中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

本 科 毕 业 设 计（论文）

题　目：中国石油大学本科毕业论文模板 (latex 实现)

学生姓名：张英杰

学　　号：

专业班级：

指导教师：

2020 年 8 月 9 日

目　录

第 1 章　基础知识

1.1 运行 latex

安装好 texlive 环境后，使用 xelatex 引擎编译 main.tex 文件来得到

1.2 组织章节、目录

一些简单的命令：

1.3 简单的文本组织

以符号%开头的行是注释。LaTeX 会完全忽略百分号后的内容，不使其输出到产生的 pdf 文件中。

latex 会自动对段落进行缩进，并且会忽略掉段前的空格。中文句子中的空格会被全部忽略，英文句子中两个单词之间的多个空格会被视为一个空格，多个空格也不会加大单词之间的间距。中英文进行混排时，xelatex 也会自动的处理空格。

示例：

源文件中的原文：

```
中　　国　　石油　　大学China　　University　　Of　　Petroleum
```

实际的打印效果：

中国石油大学 China University Of Petroleum

在 latex 中，单个换行（回车）其实就相当于一个空格，不会使文字另起一段，而是起使源代码更易阅读的作用。使用两个反斜杠\\或命令\newline可以进行换行，但不会另起一段。要另起一段，使用\par命令或者使用两个及以上的换行（多敲几次回车）。

注意，多个换行只会起到分段作用，不会在文件中输出其他的空白行。这一点与 word 不同。

示例：

源文件中的原文：

```
一二三四五六七
一二三四五六七


一二三四五六七\\
一二三四五六七
```

一二三四五六七\par
一二三四五六七

一二三四五六七一二三四五六七

一二三四五六七

一二三四五六七

一二三四五六七

一二三四五六七

空白 hspace

1.4 命令与环境

LaTeX 中的控制命令（宏）以反斜线开头，后跟命令名，并且可以传入一些参数。用来实现一些强大的功能。

latex 的许多样式依靠环境来实现

1.4.1 公式与定理环境

行内公式 $E = mc^2$ 用两个美元符号将公式内容包裹起来即可

带编号的行间公式环境

不编号的行间公式环境

$$e = mc^2$$

$$E = mc2 \tag{1-1}$$

对编号的公式进行引用

$$e = mc^2 \tag{1-2}$$

引用：式 (1-2) 。

定理类环境

定义 1.1 定义环境

定理 1.1 定理环境

注 1.1 是一个定理

引理 1.1 引理环境

命题 1.1 命题环境

推论 1.1 推论环境

例 1.1 例环境

注 1.1 注环境

定义 1.1 定理 1.1 引理 1.1 命题 1.1 推论 1.1 例 1.1 注 1.1

证明： 证明环境

□

解： 解环境

定理 1.2 证明可以放进定理类环境里面

证明： 这是证明环境

□

例 1.2 解也一样

解： 这是解

1.5 多行公式

在行间公式环境中包裹 align 环境或 aligned 环境，通过符号来进行换行，符号来进行对齐

$$\begin{aligned} a &= b + c + d + e \\ a + b &= c + d + e \end{aligned} \tag{1-3}$$

文本

`\mathbf{}` 公式环境内的粗体 `\textbf{}` 文本的粗体 `\emph{}` 斜体强调 `\hspace{1em}` 产生一段横向的空白，长度为当前字号下一个字符”m” 的宽度。`\vspace{1em}` 产生一段纵向的空白，长度为当前字号下一个字符”m” 的高度。一些常用的特殊字符被 LaTeX 作为控制字符，如果要实际的打印出他们，需要使用反斜杠\进行转义。特殊字符包括`\,#,%,$,&,_,^,{,},~`等。

1.6 代码和抄录环境

1.7 图表环境

比较复杂，另起一章说明

第 2 章　交叉引用

2.1 参考文献的引用

单个引用

多重引用

连续引用

2.2 内容的引用

通过`\label`和`\autoref`这两个命令来完成论文中元素的标记和引用。

第 3 章　模板功能

3.1 打印模式

论文答辩时需要准备三份纸质论文，而纸质论文都是双面打印。我们的论文都是单面排版的，打印的时候就需要在特定的位置加入空白页。

我们当时的要求是：在封面，中英摘要，目录，致谢，参考文献后添加空白页，并且空白页的位置必须是 **实际页码**的偶数页。纸质版论文除了空白页外其他部分与电子版论文完全一致。

本模板中暂时按此要求实现了双面打印宏 `\twoSidePrint{}` ，定义在在 main.tex 中导言区的位置，将其取消注释后编译即可获得双面打印效果支持，在需要的位置自动插入空白页。如果后续要求变更，再按照实际的要求进行修改。

具体实现的命令和行为：

`\ClearPageStyle`命令，若`\twoSidePrint{}` 未定义，无任何影响，否则添加空白页。用在无页码的封面页、中英摘要页、目录页之后。

`\PrintModeSubfile`环境，若`\twoSidePrint{}` 未定义，仅导入一子文件，否则导入子文件后根据起始页码与结束页码判断此子文件的页数，若为奇数，添加一空白页，并且空白页不会编页，页码仍与单面打印时保持一致。直接作用在章节的导入过程。

注意事项：一般来说，中英摘要都只有一页，目录可能会有多页。如果中英摘要或目录的页数是偶数，如 2 页或 4 页，`\ClearPageStyle`命令插入的空白页就会成为多余的。这时你需要手动将它注释掉来去掉这一多余的空白页，达到你实际想要的打印效果。

第 4 章　结构化编写

为了使论文的组织结构清晰明确，本模板实现了论文的分章节编译，你可以在每个单章的文件中编写内容并输出为 pdf 文件。当每个章节都编写完成后，在 main.tex 中组织和编译它们来得到完整的论文。

需要注意的部分：在对单章进行编译时，**跨文件**的交叉引用是失效的。不过只要你引用的标签是正确的，忽略产生的“There were undefined references.”警告即可，因为在主文件 main.tex 中编译时交叉引用就会恢复正常。

导入图片时，应该使用图片相对于你正编写的文件的相对路径。

第 5 章　浮动体

5.1 表格环境

普通的三线表

表 5–1　激光入射功率密度对导轨滚道表面硬化层深和显微硬度的影响

试验编号	功率密度	辐照时间	显微硬度	硬化层深
t-1	6.37×103	0.067	570，456	0.354
t-2	6.37×103	0.067	570，456	0.354
t-3	6.37×103	0.067	570，456	0.354
t-4	6.37×103	0.067	570，456	0.354
t-5	6.37×103	0.067	570，456	0.354

表 5–1

表格环境中有几个参数，h、t、b、p，这代表这个表格将会优先按照在页面的此处、顶部、底部、的顺序依次尝试插入这个表格，因为表格过大的时候可能会造成在页面在已经有了文字的情况下剩余的空间不足以放下这个表格。

tabular 环境的参数有 c/l/r 三种，分别代表此列元素按照中心/左/右对齐的格式排版

small 命令使表格内部使用小五号字体 caption 命令表示图标的标题，内部可以使用行间数学环境

当表格比较大时，`\table` 环境就无法满足我们的要求了，

第 6 章　审阅

当你进行到了论文的最终阶段，会需要和你的导师进行频繁的交流来对论文进行修改。pdf 文件的批注功能弱于 word 的审阅功能。

使用宏包 changes 可以实现简单的修改注解。

标志更改者

列出变动内容

第 7 章　自定义简单宏

论文中一般会频繁的出现一些较长的符号或专有名词，用简单的宏命令就可以方便的进行输入和全局修改。这有点像 C 语言里面的宏，LaTeX 编译器会使用命令中定义的文本内容代替实际的命令。

定义新的宏命令的方法：

```
\newcommand{commandName}[argumrntNum]{def}
```

commandName 是定义的新命令的名字，使用时以反斜线开头，区分大小写，可以使用汉字, 不能和 LaTeX 中自带的和模板中已经定义的命令名重复。

注意由于命令的名字可以是汉字，像

```
\par一些文字
```

这样的命令会出现“Undefined control sequence”错误，因为编译器将命令\par后面的汉字也识别为变量名的一部分。为了避免这种情况，应该像

```
\par 一些文字
```

这样，用空格将命令和文本分开。

方括号中的 arguumentNum 是传入参数的个数，必须为数字，最大为 9。如果不需要传参数，不写方括号即可。

def 是宏的内容，传入的参数可以在内容中使用#1,#2,…,#9来表示和区分。

简单的例子：

一个无参的例子：

```
\newcommand{vga}{VGA（Video Graphics Array）视频图形阵列}
```

2个参数的情况：

```
\newcommand{\ffield}[2]{有限域$\mathbf{\mathbb{#1}}_{#2}$}
```

命令的使用：

```
\ffield{F}{q}
\ffield{G}{a}
```

产生的效果如下：

有限域 $\mathbb{F}_q$

有限域 $\mathbb{G}_a$

建议把自定义的简单宏都放在导言区进行统一管理，不要到处乱放，以免造成混乱。模板中提供的一些简单宏命令：

字体命令

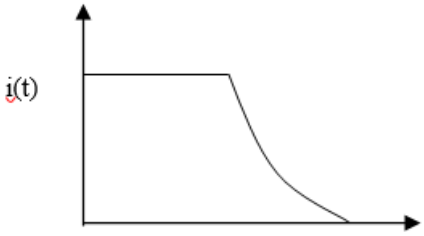

图 7–1　过阻尼响应

第 8 章　代码环境

伪代码、算法环境，内部可以使用数学模式来表达。

Algorithm 1 PARTITION(A, p, r)

1: $i = p$
2: **for** $j = p$ to r **do**
3: **if** $A[j] <= 0$ **then**
4: $swap(A[i], A[j])$//交换两个值
5: $i = \sum_{j=0}^{n} x^k$
6: **end if**
7: **end for**

算法 1

代码环境，适合简短代码的情况。

```
int main(int argc, char const *argv[])
{
        /* code */
        return 0;
}
```

从文件中插入代码（推荐），相当于对代码环境做了封装，直接从文件中导入代码。建议将代码文件放在主目录的 code/目录内，然后通过`\lstinputlisting{路径}`命令将代码引入。注意使用相对路径。

这样的好处是代码修改后可以直接编译，省去了复制粘贴的步骤，能有效防止因疏忽导致的最终论文中代码与实际代码不一致的情况。

```
#include "LinearList.h"
#include <iostream>
using namespace std;


void print(int i)
{
```

```
8           cout << i << " ";
9   }
10
11
12
13
14  // void ListInversion(LinearList &L)
15  // {
16  //      Node next, now, pre;
17  //      pre = L->head;
18  //      now = pre->next;
19
20  //      if (L->length <= 1)return;
21  //      while (now != NULL)
22  //      {
23  //              next = now->next;
24  //              now->next = pre;
25  //              pre = now;
26  //              now = next;
27  //      }
28
29  //      L->head->next->next = NULL;
30  //      L->head->next = pre;
31  // }
32
33  void ListInversion(LinearList &L)
34  {
35          LinearList L0;
36          ListInit(L0);
37
```

```
38
39          Node temp;
40          while (L->head->next != NULL)
41          {
42                  temp = L0->head->next;
43                  L0->head->next = L->head->next;
44                  L->head->next = L->head->next->next;
45                  L0->head->next->next = temp;
46          }
47
48          L->head->next = L0->head->next;
49          L0->head->next = NULL;
50          ListDestroy(L0);
51
52  }
53  int main(int argc, char const *argv[])
54  {
55
56          LinearList L;
57          int a[] = {1, 2, 3, 4};
58
59          ListCreatWithArray(L, 4, a);
60          ListTraverse(L, print);
61          cout << endl;
62          ListInversion(L);
63          ListTraverse(L, print);
64          cout << endl;
65          ListDestroy(L);
66
67          return 0;
```

```
68 }
```