色

在 L<sup>A</sup>T<sub>E</sub>X 中控制字体颜色主要依赖于 `xcolor` 宏包。而 `xcolor` 提供了几种改变颜色的方法：

- `\textcolor{color}{text}`: 将 `text` 的颜色改为 `color`。例如 `\textcolor{red}{这一句话是红色的}` 的效果为：这一句话是红色的。
- `{\color{color}text}`: 将 `text` 的颜色改为 `color`。例如 `{\color{blue}这句话是蓝色的}` 的效果为：这句话是蓝色的。

前面我们使用的 `blue`、`red` 都是 `xcolor` 的预定义颜色。当然我们可以通过 `\definecolor` 自己定义颜色。具体语法如下：

```
1 \definecolor{name}{model}{spec}
```

- `name`: 新颜色的名字。
- `model`: 颜色模型，例如 `rgb`、`cmk`、`HTML` 等。
- `spec`: 颜色的数值。

例如：

```
1 \definecolor{mypurple}{RGB}{127, 18, 255}
2 \textcolor{mypurple}{这是我定义的紫色}
```

效果为：

这是我定义的紫色

#### 2.1.2 fontspec 使用

`fontspec` 提供了几个核心命令来设置文档的字体：

- `\setmainfont{font_name}`: 设置文档的主字体（通常是衬线字体）。

- `\setsansfont{font_name}`: 设置无衬线字体。
- `\setmonofont{font_name}`: 设置等宽字体。

有时普通字体、加粗字体、斜体字体可能要求不是同一种字体，此时我们可以为字体添加额外的参数实现上述功能。代码如下（以设置正文字体为例）：

```
1 \setmainfont{Times New Roman}[
2     BoldFont = {字体1},
3     ItalicFont = {字体2},
4     SmallCapsFont = {字体3}
5 ]
```

和设置文本颜色类似，可以用 `{\fontspec{font_type} text}` 让 `text` 使用 `font_type` 字体。如果需要同时修改字体和颜色，可以参考以下示例：

```
1 % 字体为 Arial、颜色为 mypurple
2 \textcolor{mypurple}{\fontspec{Arial} Arial is purple.}
```

## 2.2 引用进阶

### 2.2.1 hyperref

`hyperref` 的核心功能是为  $\text{\LaTeX}$  文档生成可点击的超链接。它可以自动识别文档中的各种引用、目录、URL 和电子邮件地址，并将它们转换为交互式的链接，极大地提升了 PDF 文档的可用性。使用前需要用 `\usepackage[选项]{hyperref}` 加载宏包。一旦加载该宏包，`hyperref` 会自动完成以下设置：

- 目录：`\tablecontents` 生成的目录项将变为可点击的链接，可以直接跳转到对应章节。
- 交叉引用：所有使用 `\ref`、`\pageref`、`\cite` 和 `\eqref` 的引用都会变为链接。
- 页码：`\pageref` 引用的页码也会变为链接。
- URL 和邮箱：使用 `\url` 和 `\href` 命令常见的链接都会生效。

`\hyperref` 提供了丰富的选项来定制链接的样式和行为，常用选项有：

- `colorlinks=true`: 让链接以颜色高亮。
- `linkcolor`: 内部链接的颜色，比如章节、图表和公式的引用。
- `filecolor`: 指向本地文件的链接颜色。
- `urlcolor`: URL 链接的颜色。
- `citecolor`: 引文 (bibliography) 链接的颜色。
- `pdfpagemode=UseOutlines`: 让 PDF 打开时自动显示书签 (大纲)。
- `bookmarks=true`: 创建 PDF 书签。

例如：

```
1 \usepackage[
2     colorlinks=true,
3     linkcolor=pink,
4     urlcolor=purple,
5     citecolor=green
6 ]{hyperref}
```

### 2.2.2 cleveref

它的主要作用是根据引用的类型自动添加描述性文字，从而让文档中的交叉引用更加自然和清晰。使用时需要加载 `cleveref` 宏包。但是在加载时需要注意：`cleveref` 必须在 `hyperref` 之后加载。

#### cleveref 基本使用

`cleveref` 的核心命令是 `\cref{label}`。它会根据 `label` 的前缀来判断引用类型，并自动生成正确的引用文本。

常用的标签有：

- `sec`: section
- `fig`: figure
- `tab`: table
- `eq`: equation
- `ch`: chapter

例如使用：`\cref{ch:2}` 可以产生一个指向第二章的引用：chapter 2。

#### cleveref 进阶使用

多重引用：可以一次引用多个标签，`cleveref` 会自动将它们合并成一个简洁的列表，并处理单数和复数形式，语法如下：

```
1 \cref{标签1, 标前2, ..., 标签n}
```

例如我在文档中定义了两个标签：

```
1 \label{sec:交叉引用}
2 \label{sec:URL}
```

使用：`\cref{sec:交叉引用, sec:URL}` 可以产生一个“指向交叉引用”和“URL” section 的引用：sections 1.9 and 1.10

自定义引用格式：可以修改 `cref` 的默认描述文本，语法如下：

```
1 \crefname{标签类型}{单数描述}{复数描述}
```

例如使用：

```
1 \crefname{section}{章节}{章节}
```

此时再使用 `\cref{sec:交叉引用,sec:URL}`，产生的效果为：章节 1.9 and 1.10。

## 2.3 表格插入进阶

### 2.3.1 multirow 宏包

在复杂的表格中，经常需要合并行或列的单元格，这主要依赖于 `multirow` 宏包。使用前需要使用 `\usepackage{multirow}` 进行加载。

#### 合并列

`\multicolumn` 命令用于合并列，并允许你重新定义合并后的对齐方式和边界。语法如下：

```
1 \multicolumn{num}{format}{text}
```

- `num`：要合并的列数。
- `format`：合并后的列格式（例如 `l`、`c`、`r` 等）。
- `text`：合并后单元格的内容。

例如：

```
1 \begin{tabular}{|c|c|c|}  
2 \hline  
3 \multicolumn{3}{|c|}{\textbf{学生 A 面试评分}} \\ \hline  
4 90.12 & 89.14 & 93.11 \\ \hline  
5 \end{tabular}
```

会产生如下效果：

学生 A 面试评分		
90.12	89.14	93.11

#### 合并行

`\multirow` 命令用于合并行，语法如下：

```
1 \multirow{num rows}{width}{text}
```

- `num rows`：要合并的行数。
- `width`：单元格的宽度。

– 使用 \* 表示自适应宽度。

- `text`: 合并后单元格的内容。

例如:

```
1 \begin{tabular}{|c|c|}  
2 \hline  
3 \multirow{2}{*}{\textbf{合并}} & A \\ \cline{2-2}  
4 & B \\ \hline  
5 \end{tabular}
```

会产生如下效果:

合并	A
	B

一般来说, `\multirow` 命令会和 `\cline{i-j}` 一起使用。`\cline{i-j}` 的功能为:

- 只在指定的列范围 (i 到 j) 内绘制水平线。
- 不会跨越整个表格, 适合在合并单元格后局部画线。

### 2.3.2 array 宏包

使用 `\usepackage{array}` 加载 `array` 宏包后, 可以使用以下方法控制换行和对齐方式

- `p{width}`: 顶部对齐 (默认)。
- `m{width}`: 居中对齐 (垂直居中)。
- `b{width}`: 底部对齐。
- 优先级: `b{width} > p{width} > m{width}`。

注意: `tabular` 环境在对齐表格中的所有行时, 遵循一个原则: 整行内容是相对于这一行中最高的单元格进行对齐的。因此, 如果混用 `p{width}`、`m{width}`、`b{width}`, 那么可能会造成意想不到的错误。

例如

```
1 \begin{tabular}{|b{2cm}|m{4cm}|p{6cm}|}  
2 \hline  
3 \textbf{底部对齐} & \textbf{居中对齐} & \textbf{顶部对齐} \\ \hline  
4 这是一个很长的描述, 它需要跨越多行, 但左边的内容会垂直居中。  
5 & 这是一个很长的描述, 它需要跨越多行, 但左边的内容会垂直居中。  
6 & 这是一个很长的描述, 它需要跨越多行, 但左边的内容会垂直居中。 \\ \hline  
7 \end{tabular}
```



底部对齐	居中对齐	顶部对齐
这是一个很 长的描述， 它需要跨越 多行，但左 边的内容会 垂直居中。	这是一个很长的描述，它 需要跨越多行，但左边的 内容会垂直居中。	这是一个很长的描述，它需要跨越多 行，但左边的内容会垂直居中。

### 2.3.3 booktabs 宏包

基础表格的线条通常很粗且呆板。使用 `booktabs` 宏包可以创建出具有专业学术风格的、没有垂直线的表格。使用前需要用 `\usepackage{booktabs}` 加载宏包。`booktabs` 支持的线条类型有：

- `\toprule`：表格顶部的粗线。
- `\midrule`：表格中部的中等粗细线。
- `\bottomrule`：表格底部的粗线。
- `\cmidrule[thickness](trim){i-j}`
  - `thickness`：线条粗细（例如：0.5pt）
  - `trim`：端点修剪
    - \* `l`：将线条的左侧稍微缩进一点
    - \* `r`：将线条的右侧稍微缩进一点
  - `i-j`（必选）：只画第 `i` 列到第 `j` 列之间的中等粗细横线

例如：

```

1 \begin{tabular}{lcccc}
2 \toprule
3 & \multicolumn{2}{c}{实验组} & \multicolumn{2}{c}{对照组} \\
4 \cmidrule(r){2-3} \cmidrule(l){4-5}
5 方法 & 准确率 & 召回率 & 准确率 & 召回率 \\
6 \midrule
7 A & 0.92 & 0.88 & 0.85 & 0.80 \\
8 B & 0.95 & 0.91 & 0.87 & 0.83 \\
9 \bottomrule
10 \end{tabular}

```

效果如下：

	实验组		对照组	
方法	准确率	召回率	准确率	召回率
A	0.92	0.88	0.85	0.80
B	0.95	0.91	0.87	0.83

## 2.4 图片插入进阶

### 2.4.1 并排显示图片

如果需要将多张图片放在一个 `figure` 环境里面，并且每张图片都有自己的小标题，此时需要用到 `subcaption` 宏包。故需要用 `\usepackage{subcaption}` 引入宏包。

语法如下：

```
1 \begin{figure}[位置参数]
2   \centering
3   \begin{subfigure}[对齐方式]{宽度}
4     \centering
5     \includegraphics[子图宽度]{图片文件}
6     \caption{子图标题}
7     \label{子图标签}
8   \end{subfigure}
9   \hfill % 或 \quad, \qquad, \hspace{...} 添加水平间距
10  \begin{subfigure}[对齐方式]{宽度}
11    ...
12  \end{subfigure}
13  \caption{主图标题}
14  \label{主图标签}
15 \end{figure}
```

- 位置参数：
  - h: here
  - t: top
  - b: bottom
  - p: page
- 对齐方式：
  - t: 顶部对齐
  - b: 底部对齐
  - c: 居中对齐
- 宽度：指定子图环境的宽度
- `\hfill`：在子图之间添加弹性水平间距，使它们分别靠左和靠右
- 子图宽度：一般设置为：`width=linewidth`

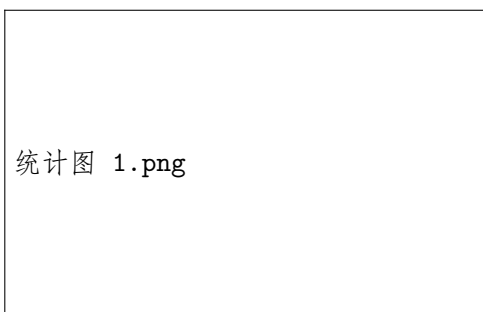
例如：

```

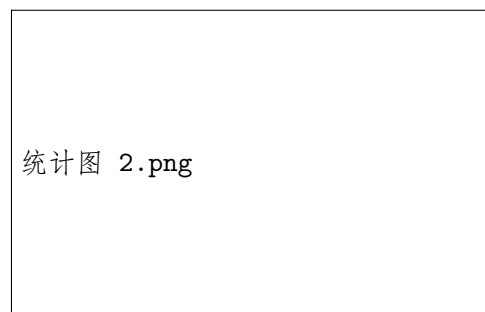
1 \begin{figure}[h]
2   \centering
3   \begin{subfigure}[b]{0.4\textwidth}
4     \centering
5     \includegraphics[width=\textwidth]{统计图1.png}
6     \caption{10000次掷骰子结果统计图}
7     \label{fig:touzi1}
8   \end{subfigure}
9   \hfill % 在两个 subfigure 之间添加水平空间
10  \begin{subfigure}[b]{0.4\textwidth}
11    \centering
12    \includegraphics[width=\textwidth]{统计图2.png}
13    \caption{100000次掷骰子结果统计图}
14    \label{fig:touzi2}
15  \end{subfigure}
16  \caption{掷骰子实验结果统计}
17  \label{fig:main}
18 \end{figure}

```

效果为:



(a) 10000 次掷骰子结果统计图



(b) 100000 次掷骰子结果统计图

图 2.1: 掷骰子实验结果统计

### 2.4.2 图片跨双栏

如果文档是双栏排版的，但某张图片需要占据整个页面宽度，则必须使用带星号的 `figure*` 环境。它通常会放在页面的顶部或单独一页。语法如下：

```

1 \begin{figure*}
2   \centering
3   \includegraphics[图片大小]{图片路径}
4   \caption{图片标题}
5   \label{标签}

```

## 2.5 tcolorbox

`tcolorbox` 是 L<sup>A</sup>T<sub>E</sub>X 中一个功能极其强大和灵活的工具，专门用于创建带有背景色、边框、标题和阴影的彩色盒子。它极大地简化了美观提示、定义、定理、代码框等复杂环境的创建。

### 2.5.1 创建单个盒子

创建单个盒子的语法为：

```
1 \begin{tcolorbox}[选项]
2 文本内容
3 \end{tcolorbox}
```

常用选项如下：

- 颜色：
  - `colback`：背景色。
  - `colframe`：边框色。
  - `coltitle`：标题颜色。
- 边框：
  - `boxrule`：边框粗细。
  - `sharp corners`：使用直角（默认圆角）。
  - `arc`：边框圆角的半径。
- 标题：
  - `title`：标题文本
  - `fonttitle`：标题字体样式
- 尺寸：
  - `width`：强制设置盒子固定宽度
  - `auto adjust textwidth=true`：让盒子自动适应文本宽度

例如：

```
1 \begin{tcolorbox}[
2   colback=red!5!white,           % 背景色：5% 红色 + 95% 白色（淡红）
3   colframe=red!75!black,         % 边框色：75% 红色 + 25% 黑色（深红）
4   title=\textbf{注意（Attention）}, % 盒子标题，自动居中
5   boxrule=1.5pt,                % 边框粗细
```

```

6     sharp corners,           % 边框使用直角（默认是圆角）
7 ]
8     这个盒子用来强调重要的信息。
9 \end{tcolorbox}

```

效果为：

### 注意 (Attention)

这个盒子用来强调重要的信息。

## 2.5.2 创建自定义可重复盒子

在文档中，通常需要多次使用相同样式的盒子。最佳实践是使用 `\newtcolorbox` 在导言区定义一个盒子。语法为：

```

1 % 现阶段先不考虑参数的用法
2 \newtcolorbox{盒子名称}[参数的数量][默认参数]{选项}

```

使用语法为：

```

1 \begin{盒子名称}
2 文本内容
3 \end{盒子名称}

```

## 2.6 算法表示

为了在  $\text{\LaTeX}$  文档中清晰地表示算法，通常使用 `algorithm2e`。

使用前需要在导言区用 `\usepackage[ruled, linesnumbered]{algorithm2e}` 加载宏包。

- `ruled`：为算法添加顶部和底部的横线。
- `linesnumbered`：为每一行代码添加行号。

之后就可以用 `algorithm` 环境来编写算法。语法如下：

```

1 \begin{algorithm}
2   \caption{算法标题}
3   \label{标签}
4   \KwData{输入}
5   \KwResult{输出}
6
7   算法伪代码
8 \end{algorithm}

```

算法伪代码中常用命令如下：

- `\BlankLine`: 插入一个空白行, 用于视觉分组。
- `\tcp{...}`: 注释。
- `\KwSty{...}`: 手动设置粗体关键字。
- `\KwTo`: for 循环中表示“到”。
- `\If{condition}{if-statement}\Else{else-statement}`: 条件判断。
- `\While{condition}{while-statement}`: while 循环。
- `\For{init; cond; step}{for-statement}`: for 循环。
- `\Foreach{item \KwIn list}{for-each-statement}`: for-each 循环。
- `Repeat{statement}\Until{condition}`: repeat 循环。
- `\Function{Name}{Params}{function-body}`: 函数。
- `\Return{...}`: 返回值。

例如:

```

1 \begin{algorithm}[h!]
2   \SetAlgoLined % 结构块带线
3   \caption{迭代斐波那契数列}
4   \label{alg:fibonacci_iter}
5
6   \KwIn{非负整数 $n$}
7   \KwOut{斐波那契数列的第 $n$ 项 $F_n$}
8
9   \If{$n \leq 1$}{
10     \Return{$n$}\;
11   }
12
13   $a \leftarrow 0$;
14   $b \leftarrow 1$;
15   $i \leftarrow 2$;
16
17   \While{$i \leq n$}{
18     $temp \leftarrow a + b$;
19     $a \leftarrow b$;
20     $b \leftarrow temp$;
21     $i \leftarrow i + 1$;
22   }
23
24   \Return{$b$}\;

```

```
25  
26 \end{algorithm}
```

产生如下效果：

---

**Algorithm 1:** 迭代斐波那契数列

---

**Input:** 非负整数  $n$

**Output:** 斐波那契数列的第  $n$  项  $F_n$

```
1 if  $n \leq 1$  then  
2   | return  $n$ ;  
3 end  
4  $a \leftarrow 0$ ;  
5  $b \leftarrow 1$ ;  
6  $i \leftarrow 2$ ;  
7 while  $i \leq n$  do  
8   |  $temp \leftarrow a + b$ ;  
9   |  $a \leftarrow b$ ;  
10  |  $b \leftarrow temp$ ;  
11  |  $i \leftarrow i + 1$ ;  
12 end  
13 return  $b$ ;
```

---

## 2.7 代码块

可以使用 `listings` 宏包来插入代码块。

使用前需要在导言区用 `\usepackage{listings}` 加载宏包。

### 2.7.1 插入简单代码

对于行内代码，可以用 `\lstinline{代码}` 完成插入。例如 `\lstinline{int x = 1;}` 的效果为： `int x = 1;`

更常用的是独立代码块，语法如下： `\begin{lstlisting}` 代码 `\end{lstlisting}`。此处的代码可以换行。

### 2.7.2 全局设置

有时我们会重复用到一个代码块样式，此时可以通过 `\lstset{}` 全局设置样式。常见设置如下：

- `basicstyle`: 基础字体款式（例如： `\ttfamily\small`）
- `keywordstyle`: 关键字颜色
- `commentstyle`: 注释样式

- `stringstyle`: 字符串颜色
- `numbers`: 行号的位置 (left 或 right)
- `numberstyle`: 行号样式
- `frame`: 添加边框
- `tabsize`: Tab 等于多少个空格
- `breaklines`: 自动换行
- `language`: 默认编程语言
- `captionpos`: 标题位置

例如:

```

1 \lstset{
2     basicstyle=\ttfamily\small,
3     keywordstyle=\color{blue},
4     commentstyle=\color{gray}\itshape,
5     stringstyle=\color{red},
6     numbers=left,
7     numberstyle=\tiny\color{gray},
8     frame=single,
9     tabsize=4,
10    breaklines=true,
11    breakatwhitespace=false,
12    showspaces=false,
13    showstringspaces=false,
14    language=Python,
15    captionpos=b,
16    escapeinside={(*@){}@*},
17 }
```

### 2.7.3 从外部文件插入代码

语法如下:

```

1 \lstinputlisting[language=编程语言, caption={标题}]{代码源文件位置}
```

## 2.8 数学公式进阶

### 2.8.1 数理逻辑

大部分逻辑符号都可以通过基本的 L<sup>A</sup>T<sub>E</sub>X 数学环境或结合 `amssymb`、`amsmath` 来实现。使用前需要导入宏包。常见的符号有:



- `\neg`:  $\neg$
- `\land`:  $\wedge$
- `\lor`:  $\vee$
- `\oplus`:  $\oplus$
- `\to`:  $\rightarrow$
- `\implies`:  $\implies$
- `\leftrightharpoonup`:  $\leftrightarrow$
- `\iff`:  $\iff$
- `\forall`:  $\forall$
- `\exists`:  $\exists$
- `\exists!`:  $\exists!$
- `\nexists`:  $\nexists$
- `\vdash`:  $\vdash$
- `\models`:  $\models$
- `\because`:  $\because$
- `\therefore`:  $\therefore$

## 2.8.2 导数和微分

### 基础的微分符号

- `\mathrm{d}x`:  $dx$
- `\partial x`:  $\partial x$

### 导数的莱布尼茨记法

- `\frac{\mathrm{d} y}{\mathrm{d} x}`:  $\frac{dy}{dx}$
- `\frac{\mathrm{d}^n y}{\mathrm{d} x^n}`:  $\frac{d^n y}{dx^n}$
- `\frac{\partial f}{\partial x}`:  $\frac{\partial f}{\partial x}$
- `\frac{\partial^n f}{\partial x^n}`:  $\frac{\partial^n f}{\partial x^n}$
- `\frac{\partial^2 f}{\partial x \partial y}`:  $\frac{\partial^2 f}{\partial x \partial y}$

对于导数在特定点的值，可以使用定界符 `\left.` 和 `\right|_.`。

- `\left.`: 是一个空的左定界符，确保右侧的 `|` 能够正确垂直拉伸。

- `\right|_:` 会生成一个垂直线，并在其底部添加下标。

例如：

```
1 \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{x=a} = 2a
```

表示：

$$\left.\frac{df}{dx}\right|_{x=a} = 2a$$

### 导数的拉格朗日记法

- $f'(x)$ :  $f'(x)$
- $f''(x)$ :  $f''(x)$
- $f^{(n)}(x)$ :  $f^{(n)}(x)$

对于导数在  $x_0$  的值，直接将  $x$  替换为  $x_0$  即可。

### 2.8.3 自定义符号

最基本的自定义命令的语法如下：

```
1 \newcommand{命令名称}[参数个数][参数默认值]{被替换的代码（可能包含被替换的内容）}
```

- 参数在被替换的代码中以 #1, #2, ... 的形式引用。
- 总参数个数：包含可选参数在内的所有参数数量。
- 默认值：可选参数的默认值。

例如：

- `\newcommand{\R}{$\mathbb{R}$}`: 此后可以用 `\R` 表示  $\mathbb{R}$ 。
- `\newcommand{\set}[1]{\left\{ #1 \right\}}`: 此后可以用 `\set{集合元素}` 表示集合。例如：`\set{1, 2, 3, 4, 5}` 表示  $\{1, 2, 3, 4, 5\}$ 。

如果我们定义了一个已经存在的命令，就会出现错误，为避免该错误，可用 `\providecommand`。它只会在命令尚未定义时创建它，否则不做任何操作。语法如下：

```
1 \providecommand{命令名称}[参数个数]{替换内容}
```

有时我们需要修改一个已存在的命令（例如修改 `\chapter`、`\section` 的外观，或者替换一个简单的符号），此时需要使用 `\renewcommand`，语法如下。

```
1 \renewcommand{已存在命令名称}[参数个数]{新替换内容}
```

例如可以使用下面命令将无序列表符号从圆形改为方块：

```
1 \renewcommand{\labelitemi}{\ensuremath{\blacksquare}}
```

## 2.8.4 公式中的文本

最常用且推荐的在数学公式中插入文本的命令是 `\text{...}`，使用前需要加载 `amsmath` 宏包。它有如下特点：

- 字体切换：`\text{...}` 会暂时跳出数学模式，切换回当前环境下的文本字体。这意味着它会保持正常的文本样式、粗细和大小。
- 保留空格：`\text{...}` 内部的空格会被保留。
- 自动调整大小：在上下标或分数中，`\text` 内部的字体大小会根据上下文自动缩放。

例如：

```
1 $$
2 \sum_{i=1}^n \left( \frac{1}{i^2} \right) \quad \text{和} \quad \sum_{i=1}^n \left( \frac{1}{i^2} \right)
3 $$
```

效果为：

$$\sum_{i=1}^n \left( \frac{1}{i^2} \right) \quad \text{和} \quad \sum_{i=1}^n \left( \frac{1}{i^2} \right)$$

## 2.8.5 符号和字体的精细控制

### 常规数学字体

在数学模式中，变量默认是斜体，而函数名和单位默认是正体。为了更好地排版， $\text{\LaTeX}$  提供了一系列命令来手动切换数学公式中字母和数字的字体样式。

- `\mathrm{...}` (罗马正体)：用于微分符号、单位、或自定义函数名。例如 `\mathrm{d}x` 的效果为： $dx$ 。
- `\mathbf{...}` (数字粗体)：用于粗体向量或矩阵（仅对英文字母有效）。例如 `\mathbf{v}` 的效果为： $\mathbf{v}$ 。
- `\mathsf{...}` (无衬线)：例如 `\mathsf{E=mc^2}` 的效果为： $E = mc^2$ 。
- `\mathtt{...}` (等宽打印机字体)：例如 `\mathtt{Code}` 的效果为： $\text{Code}$ 。

### 特殊数学字体

- `\mathcal{...}`：常用于集合、算子等（仅限于大写字母）。例如 `\mathcal{T}` 的效果为： $\mathcal{T}$ 。
- `\mathbb{...}`：常用于表示数集，如实数、整数等（仅限于大写字母）。例如 `\mathbb{R}` 的效果为： $\mathbb{R}$ 。
- `\mathfrak{...}`：常用于李代数等。例如 `\mathfrak{A}` 的效果为： $\mathfrak{A}$ 。

## 希腊字母和符号的粗体控制

`\mathbf{...}` 无法使希腊字母和大多数数学符号变粗。此时可以使用 `bm` 宏包。注意，在加载 `bm` 宏包时，确保 `bm` 宏包在 `amsmath` 宏包之后加载（如果同时使用）。语法如下：

```
1 \bm{需要加粗的内容}
```

例如 `$\bm{\sigma}$` 的效果为： $\sigma$

## 2.8.6 定理类环境与结构化数学写作

在  $\text{\LaTeX}$  中，定理类环境是结构化数学写作的基石。它们用于清晰地展示定理、引理、定义、命题等关键概念，并实现自动编号和统一的格式。

这个功能主要由 `amsmath` 宏包的配套宏包 `amsthm` 实现。因此使用前需要先引入宏包 `amsthm`。

### `newtheorem`

所有定理类环境都必须在文档的导言区使用、`\newtheorem` 命令进行定义。基本语法如下：

```
1 \newtheorem{thm/defn/prop/lem}[共享计数器对象]{打印名称}[编号方式]
```

- 共享计数器：和共享计数器对象共享一个计数器。
- 打印名称：实际显示在文档中的文字，例如“定理”、“定义”、“引理”等。
- 编号方式：将定理编号与文档结构（例如 `section`、`chapter` 等）挂钩，并在每章/节开始时重置计数器。

例如：

```
1 % 定义一个名为 'thm' 的环境，打印为 '定理'
2 \newtheorem{thm}{定理}[section]
3 % 定义一个名为 'defn' 的环境，打印为 '定义'
4 \newtheorem{defn}{定义}
5 % 2. 定义引理，使用定理的计数器 [thm]
6 \newtheorem{lem}[thm]{引理}
```

一旦定义，就可以在文档主体中使用，使用方法如下：

```
1 \begin{thm/defn/prop/lem}[该thm/defn/prop/lem的名字]
2 具体内容
3 \end{thm/defn/prop/lem}
```

例如：

```
1 \begin{thm}[连续和可积的关系]
2 如果函数  $f$  在区间  $[a, b]$  上连续，那么它在该区间上必可积。
3 \end{thm}
4
```

```

5 \begin{defn}
6 称函数  $f(x)$  是连续的, 如果对任意  $\epsilon > 0$ , 存在  $\delta > 0$ ,
   使得当  $|x - x_0| < \delta$  时, 有  $|f(x) - f(x_0)| < \epsilon$ 。
7 \end{defn}
8
9 \begin{lem} % 结果: 引理 2.8.2
10 这是一条引理
11 \end{lem}

```

表示:

**定理 2.8.1** (连续和可积的关系). 如果函数  $f$  在区间  $[a, b]$  上连续, 那么它在该区间上必可积。

**定义 1.** 称函数  $f(x)$  是连续的, 如果对任意  $\epsilon > 0$ , 存在  $\delta > 0$ , 使得当  $|x - x_0| < \delta$  时, 有  $|f(x) - f(x_0)| < \epsilon$ 。

**引理 2.8.2.** 这是一条引理

## 样式控制

`amsthm` 宏包提供了 `\theoremstyle` 命令来修改后续 `\newtheorem` 环境的默认外观。它必须在 `\newtheorem` 之前使用。使用方法如下:

```

1 \theoremstyle{plain/definition/remark}
2 \newtheorem{...}

```

- `plain` (默认样式): 标题粗体, 正文斜体。主要用于定理、引理、命题、推论。
- `definition`: 标题粗体, 正文正体。主要用于定义、例题、性质、公理。
- `remark`: 标题斜体、正文正体。主要用于注释、评注、例子。

## 证明环境 `proof`

`amsthm` 宏包提供了一个专门的 `proof` 环境, 用于排版证明文本。使用方法如下:

```

1 \begin{proof}[证明标题]
2 证明内容
3 \end{proof}

```

它会自动在开头打印“证明”(或“Proof.”), 并在末尾添加一个证毕符号 (Q.E.D. 符号, 通常是一个实心小方块)。

例如:

```

1 \begin{proof}
2   证明方式如下...
3 \end{proof}

```

效果为:

证明. 证明方式如下...

□

如果证明的最后一行是一个行间公式环境（如 `align*` 或 `equation*`），证毕符号默认会出现在下一行。为了让它紧跟在公式后面，需要在公式环境内使用 `\qedhere`。

例如：

```
1 \begin{proof}
2   ... 所以我们得到：
3   \begin{equation*}
4     a^2 + b^2 = c^2 \qedhere
5   \end{equation*}
6 \end{proof}
```

效果为：

证明. ... 所以我们得到：

$$a^2 + b^2 = c^2$$

□

## 2.9 文献管理与引用——BibTeX

### 2.9.1 BibTeX 简单介绍

BibTeX 是 L<sup>A</sup>T<sub>E</sub>X 生态系统中处理参考文献和引用的经典且强大的工具。极大地简化了学术写作中的引用管理工作。整个 BibTeX 引用系统涉及三个文件。

- 数据库文件：包含所有引文信息（作者、标题、年份）的纯文本文件。
- 样式文件：定义参考文献列表和正文引用格式的文件。
- 主文档文件：L<sup>A</sup>T<sub>E</sub>X 源代码文件，包含正文和引用命令。

### 2.9.2 编译流程

编译流程为：pdf<sub>l</sub>atex → bib<sub>t</sub>ex → pdf<sub>l</sub>atex → pdf<sub>l</sub>atex

### 2.9.3 .bib 数据库文件

.bib 文件是一个纯文本文件，存储您的所有文献条目。

每个条目都以 @ 符号开头，包含 3 部分：

- 条目类型：
  - @article：期刊文章。
  - @book：整本书。
  - @inproceedings：会议论文。
  - @incollection：书籍中的章节/论文。
  - @phdthesis：学位论文。

– @misc: 无法归类的文献。

- 引用键: 花括号中的第一个参数。这是在 .tex 文件中用来引用的唯一标识符 (cite 命令会用到)。
- 字段: 作者、标题、年份等信息。用逗号分隔。

多个作者使用 AND 连接。

例如:

```
1 @article{
2   feynman1965quantum,
3   author    = {Feynman, Richard P. and Vernon, Frank L. and Hellwarth,
4               Robert W.},
5   title     = {Geometrical representation of the Schrödinger equation
6               for a two-level system},
7   journal   = {Journal of Applied Physics},
8   volume    = {36},
9   number    = {11},
10  pages     = {3560--3562},
11  year      = {1965},
12  publisher = {AIP Publishing}
13 }
```

有关 .bib 条目的信息了解即可。

## 2.9.4 .bst 样式文件

.bst 样式文件定义了引文和参考文献列表的排版规则。它们决定了:

- 文献列表的排序方式 (按作者字母、引用顺序等)。
- 正文引用的格式 (数字、作者-年份)。
- 参考文献条目的具体格式 (粗体、斜体、逗号、句号)。

常见的样式有:

- plain:
  - 按作者字母顺序排序, 正文引用格式为数字。
  - 标准、简介的科技论文格式。
- unsrt:
  - 按引用出现的顺序排序, 正文引用格式为数字。
  - 适合实验报告或按重要性排序的文档。
- abbrev:

- 按作者字母顺序排序，正文引用格式为数字。
- 类似plain，但缩写为期刊名和月份。
- alpha:
  - 按作者字母顺序排序，正文引用格式为字母/数字组合。
  - 使用作者姓氏和年份的组合作为标签。

### 2.9.5 .tex 中实现引用

使用 `cite` 命令将引文插入到正文中。语法如下：

```
1 \cite{文献的引用键}
```

通常需要在文档末尾生成参考文献列表。为实现这一功能，需要在文档末尾（`\appendix` 或 `\end{document}` 之前），添加以下两个核心命令：

1. `\bibliographystyle{样式名}`：指定用于格式化的 `.bst` 文件。
  - 例如：`\bibliographystyle{plain}`。
2. `\bibliography{bib文件名}`：指定你的 `.bib` 文件名（不包含 `.bib` 后缀）。
  - 如果有一个数据库文件名为 `myref.bib`，则使用 `\bibliography{myref}`。

一个例子如下：

```
1 \begin{document}
2   % ... 正文内容 ...
3
4   \nocite{*} % 可选：将 .bib 文件中的所有文献都列出，即使没有在正文中引用。
5
6   \newpage
7   \bibliographystyle{plain}
8   \bibliography{myreferences}
9
10  \end{document}
```

另一个例子：

```
1 参考这篇论文 \cite{10.5555/3295222.3295349}
```

效果为：参考这篇论文 [1]

### 2.9.6 natbib 宏包

虽然 BibTeX 本身功能强大，但其原生的 `\cite` 命令比较单一。`natbib` 宏包（需在导言区加载 `\usepackage{natbib}`）是增强经典 BibTeX 引用的标准工具，它允许更灵活的引用格式：



- `\citep{key}`: 括号引用。
  - 默认数字样式输出: [1]
  - 默认作者-年份样式输出: (Einstein, 1905)
- `\citet{key}`: 作者引用。
  - 默认数字样式输出: Einstein [1]
  - 默认作者-年份样式输出: Einstein(1905)
- `\citeauthor{key}`: 仅作者名。
  - 默认数字样式输出: Einstein
  - 默认作者-年份样式输出: Einstein
- `\citeyear{key}`: 仅年份。
  - 默认数字样式输出: 1905
  - 默认作者-年份样式输出: 1905

# 参考文献

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.