

# 使用 Quarto 编写学术和技术文档

Strik0r

2025-10-27

# 目录

前言	1	4.2 页面布局设置 . . . . .	17
关于本书 . . . . .	1	4.3 字体配置 . . . . .	18
主要任务 . . . . .	1	4.4 数学公式设置 . . . . .	18
目标读者 & 前置知识储备 . . . . .	1	4.5 总结 . . . . .	18
致谢 . . . . .	2		
第一章 环境配置	3	第五章 TikZ 相关设置	21
		5.1 准备工作 . . . . .	22
第二章 项目初始化与项目结构	7	5.2 工作流 . . . . .	23
2.1 项目配置文件示例 . . . . .	9	5.3 完整示例演示 . . . . .	23
第三章 Markdown 基础	13	第六章 Mermaid 基础	27
3.1 标题 . . . . .	13		
3.2 段落 . . . . .	13	第七章 内容渲染与分发	29
3.3 列表 . . . . .	14	7.1 渲染 . . . . .	29
第四章 配置 LaTeX	17	7.2 把内容发布到 GitHub Pages . . . . .	29
4.1 PDF 输出格式配置 . . . . .	17	参考文献	31

# 前言

## 关于本书

This is a Quarto book. To learn more about Quarto books visit <https://quarto.org/docs/books>.

## 主要任务

Markdown 作为一种轻量级的标记语言，在日常的文档编辑中被广泛使用。然而，Markdown 的功能有些“简陋”，并不能满足学术文档、技术文档的较复杂的排版需求。

LaTeX 作为一种专业的排版系统，在学术文档、技术文档的排版中被广泛使用。然而，LaTeX 的语法较为复杂，学习成本较高，对于初学者来说并不友好，即使熟练使用，也会花费大量的时间，效率不高。

Quarto 是 Markdown 的诸多扩展之一，在保持 Markdown 的简洁、易用的基础上，提供了丰富的功能以满足学术文档、技术文档的排版需求。本项目的工作就是把之前基于 LaTeX 的文档内容无缝地迁移到 Quarto 上来，从而实现一份源代码，各种输出格式的高效内容编辑 workflow。

### 成果展示

最近在看 Cohen (2022)，全当是对线性代数做一个偏应用的复习了。复习过程中所做的一些记录（当然是融合了以前的偏理论的内容）可以在 <https://strik0rium.github.io/linear-algebra-web/> 中查看。

## 目标读者 & 前置知识储备

本书不适合在日常写作中不涉及大量公式、或更偏向于 Markdown 特性的读者，也不适合主要使用 LaTeX 且无需 Markdown 特性的用户，或者仅有 PDF 生成需求、没有 HTML 等格式输出和内容分发需求的人群，以及已经使用 Docusaurus 等其他基于 Markdown 的文档工具的读者。

本书适合已经具备一定 Markdown 和 LaTeX 基础，并且希望结合两者优势、提升写作效率的读者。需要使用 LaTeX 的读者，通常会在日常写作中涉及大量公式排版、TikZ 绘图，并希望通过各类 LaTeX 环境来对文档结构进行更灵活的划分。而对 Markdown 有需求的读者，则更希望凭借其简洁的语法，实现高效的图文混排、列表编辑<sup>1</sup>、代码块编辑，并用 callout 特性生成醒目的提示信息等内容。

- 需要对 Markdown 有一定的了解，熟悉 Markdown 的语法，能够使用 Markdown 编写文档。

---

<sup>1</sup>这里的列表指的是无序列表和有序列表

- 需要读者完成 LaTeX 的环境配置，但这并不是必须的。没有完成 LaTeX 配置的读者可以参考 Quarto 的官方文档完成配置。如果你不确定你的 LaTeX 环境是否已经配置完成，可以尝试在系统终端中执行如下命令：

```
$ pdflatex --version
$ xelatex --version
$ lualatex --version
$ latexmk --version
```

如果上述命令都能正确输出版本信息，则说明你的 LaTeX 环境已经配置完成。否则，请参考 Quarto 的官方文档或者 [这个视频](#) 完成配置。

## 致谢

感谢 Cohen (2022) 的作者，让我从此多了一个给偏 CS、DataSci、AI 和 FinTech 方向的同学推荐线性代数学习资料的选择，拒绝同济线性代数、北大高等代数等诸多防自学教材的理由又多了一个。

感谢 ChatGPT 和 Cursor，他们的编码能力实在是很强，帮我节省了许多工作量（但同时也让我多走了很多弯路）。

---

# 第 1 章

## 环境配置

访问 <https://quarto.org/docs/get-started/>，点击 `Download Quarto CLI` 按钮下载 Quarto 的命令行工具，浏览器会自动下载适合你的操作系统的安装包<sup>1</sup>。接下来，在该网站的 Step 2 中选择你喜欢的工具，我们这里使用 VS Code (Cursor) 作为编辑器。

---

<sup>1</sup>如果你不放心的话，也可以在下面的表格里手动选择安装包。

[Overview](#)
[Get Started](#)
[Guide](#)
[Extensions](#)
[Reference](#)
[Gallery](#)
[Blog](#)
[Help](#)

[Get Started](#)  
[Tutorial: Hello, Quarto](#)  
[Tutorial: Computations](#)  
[Tutorial: Authoring](#)

## Get Started

Install Quarto, then check out the tutorials to learn the basics.

### Step 1

Install Quarto

Download Quarto CLI  
1.8.25 (Mac OS) | Date: Sep 30, 2025

Platform	Download	Size	SHA-256
Ubuntu 18+/Debian 10+	<a href="#">quarto-1.8.25-linux-amd64.deb</a>	118.03 MB	<a href="#">48819e4</a>
Linux Arm64	<a href="#">quarto-1.8.25-linux-arm64.deb</a>	118.2 MB	<a href="#">ed5d9f8</a>
Mac OS	<a href="#">quarto-1.8.25-macos.pkg</a>	199.38 MB	<a href="#">31c73d0</a>
Windows	<a href="#">quarto-1.8.25-win.msi</a>	123.25 MB	<a href="#">c449373</a>

[Release notes and more downloads...](#)

### Step 2

Choose your tool and get started

图 1.1: Quarto official site &gt; Get Started

下载到 Quarto CLI 的安装包后，根据你的操作系统上的安装导引完成安装。安装完成后，打开终端，输入 `quarto --version` 命令，如果显示 Quarto 的版本信息，则说明安装成功。

```
$ quarto --version
1.8.25
```

A terminal window titled "strik0r — zsh — 80x24" with standard macOS window controls. The terminal output shows a successful login and the execution of the command "quarto --version", which returns "1.8.25".

```
Last login: Mon Oct 27 09:57:21 on ttys123
(base) strik0r@Strik0rs-HUAWEI-MateBook-Pro ~ % quarto --version
1.8.25
(base) strik0r@Strik0rs-HUAWEI-MateBook-Pro ~ % █
```

图 1.2: 成功安装 Quarto CLI

接下来，打开 VS Code，在插件市场中搜索 Quarto 插件，安装后重启 VS Code 即可。

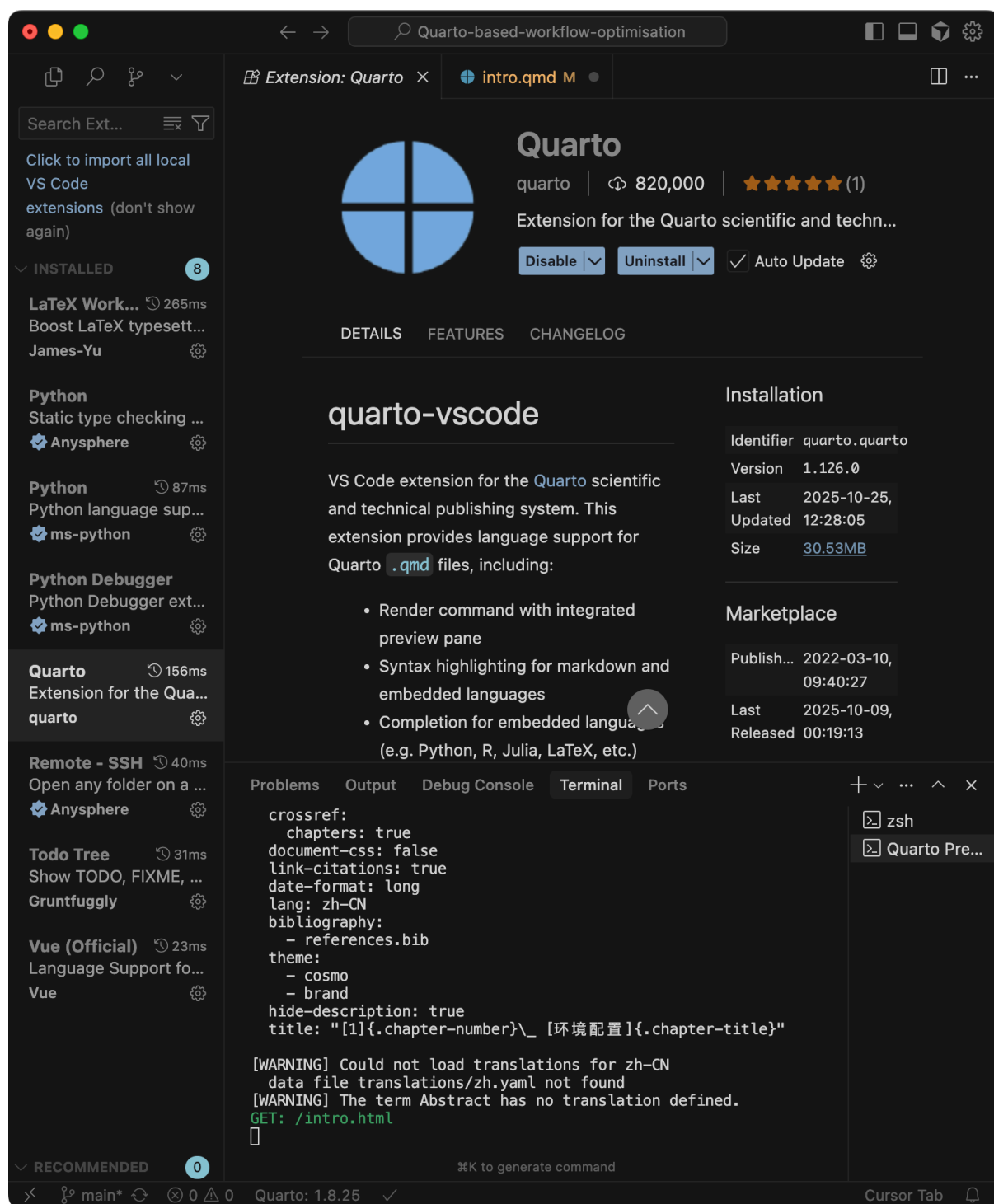


图 1.3: 插件市场 &gt; Quarto



---

## 第 2 章

# 项目初始化与项目结构

在上一节中，我们完成了 Quarto 的安装与配置。接下来，我们将开始创建我们的项目。在 VS Code 中，调用命令面板，快捷键是 `Cmd Shift P`<sup>1</sup>。在命令面板中，输入 `Quarto: New Project` 命令，选择项目模板和项目路径，我们这里选择 `Book` 模板。

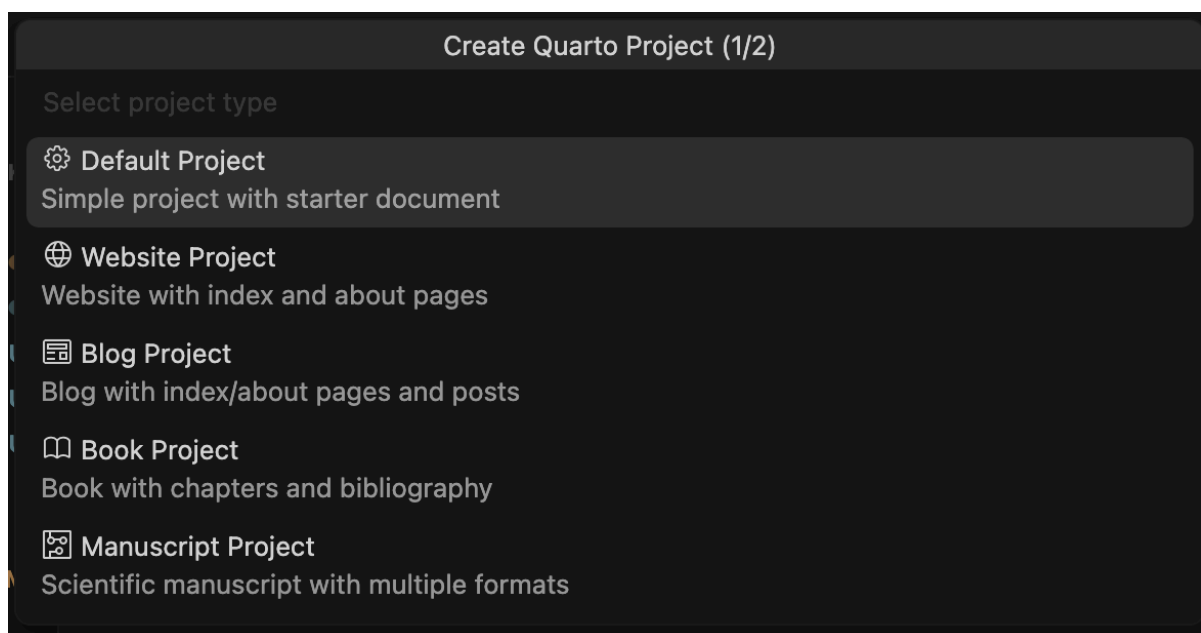


图 2.1: 命令面板 > New Project

新建项目以后，我们可以看到 Quarto 为我们生成了一个项目的基本结构：

---

<sup>1</sup>对于 Windows 用户，快捷键是 `Ctrl Shift P`。

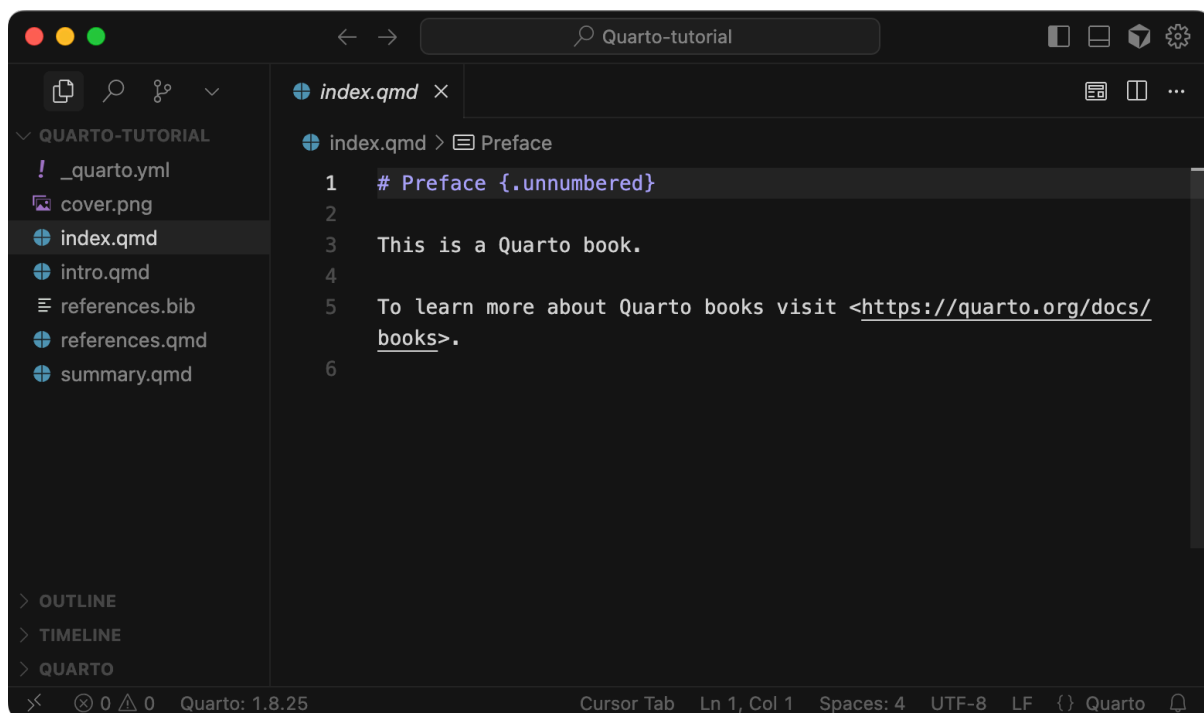


图 2.2: 项目结构

- `_quarto.yml` 是 Quarto 的配置文件，我们的大多数项目配置都在这个文件中进行。
- `cover.png` 是封面图片，它实际上只是一个占位符，我想不会有人恶趣味到选择用这个东西作为自己的书籍的封面。
- `index.qmd` 是书籍的首页，首页中应该包含前言、关于本书、致谢等内容。
- 文件名为 `references` 的两个文件是与参考文献有关的。`references.bib` 是参考文献的 BibTeX 文件，用于生成参考文献列表；`references.qmd` 是参考文献的列表页，用于生成参考文献列表。
- 剩余的 `intro.qmd` 和 `summary.qmd` 是项目初始化中生成的两个示例章节。

在 VS Code 的顶部菜单栏选择 `Terminal > New Terminal` 打开终端（快捷键是 `Control Shift ``），执行以下命令以预览项目：

```
$ quarto preview
```

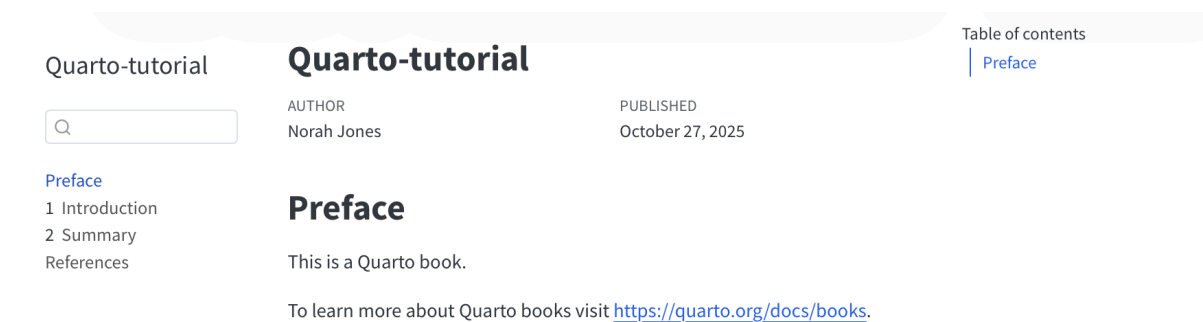


图 2.3: 预览项目

你可以在 `_quarto.yml` 文件中修改项目的语言：

```
lang: zh-CN
```

## 2.1 项目配置文件示例

本项目采用的 `_quarto.yml` 文件内容如下，我们会在稍后的章节中详细介绍这个文件中的各个配置项。

```
project:
  type: book

lang: zh-CN
callout-appearance: simple
bibliography: references.bib
number-sections: false

book:
  title: " 使用 Quarto 编写学术和技术文档"
  author: "Strik0r"
  date: "10/27/2025"
  chapters:
    - index.qmd
    - intro.qmd
    - init-project.qmd
    - markdown-basic.qmd
    - latex-related-settings.qmd
    - tikz-related-settings.qmd
    - mermaid.qmd
    - publish.qmd
    # - summary.qmd
    - references.qmd
```

```
format:
  html:
    include-in-header:
      text: |
        <script>
        window.MathJax = {
          tex: { inlineMath: [['$', '$'], ['\\(', '\\)']] },
          chtml: { scale: 0.80 } // 1.0 = 100%, adjust as you wish
        };
        </script>
    respect-user-color-scheme: true
    css: format/custom.css
    theme:
      # cosmo
      # brand
      light: [cosmo, format/theme.scss]
      dark: [cosmo, format/theme-dark.scss]
  pdf:
    documentclass: ctexbook
    fontsize: 10pt
    # geometry:
    #   - paperwidth=177.8mm
    #   - paperheight=233.4mm
    #   - top=32mm
    #   - bottom=22mm
    #   - left=18mm
    #   - right=18mm
    geometry:
      - paperwidth=210mm
      - paperheight=297mm
      - top=36.8mm
      - bottom=31.8mm
      - left=25.4mm
      - right=25.4mm
    pdf-engine: xelatex
    include-in-header:
      text: |
        \graphicspath{{tikzplots/png/}}
        \setmonofont{JetBrains Mono}
```

```
\input{tex_utils/styles/documentsettings}  
\input{tex_utils/styles/titlesettings}
```



---

## 第 3 章


# Markdown 基础

Markdown 是一种轻量级的标记语言，用于将纯文本转换为格式化的文档。它最初由 John Gruber 和 Aaron Swartz 于 2004 年创建，旨在简化文本格式化过程。Markdown 的语法非常简单，易于学习和使用，因此被广泛应用于各种文档、博客、论坛和代码编辑器中。本章节主要介绍了 Markdown 的基础知识。

### 3.1 标题

Markdown 的标题使用 # 符号来表示，# 的数量表示标题的级别，# 越多，标题的级别越低。例如：

```
# 一级标题
## 二级标题
### 三级标题
#### 四级标题
##### 五级标题
```

 标题级别不建议超过 4 级别，否则可能对 PDF 版本的 LaTeX 渲染产生影响。通常来说，一级标题表示 `chapter`，二级标题表示 `section`，三级标题表示 `subsection`，四级标题表示 `subsubsection`，照这样下去，五级标题就是 `paragraph`。但是经过测试，渲染含有五级标题的 PDF 文档时会报错。

### 3.2 段落

Markdown 的段落使用空行来表示，段落之间使用空行来分隔。

```
这是一个段落，
代码中的换行会被解释为一个空格。
```

```
这是另一个段落，
同样，代码中的换行只能被认为是空格。
```

这是一个段落，代码中的换行会被解释为一个空格。

这是另一个段落，同样，代码中的换行只能被认为是空格。

### 3.3 列表

Quarto Markdown 支持多种列表类型，也支持混用不同样式的列表。

- 无序列表项 1
- 无序列表项 2
- 无序列表项 3

- 无序列表项 1
- 无序列表项 2
- 无序列表项 3

1. 有序列表项 1
2. 有序列表项 2
3. 有序列表项 3

1. 有序列表项 1
2. 有序列表项 2
3. 有序列表项 3

Quarto Markdown 也支持大写字母列表，这一列表使用大写拉丁字母作为标签，非常适合用于选择题的排版：

- A. 大写字母列表项 1
- A. 大写字母列表项 2
- A. 大写字母列表项 3

- A. 大写字母列表项 1
- B. 大写字母列表项 2
- C. 大写字母列表项 3

🔥 大写字母列表项的标签有两个空格。

- A. 只有一个空格
- A. 有两个空格 <!-- 正确 -->

只打一个空格是不能触发 Quarto 的大写字母列表的：

- A. 只有一个空格
- A. 只有一个空格

A. 只有一个空格 A. 只有一个空格

注意到上述代码的渲染结果是一个段落，而不是一个以大写字母为标签的列表。



类似地，还可以将小写字母和复选框作为列表项的标签：

```
a. 小写字母列表项 <!-- 两个空格 -->
- [ ] 复选框列表项
- [x] 复选框列表项（已勾选）
```

a. 小写字母列表项

☐ 复选框列表项

☒ 复选框列表项（已勾选）



---

## 第 4 章

# 配置 LaTeX

本章节详细介绍了本项目中的 LaTeX 相关配置，包括 PDF 输出格式设置、文档类配置、页面布局、字体设置以及各种宏包的使用。这些设置确保了生成的 PDF 文档具有专业的外观和良好的排版效果。

### 4.1 PDF 输出格式配置

在 Quarto 项目的 `_quarto.yml` 配置文件中，通过 `pdf` 相关设置可以灵活地指定 PDF 输出的排版样式、文档类、页面尺寸和各种格式参数，从而实现专业而定制化的排版效果。

本项目使用 `ctexbook` 作为 PDF 输出的文档类，这是专门为中文文档设计的 LaTeX 文档类：

```
pdf:
  documentclass: ctexbook
  fontsize: 10pt
  pdf-engine: xelatex
```

`ctexbook` 是专门为中文文档设计的 LaTeX 文档类，支持中文排版，自动处理中文字体、标点符号等。

`fontsize: 10pt` 设置基础字体大小为 10 点，适合学术文档阅读。

`xelatex` 使用 `xelatex` 作为 PDF 生成引擎，`xelatex` 对 Unicode 和现代字体支持更好，特别适合处理中文字符和自定义字体。

### 4.2 页面布局设置

项目配置了标准的 A4 页面尺寸和合适的边距：

```
geometry:
  - paperwidth=210mm
  - paperheight=297mm
  - top=36.8mm
  - bottom=31.8mm
```

```
- left=25.4mm  
- right=25.4mm
```

- 页面尺寸: 210mm × 297mm (A4 标准)
- 上边距: 36.8mm, 为页眉和章节标题留出空间
- 下边距: 31.8mm, 为页脚留出空间
- 左右边距: 25.4mm, 确保文本不会过于靠近页面边缘

### 4.3 字体配置

在 `_quarto.yml` 文件的 `pdf` 配置中, `include-in-header` 用于向生成的 LaTeX 文档头部插入自定义的 LaTeX 代码片段。这些代码会在每次编译 PDF 时自动插入到 LaTeX 源文件的 `\begin{document}` 之前, 常用于字体设置、引入额外的宏包、定义自定义命令或调整全局排版风格等。例如可以用它为整个文档设定等宽字体、插入自定义标题样式、统一公式渲染模式等配置。从而更灵活地微调最终输出文档的外观与功能。

```
include-in-header:  
  text: |  
    \setmonofont{JetBrains Mono}
```

使用 JetBrains Mono 作为等宽字体, 这是一个专为开发者设计的现代等宽字体, 具有良好的可读性。你可以访问 <https://www.jetbrains.com/idea/mono/> 了解更多关于 JetBrains Mono 的信息和下载该字体。

### 4.4 数学公式设置

```
include-in-header:  
  text: |  
    \everymath{\displaystyle}
```

将所有行内数学公式显示为显示模式, 使公式更加清晰。

### 4.5 总结

本项目的 LaTeX 配置涵盖了:

- 中文文档支持 (ctexbook)
- 现代字体设置 (JetBrains Mono)
- 专业的页面布局
- 完整的数学公式支持
- 规范的标题格式
- 清晰的目录结构

```
pdf:  
  documentclass: ctexbook  
  fontsize: 10pt  
  geometry:
```

```
- paperwidth=210mm
- paperheight=297mm
- top=36.8mm
- bottom=31.8mm
- left=25.4mm
- right=25.4mm
pdf-engine: xelatex
include-in-header:
  text: |
    \setmonofont{JetBrains Mono}
    \everymath{\displaystyle}
```

这些配置确保了生成的 PDF 文档具有专业的外观和良好的可读性，特别适合学术和技术文档的排版需求。



---

## 第 5 章

# TikZ 相关设置

TikZ 是一种强大的绘图工具，可以用于绘制各种复杂的图形，在我的 LaTeX 文档中有大量的图表都是由 TikZ 绘制的。但是对 TikZ 的支持是我在工作流迁移中的最大难点。

⚠ 本方案的缺点与不足

我尝试了所有能匹配 TikZ 关键词的 Quarto 扩展，包括 `diagram`、`imagify` 和 `tikz`，也查阅了 Stack Overflow 上的有关讨论，但是这些方案都不能成功就地编译 TikZ 代码并将绘图结果作为图像输出。并且这些项目的 GitHub 仓库都已经有了十几个月没有更新了，也许是烂尾了？再经过长达 12 个小时的爆肝之后，我于 2025 年 10 月 26 日凌晨 3 点放弃了这看起来有些不切实际的尝试。

Shortcodes and Filters

🔍 tikz

Name	Description	Author
<code>diagram</code>	Generate diagrams from embedded code; supports Mermaid, Dot/GraphViz, PlantUML, Asymptote, and TikZ.	<a href="#">Albert Krewinkel</a>
<code>imagify</code>	Convert embedded LaTeX to images and use .tex/.tikz files as image sources.	<a href="#">Dialoa</a>
<code>tikz</code>	A filter that renders PGF/TikZ diagrams in HTML as SVG.	<a href="#">danmackinlay</a>

1

图 5.1: Quarto 扩展搜索结果

我们的方案采用的是一种比较“笨”的方法，即：将 TikZ 代码编译为 PDF 文件，然后将其转换为 PNG 图像，最后将 PNG 图像插入到 Quarto 文档中。这也算是一种无奈之举了。

! 悬而未决的问题

如果你有机会解决这个问题，请通过 [strik0rium@gmail.com](mailto:strik0rium@gmail.com) 联系我！

5.1 准备工作

在项目的根目录新建一个名为 `tikzplots` 的目录和三个子目录，其中 `tex/` 目录用于存放 TikZ 代码，`pdf/` 目录用于存放 PDF 文件，`png/` 目录用于存放 PNG 文件。

```
tikzplots/  
  tex/  
  pdf/  
  png/
```



## 5.2 工作流

在 `tikzplots/tex/` 目录下新建一个名为 `your_filename.tex` 的文件，并输入你的 TikZ 代码。建议选择 `standalone` 文档类，并进行绘图代码的编写。编写完成后，在终端中执行以下命令以编译 TikZ 代码，并将其转换为 PNG 图像，最后将 PDF 文件和 PNG 图像移动到 `tikzplots/pdf/` 和 `tikzplots/png/` 目录中。

```
cd tikzplots/tex
pdflatex -interaction=nonstopmode your_filename.tex
sips -s format png -Z 1200 your_filename.pdf --out ../png/your_filename.png
mv your_filename.pdf ../pdf/
rm -f your_filename.aux your_filename.log your_filename.out
```

### i 格式转换工具的选择

在上述命令中，我们使用了 `sips` 命令来转换 PDF 文件为 PNG 图像。`sips` 是 macOS 自带的格式转换工具，也可以使用其他工具。

### 💡 最佳实践

你可以根据你选用的工具和具体的执行方案，编写一个 `shell` 脚本来自动化你的工作流，你的 AI 助手会很乐意帮你完成这项工作。

## 5.3 完整示例演示

首先创建几个相关的目录：

```
$ mkdir tikzplots
$ cd tikzplots
$ mkdir -p tex pdf png
```

接下来，在 `tikzplots/tex/rlc-circuit.tex` 文件中输入以下代码：

```
\documentclass{standalone}
\usepackage{tikz}
\usetikzlibrary{arrows, shapes, calc, positioning}
\usepackage{pgfplots}
\usepackage{circuitikz}
\usepackage{amsmath, amssymb, amsfonts, amsthm, mathtools, mathrsfs}
\pgfplotsset{compat=1.18}

\begin{document}
\begin{circuitikz}[scale=0.7]
  \draw (0, 0) to[vsource=$v_{\text{s}}(t)$] (0, 3)
    to[R=1<ohm>] (4, 3)
    to[L=1<henry>] (4, 0)
```

```
to[C=1<farad>] (0, 0);  
\end{circuitikz}  
\end{document}
```

然后执行这些命令编译 TikZ 代码, 并将其转换为 PNG 图像, 最后将 PDF 文件和 PNG 图像移动到 tikzplots/pdf/ 和 tikzplots/png/ 目录中:

```
$ cd tex  
$ pdflatex -interaction=nonstopmode rlc-circuit.tex  
This is pdfTeX, Version 3.141592653-2.6-1.40.26 (TeX Live 2024) (preloaded format=pdflatex)  
...  
Output written on rlc-circuit.pdf (1 page, 31309 bytes).  
Transcript written on rlc-circuit.log.  
$ sips -s format png -Z 1200 rlc-circuit.pdf --out ../png/rlc-circuit.png  
$ mv rlc-circuit.pdf ../pdf/  
$ rm -f rlc-circuit.aux rlc-circuit.log rlc-circuit.out
```

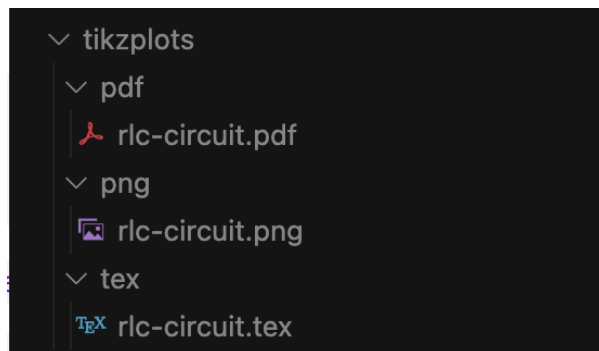


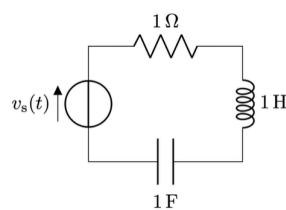
图 5.2: 上述命令行命令的执行效果

接下来在项目正文中插入新生成的 PNG 图像。

```
6
7 RLC circuits are a type of electrical circuit that consists of a
8 resistor, an inductor, and a capacitor connected in series or
9 parallel. They are used to filter signals and to store energy. Here
  is an example of an RLC circuit:
```

(a) 插入 PNG 图像

RLC circuits are a type of electrical circuit that consists of a resistor, an inductor, and a capacitor connected in series or parallel. They are used to filter signals and to store energy. Here is an example of an RLC circuit:



An RLC circuit

(b) 预览效果

图 5.3: 在项目正文中插入新生成的 PNG 图像



---

## 第 6 章

# Mermaid 基础

Blocked Blocked -> Ready Ready -> Running Running -> Ready

```
`<!-- quarto-file-metadata: eyJyZXNvdXJjZURpciI6Ii4ifQ== -->`{=html}
```

```
````{=html}
```

```
<!-- quarto-file-metadata: eyJyZXNvdXJjZURpciI6Ii4iLCJib29rSXRlbVR5cGUiOiJjaGFwdGVyIiwiaWm9va0l0ZW10dW1lZXIiOiJjcsI
```



---

## 第 7 章

# 内容渲染与分发

### 7.1 渲染

使用如下命令渲染内容：

```
$ quarto render
```

### 7.2 把内容发布到 GitHub Pages

新建一个 GitHub 仓库，然后执行

```
$ quarto publish gh-pages
```





## 参考文献

Cohen, M. X. 2022. *Practical Linear Algebra for Data Science*. O'Reilly Media. <https://books.google.com.hk/books?id=GHmHEAAAQBAJ>.

