



Text to Speech

ElegantL^AT_EX 经典之作

作者：陈梦楠

组织：ElegantL^AT_EX Program

时间：May 2, 2022

版本：4.1

自定义：信息



各人自扫门前雪，休管他人瓦上霜。——陈元靓

特别声明

自 2019 年 ElegantL^AT_EX 系列模板上线 GitHub、CTAN 以来，受到很多用户的喜爱。

2020 年，我打算做 ElegantL^AT_EX 的最后一个版本，也就是原定计划 ElegantBook 4.x 版本为 ElegantL^AT_EX 系列模板的终止符。基于我想把 4.x 做成一个最终版本，我计划了很多事情，包括将代码转为 dtx，将三个模板的文档打包进 dtx 里面，然后重新设计封面，补充各种页面，增加元素等等。我想的很多，但是做起来并不是很顺利，中间也发生了很多事情，不想解释。直至今年 4 月，我决定，不论如何，先把 4.1 发布出来。

另外，在临近 ElegantL^AT_EX 模板告别之际，我想和各位用户说：多分享，多奉献。

如果你无法认同我的想法，建议直接删除本模板。

Ethan Deng
May 2, 2021

目录

1 语音合成概述	1
1.1 Awesome List	1
1.2 参考书籍	1
1.3 语音相关的会议、期刊、比赛和公司	1
1.3.1 会议	1
1.3.2 期刊	1
1.3.3 最新论文	1
1.3.4 比赛	2
1.3.5 公司	2
1.3.6 微信公众号	2
1.3.7 CCF 语音对话与听觉专业组	2
1.4 数据集	2
1.4.1 中文数据集	2
1.4.2 英文数据集	3
1.4.3 情感数据集	3
1.4.4 其它	3
1.4.5 开源工具	3
1.4.6 开源项目	3
1.5 语音合成评价指标	4
1.6 平均意见得分的测评要求与方法	4
1.6.1 实验要求	5
1.6.2 实验方法	5
1.6.3 实验步骤	5
1.6.4 实验设计	5
1.6.5 实验数据处理	6
2 语音信号基础	7
2.1 参考资料	7
2.2 基本概念	7
2.3 音频格式	8
2.4 发音学	8
2.5 数字信号处理	8
2.5.1 模拟信号到数字信号的转换 (Analog to Digital Converter, ADC, 模数转换)	8
2.5.2 傅里叶变换	8
2.5.3 频谱泄露	8
2.5.4 频率分辨率	8
3 语音特征提取	9
3.1 预处理	9
3.1.1 预加重	9
3.1.2 分帧	9
3.1.3 加窗	9

3.2 短时傅里叶变换	10
3.3 听觉特性	11
3.3.1 梅尔滤波	11
3.3.2 Bark 滤波	12
3.4 倒谱分析	12
3.5 常见的声学特征	13
3.5.1 FBank	13
3.5.2 MFCC	14
3.6 具体操作	14
3.6.1 利用 librosa 读取音频	14
3.6.2 提取梅尔频谱	15
3.6.3 提取 MFCC	16
4 声学模型	18
4.1 Tacotron	18
4.2 FastSpeech	20
4.2.1 模型结构	20
4.2.2 损失函数	24
4.2.3 小结	25
4.3 VITS	25
4.3.1 模型整体结构	26
4.3.2 变分推断	26
4.3.3 对齐估计	27
4.3.4 对抗训练	27
4.3.5 总体损失	28
4.3.6 总结	28
5 声码器	29
5.1 Griffin Lim 声码器	29
5.1.1 代码实现	29
5.2 ElegantBook 更新说明	30
5.3 模板安装与更新	30
5.3.1 在线使用模板	30
5.3.2 本地免安装使用	30
5.3.3 发行版安装使用	31
5.3.4 更新问题	31
5.3.5 其他发行版本	31
5.4 关于提交	31
6 ElegantBook 设置说明	32
6.1 语言模式	32
6.2 设备选项	32
6.3 颜色主题	32
6.4 封面	33
6.4.1 封面个性化	33
6.4.2 封面图	33

6.4.3 徽标	34
6.4.4 自定义封面	34
6.5 章标题	34
6.6 数学环境简介	34
6.6.1 定理类环境的使用	35
6.6.2 其他环境的使用	35
6.7 列表环境	35
6.8 参考文献	36
6.9 添加序章	36
6.10 目录选项与深度	36
6.11 章节摘要	37
6.12 章后习题	37
第 6 章 练习	37
6.13 旁注	38
7 字体选项	39
7.1 数学字体选项	39
7.2 使用 newtx 系列字体	39
7.2.1 连字符	39
7.2.2 宏包冲突	39
7.3 中文字体选项	40
7.3.1 方正字体选项	40
7.3.2 其他中文字体	40
8 ElegantBook 写作示例	42
8.1 Lebesgue 积分	42
8.1.1 积分的定义	42
第 8 章 练习	44
9 常见问题集	45
10 版本更新历史	46
A 基本数学工具	49
A.1 求和算子与描述统计量	49

第1章 语音合成概述

Elegant \LaTeX 项目组致力于打造一系列美观、优雅、简便的模板方便用户使用。目前由 **ElegantNote**, **ElegantBook**, **ElegantPaper** 组成，分别用于排版笔记，书籍和工作论文。强烈推荐使用最新正式版本！本文将介绍本模板的一些设置内容以及基本使用方法。如果您有其他问题，建议或者意见，欢迎在 GitHub 上给我们提交 issues 或者邮件联系我们。

我们的联系方式如下，建议加入用户 QQ 群提问，这样能更快获得准确的反馈，加群时请备注 \LaTeX 或者 Elegant \LaTeX 相关内容。

- 官网：<https://elegantlatex.org/>
- GitHub 网址：<https://github.com/ElegantLaTeX/>
- CTAN 地址：<https://ctan.org/pkg/elegantbook>
- 下载地址：正式发行版，最新版
- 微博：Elegant \LaTeX
- 微信公众号：Elegant \LaTeX
- 用户 QQ 群：692108391
- 邮件：elegantlatex2e@gmail.com

1.1 Awesome List

1. <https://github.com/wenet-e2e/speech-synthesis-paper>
2. <https://github.com/dllBoJack/Speech-Resources>
3. <https://github.com/sindresorhus/awesome>

1.2 参考书籍

1. 神经网络与深度学习
2. Tan X, Qin T, Soong F, et al. A survey on neural speech synthesis[J]. arXiv preprint arXiv:2106.15561, 2021.

1.3 语音相关的会议、期刊、比赛和公司

1.3.1 会议

1. INTERSPEECH (Conference of the International Speech Communication Association)
2. ICASSP (IEEE International Conference on Acoustics, Speech and Signal Processing)
3. ASRU (IEEE Automatic Speech Recognition and Understanding Workshop)
4. ISCSLP (International Symposium on Chinese Spoken Language Processing)
5. ACL (Association of Computational Linguistics)

1.3.2 期刊

1. Computer Speech and Language

1.3.3 最新论文

1. <https://arxiv.org/list/eess.AS/recent>

2. <https://arxiv.org/list/cs.SD/recent>
3. <https://arxiv.org/list/cs.CL/recent>
4. <https://arxiv.org/list/cs.MM/recent>

1.3.4 比赛

1. Blizzard Challenge
2. CHiME: Computational Hearing in Multisource Environment
3. NIST

1.3.5 公司

1. 微软
2. 谷歌云
3. 捷通华声
4. Nuance
5. Amazon polly
6. 百度（翻译）
7. 搜狗开发平台
8. 搜狗（翻译）
9. 有道开放平台
10. 有道（翻译）
11. 微软（翻译）
12. Google 翻译

1.3.6 微信公众号

1.3.7 CCF 语音对话与听觉专业组

1. <https://history.ccf.org.cn/tfsdap/index.html>

1.4 数据集

1.4.1 中文数据集

1. **标贝中文标准女声音库**: 中文单说话人语音合成数据集，质量高。
2. **THCHS-30**: 中文多说话人数据集，原为语音识别练手级别的数据集，也可用于多说话人中文语音合成。
3. **Free ST Chinese Mandarin Corpus**: 855 个说话人，每个说话人 120 句话，有对应人工核对的文本，共 102600 句话。
4. **zhvoice**: zhvoice 语料由 8 个开源数据集，经过降噪和去除静音处理而成，说话人约 3200 个，音频约 900 小时，文本约 113 万条，共有约 1300 万字。
5. **滴滴 800+ 小时 DiDiSpeech 语音数据集**: DiDi 开源数据集，800 小时，48kHz，6000 说话人，存在对应文本，背景噪音干净，适用于音色转换、多说话人语音合成和语音识别，参见：<https://zhuanlan.zhihu.com/p/268425880>。
6. **SpiCE-Corpus**: SpiCE 是粤语和英语会话双语语料库。

1.4.2 英文数据集

1. **LJSpeech**: 英文单说话人语音合成数据集，质量较高。
2. **VCTK**: 英文多说话人语音数据集，109个说话人，每人400句话，采样率48kHz，位深16bits。
3. **TIMIT**: 630个说话人，8个美式英语口音，每人10句话，采样率16kHz，位深16bits。[这里是具体下载地址](#)，下载方法：首先下载种子，然后执行：


```
1 ctorrent *.torrent
```
4. **CMU ARCTIC**: 共18个说话人。语音质量较高，可以用于英文多说话人的训练。

1.4.3 情感数据集

1. **ESD**: 用于语音合成和语音转换的情感数据集。
2. **情感数据和实验总结**: 实际是情感语音合成的实验总结，包含了一些情感数据集的总结。

1.4.4 其它

1. **Opencpop**: 高质量歌唱合成数据集。
2. **好未来开源数据集**: 目前主要开源了3个大的语音数据集，分别是语音识别数据集，语音情感数据集和中英文混合语音数据集，都是多说话人教师授课音频。
3. **AISHELL-2**: 希尔贝壳开源了不少中文语音数据集，AISHELL-2是最近开源的一个1000小时的语音数据库，禁止商用。官网上还有其它领域，比如用于语音识别的4个开源数据集。
4. **CSS10**: 十个语种的单说话人语音数据的集合。
5. **OpenSLR**: OpenSLR是一个专门托管语音和语言资源的网站，例如语音识别训练语料库和与语音识别相关的软件。迄今为止，已经有100+语音相关的语料。
6. **voice datasets**: Github上较为全面的开源语音和音乐数据集列表，包括语音合成、语音识别、情感语音数据集、语音分离、歌唱等语料，找不到语料可以到这里看看。
7. **Open Speech Corpora**: 开放式语音数据库列表，特点是包含多个语种的语料。
8. **EMIME**: 包含一些TTS和ASR模型，以及一个中文/英语，法语/英语，德语/英语双语数据集。
9. **Celebrity Audio Extraction**: 中国名人数据集，包含中国名人语音和图像数据。

1.4.5 开源工具

1. **sonic**: 语音升降速工具。
2. **MFA**: 从语音识别工具Kaldi中提取出来的音素-音频对齐工具，可以利用MFA获取每一个音素的时长，供预标注或时长模型使用。
3. **宾西法尼亚大学强制对齐标注软件（P2FA）**: [这里有相关的介绍](#)，对于噪音数据鲁棒性差。
4. **ABXpy**: 语音等测评ABX测试网页。
5. **SpeechSubjectiveTest**: 主观测评工具，包括用于语音合成和转换的MOS、PK（倾向性测听）、说话人相似度测试和ABX测试。
6. **Matools**: 机器学习环境配置工具库
7. **MyTinySTL**: 基于C++11的迷你STL。
8. **CppPrimerPractice**: 《C++ Primer中文版（第5版）》学习仓库。
9. **git-tips**: Git的奇技淫巧。

1.4.6 开源项目

1. **coqui-ai TTS**: 采用最新研究成果构建的语音合成后端工具集。

2. **ESPNet**: 语音合成和识别工具集，主要集成后端模型。
3. **fairseq**: 序列到序列建模工具集，包含语音识别、合成、机器翻译等模型。
4. **eSpeak NG Text-to-Speech**: 共振峰生成的语音合成模型，集成超过 100 个语种和口音的语音合成系统。
5. **Epitrans**: 将文本转换为 IPA 的工具，支持众多语种。
6. **Tacotron-2**: Tensorflow 版本的 Tacotron-2。
7. **Text-to-speech in (partially) C++ using Tacotron model + Tensorflow**: 采用 Tensorflow C++ API 运行 Tacotron 模型。

1.5 语音合成评价指标

对合成语音的质量评价，主要可以分为主观和客观评价。主观评价是通过人类对语音进行打分，比如平均意见得分（Mean Opinion Score, MOS）、众包平均意见得分（CrowdMOS, CMOS）和 ABX 测试。客观评价是通过计算机自动给出语音音质的评估，在语音合成领域研究的比较少，论文中常常通过展示频谱细节，计算梅尔倒谱失真（Mel Cepstral Distortion, MCD）等方法作为客观评价。客观评价还可以分为有参考和无参考质量评估，这两者的主要判别依据在于该方法是否需要标准信号。有参考评估方法除了待评测信号，还需要一个音质优异的，可以认为没有损伤的参考信号。常见的有参考质量评估主要有 ITU-T P.861 (MNB)、ITU-T P.862 (PESQ)、ITU-T P.863 (POLQA)、STOI 和 BSSEval。无参考评估方法则不需要参考信号，直接根据待评估信号，给出质量评分，无参考评估方法还可以分为基于信号、基于参数以及基于深度学习的质量评估方法。常见的基于信号的无参考质量评估包括 ITU-T P.563 和 ANIQUE+，基于参数的方法有 ITU-T G.107(E-Model)。近年来，深度学习也逐步应用到无参考质量评估中，如：AutoMOS、QualityNet、NISQA 和 MOSNet。

主观评价中的 MOS 评测是一种较为宽泛的说法，由于给出评测分数的主体是人类，因此可以灵活测试语音的不同方面。比如在语音合成领域，主要有自然度 MOS (MOS of Naturalness) 和相似度 MOS (MOS of Similarity)。但是人类给出的评分结果受到的干扰因素较多，谷歌对合成语音的主观评估方法进行了比较，在评估较长语音中的单个句子时，音频样本的呈现形式会显著影响参与人员给出的结果。比如仅提供单个句子而不提供上下文，与相同句子给出语境相比，被测人员给出的评分差异显著。国际电信联盟 (International Telecommunication Union, ITU) 将 MOS 评测规范化为 ITU-T P.800，其中绝对等级评分 (Absolute Category Rating, ACR) 应用最为广泛，同时也是本文中采用的语音质量评估标准，ACR 的详细评估标准如下表所示。

表 1.1: 主观意见得分的评估标准

音频级别	平均意见得分	评价标准
优	5.0	很好，听得清楚；延迟小，交流流畅
良	4.0	稍差，听得清楚；延迟小，交流欠流畅，有点杂音
中	3.0	还可以，听不太清；有一定延迟，可以交流
差	2.0	勉强，听不太清；延迟较大，交流需要重复多遍
劣	1.0	极差，听不懂；延迟大，交流不通畅

在使用 ACR 方法对语音质量进行评价时，参与评测的人员（简称被试）对语音整体质量进行打分，分值范围为 1~5 分，分数越大表示语音质量越好。MOS 大于 4 时，可以认为该音质受到大部分被试的认可，音质较好；若 MOS 低于 3，则该语音有比较大的缺陷，大部分被试并不满意该音质。

1.6 平均意见得分的测评要求与方法

语音合成的最终目标是，合成语音应尽可能接近真实发音，以至于人类无法区分合成和真实语音。因此让人类对合成语音进行评价打分是最为直观的评价方法，评分经处理之后即可获得平均意见得分。平均意见得分是语音合成系统最重要的性能指标之一，能够直接反映合成语音的自然度、清晰度以及可懂度。

1.6.1 实验要求

获取多样化且数量足够大的音频样本，以确保结果在统计上的显著，测评在具有特定声学特性的设备上进行，控制每个被试遵循同样的评估标准，并且确保每个被试的实验环境保持一致。

1.6.2 实验方法

为了达到实验要求，可以通过两种方法获得足够精确的测评结果。第一种是实验室方式，该方式让被试在实验室环境中进行测评，在试听过程中环境噪音必须低于 35dB，测试语音数量至少保持 30 个以上，且覆盖该语种所有音素和音素组合，参与评测的被试应尽可能熟练掌握待测合成语音的语种，最好以合成语音的语种为母语。该方法的优点是测试要素容易控制，能够稳定保证实验环境达到测评要求；缺点则主要是需要被试在固定场所完成试听，人力成本高。第二种是众包，也就是将任务发布到网络上，让具有条件的被试在任何地方进行测评。该方法主要优点是易于获得较为有效的评估结果；而缺点则体现在无法确保试听条件。

1.6.3 实验步骤

1. 收集合成语音和录制的真实语音；
2. 确保文本和语音一一对应，去除发音明显错误的音频样本；
3. 生成问卷，将合成语音和真实语音交叉打乱，确保打乱的顺序没有规律，合成语音和真实语音不可让被试提前探知到；
4. 开始任务前，被试试听示例语音，并告知其对应的大致得分；
5. 被试开始对给定音频打分，前三条语音可以作为被试进入平稳打分状态的铺垫，不计入最终结果；
6. 回收问卷，舍弃有明显偏差的评价数据，统计最终得分。

1.6.4 实验设计

1. 准备测试语音数据。(1)从各领域和语音合成系统实际应用场景中，摘选常规文本作为测试语料，选取的语句一般尽可能排除生僻字；(2)用于测试的句子一般是未出现在训练集中的；(3)被试必须使用耳机试听语音，以便于判断更为细微的差别；(4)为了避免被试的疲惫，待测评系统和语料数量不可太多，需要控制测评时间；(5)一个句子需要由多个被试打分。
2. 设置实验参数。在准备测试语音时，需要提前设置好训练语料、待测系统、参与测试的句子数量、每个句子被试听的次数等。以中文语音合成系统的语音评估为例，测评设置如下表所示。

表 1.2: 语音测评设置

训练集	待测系统	句子数量	每个句子被测次数
内部数据集	真实语音	40	12
内部数据集	Tacotron-2	40	12
内部数据集	FastSpeech-2	40	12

3. 准备 HTML 文档等展示材料，向被试介绍该测试。该 HTML 文档至少包括：(1) 测试注意事项，如被试应该使用何种设备，在何种环境下试听，试听时应该排除的干扰因素等；(2) 测试任务，向被试介绍本次试听的测试目标，应关注的侧重点，如：可懂度、相似度、清晰度等方面；(3) 参考音频，可以放置一些示例音频，如 MOS=5 的优质语音，MOS=1 的低劣音频，以便被试更好地对音频打分；(4) 测试音频，根据不同任务，放置合理的测试音频，真实和合成音频应提前打乱，并且不可告知被试打乱的顺序。

1.6.5 实验数据处理

1. 数据筛选。由于被试有可能没有受到监督，因此需要对收集到的评分进行事后检查，如删除使用扬声器试听的评分。另外，为了控制个体因素对整体结果的影响，减少偏离整体数据的异常值，需要计算每个人的评分与总体得分序列的相关性，相关性的度量使用相关系数来实现，如果相关系数 r 大于 0.25，则保留；否则拒绝该被试的所有评分。相关系数 r 的计算方法如下：

$$r = \frac{cov(\mu_{1n}, \dots, \mu_{Mn}; \mu_1, \dots, \mu_M)}{\sqrt{var(\mu_{1n}, \dots, \mu_{Mn})} \cdot \sqrt{var(\mu_1, \dots, \mu_M)}} \quad (1.1)$$

其中， M 为句子数量， N 为被试数量， μ_{mn} 为被试 n 对句子 m 给出的评分， $1 \leq m \leq M, 1 \leq n \leq N$ ， $\mu_m = 1/N \sum_{n=1}^N \mu_{mn}$ 为句子 m 的总体平均分， cov 为协方差， var 为方差。

第2章 语音信号基础

2.1 参考资料

1. An Introduction to Signal Processing for Speech
2. 爱丁堡大学课件

2.2 基本概念

声波通过空气传播，被麦克风接收，通过采样、量化、编码转换为离散的数字信号，即波形文件。

1. 波形 (waveform)：声音是由声源振动产生的波
2. 音高 (pitch)：与频率 (frequency) 有关，是频谱上的第一共振峰。音高 (pitch) 是稀疏离散化的基频 (f0)。
3. 响度：与振幅 (amplitude) 有关。
4. 音色：由频谱上的谐波决定。音高、响度和音色被称作声音的三要素。
5. 信道：通信的通道 Channel，是信号传输的媒介。
6. 声道：声音在录制和播放时，在不同空间位置采集或回放相互独立的音频信号，因此声道数也就是声音录制时的音源数量或回放时相应的扬声器数量。
7. 采样率：单位时间内从连续信号中提取并组成离散信号的采样个数，音频常用单位 kHz。
8. 采样位数/采样深度：数字信号的二进制位数，与每次采样的可能值个数有关，音频常用单位 bit。常见的音频格式：16kHz，16bit 中 16kHz 指的是采样率，16bit 表示采样位深。
9. 信噪比 (SNR)：和声压级类似，单位仍采用分贝，数值越高，表示声音越干净，噪音比例越小。
10. 音素 (phoneme)：声音的最小单位，同一音素由不同人/环境阅读，可以形成不同的发音。
11. 字素 (grapheme)：音素对应的文本。具体的区别参见：[Phonemes, Graphemes, and Morphemes!](#)
12. phone：某个音素的具体发音。
13. 过零率
14. 短时能量
15. LPC。线性预测系数。LPC 的基本思想是，当前时刻的信号可以用若干历史时刻信号的线性组合来估计，通过使实际语音的采样值和线性预测的采样值之间达到均方差最小，即可得到一组线性预测系数。求解 LPC 系数可以采用自相关法、协方差法、格型法等快速算法。
② **笔记** 语音信号的数字表示可以分为两类：波形表示和参数表示，波形表示仅通过采样和量化保存模拟信号的波形；而参数表示将语音信号表示为某种语音产生模型的输出，是对数字化语音进行分析和处理之后得到的。
- ② **笔记** 利用同态处理方法，对语音信号求离散傅里叶变换之后取对数，再求反变换就可以得到倒谱系数。其中，LPC 倒谱 (LPCCEP) 是建立在 LPC 谱上的，而梅尔倒谱系数 (Mel Frequency Cepstrum Coefficient, MFCC) 则是基于梅尔频谱的。
16. LPCC。LPCC 特征假定信号存在一种线性预测的结构，这对于周期特性的浊音描述比较准确，而对于辅音则相当于强加了一种错误的结构。MFCC 相邻帧特征近乎独立，所以能够比较好地描述辅音，但忽略了信号可能的内在结构，如相邻帧之间的关联，经验表明 MFCC 更好用，并且经常会加入差分特征以减弱其独立性。
② **笔记** 线性预测倒谱系数 (LPCC) 是根据声管模型建立的特征参数，是对声道响应的特征表征。梅尔频谱倒谱系数 (MFCC) 是基于人类听觉机理提取出的特征参数，是对人耳听觉的特征表征。

对于一段 1 秒的波形，假设采样率 16kHz，采样位深 16bit，则包含样本点 $1 \times 16000 = 16000$ 个，所占容量 $1 \times 16000 \times 16/8 = 32000$ 字节 (B)。

2.3 音频格式

1. *.wav: 波形无损压缩格式，是语音合成中音频语料的常用格式，主要的三个参数：采样率，量化位数和通道数。一般来说，合成语音的采样率采用 16kHz、22050Hz、24kHz，对于歌唱合成等高质量合成场景采样率可达到 48kHz；量化位数采用 16bit；通道数采用 1。
2. *.flac: Free Lossless Audio Codec，无损音频压缩编码。
3. *.mp3: Moving Picture Experts Group Audio Player III，有损压缩。
4. *.wma: Window Media Audio，有损压缩。
5. *.avi: Audio Video Interleaved，avi 文件将音频和视频包含在一个文件容器中，允许音视频同步播放。

2.4 发音学

2.5 数字信号处理

2.5.1 模拟信号到数字信号的转换 (Analog to Digital Converter, ADC, 模数转换)

奈奎斯特采样定理：要从抽样信号中无失真地恢复原信号，抽样频率应大于 2 倍信号最高频率。抽样频率小于 2 倍频谱最高频率时，信号的频谱有混叠。抽样频率大于 2 倍频谱最高频率时，信号的频谱无混叠。如果对语音模拟信号进行采样率为 16000Hz 的采样，得到的离散信号中包含的最大频率为 8000Hz。

2.5.2 傅里叶变换

2.5.3 频谱泄露

音频处理中，经常需要利用傅里叶变换将时域信号转换到频域，而一次快速傅里叶变换（FFT）只能处理有限长的时域信号，但语音信号通常是长的，所以需要将原始语音截断成一帧一帧长度的数据块。这个过程叫信号截断，也叫分帧。分完帧后再对每帧做 FFT，得到对应的频域信号。FFT 是离散傅里叶变换（DFT）的快速计算方式，而做 DFT 有一个先验条件：分帧得到的数据块必须是整数周期的信号，也即是每次截断得到的信号要求是周期主值序列。

但做分帧时，很难满足周期截断，因此就会导致频谱泄露。要解决非周期截断导致的频谱泄露是比较困难的，可以通过加窗尽可能减少频谱泄露带来的影响。窗类型可以分为汉宁窗、汉明窗、平顶窗等。虽然加窗能够减少频谱泄露，但加窗衰减了每帧信号的能量，特别是边界处的能量，这时加一个合成窗，且 overlap-add，便可以补回能量。参见：[频谱泄露和加窗](#)。

2.5.4 频率分辨率

频率分辨率是指将两个相邻谱峰分开的能力，在实际应用中是指分辨两个不同频率信号的最小间隔。

第3章 语音特征提取

原始信号是不定长的时序信号，不适合作为机器学习的输入。因此一般需要将原始波形转换为特定的特征向量表示，该过程称为语音特征提取。

3.1 预处理

包括预加重、分帧和加窗。

3.1.1 预加重

语音经过说话人的口唇辐射发出，受到唇端辐射抑制，高频能量明显降低。一般来说，当语音信号的频率提高两倍时，其功率谱的幅度下降约 6dB，即语音信号的高频部分受到的抑制影响较大。在进行语音信号的分析和处理时，可采用预加重（pre-emphasis）的方法补偿语音信号高频部分的振幅，本质是施加低通滤波器。假设输入信号第 n 个采样点为 $x[n]$ ，则预加重公式如下：

$$x'[n] = x[n] - a \times x[n - 1] \quad (3.1)$$

其中， a 是预加重系数，一般取 $a = 0.97$ 。

3.1.2 分帧

语音信号是非平稳信号，考虑到发浊音时声带有规律振动，即基音频率在短时范围内相对固定的，因此可以认为语音信号具有短时平稳特性，一般认为 10ms~50ms 的语音信号片段是一个准稳态过程。短时分析采用分帧方式，一般每帧帧长为 20ms 或 50ms。假设语音采样率为 16kHz，帧长为 20ms，则一帧有 $16000 \times 0.02 = 320$ 个样本点。

相邻两帧之间的基音有可能发生变化，如两个音节之间，或者声母向韵母过渡。为确保声学特征参数的平滑性，一般采用重叠取帧的方式，即相邻帧之间存在重叠部分。一般来说，帧长和帧移的比例为 1:4 或 1:5。

3.1.3 加窗

分帧相当于对语音信号加矩形窗，矩形窗在时域上对信号进行截断，在边界处存在多个旁瓣，会发生频谱泄露。为了减少频谱泄露，通常对分帧之后的信号进行其它形式的加窗操作。常用的窗函数有：汉明（Hamming）窗、汉宁（Hanning）窗和布莱克曼（Blackman）窗等。

汉明窗的窗函数为：

$$W_{ham}[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N} - 1\right) \quad (3.2)$$

其中， $0 \leq n \leq N - 1$ ， N 是窗的长度。

汉宁窗的窗函数为：

$$W_{han}[n] = 0.5[1 - \cos\left(\frac{2\pi n}{N} - 1\right)] \quad (3.3)$$

其中， $0 \leq n \leq N - 1$ ， N 是窗的长度。

3.2 短时傅里叶变换

人类听觉系统与频谱分析紧密相关，对语音信号进行频谱分析，是认识和处理语音信号的重要方法。声音从频率上可以分为纯音和复合音，纯音只包含一种频率的声音（基音），而没有倍音。复合音是除了基音之外，还包含多种倍音的声音。大部分语音都是复合音，涉及多个频率段，可以通过傅里叶变换进行频谱分析。

每个频率的信号可以用正弦波表示，采用正弦函数建模。基于欧拉公式，可以将正弦函数对应到统一的指数形式：

$$e^{j\omega n} = \cos(\omega n) + j \sin(\omega n) \quad (3.4)$$

正弦函数具有正交性，即任意两个不同频率的正弦波乘积，在两者的公共周期内积分等于零。正交性用复指数运算表示如下：

$$\int_{-\infty}^{+\infty} e^{j\alpha t} e^{-j\beta t} dt = 0, \quad \text{if } \alpha \neq \beta \quad (3.5)$$

基于正弦函数的正交性，通过相关处理可以从语音信号分离出对应不同频率的正弦信号。对于离散采样的语音信号，可以采用离散傅里叶变换（DFT）。DFT 的第 k 个点计算如下：

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{K}}, \quad k = 0, 1, \dots, K-1 \quad (3.6)$$

其中， $x[n]$ 是时域波形第 n 个采样点值， $X[k]$ 是第 k 个傅里叶频谱值， N 是采样点序列的点数， K 是频谱系数的点数，且 $K \geq N$ 。利用 DFT 获得的频谱值通常是复数形式，这是因为上式中，

$$e^{-\frac{j2\pi kn}{K}} = \cos\left(\frac{2\pi kn}{K}\right) - j \sin\left(\frac{2\pi kn}{K}\right) \quad (3.7)$$

则

$$X[k] = X_{real}[k] - j X_{imag}[k] \quad (3.8)$$

其中，

$$X_{real}[k] = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi kn}{K}\right) \quad (3.9)$$

$$X_{imag}[k] = \sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi kn}{K}\right) \quad (3.10)$$

N 个采样点序列组成的时域信号经过 DFT 之后，对应 K 个频率点。经 DFT 变换得到信号的频谱表示，其频谱幅值和相位随着频率变化而变化。

在语音信号处理中主要关注信号的频谱幅值，也称为振幅频谱/振幅谱：

$$X_{magnitude}[k] = \sqrt{X_{real}[k]^2 + X_{imag}[k]^2} \quad (3.11)$$

能量频谱/能量谱是振幅频谱的平方：

$$X_{power}[k] = X_{real}[k]^2 + X_{imag}[k]^2 \quad (3.12)$$

各种声源发出的声音大多由许多不同强度、不同频率的声音组成复合音，在复合音中，不同频率成分与能

量分布的关系称为声音的频谱，利用频谱图表示各频率成分与能量分布之间的关系，频谱图横轴是频率（Hz），纵轴是幅度（dB）。

通过对频域信号进行逆傅里叶变换（IDFT），可以恢复时域信号：

$$x[n] = \frac{1}{K} \sum_{k=0}^{K-1} X[k] e^{\frac{j2\pi kn}{N}}, \quad n = 0, 1, \dots, N-1 \quad (3.13)$$

离散傅里叶变换（DFT）的计算复杂度为 $O(N^2)$ ，可以采用快速傅里叶变换（FFT），简化计算复杂度，在 $O(N \log_2 N)$ 的时间内计算出 DFT。在实际应用中，对语音信号进行分帧加窗处理，将其分割成一帧帧的离散序列，可视为短时傅里叶变换（STFT）：

$$X[k, l] = \sum_{n=0}^{N-1} x_l[n] e^{-\frac{j2\pi nk}{K}} = \sum_{n=0}^{N-1} w[n] x[n + lL] e^{-\frac{j2\pi nk}{K}} \quad (3.14)$$

其中， K 是 DFT 后的频率点个数， k 是频率索引， $0 \leq k < K$ 。 $X[k, l]$ 建立起索引为 lL 的时域信号，与索引为 k 的频域信号之间的关系。

3.3 听觉特性

3.3.1 梅尔滤波

人类对不同频率的语音有不同的感知能力：

1. 1kHz 以下，人耳感知与频率成线性关系。
2. 1kHz 以上，人耳感知与频率成对数关系。

因此，人耳对低频信号比高频信号更为敏感。因此根据人耳的特性提出了一种 mel 刻度，即定义 1 个 mel 刻度相当于人对 1kHz 音频感知程度的千分之一，mel 刻度表达的是，从线性频率到“感知频率”的转换关系：

$$mel(f) = 2595 \lg(1 + \frac{f}{700}) \quad (3.15)$$

```

1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  x = np.linspace(0, 5000, 50000)
5  y = 2595 * np.log10(1+x/700)
6  x0 = 1000
7  y0 = 2595 * np.log10(1+x0/700)
8  plt.plot(x, y)
9  plt.scatter(x0, y0)
10 plt.plot([x0, x0], [0, y0], 'k--')
11 plt.plot([0, x0], [x0, y0], 'k--')
12 plt.xlabel('f (Hz)')
13 plt.ylabel('Mel(f)')
14 plt.title('relationship between linear and mel scale')
15 plt.xlim(0, x[-1])
16 plt.ylim(0, y[-1])
17 plt.savefig('mel_vs_f.png')
18 plt.show()

```

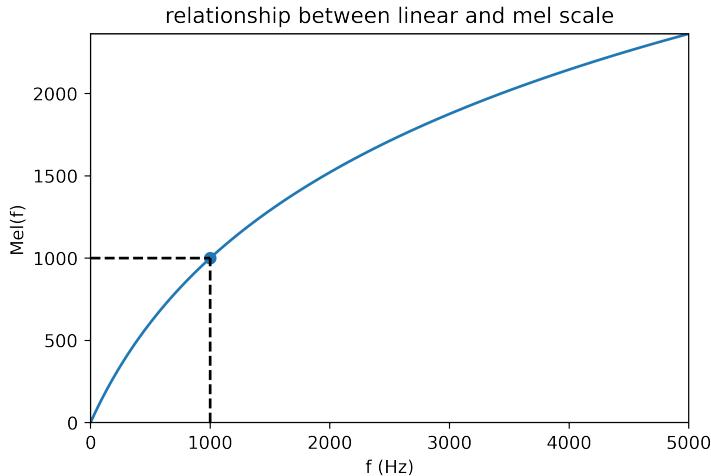


图 3.1: mel 刻度和频率之间的关系

人们根据一系列心理声学实验得到了类似耳蜗作用的滤波器组，用于模拟人耳不同频段声音的感知能力，也就是多个三角滤波器组成的 mel 频率滤波器组。每个滤波器带宽不等，线性频率小于 1000Hz 的部分为线性间隔，而线性频率大于 1000Hz 的部分为对数间隔。同样地，将梅尔频率转换到线性频率的公式为：

$$f_{mel}^{-1} = 700 \cdot (10^{\frac{f_{mel}}{2595}} - 1) \quad (3.16)$$

3.3.2 Bark 滤波

声音的响度，反映人对不同频率成分声强/声音强弱的主观感受。响度与声强、频率的关系可以用等响度轮廓曲线表示。

人耳对响度的感知有一个范围，当声音低于某个响度时，人耳是无法感知到的，这个响度值称为听觉阈值，或称听阈。在实际环境中，但一个较强信号（掩蔽音）存在时，听阈就不等于安静时的阈值，而是有所提高。这意味着，邻近频率的两个声音信号，弱响度的声音信号会被强响度的声音信号所掩蔽（Mask），这就是频域掩蔽。

根据听觉频域分辨率和频域掩蔽的特点，定义能够引起听觉主观变化的频率带宽为一个临界频带。一个临界频带的宽度被称为一个 Bark，Bark 频率 $Z(f)$ 和线性频率 f 的对应关系定义如下：

$$Z(f) = 6 \ln\left(\frac{f}{600} + \left(\left(\frac{f}{600}\right)^2 + 1\right)^{\frac{1}{2}}\right) \quad (3.17)$$

其中，线性频率 f 的单位为 Hz，临界频带 $Z(f)$ 的单位为 Bark。

3.4 倒谱分析

语音信号的产生模型包括发生源（Source）和滤波器（Filter）。人在发声时，肺部空气受到挤压形成气流，气流通过声门（声带）振动产生声门源激励 $e[n]$ 。对于浊音，激励 $e[n]$ 是以基音周期重复的单位冲激；对于清音， $e[n]$ 是平稳白噪声。该激励信号 $e[n]$ 经过咽喉、口腔形成声道的共振和调制，特别是舌头能够改变声道的容积，从而改变发音，形成不同频率的声音。气流、声门可以等效为一个激励源，声道等效为一个时变滤波器，语音信号 $x[n]$ 可以被看成激励信号 $e[n]$ 与时变滤波器的单位响应 $v[n]$ 的卷积：

$$x[n] = e[n] * v[n] \quad (3.18)$$

已知语音信号 $x[n]$ ，待求出上式中参与卷积的各个信号分量，也就是解卷积处理。除了线性预测方法外，还

可以采用倒谱分析实现解卷积处理。倒谱分析，又称为同态滤波，采用时频变换，得到对数功率谱，再进行逆变换，分析出倒谱域的倒谱系数。

同态滤波的处理过程如下：

1. 傅里叶变换。将时域的卷积信号转换为频域的乘积信号：

$$\text{DFT}(x[n]) = X[z] = E[z]V[z] \quad (3.19)$$

2. 对数运算。将乘积信号转换为加性信号：

$$\log X[z] = \log E[z] + \log V[z] = \hat{E}[z] + \hat{V}[z] = \hat{X}[z] \quad (3.20)$$

3. 傅里叶反变换。得到时域的语音信号倒谱。

$$Z^{-1}(\hat{X}[z]) = Z^{-1}(\hat{E}[z] + \hat{V}[z]) = \hat{e}[n] + \hat{v}[z] \approx \hat{x}[n] \quad (3.21)$$

在实际应用中，考虑到离散余弦变换（DCT）具有最优的去相关性能，能够将信号能量集中到极少数的变换系数上，特别是能够将大多数的自然信号（包括声音和图像）的能量都集中在离散余弦变换后的低频部分。一般采用 DCT 反变换代替傅里叶反变换，上式可以改写成：

$$\hat{c}[m] = \sum_{k=1}^N \log X[k] \cos\left(\frac{\pi(k-0.5)m}{N}\right), \quad m = 1, 2, \dots, M \quad (3.22)$$

其中， $X[k]$ 是 DFT 变换系数， N 是 DFT 系数的个数， M 是 DCT 变换的个数。

此时， $\hat{x}[n]$ 是复倒谱信号，可采用逆运算，恢复出语音信号，但 DCT 不可逆，从倒谱信号 $\hat{c}[m]$ 不可还原出语音 $x[n]$ 。

3.5 常见的声学特征

在语音合成中，常用的声学特征有梅尔频谱（Mel-Spectrogram）/滤波器组（Filter-bank, Fbank），梅尔频率倒谱系数（Mel-Frequency Cepstral Coefficient, MFCC）等。

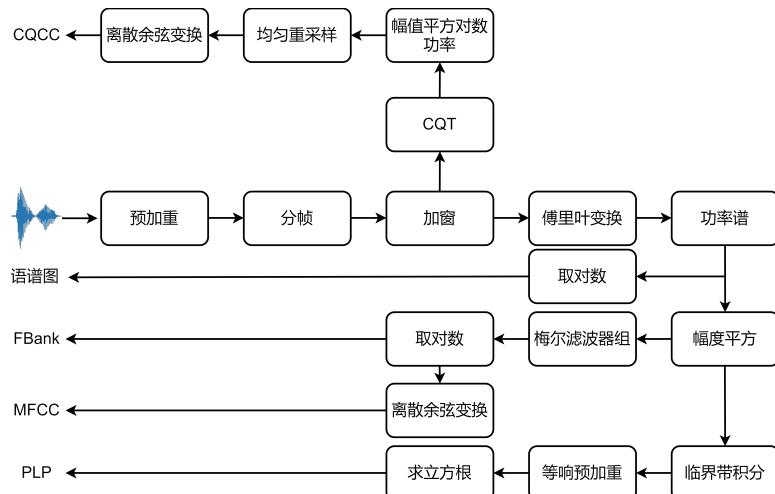


图 3.2: 常用的声学特征

接下来重点介绍 FBank 和 MFCC 的计算过程。

3.5.1 FBank

FBank 的特征提取过程如下：

1. 将信号进行预加重、分帧、加窗，然后进行短时傅里叶变换（STFT）获得对应的频谱。

2. 求频谱的平方，即能量谱。进行梅尔滤波，即将每个滤波频带内的能量进行叠加，第 k 个滤波器输出功率谱为 $X[k]$ 。
3. 将每个滤波器的输出取对数，得到相应频带的对数功率谱。

$$Y_{\text{FBank}}[k] = \log X[k] \quad (3.23)$$

FBank 特征本质上是对数功率谱，包括低频和高频信息。相比于语谱图，FBank 经过了梅尔滤波，依据人耳听觉特性进行了压缩，抑制了一部分人耳无法感知的冗余信息。

3.5.2 MFCC

MFCC 和 FBank 唯一的不同就在于，获得 FBank 特征之后，再经过反离散余弦变换，就得到 L 个 MFCC 系数。在实际操作中，得到的 L 个 MFCC 特征值可以作为静态特征，再对这些静态特征做一阶和二阶差分，得到相应的动态特征。

3.6 具体操作

3.6.1 利用 librosa 读取音频

```

1  from matplotlib import pyplot as plt
2  import numpy as np
3  import librosa
4
5  # 利用 librosa 读取音频
6  input_wav_path = r'test.wav'
7  y, sr = librosa.load(input_wav_path)
8  y_num = np.arange(len(y))
9
10 # 截取前 0.3s 的音频
11 sample_signal = y[0:int(sr*0.3)]
12 sample_num = np.arange(len(sample_signal))
13
14 plt.figure(figsize=(11, 7), dpi=500)
15 plt.subplot(211)
16 plt.plot(y_num/sr, y, color='black')
17 plt.plot(sample_num/sr, sample_signal, color='blue')
18 plt.xlabel('Time (sec)')
19 plt.ylabel('Amplitude')
20 plt.title('Waveform')
21
22 plt.subplot(212)
23 plt.plot(sample_num/sr, sample_signal, color='blue')
24 plt.xlabel('Time (sec)')
25 plt.ylabel('Amplitude')
26 plt.title('0~0.3s waveform')
27 plt.tight_layout()
28 plt.savefig('waveform.png', dpi=500)
29 plt.show()
```

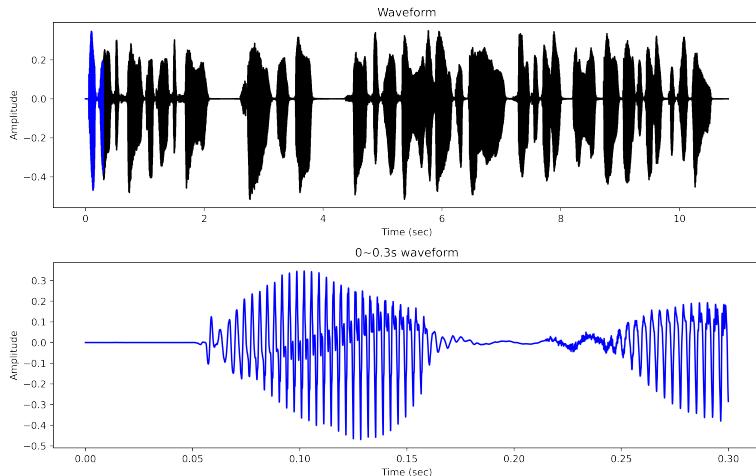


图 3.3: 波形图

音频有不同的编码类型，librosa 默认采取浮点格式读取，即读取的样本点均是 $[-1, -1]$ 之间的浮点值。更详细的文档参见 SoX 的 Input & Output File Format Options 部分。

表 3.1: SoX 音频个数

选项	描述	常见可选项
b	每个编码样本所占的数据位数（位深）	8/16/32
c	音频文件包含的通道数	1/2
e	音频文件的编码类型	signed-integer/unsigned-integer/floating-point
r	音频文件的采样率	16k/16000/22050
t	音频文件的文件类型	raw/mp3

3.6.2 提取梅尔频谱

```

1 sample_rate = 16000
2 preemphasis = 0.97
3 n_fft = 1024
4 frame_length = 0.05 # ms
5 frame_shift = 0.01 # ms
6 fmin = 0
7 fmax = sample_rate/2
8 eps = 1e-10
9 n_mel = 80
10 win_length = int(sample_rate*frame_length)
11 hop_length = int(sample_rate*frame_shift)
12 mel_basis = librosa.filters.mel(
13     sample_rate, n_fft, n_mel, fmin=fmin, fmax=fmax)
14
15
16 def get_spectrogram(input_wav_path):
17     y, sr = librosa.load(input_wav_path)
18     y = np.append(y[0], y[1:]-preemphasis*y[:-1])
19     linear = librosa.stft(
20         y=y, n_fft=n_fft, hop_length=hop_length, win_length=win_length)
21     mag = np.abs(linear)

```

```

22     mel = np.dot(mel_basis, mag)
23     mel = np.log10(np.maximum(eps, mel))
24     mel = mel.T.astype(np.float32) # (T, n_mels)
25     return mel
26
27 # plt.switch_backend('agg')
28
29
30 def plot_spectrogram(spectrogram, file_path):
31     spectrogram = spectrogram.T
32     fig = plt.figure(figsize=(16, 9))
33     plt.imshow(spectrogram, aspect='auto', origin='lower')
34     plt.colorbar()
35     plt.xlabel('frames')
36     plt.tight_layout()
37     plt.savefig(file_path, dpi=500)
38     plt.show()
39
40
41 mel_spec = get_spectrogram(input_wav_path)
42 plot_spectrogram(mel_spec, 'mel_spectrogram.png')

```

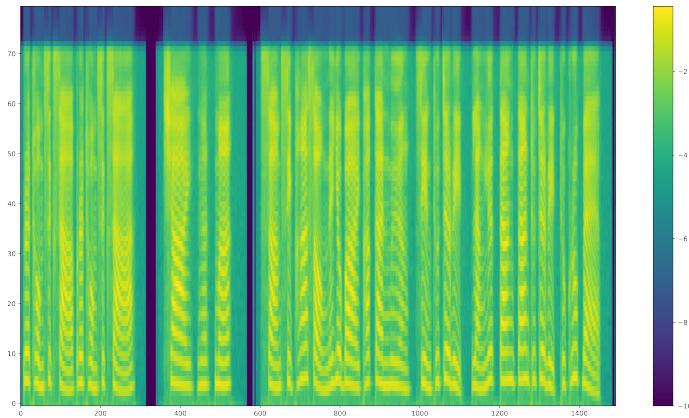


图 3.4: 梅尔频谱

3.6.3 提取 MFCC

```

1 mfcc = dct(mel_spec)
2 plot_spectrogram(mfcc, 'mfcc.png')

```

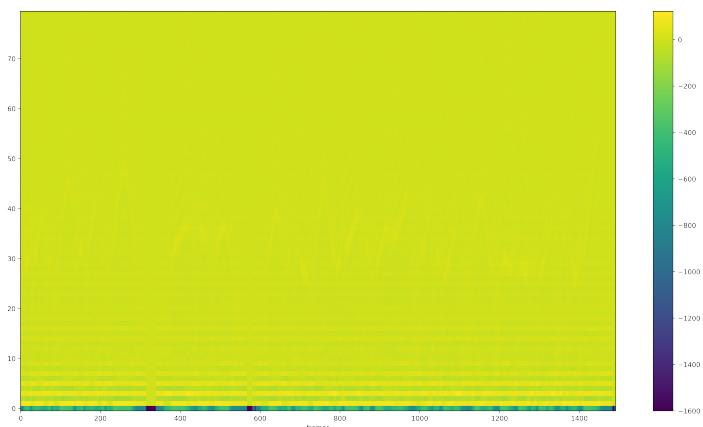


图 3.5: MFCC

第4章 声学模型

现代工业级神经网络语音合成系统主要包括三个部分：文本前端、声学模型和声码器，文本输入到文本前端中，将文本转换为音素、韵律边界等文本特征。文本特征输入到声学模型，转换为对应的声学特征。声学特征输入到声码器，重建为原始波形。



图 4.1：神经网络 TTS 的三个主要部件

主要采用的声学模型包括 Tacotron 系列、FastSpeech 系列等，目前同样出现了一些完全端到端的语音合成模型，也即是直接由字符/音素映射为波形。

4.1 Tacotron

以经常使用的 Tacotron-2 声学模型为例。原始论文参见：

1. Tacotron: Towards End-to-End Speech Synthesis
2. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions

此外，谷歌在语音合成领域，特别是端到端语音合成领域做出了开创性的共享，该组会将最新的论文汇总在 Tacotron (/täkträn/): An end-to-end speech synthesis system by Google.

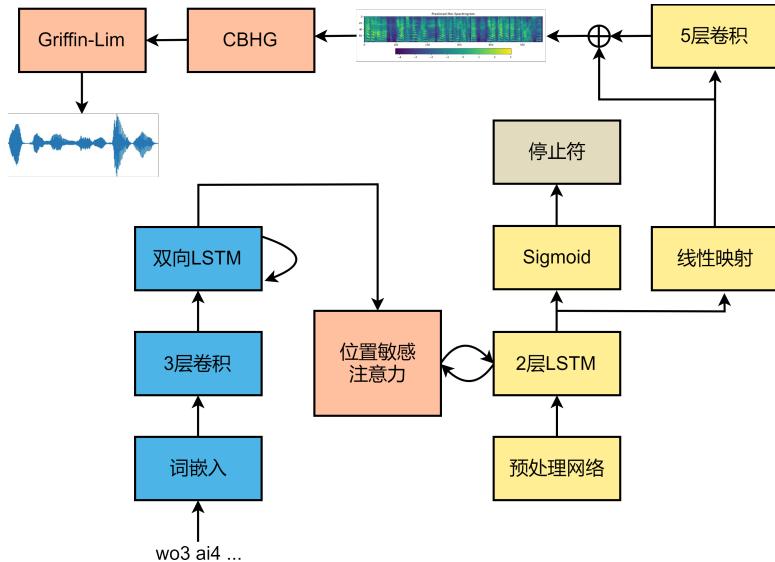


图 4.2：Tacotron-2 模型结构

Tacotron-2 的声学模型部分采用典型的序列到序列结构。编码器是 3 个卷积层和一个双向 LSTM 层组成的模块，卷积层给予了模型类似于 N-gram 感知上下文的能力，并且对不发音字符更加鲁棒。经词嵌入的注音序列首先进入卷积层提取上下文信息，然后送入双向 LSTM 生成编码器隐状态。编码器隐状态生成后，就会被送入注意力机制，以生成编码向量。我们利用了一种被称为位置敏感注意力 (Location Sensitive Attention, LSA)，该注意力机制的对齐函数为：

$$score(s_{i-1}, h_j) = v_a^T \tanh(W s_{i-1} + V h_j + U f_{i,j} + b) \quad (4.1)$$

其中, v_a, W, V, U 为待训练参数, b 是偏置值, s_{i-1} 为上一时间步 $i-1$ 的解码器隐状态, h_j 为当前时间步 j 的编码器隐状态, $f_{i,j}$ 为上一个解码步的注意力权重 α_{i-1} 经卷积获得的位置特征, 如下式:

$$f_{i,j} = F * \alpha_{i-1} \quad (4.2)$$

其中, α_{i-1} 是经过 softmax 的注意力权重的累加和。位置敏感注意力机制不但综合了内容方面的信息, 而且关注了位置特征。解码过程从输入上一解码步或者真实音频的频谱进入解码器预处理网络开始, 到线性映射输出该时间步上的频谱帧结束, 模型的解码过程如下图所示。

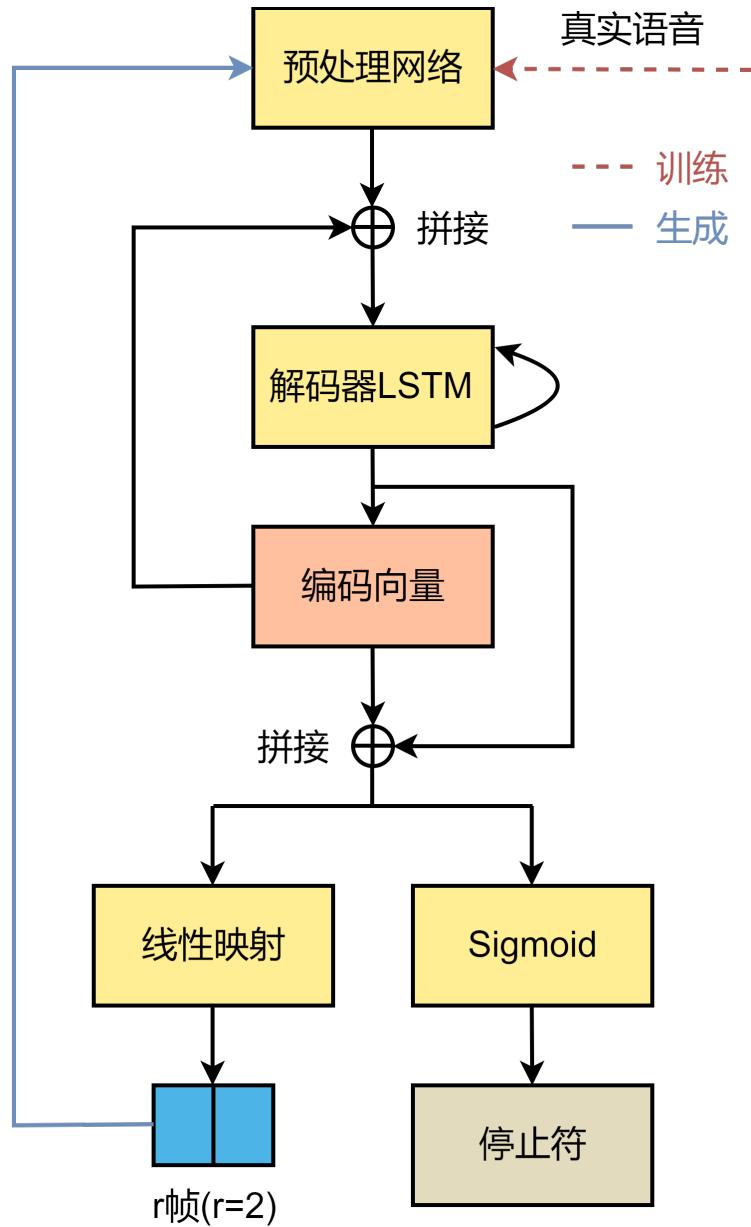


图 4.3: Tacotron2 解码过程

频谱生成网络的解码器将预处理网络的输出和注意力机制的编码向量做拼接, 然后整体送入 LSTM 中, LSTM 的输出用来计算新的编码向量, 最后新计算出来的编码向量与 LSTM 输出做拼接, 送入映射层以计算输出。输出有两种形式, 一种是频谱帧, 另一种是停止符的概率, 后者是一个简单二分类问题, 决定解码过程是否结束。为了能够有效加速计算, 减小内存占用, 引入缩减因子 r (Reduction Factor), 即每一个时间步允许解码器预测 r 个频谱帧进行输出。解码完成后, 送入后处理网络处理以生成最终的梅尔频谱, 如下式所示。

$$s_{final} = s_i + s'_i \quad (4.3)$$

其中， s_i 是解码器输出， s_{final} 表示最终输出的梅尔频谱， s'_i 是后处理网络的输出，解码器的输出经过后处理网络之后获得 s'_i 。

在 Tacotron-2 原始论文中，直接将梅尔频谱送入声码器 WaveNet 生成最终的时域波形。但是 WaveNet 计算复杂度过高，几乎无法实际使用，因此可以使用其它声码器，比如 Griffin-Lim、HiFiGAN 等。

Tacotron2 的损失函数主要包括以下 4 个方面：

1. 进入后处理网络前后的平方损失。

$$\text{MelLoss} = \frac{1}{n} \sum_{i=1}^n (y_{real,i}^{mel} - y_{before,i}^{mel})^2 + \frac{1}{n} \sum_{i=1}^n (y_{real,i}^{mel} - y_{after,i}^{mel})^2 \quad (4.4)$$

其中， $y_{real,i}^{mel}$ 表示从音频中提取的真实频谱， $y_{before,i}^{mel}, y_{after,i}^{mel}$ 分别为进入后处理网络前、后的解码器输出， n 为每批的样本数。

2. 从 CBHG 模块中输出线性谱的平方损失。

$$\text{LinearLoss} = \frac{1}{n} \sum_{i=1}^n (y_{real,i}^{linear} - y_i^{linear})^2 \quad (4.5)$$

其中， $y_{real,i}^{linear}$ 是从真实语音中计算获得的线性谱， y_i^{linear} 是从 CBHG 模块输出的线性谱。

3. 停止符交叉熵

$$\text{StopTokenLoss} = -[y \cdot \log(p) + (1 - y) \cdot \log(1 - p)] \quad (4.6)$$

其中， y 为停止符真实概率分布， p 是解码器线性映射输出的预测分布。

4. L2 正则化

$$\text{RegulationLoss} = \frac{1}{K} \sum_{k=1}^K w_k^2 \quad (4.7)$$

其中， K 为参数总数， w_k 为模型中的参数，这里排除偏置值、RNN 以及线性映射中的参数。最终的损失函数为上述 4 个部分的损失之和，如下式：

$$\text{Loss} = \text{MelLoss} + \text{LinearLoss} + \text{StopTokenLoss} + \text{RegulationLoss} \quad (4.8)$$

4.2 FastSpeech

FastSpeech 是基于 Transformer 显式时长建模的声学模型，由微软和浙大提出。原始论文参见：

1. [FastSpeech: Fast, Robust and Controllable Text to Speech](#)
2. [FastSpeech 2: Fast and High-Quality End-to-End Text to Speech](#)

相对应地，微软在语音合成领域的论文常常发布在 [Microsoft-Speech Research](#)。

4.2.1 模型结构

FastSpeech 2 和上代 FastSpeech 的编解码器均是采用 FFT (feed-forward Transformer, 前馈 Transformer) 块。编解码器的输入首先进行位置编码，之后进入 FFT 块。FFT 块主要包括多头注意力模块和位置前馈网络，位置前馈网络可以由若干层 Conv1d、LayerNorm 和 Dropout 组成。

论文中提到语音合成是典型的一对多问题，同样的文本可以合成无数种语音。上一代 FastSpeech 主要通过目标侧使用教师模型的合成频谱而非真实频谱，以简化数据偏差，减少语音中的多样性，从而降低训练难度；向模型提供额外的时长信息两个途径解决一对多的问题。在语音中，音素时长自不必说，直接影响发音长度和整体韵律；音调则是影响情感和韵律的另一个特征；能量则影响频谱的幅度，直接影响音频的音量。在 FastSpeech 2 中对这三个最重要的语音属性单独建模，从而缓解一对多带来的模型学习目标不确定的问题。

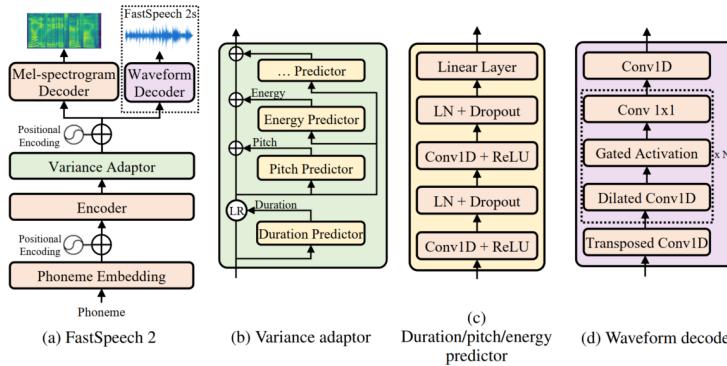


Figure 1: The overall architecture for FastSpeech 2 and 2s. LR in subfigure (b) denotes the length regulator proposed in FastSpeech. LN in subfigure (c) denotes layer normalization.

图 4.4: FastSpeech 2 整体结构

在对时长、基频和能量单独建模时，所使用的网络结构实际是相似的，在论文中称这种语音属性建模网络为变量适配器（Variance Adaptor）。时长预测的输出也作为基频和能量预测的输入。最后，基频预测和能量预测的输出，以及依靠时长信息展开的编码器输入元素加起来，作为下游网络的输入。变量适配器主要是由 2 层卷积和 1 层线性映射层组成，每层卷积后加 ReLU 激活、LayerNorm 和 Dropout。代码摘抄自 [FastSpeech2](#)，添加了一些注释。

```

1 class VariancePredictor(nn.Module):
2     """ Duration , Pitch and Energy Predictor """
3     def __init__(self):
4         super(VariancePredictor, self).__init__()
5
6         self.input_size = hp.encoder_hidden
7         self.filter_size = hp.variance_predictor_filter_size
8         self.kernel = hp.variance_predictor_kernel_size
9         self.conv_output_size = hp.variance_predictor_filter_size
10        self.dropout = hp.variance_predictor_dropout
11
12        self.conv_layer = nn.Sequential(OrderedDict([
13            ("conv1d_1", Conv(self.input_size,
14                               self.filter_size,
15                               kernel_size=self.kernel,
16                               padding=(self.kernel-1)//2)),
17            ("relu_1", nn.ReLU()),
18            ("layer_norm_1", nn.LayerNorm(self.filter_size)),
19            ("dropout_1", nn.Dropout(self.dropout)),
20            ("conv1d_2", Conv(self.filter_size,
21                               self.filter_size,
22                               kernel_size=self.kernel,
23                               padding=1)),
24            ("relu_2", nn.ReLU()),
25            ("layer_norm_2", nn.LayerNorm(self.filter_size)),
26            ("dropout_2", nn.Dropout(self.dropout))
27        ]))
28
29        self.linear_layer = nn.Linear(self.conv_output_size, 1)
30

```

```

31     def forward(self, encoder_output, mask):
32         ...
33         :param encoder_output: Output of encoder. [batch_size, seq_len, encoder_hidden]
34         :param mask: Mask for encoder. [batch_size, seq_len]
35         ...
36         out = self.conv_layer(encoder_output)
37         out = self.linear_layer(out)
38         out = out.squeeze(-1)
39
40         if mask is not None:
41             out = out.masked_fill(mask, 0.)
42
43     return out

```

利用该变量适配器对时长、基频和能量进行建模。

```

1  class VarianceAdaptor(nn.Module):
2      """ Variance Adaptor """
3
4      def __init__(self):
5          super(VarianceAdaptor, self).__init__()
6          self.duration_predictor = VariancePredictor()
7          self.length_regulator = LengthRegulator()
8          self.pitch_predictor = VariancePredictor()
9          self.energy_predictor = VariancePredictor()
10
11         self.pitch_bins = nn.Parameter(torch.exp(torch.linspace(
12             np.log(hp.f0_min), np.log(hp.f0_max), hp.n_bins-1)), requires_grad=False)
13         self.energy_bins = nn.Parameter(torch.linspace(
14             hp.energy_min, hp.energy_max, hp.n_bins-1), requires_grad=False)
15         self.pitch_embedding = nn.Embedding(hp.n_bins, hp.encoder_hidden)
16         self.energy_embedding = nn.Embedding(hp.n_bins, hp.encoder_hidden)
17
18     def forward(self, x, src_mask, mel_mask=None, duration_target=None, pitch_target=
19                 None, energy_target=None, max_len=None, d_control=1.0, p_control=1.0, e_control
20                 =1.0):
21         ...
22         :param x: Output of encoder. [batch_size, seq_len, encoder_hidden]
23         :param src_mask: Mask of encoder, can get src_mask from input_lengths. [
24             batch_size, seq_len]
25         :param duration_target, pitch_target, energy_target: Ground-truth when training
26             , None when synthesis. [batch_size, seq_len]
27         ...
28         log_duration_prediction = self.duration_predictor(x, src_mask)
29         if duration_target is not None:
30             x, mel_len = self.length_regulator(x, duration_target, max_len)
31         else:
32             duration_rounded = torch.clamp(
33                 (torch.round(torch.exp(log_duration_prediction))-hp.log_offset)*
34                 d_control), min=0)

```

```

30     x, mel_len = self.length_regulator(x, duration_rounded, max_len)
31     mel_mask = utils.get_mask_from_lengths(mel_len)
32
33     pitch_prediction = self.pitch_predictor(x, mel_mask)
34     if pitch_target is not None:
35         pitch_embedding = self.pitch_embedding(
36             torch.bucketize(pitch_target, self.pitch_bins))
37     else:
38         pitch_prediction = pitch_prediction*p_control
39         pitch_embedding = self.pitch_embedding(
40             torch.bucketize(pitch_prediction, self.pitch_bins))
41
42     energy_prediction = self.energy_predictor(x, mel_mask)
43     if energy_target is not None:
44         energy_embedding = self.energy_embedding(
45             torch.bucketize(energy_target, self.energy_bins))
46     else:
47         energy_prediction = energy_prediction*e_control
48         energy_embedding = self.energy_embedding(
49             torch.bucketize(energy_prediction, self.energy_bins))
50
51     x = x + pitch_embedding + energy_embedding
52
53     return x, log_duration_prediction, pitch_prediction, energy_prediction, mel_len
           , mel_mask

```

同样是通过长度调节器 (Length Regulator)，利用时长信息将编码器输出长度扩展到频谱长度。具体实现就是根据 duration 的具体值，直接上采样。一个音素时长为 2，就将编码器输出复制 2 份，给 3 就直接复制 3 份，拼接之后作为最终的输出。实现代码：

```

1 class LengthRegulator(nn.Module):
2     """ Length Regulator """
3
4     def __init__(self):
5         super(LengthRegulator, self).__init__()
6
7     def LR(self, x, duration, max_len):
8         ...
9         :param x: Output of encoder. [batch_size, phoneme_seq_len, encoder_hidden]
10        :param duration: Duration for phonemes. [batch_size, phoneme_seq_len]
11        :param max_len: Max length for mel-frames. scaler
12
13    Return:
14        output: Expanded output of encoder. [batch_size, mel_len, encoder_hidden]
15        ...
16        output = list()
17        mel_len = list()
18        for batch, expand_target in zip(x, duration):
19            # batch: [seq_len, encoder_hidden]
20            # expand_target: [seq_len]

```

```

21     expanded = self.expand(batch, expand_target)
22     output.append(expanded)
23     mel_len.append(expanded.shape[0])
24
25     if max_len is not None:
26         output = utils.pad(output, max_len)
27     else:
28         output = utils.pad(output)
29
30     return output, torch.LongTensor(mel_len).to(device)
31
32 def expand(self, batch, predicted):
33     out = list()
34
35     for i, vec in enumerate(batch):
36         # expand_size: scalar
37         expand_size = predicted[i].item()
38         # Passing -1 as the size for a dimension means not changing the size of
39         # that dimension.
40         out.append(vec.expand(int(expand_size), -1))
41     out = torch.cat(out, 0)
42
43     return out
44
45 def forward(self, x, duration, max_len):
46     output, mel_len = self.LR(x, duration, max_len)
47     return output, mel_len

```

对于音高和能量的预测，模块的主干网络相似，但使用方法有所不同。以音高为例，能量的使用方式相似。首先对预测出的实数域音高值进行分桶，映射为一定范围内的自然数集，然后做嵌入。

```

1 pitch_prediction = self.pitch_predictor(x, mel_mask)
2 if pitch_target is not None:
3     pitch_embedding = self.pitch_embedding(
4         torch.bucketize(pitch_target, self.pitch_bins))
5 else:
6     pitch_prediction = pitch_prediction*p_control
7     pitch_embedding = self.pitch_embedding(
8         torch.bucketize(pitch_prediction, self.pitch_bins))

```

这里用到了 Pytorch 中一个不是特别常见的函数`torch.bucketize`。这是 Pytorch 中的分桶函数，boundaries 确定了各个桶的边界，是一个单调递增向量，用于划分 input，并返回 input 所属桶的索引，桶索引从 0 开始。

能量嵌入向量的计算方法与之类似。至此，获得了展开之后的编码器输出 x，基频嵌入向量`pitch_embedding`和能量嵌入向量`energy_embedding`之后，元素加获得最终编解码器的输入。

4.2.2 损失函数

FastSpeech 2 的目标函数由 PostNet 前后的频谱均方差，时长、音高和能量的均方差组成。时长映射到指数组（时长预测器输出的数值 x 作为指数，最终的预测时长为 e^x ），音高映射到对数域（音高预测器输出的数值 x 做对数，作为最终的音高 $\log x$ ），而能量直接采用能量预测器的输出值。整体的损失函数为：

$$\text{Loss} = \text{Loss}_{mel} + \text{Loss}_{mel}^{post} + \text{Loss}_{duration} + \text{Loss}_{pitch} + \text{Loss}_{energy} \quad (4.9)$$

频谱的损失函数形式采用均方差 (MSE)，时长、基频和能量采用平均绝对误差 (MAE)，具体的实现如下：

```

1 log_d_target.requires_grad = False
2 p_target.requires_grad = False
3 e_target.requires_grad = False
4 mel_target.requires_grad = False
5
6 log_d_predicted = log_d_predicted.masked_select(src_mask)
7 log_d_target = log_d_target.masked_select(src_mask)
8 p_predicted = p_predicted.masked_select(mel_mask)
9 p_target = p_target.masked_select(mel_mask)
10 e_predicted = e_predicted.masked_select(mel_mask)
11 e_target = e_target.masked_select(mel_mask)
12
13 mel = mel.masked_select(mel_mask.unsqueeze(-1))
14 mel_postnet = mel_postnet.masked_select(mel_mask.unsqueeze(-1))
15 mel_target = mel_target.masked_select(mel_mask.unsqueeze(-1))
16
17 mel_loss = self.mse_loss(mel, mel_target)
18 mel_postnet_loss = self.mse_loss(mel_postnet, mel_target)
19
20 d_loss = self.mae_loss(log_d_predicted, log_d_target)
21 p_loss = self.mae_loss(p_predicted, p_target)
22 e_loss = self.mae_loss(e_predicted, e_target)
23
24 total_loss = mel_loss + mel_postnet_loss + d_loss + p_loss + e_loss

```

4.2.3 小结

FastSpeech 系列的声学模型将 Transformer 引入语音合成领域，并且显式建模语音中的重要特征，比如时长、音高和能量等。实际上，微软首次在 [Neural Speech Synthesis with Transformer Network](#) 将 Transformer 作为主干网络，实现语音合成的声学模型，这一思想同样被 [FastPitch: Parallel Text-to-speech with Pitch Prediction](#) 采用，相关的开源代码：[as-ideas/TransformerTTS](#)。

4.3 VITS

VITS (Variational Inference with adversarial learning for end-to-end Text-to-Speech) 是一种结合变分推理 (variational inference)、标准化流 (normalizing flows) 和对抗训练的高表现力语音合成模型。和 Tacotron 和 FastSpeech 不同，Tacotron / FastSpeech 实际是将字符或音素映射为中间声学表征，比如梅尔频谱，然后通过声码器将梅尔频谱还原为波形，而 VITS 则直接将字符或音素映射为波形，不需要额外的声码器重建波形，真正的端到端语音合成模型。VITS 通过隐变量而非之前的频谱串联语音合成中的声学模型和声码器，在隐变量上进行建模并利用随机时长预测器，提高了合成语音的多样性，输入同样的文本，能够合成不同声调和韵律的语音。VITS 合成音质较高，并且可以借鉴之前的 FastSpeech，单独对音高等特征进行建模，以进一步提升合成语音的质量，是一种非常有潜力的语音合成模型。

4.3.1 模型整体结构

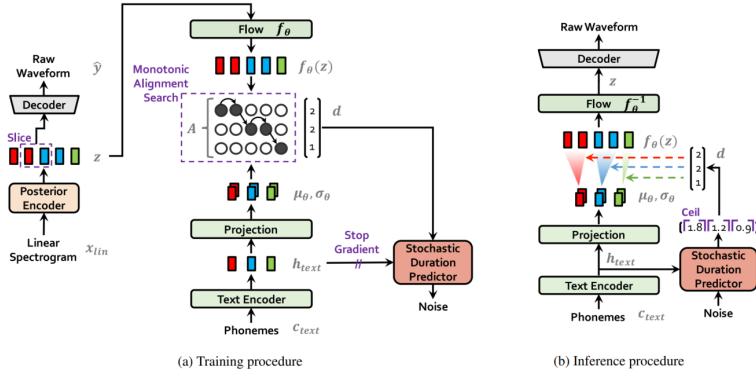


Figure 1. System diagram depicting (a) training procedure and (b) inference procedure. The proposed model can be viewed as a conditional VAE; a posterior encoder, decoder, and conditional prior (green blocks: a normalizing flow, linear projection layer, and text encoder) with a flow-based stochastic duration predictor.

图 4.5: VITS 整体结构

VITS 包括三个部分：

1. 后验编码器。如上图 (a) 的左下部分所示，在训练时输入线性谱，输出隐变量 z ，推断时隐变量 z 则由 f_θ 产生。VITS 的后验编码器采用 WaveGlow 和 Glow-TTS 中的非因果 WaveNet 残差模块。应用于多人模型时，将说话人嵌入向量添加进残差模块，仅用于训练。这里的隐变量 z 可以理解为 Tacotron / FastSpeech 中的梅尔频谱。
2. 解码器。如上图 (a) 左上部分所示，解码器从提取的隐变量 z 中生成语音波形，这个解码器实际就是声码器 HiFi-GAN V1 的生成器。应用于多人模型时，在说话人嵌入向量之后添加一个线性层，拼接到 f_θ 的输出隐变量 z 。
3. 先验编码器。如上图 (a) 右侧部分所示，先验编码器结构比较复杂，作用类似于 Tacotron / FastSpeech 的声学模型，只不过 VITS 是将音素映射为中间表示 z ，而不是将音素映射为频谱。包括文本编码器和提升先验分布复杂度的标准流 f_θ 。应用于多人模型时，向标准流的残差模块中添加说话人嵌入向量。
4. 随机时长预测器。如上图 (a) 右侧中间橙色部分。从条件输入 h_{text} 估算音素时长的分布。应用于多人模型时，在说话人嵌入向量之后添加一个线性层，并将其拼接到文本编码器的输出 h_{text} 。
5. 判别器。实际就是 HiFi-GAN 的多周期判别器，在上图中未画出，仅用于训练。目前看来，对于任意语音合成模型，加入判别器辅助都可以显著提升表现。

4.3.2 变分推断

VITS 可以看作是一个最大化变分下界，也即 ELBO (Evidence Lower Bound) 的条件 VAE。

1. 重建损失

VITS 在训练时实际还是会生成梅尔频谱以指导模型的训练，重建损失中的目标使用的是梅尔频谱而非原始波形：

$$L_{recon} = \|x_{mel} - \hat{x}_{mel}\|_1 \quad (4.10)$$

但在推断时并不需要生成梅尔频谱。在实现上，不上采样整个隐变量 z ，而只是使用部分序列作为解码器的输入。

2. KL 散度

先验编码器 c 的输入包括从文本生成的音素 c_{text} ，和音素、隐变量之间的对齐 A 。所谓的对齐就是 $|c_{text}| \times |z|$ 大小的严格单调注意力矩阵，表示每一个音素的发音时长。因此 KL 散度是：

$$L_{kl} = \log q_\phi(z|x_{lin}) - \log p_\theta(z|c_{text}, A) \quad (4.11)$$

其中, $q_\phi(z|x_{lin})$ 表示给定输入 x 的后验分布, $p_\theta(z|c)$ 表示给定条件 c 的隐变量 z 的先验分布。其中隐变量 z 为:

$$z \sim q_\phi(z|x_{lin}) = \mathbb{N}(z; \mu_\phi(x_{lin}), \sigma_\phi(x_{lin})) \quad (4.12)$$

为了给后验编码器提供更高分辨率的信息, 使用线性谱而非梅尔频谱作为后验编码器 ϕ_θ 的输入。同时, 为了生成更加逼真的样本, 提高先验分布的表达能力比较重要, 因此引入标准化流, 在文本编码器产生的简单分布和复杂分布间进行可逆变换。也就是说, 在经过上采样的编码器输出之后, 加入一系列可逆变换:

$$p_\theta(z|c) = \mathbb{N}(f_\theta(z); \mu_\theta(c), \sigma_\theta(c)) |\det \frac{\partial f_\theta(z)}{\partial z}| \quad (4.13)$$

其中, 上式中的 c 就是上采样的编码器输出:

$$c = [c_{text}, A] \quad (4.14)$$

4.3.3 对齐估计

由于在训练时没有对齐的真实标签, 因此在训练的每一次迭代时都需要估计对齐。

1. 单调对齐搜索

为了估计文本和语音之间的对齐 A , VITS 采用了类似于 Glow-TTS 中的单调对齐搜索 (Monotonic Alignment Search, MAS) 方法, 该方法寻找一个最优的对齐路径以最大化利用标准化流 f 参数化数据的对数似然:

$$\underset{\hat{A}}{\operatorname{argmax}} \log p_\theta(x|c_{text}, \hat{A}) = \underset{\hat{A}}{\operatorname{argmax}} \log \mathbb{N}(f(x); \mu(c_{text}, \hat{A}), \sigma(c_{text}, \hat{A})) \quad (4.15)$$

MAS 约束获得的最优对齐必须是单调且无跳过的。但是无法直接将 MAS 直接应用到 VITS, 因为 VITS 优化目标是 ELBO 而非确定的隐变量 z 的对数似然, 因此稍微改变了一下 MAS, 寻找最优的对齐路径以最大化 ELBO:

$$\underset{\hat{A}}{\operatorname{argmax}} \log p_\theta(x_{mel}|z) - \log \frac{q_\theta(z|x_{lin})}{p_\theta(z|c_{text}, \hat{A})} \quad (4.16)$$

2. 随机时长预测器

随机时长预测器是一个基于流的生成模型, 训练目标为音素时长对数似然的变分下界:

$$\log p_\theta(d|c_{text}) \geq \mathbb{E}_{q_\theta(u, v|d, c_{text})} [\log \frac{p_\theta(d-u, v|c_{text})}{q_\phi(u, v|d, c_{text})}] \quad (4.17)$$

在训练时, 断开随机时长预测器的梯度反传, 以防止该部分的梯度影响到其它模块。音素时长通过随机时长预测器的可逆变换从随机噪音中采样获得, 之后转换为整型值。

4.3.4 对抗训练

引入判别器 D 判断输出是解码器 G 的输出, 还是真实的波形 y 。VITS 用于对抗训练的损失函数包括两个部分, 第一部分是用于对抗训练的最小二乘损失函数 (least-squares loss function):

$$\mathcal{L}_{adv}(D) = \mathbb{E}_{(y, z)} [(D(y) - 1)^2 + (D(G(z)))^2] \quad (4.18)$$

$$\mathcal{L}_{adv}(G) = \mathbb{E}_z [(D(G(z)) - 1)^2] \quad (4.19)$$

第二部分是仅作用于生成器的特征匹配损失 (feature-matching loss):

$$\mathcal{L}_{fm}(G) = \mathbb{E}_{(y, c)} [\sum_{l=1}^T \frac{1}{N_l} \|D^l(y) - D^l(G(z))\|_1] \quad (4.20)$$

其中, T 表示判别器的层数, D^l 表示第 l 层判别器的特征图 (feature map), N_l 表示特征图的数量。特征匹配损失可以看作是重建损失, 用于约束判别器中间层的输出。

4.3.5 总体损失

VITS 可以看作是 VAE 和 GAN 的联合训练，因此总体损失为：

$$L_{vae} = L_{recon} + L_{kl} + L_{dur} + L_{adv} + L_{fm}(G) \quad (4.21)$$

4.3.6 总结

VITS 是一种由字符或音素直接映射为波形的端到端语音合成模型，该语音合成模型采用对抗训练的模式，生成器多个模块基于标准化流。模型较大，合成质量优异。VITS 的想法相当有启发，但是理解起来确实比较难，特别是标准化流，可参考：[Awesome Normalizing Flows](#)。

第5章 声码器

声码器（Vocoder），又称语音信号分析合成系统，负责对声音进行分析和合成，主要用于合成人类的语音。声码器主要由以下功能：

1. 分析 Analysis
2. 操纵 Manipulation
3. 合成 Synthesis

分析过程主要是从一段原始声音波形中提取声学特征，比如线性谱、MFCC；操纵过程是指对提取的原始声学特征进行压缩等降维处理，使其表征能力进一步提升；合成过程是指将此声学特征恢复至原始波形。人类发声机理可以用经典的源-滤波器模型建模，也就是输入的激励部分通过线性时不变进行操作，输出的声音谐振部分作为合成语音。输入部分被称为激励部分（Source Excitation Part），激励部分对应肺部气流与声带共同作用形成的激励，输出结果被称为声道谐振部分（Vocal Tract Resonance Part），对应人类发音结构，而声道谐振部分对应于声道的调音部分，对声音进行调制。

声码器的发展可以分为两个阶段，包括用于统计参数语音合成（Statistical Parameteric Speech Synthesis, SPSS）基于信号处理的声码器，和基于神经网络的声码器。常用基于信号处理的声码器包括 Griffin-Lim¹, STRAIGHT²和 WORLD³。早期神经声码器包括 WaveNet、WaveRNN 等，近年来神经声码器发展迅速，涌现出包括 MelGAN、HiFiGAN、LPCNet、NHV 等优秀的工作。

5.1 Griffin Lim 声码器

在早期的很多 Tacotron 开源语音合成模型中均采用 Griffin Lim 声码器，同时也有一些专门的开源实现，比如 GriffinLim。

GriffinLim 将幅度谱恢复为原始波形，但是相比原始波形，幅度谱缺失了原始相位谱信息。音频一般采用的是短视傅里叶变化，也就是将音频分割成帧（每帧 0.25ms-0.35ms），再进行傅里叶变化，帧与帧之间是有重叠的（可百度了解 stft）。

griffin lim 算法是利用两帧之间的重叠部分重构信号，而且中间用了 hanning 窗等，所以如果要 griffin lim 算法还原的信号质量较好，尽量保证两帧之间重叠越多越好（一般帧移为每一帧长的 25% 左右，也就是 75% 的重叠为宜）。

5.1.1 代码实现

摘抄自 Build End-To-End TTS Tacotron: Griffin Lim 信号估计算法。

```
1 def griffin_lim(stftm_matrix, shape, min_iter=20, max_iter=50, delta=20):
2     y = np.random.random(shape)
3     y_iter = []
4
5     for i in range(max_iter):
6         if i >= min_iter and (i - min_iter) % delta == 0:
7             y_iter.append((y, i))
```

¹Griffin D. and Lim J. (1984). "Signal Estimation from Modified Short-Time Fourier Transform". IEEE Transactions on Acoustics, Speech and Signal Processing. 32 (2): 236–243. doi:10.1109/TASSP.1984.1164317

²Kawahara H. Speech representation and transformation using adaptive interpolation of weighted spectrum: vocoder revisited[C]. 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing. IEEE, 1997, 2: 1303-1306.

³Morise M, Yokomori F, Ozawa K. World: a vocoder-based high-quality speech synthesis system for real-time applications[J]. IEICE TRANSACTIONS on Information and Systems, 2016, 99(7): 1877-1884.

```

8     stft_matrix = librosa.core.stft(y)
9     stft_matrix = stftm_matrix * stft_matrix / np.abs(stft_matrix)
10    y = librosa.core.istft(stft_matrix)
11    y_iter.append((y, max_iter))
12
13    return y_iter

```

具体使用：

```

1 # assume 1 channel wav file
2 sr, data = scipy.io.wavfile.read(argv[0])
3
4 # 由 STFT -> STFT magnitude
5 stftm_matrix = np.abs(librosa.core.stft(data))
6 # + random 模拟 modification
7 stftm_matrix_modified = stftm_matrix + np.random.random(stftm_matrix.shape)
8
9 # Griffin-Lim 估计音频信号
10 y_iters = griffin_lim(stftm_matrix_modified, data.shape)

```

<https://zhuanlan.zhihu.com/p/424846347> <https://zhuanlan.zhihu.com/p/25002923> <https://zhuanlan.zhihu.com/p/102539783>
<https://zhuanlan.zhihu.com/p/32321773> <https://zhuanlan.zhihu.com/p/66809424>

5.2 ElegantBook 更新说明

此次为 4.x 第一个版本，在 3.x 基础上，主要更新了定理以及参考文献的支持方式，具体内容有：

1. **重要改动**：由原先的 `BIBTEX` 改为 `biblatex` 编译方式（后端为 `biber`），请注意两者之间的差异；
2. **重要改进**：修改对于定理写法兼容方式，提高数学公式代码的兼容性；
3. 页面设置改动，默认页面更宽；方便书写和阅读；
4. 支持目录文字以及页码跳转；
5. 不再维护 `pdflATEX` 中文支持方式，请务必使用 `XeLATEX` 编译中文文稿。
6. 增加多语言选项，法语 `lang=fr`、德语 `lang=de`、荷兰语 `lang=nl`、匈牙利语 `lang=hu`、西班牙语 `lang=es`、蒙古语 `lang=mn` 等。

 **笔记** 如果你使用旧版本切换到新版本时，遇到问题时，请核对文档中是否有 `pageanchor` 字样。如果有，请删除文档中的 `\hypersetup{pageanchor=true}`，并且在 `\maketitle` 和 `\tableofcontents` 之间添加 `\frontmatter`。
2.x 版本的用户请仔细查看跨版本转换。

5.3 模板安装与更新

你可以通过免安装的方式使用本模板，包括在线使用和本地（文件夹内）使用两种方式，也可以通过 `TeX` 发行版安装使用。

5.3.1 在线使用模板

我们把三套模板全部上传到 `Overleaf` 上了，网络便利的用户可以直接通过 `Overleaf` 在线使用我们的模板。使用 `Overleaf` 的好处是无需安装 `TeX Live 2020`，可以随时随地访问自己的文件。查找模板，请在 `Overleaf` 模板库里面搜索 `elegantlatex` 即可，你也可以直接访问 [搜索结果](#)。选择适当的模板之后，将其 `Open as Template`，即

可把模板存到自己账户下，然后可以自由编辑以及与别人一起协作。更多关于 Overleaf 的介绍和使用，请参考 Overleaf 的[官方文档](#)。

注 Overleaf 上，中文需要使用 X_EL^AT_EX 进行编译，英文建议使用 pdfL^AT_EX 编译。

5.3.2 本地免安装使用

免安装使用方法如下，从 GitHub 或者 CTAN 下载最新版，严格意义上只需要类文件 `elegantbook.cls`。然后将模板文件放在你的工作目录下即可使用。这样使用的好处是，无需安装，简便；缺点是，当模板更新之后，你需要手动替换 `cls` 文件。

5.3.3 发行版安装使用

本模板测试环境为 Win10 和 TeX Live 2021，如果你刚安装 TeX Live 2021 用户，安装后建议升级全部宏包，升级方法：使用 cmd 运行 `tlmgr update --all`，如果 tlmgr 需要更新，请使用 cmd 运行 `tlmgr update --self`，如果更新过程中出现了中断，请改用 `tlmgr update --self --all --reinstall-forcibly-removed` 更新。

5.3.4 更新问题

如果使用 tlshell 无法更新模板，请使用命令行全部更新全部宏包或者使用免安装的方法使用本模板。

通过命令行（管理员权限）输入下面的命令对 tlmgr 自身和全部宏包进行更新。

```
1 tlmgr update --self
2 tlmgr update --all
```

更多的内容请参考 [How do I update my TeX distribution?](#)

5.3.5 其他发行版本

由于宏包版本问题，本模板不支持 C_TE_X 套装，请务必安装 TeX Live。更多关于 TeX Live 的安装使用以及 C_TE_X 与 TeX Live 的兼容、系统路径问题，请参考[官方文档](#)以及[一份简短的安装 L^AT_EX 的介绍](#)。

5.4 关于提交

出于某些因素的考虑，ElegantL^AT_EX 项目自 2019 年 5 月 20 日开始，**不再接受任何非作者预约性质的提交**（pull request）！如果你想改进模板，你可以给我们提交 issues，或者可以在遵循协议（LPPL-1.3c）的情况下，克隆到自己仓库下进行修改。

第6章 ElegantBook 设置说明

本模板基于基础的 book 文类，所以 book 的选项对于本模板也是有效的（纸张无效，因为模板有设备选项）。默认编码为 UTF-8，推荐使用 TeX Live 编译。本文编写环境为 Win10 (64bit) + TeX Live 2021，英文支持 pdfLATEX，中文仅支持 XE_LA^TE_X 编译。

6.1 语言模式

本模板内含两套基础语言环境 lang=cn、lang=en。改变语言环境会改变图表标题的引导词（图，表），文章结构词（比如目录，参考文献等），以及定理环境中的引导词（比如定理，引理等）。不同语言模式的启用如下：

```
1 \documentclass[en]{elegantbook}
2 \documentclass[lang=cn]{elegantbook}
```

除模板自带的两套语言设定之外，由网友提供的其他语言环境设置如下：

- 由 VincentMVV 提供的意大利语翻译 lang=it，相关讨论见 [Italian translation](#)；
- 由 abfek66 提供的法语翻译 lang=fr，相关讨论见 [Italian translation](#)；
- 由 inktvis75 提供的荷兰语翻译 lang=nl，相关讨论见 [Dutch Translation](#)；
- 由 palkotamas 提供的匈牙利语翻译 lang=hu，相关讨论见 [Hungarian translation](#)；
- 由 Lisa 提供的德语翻译 lang=de，相关讨论见 [Deutsch translation](#)；
- 由 Gustavo A. Corradi 提供的西班牙语的翻译 lang=es，相关讨论见 [Spanish translation](#)；
- 由 Altantssooj 提供的蒙古语的翻译 lang=mn，相关讨论见 [Mongolian translation](#)。

注 以上各个语言的设定均为网友设定，我们未对上述翻译进行过校对，如果有问题，请在对应的 issue 下评论。并且，只有中文环境 (lang=cn) 才可以输入中文。

6.2 设备选项

最早我们在 ElegantNote 模板中加入了设备选项 (device)，后来，我们认为这个设备选项的设置可以应用到 ElegantBook 中¹，而且 Book 一般内容比较多，如果在 iPad 上看无需切边，放大，那用户的阅读体验将会得到巨大提升。你可以使用下面的选项将版面设置为 iPad 设备模式²

```
1 \documentclass[pad]{elegantbook} %or
2 \documentclass[device=pad]{elegantbook}
```

6.3 颜色主题

本模板内置 5 组颜色主题，分别为 green³、cyan、blue（默认）、gray、black。另外还有一个自定义的选项 nocolor。调用颜色主题 green 的方法为

```
1 \documentclass[green]{elegantbook} %or
2 \documentclass[color=green]{elegantbook}
```

¹不过因为 ElegantBook 模板封面图片的存在，在修改页面设计时，需要对图片进行裁剪。

²默认为 normal 模式，也即 A4 纸张大小。

³为原先默认主题。

表 6.1: ElegantBook 模板中的颜色主题

	green	cyan	blue	gray	black	主要使用的环境
structure						chapter section subsection
main						definition exercise problem
second						theorem lemma corollary
third						proposition

如果需要自定义颜色的话请选择 `nocolor` 选项或者使用 `color=none`, 然后在导言区定义 `structurecolor`、`main`、`second`、`third` 颜色, 具体方法如下:

```
1 \definecolor{structurecolor}{RGB}{0,0,0}
2 \definecolor{main}{RGB}{70,70,70}
3 \definecolor{second}{RGB}{115,45,2}
4 \definecolor{third}{RGB}{0,80,80}
```

6.4 封面

6.4.1 封面个性化

从 3.10 版本开始, 封面更加弹性化, 用户可以自行选择输出的内容, 包括 `\title` 在内的所有封面元素都可为空。目前封面的元素有

表 6.2: 封面元素信息

信息	命令	信息	命令	信息	命令
标题	<code>\title</code>	副标题	<code>\subtitle</code>	作者	<code>\author</code>
机构	<code>\institute</code>	日期	<code>\date</code>	版本	<code>\version</code>
箴言	<code>\extrainfo</code>	封面图	<code>\cover</code>	徽标	<code>\logo</code>

另外, 额外增加一个 `\bioinfo` 命令, 有两个选项, 分别是信息标题以及信息内容。比如需要显示 User Name: 111520, 则可以使用

```
1 \bioinfo{User Name}{111520}
```

封面中间位置的色块的颜色可以使用下面命令进行修改:

```
1 \definecolor{customcolor}{RGB}{32,178,170}
2 \colorlet{coverlinecolor}{customcolor}
```

6.4.2 封面图

本模板使用的封面图片来源于 pixabay.com⁴, 图片完全免费, 可用于任何场景。封面图片的尺寸为 1280×1024, 更换图片的时候请严格按照封面图片尺寸进行裁剪。推荐一个免费的在线图片裁剪网站 fotor.com。用户 QQ 群

⁴感谢 ChinaTeX 提供免费图源网站, 另外还推荐 pexels.com。

内有一些合适尺寸的封面，欢迎取用。

6.4.3 徽标

本文用到的 Logo 比例为 1:1，也即正方形图片，在更换图片的时候请选择合适的图片进行替换。

6.4.4 自定义封面

另外，如果使用自定义的封面，比如 Adobe illustrator 或者其他软件制作的 A4 PDF 文档，请把 `\maketitle` 注释掉，然后借助 `pdfpages` 宏包将自制封面插入即可。如果使用 `titlepage` 环境，也是类似。如果需要 2.x 版本的封面，请参考 `etitlepage`。

6.5 章标标题

本模板内置 2 套章标题显示风格，包含 `hang`（默认）与 `display` 两种风格，区别在于章标题单行显示（`hang`）与双行显示（`display`），本说明使用了 `hang`。调用方式为

```
1 \documentclass[hang]{elegantbook} %or
2 \documentclass[titlestyle=hang]{elegantbook}
```

在章标题内，章节编号默认是以数字显示，也即第 1 章，第 2 章等等，如果想要把数字改为中文，可以使用

```
1 \documentclass[chinese]{elegantbook} %or
2 \documentclass[scheme=chinese]{elegantbook}
```

6.6 数学环境简介

在我们这个模板中，我们定义了两种不同的定理模式 `mode`，包括简单模式（`simple`）和炫彩模式（`fancy`），默认为 `fancy` 模式，不同模式的选择为

```
1 \documentclass[simple]{elegantbook} %or
2 \documentclass[mode=simple]{elegantbook}
```

本模板定义了四大类环境

- 定理类环境，包含标题和内容两部分，全部定理类环境的编号均以章节编号。根据格式的不同分为 3 种
 - `definition` 环境，颜色为 `main`；
 - `theorem`、`lemma`、`corollary` 环境，颜色为 `second`；
 - `proposition` 环境，颜色为 `third`。
- 示例类环境，有 `example`、`problem`、`exercise` 环境（对应于例、例题、练习），自动编号，编号以章节为单位，其中 `exercise` 有提示符。
- 提示类环境，有 `note` 环境，特点是：无编号，有引导符。
- 结论类环境，有 `conclusion`、`assumption`、`property`、`remark`、`solution` 环境⁵，三者均以粗体的引导词为开头，和普通段落格式一致。

⁵本模板还添加了一个 `result` 选项，用于隐藏 `solution` 和 `proof` 环境，默認為显示 (`result=answer`)，隐藏使用 `result=noanswer`。

6.6.1 定理类环境的使用

由于本模板使用了 `tcolorbox` 宏包来定制定理类环境，所以和普通的定理环境的使用有些许区别，定理的使用方法如下：

```
1 \begin{theorem}{theorem name}{label}
2   The content of theorem.
3 \end{theorem}
```

第一个必选项 `theorem name` 是定理的名字，第二个必选项 `label` 是交叉引用时所用到的标签，交叉引用的方法为 `\ref{thm:label}`。请注意，交叉引用时必须加上前缀 `thm:`。

在用户多次反馈下，4.x 之后，引入了原生定理的支持方式，也就是使用可选项方式：

```
1 \begin{theorem}[theorem name] \label{thm:theorem-label}
2   The content of theorem.
3 \end{theorem}
4 % or
5 \begin{theorem} \label{thm:theorem-without-name}
6   The content of theorem without name.
7 \end{theorem}
```

其他相同用法的定理类环境有：

表 6.3: 定理类环境

环境名	标签名	前缀	交叉引用
definition	label	def	<code>\ref{def:label}</code>
theorem	label	thm	<code>\ref{thm:label}</code>
lemma	label	lem	<code>\ref{lem:label}</code>
corollary	label	cor	<code>\ref{cor:label}</code>
proposition	label	pro	<code>\ref{pro:label}</code>

6.6.2 其他环境的使用

其他三种环境没有选项，可以直接使用，比如 `example` 环境的使用方法与效果：

```
1 \begin{example}
2   This is the content of example environment.
3 \end{example}
```

这几个都是同一类环境，区别在于

- 示例环境（example）、练习（exercise）与例题（problem）章节自动编号；
- 注意（note），练习（exercise）环境有提醒引导符；
- 结论（conclusion）等环境都是普通段落环境，引导词加粗。

6.7 列表环境

本模板借助于 `tikz` 定制了 `itemize` 和 `enumerate` 环境，其中 `itemize` 环境修改了 3 层嵌套，而 `enumerate` 环境修改了 4 层嵌套（仅改变颜色）。示例如下

- first item of nesti;
 - second item of nesti;
 - first item of nestii;
 - second item of nestii;
 - first item of nestiii;
 - second item of nestiii.
1. first item of nesti;
 2. second item of nesti;
 - (a). first item of nestii;
 - (b). second item of nestii;
 - I. first item of nestiii;
 - II. second item of nestiii.

6.8 参考文献

此模板使用了 biber 来生成参考文献，也即使用 biblatex 宏包，在中文示例中，使用了 gbt7714 宏包。参考文献示例:cn1,en2,en3 使用了中国一个大型的 P2P 平台（人人贷）的数据来检验男性投资者和女性投资者在投资表现上是否有显著差异。

你可以在谷歌学术, Mendeley, Endnote 中获得文献条目 (bib item)，然后把它们添加到 reference.bib 中。在文中引用的时候，引用它们的键值 (bib key) 即可。注意需要在编译的过程中添加 biber 编译。

为了方便文献样式修改，模板引入了 bibstyle 和 citestyle 选项，默认均为数字格式 (numeric)，如果需要设置为国标 GB7714-2015，需要使用：

```
1 \documentclass[citestyle=gb7714-2015, bibstyle=gb7714-2015]{elegantbook}
```

如果需要添加排序方式，可以在导言区加入

```
1 \ExecuteBibliographyOptions{sorting=ynt}
```

启用国标之后，可以加入 sorting=gb7714-2015。

6.9 添加序章

如果你想在第一章前面添序章，不改变原本章节序号，可以在第一章内容前面使用

```
1 \chapter*{Introduction}
2 \markboth{Introduction}{Introduction}
3 The content of introduction.
```

6.10 目录选项与深度

本模板添加了一个目录选项 toc，可以设置目录为单栏 (onecol) 和双栏 (twocol) 显示，比如双栏显示可以使用

```
1 \documentclass[twocol]{elegantbook}
2 \documentclass[toc=twocol]{elegantbook}
```

默认本模板目录深度为 1，你可以在导言区使用

```
1 \setcounter{tocdepth}{2}
```

将其修改为 2 级目录（章与节）显示。

6.11 章节摘要

模板新增了一个章节摘要环境 (introduction)，使用示例

```

1 \begin{introduction}
2   \item Definition of Theorem
3   \item Ask for help
4   \item Optimization Problem
5   \item Property of Cauchy Series
6   \item Angle of Corner
7 \end{introduction}
```

效果如下：

内容提要

- | | |
|--|--|
| <input type="checkbox"/> Definition of Theorem
<input type="checkbox"/> Ask for help
<input type="checkbox"/> Optimization Problem | <input type="checkbox"/> Property of Cauchy Series
<input type="checkbox"/> Angle of Corner |
|--|--|

环境的标题文字可以通过这个环境的可选参数进行修改，修改方法为：

```

1 \begin{introduction}[Brief Introduction]
2 ...
3 \end{introduction}
```

6.12 章后习题

前面我们介绍了例题和练习两个环境，这里我们再加一个，章后习题 (problemset) 环境，用于在每一章结尾，显示本章的练习。使用方法如下

```

1 \begin{problemset}
2   \item exercise 1
3   \item exercise 2
4   \item exercise 3
5 \end{problemset}
```

效果如下：

第6章 练习

1. exercise 1
2. exercise 2
3. exercise 3
4. 测试数学公式

$$a^2 + b^2 = c_{2i}(1, 2)[1, 23] \quad (6.1)$$

注 如果你想把 problemset 环境的标题改其他文字，你可以类似于 introduction 环境修改 problemset 的可选参数。另外，目前这个环境会自动出现在目录中，但是不会出现在页眉页脚信息中（待解决）。

解 如果你想把 problemset 环境的标题改其他文字，你可以类似于 introduction 环境修改 problemset 的可选参数。另外，目前这个环境会自动出现在目录中，但是不会出现在页眉页脚信息中（待解决）。

6.13 旁注

在 3.08 版本中，我们引入了旁注设置选项 `marginpar=margintrue` 以及测试命令 `\elegantpar`，但是由此带来一堆问题。我们决定在 3.09 版本中将其删除，并且，在旁注命令得到大幅度优化之前，不会将此命令再次引入书籍模板中。对此造成各位用户的不方便，非常抱歉！不过我们保留了 `marginpar` 这个选项，你可以使用 `marginpar=margintrue` 获得保留右侧旁注的版面设计。然后使用系统自带的 `\marginpar` 或者 `marginnote` 宏包的 `\marginnote` 命令。

注 在使用旁注的时候，需要注意的是，文本和公式可以直接在旁注中使用。

```

1 % text
2 \marginpar{margin paragraph text}
3
4 % equation
5 \marginpar{
6   \begin{equation}
7     a^2 + b^2 = c^2
8   \end{equation}
9 }
```

但是浮动体（表格、图片）需要注意，不能用浮动体环境，需要使用直接插图命令或者表格命令环境。然后使用 `\captionof` 为其设置标题。为了得到居中的图表，可以使用 `\centerline` 命令或者 `center` 环境。更多详情请参考：[Caption of Figure in Marginpar](#)。

```

1 % graph with centerline command
2 \marginpar{
3   \centerline{
4     \includegraphics[width=0.2\textwidth]{logo.png}
5   }
6   \captionof{figure}{your figure caption}
7 }
8
9 % graph with center environment
10 \marginpar{
11   \begin{center}
12     \includegraphics[width=0.2\textwidth]{logo.png}
13     \captionof{figure}{your figure caption}
14   \end{center}
15 }
```

第 7 章 字体选项

字体选项独立成章的原因是，我们希望本模板的用户关心模板使用的字体，知晓自己使用的字体以及遇到字体相关的问题能更加便捷地找到答案。

重要提示：从 3.10 版本更新之后，沿用至今的 newtx 系列字体被重新更改为 cm 字体。并且新增中文字体 (chinesefont) 选项。

7.1 数学字体选项

本模板定义了一个数学字体选项 (math)，可选项有三个：

1. math=cm (默认)，使用 L^AT_EX 默认数学字体（推荐，无需声明）；
2. math=newtx，使用 newtxmath 设置数学字体（潜在问题比较多）。
3. math=mtpro2，使用 mtpro2 宏包设置数学字体，要求用户已经成功安装此宏包。

7.2 使用 newtx 系列字体

如果需要使用原先版本的 newtx 系列字体，可以通过显示声明数学字体：

```
1 \documentclass [math=newtx]{elegantbook}
```

7.2.1 连字符

如果使用 newtx 系列字体宏包，需要注意下连字符的问题。

$$\int_{R^q} f(x, y) dy. off \quad (7.1)$$

的代码为

```
1 \begin{equation}
2   \int_{R^q} f(x, y) dy. \emph{of} \kern0pt f
3 \end{equation}
```

7.2.2 宏包冲突

另外在 3.08 版本中，有用户反馈模板在和 yhmath 以及 esvect 等宏包搭配使用的时候会出现报错：

```
1 LaTeX Error:
2 Too many symbol fonts declared.
```

原因是在使用 newtxmath 宏包时，重新定义了数学字体用于大型操作符，达到了最多 16 个数学字体的上限，在调用其他宏包的时候，无法新增数学字体。为了减少调用非常用宏包，在此给出如何调用 yhmath 以及 esvect 宏包的方法。

请在 elegantbook.cls 内搜索 yhmath 或者 esvect，将你所需要的宏包加载语句取消注释即可。

```
1 %% use yhmath pkg, uncomment following code
2 % \let\oldwidering\widering
3 % \let\widering\undefined
4 % \RequirePackage{yhmath}
```

```

5 % \let\widering\oldwidering
6
7 %%% use esvect pkg, uncomment following code
8 % \RequirePackage{esvect}

```

7.3 中文字体选项

模板从 3.10 版本提供中文字体选项 `chinesefont`, 可选项有

1. `ctexfont`: 默认选项, 使用 `ctex` 宏包根据系统自行选择字体, 可能存在字体缺失的问题, 更多内容参考 [ctex 宏包官方文档¹](#)。
2. `founder`: 方正字体选项, 调用 `ctex` 宏包并且使用 `fontset=none` 选项, 然后设置字体为方正四款免费字体, 方正字体下载注意事项见后文。
3. `nofont`: 调用 `ctex` 宏包并且使用 `fontset=none` 选项, 不设定中文字体, 用户可以自行设置中文字体, 具体见后文。

注 使用 `founder` 选项或者 `nofont` 时, 必须使用 `XeLaTeX` 进行编译。

7.3.1 方正字体选项

由于使用 `ctex` 宏包默认调用系统已有的字体, 部分系统字体缺失严重, 因此, 用户希望能够使用其它字体, 我们推荐使用方正字体。方正的方正黑体、方正宋、方正楷体、方正仿宋四款字体均可免费试用, 且可用于商业用途。用户可以自行从[方正字体官网](#)下载此四款字体, 在下载的时候请**务必**注意选择 GBK 字符集, 也可以使用 [LaTeX 工作室](#)提供的**方正字体**, 提取码为: `njy9` 进行安装。安装时, Win 10 用户请右键选择为全部用户安装, 否则会找不到字体。

字体名称	编码	单价	实付价	交易状态	操作
订单号: C20200204164821OW1F				2020-02-04 16:48:21	
方正仿宋_GBK • 免费商用	简繁扩展(GBK)	¥ 0.00			
方正黑体_GBK • 免费商用	简繁扩展(GBK)	¥ 0.00		免费	已完成
方正书宋_GBK • 免费商用	简繁扩展(GBK)	¥ 0.00			
方正楷体_GBK • 免费商用	简繁扩展(GBK)	¥ 0.00			

7.3.2 其他中文字体

如果你想完全自定义字体², 你可以选择 `chinesefont=nofont`, 然后在导言区设置

```

1 \setCJKmainfont [BoldFont={FZHei-B01}, ItalicFont={FZKai-Z03}]{FZShuSong-Z01}

```

¹可以使用命令提示符, 输入 `texdoc ctex` 调出本地 `ctex` 宏包文档

²这里仍然以方正字体为例。

```
2 \setCJKsansfont [BoldFont={FZHei-B01},ItalicFont={FZHei-B01}]{FZHei-B01}
3 \setCJKmonofont [BoldFont={FZHei-B01},ItalicFont={FZHei-B01}]{FZFangSong-Z02}
4 \setCJKfamilyfont{zhsong}{FZShuSong-Z01}
5 \setCJKfamilyfont{zhhei}{FZHei-B01}
6 \setCJKfamilyfont{zhkai}{FZKai-Z03}
7 \setCJKfamilyfont{zhfs}{FZFangSong-Z02}
8 \newcommand*\songti{\CJKfamily{zhsong}}
9 \newcommand*\heiti{\CJKfamily{zhhei}}
10 \newcommand*\kaishu{\CJKfamily{zhkai}}
11 \newcommand*\fangsong{\CJKfamily{zhfs}}
```

第8章 ElegantBook 写作示例

内容提要

- 积分定义 8.1
- 柯西列性质 8.1.1
- Fubini 定理 8.1
- 韦达定理
- 最优化原理 8.1

8.1 Lebesgue 积分

在前面各章做了必要的准备后，本章开始介绍新的积分。在 Lebesgue 测度理论的基础上建立了 Lebesgue 积分，其被积函数和积分域更一般，可以对有界函数和无界函数统一处理。正是由于 Lebesgue 积分的这些特点，使得 Lebesgue 积分比 Riemann 积分具有在更一般条件下的极限定理和累次积分交换积分顺序的定理，这使得 Lebesgue 积分不仅在理论上更完善，而且在计算上更灵活有效。

Lebesgue 积分有几种不同的定义方式。我们将采用逐步定义非负简单函数，非负可测函数和一般可测函数积分的方式。

由于现代数学的许多分支如概率论、泛函分析、调和分析等常常用到一般空间上的测度与积分理论，在本章最后一节将介绍一般的测度空间上的积分。

8.1.1 积分的定义

我们将通过三个步骤定义可测函数的积分。首先定义非负简单函数的积分。以下设 E 是 \mathcal{R}^n 中的可测集。

定义 8.1 (可积性)

设 $f(x) = \sum_{i=1}^k a_i \chi_{A_i}(x)$ 是 E 上的非负简单函数，其中 $\{A_1, A_2, \dots, A_k\}$ 是 E 上的一个可测分割， a_1, a_2, \dots, a_k 是非负实数。定义 f 在 E 上的积分为 $\int_a^b f(x) dx$

$$\int_E f dx = \sum_{i=1}^k a_i m(A_i) \pi \alpha \beta \sigma \gamma \nu \xi \epsilon \varepsilon. \oint_a^b \oint_a^b \prod_{i=1}^n$$
 (8.1)

一般情况下 $0 \leq \int_E f dx \leq \infty$ 。若 $\int_E f dx < \infty$ ，则称 f 在 E 上可积。

一个自然的问题是，Lebesgue 积分与我们所熟悉的 Riemann 积分有什么联系和区别？在 4.4 在我们将详细讨论 Riemann 积分与 Lebesgue 积分的关系。这里只看一个简单的例子。设 $D(x)$ 是区间 $[0, 1]$ 上的 Dirichlet 函数。即 $D(x) = \chi_{Q_0}(x)$ ，其中 Q_0 表示 $[0, 1]$ 中的有理数的全体。根据非负简单函数积分的定义， $D(x)$ 在 $[0, 1]$ 上的 Lebesgue 积分为

$$\int_0^1 D(x) dx = \int_0^1 \chi_{Q_0}(x) dx = m(Q_0) = 0$$
 (8.2)

即 $D(x)$ 在 $[0, 1]$ 上是 Lebesgue 可积的并且积分值为零。但 $D(x)$ 在 $[0, 1]$ 上不是 Riemann 可积的。

有界变差函数是与单调函数有密切联系的一类函数。有界变差函数可以表示为两个单调递增函数之差。与单调函数一样，有界变差函数几乎处处可导。与单调函数不同，有界变差函数类对线性运算是封闭的，它们构成一线空间。练习题 8.1 是一个性质的证明。

练习 8.1 设 $f \notin L(\mathcal{R}^1)$ ， g 是 \mathcal{R}^1 上的有界可测函数。证明函数

$$I(t) = \int_{\mathcal{R}^1} f(x+t) g(x) dx \quad t \in \mathcal{R}^1$$
 (8.3)

是 \mathcal{R}^1 上的连续函数。

解 即 $D(x)$ 在 $[0, 1]$ 上是 Lebesgue 可积的并且积分值为零。但 $D(x)$ 在 $[0, 1]$ 上不是 Riemann 可积的。

证明 即 $D(x)$ 在 $[0, 1]$ 上是 Lebesgue 可积的并且积分值为零。但 $D(x)$ 在 $[0, 1]$ 上不是 Riemann 可积的。

定理 8.1 (Fubini 定理)

(1) 若 $f(x, y)$ 是 $\mathcal{R}^p \times \mathcal{R}^q$ 上的非负可测函数，则对几乎处处的 $x \in \mathcal{R}^p$, $f(x, y)$ 作为 y 的函数是 \mathcal{R}^q 上的非负可测函数， $g(x) = \int_{\mathcal{R}^q} f(x, y) dy$ 是 \mathcal{R}^p 上的非负可测函数。并且

$$\int_{\mathcal{R}^p \times \mathcal{R}^q} f(x, y) dx dy = \int_{\mathcal{R}^p} \left(\int_{\mathcal{R}^q} f(x, y) dy \right) dx. \quad (8.4)$$

(2) 若 $f(x, y)$ 是 $\mathcal{R}^p \times \mathcal{R}^q$ 上的可积函数，则对几乎处处的 $x \in \mathcal{R}^p$, $f(x, y)$ 作为 y 的函数是 \mathcal{R}^q 上的可积函数，并且 $g(x) = \int_{\mathcal{R}^q} f(x, y) dy$ 是 \mathcal{R}^p 上的可积函数。而且 8.4 成立。



8.1

笔记 在本模板中，引理 (lemma), 推论 (corollary) 的样式和定理 8.1 的样式一致，包括颜色，仅仅只有计数器的设置不一样。

我们说一个实变或者复变量的实值或者复值函数是在区间上平方可积的，如果其绝对值的平方在该区间上的积分是有限的。所有在勒贝格积分意义下平方可积的可测函数构成一个希尔伯特空间，也就是所谓的 L^2 空间，几乎处处相等的函数归为同一等价类。形式上， L^2 是平方可积函数的空间和几乎处处为 0 的函数空间的商空间。

命题 8.1 (最优化原理)

如果 u^* 在 $[s, T]$ 上为最优解，则 u^* 在 $[s, T]$ 任意子区间都是最优解，假设区间为 $[t_0, t_1]$ 的最优解为 u^* ，则 $u(t_0) = u^*(t_0)$ ，即初始条件必须还是在 u^* 上。



我们知道最小二乘法可以用来处理一组数据，可以从一组测定的数据中寻求变量之间的依赖关系，这种函数关系称为经验公式。本课题将介绍最小二乘法的精确定义及如何寻求点与点之间近似成线性关系时的经验公式。假定实验测得变量之间的 n 个数据，则在平面上，可以得到 n 个点，这种图形称为“散点图”，从图中可以粗略看出这些点大致散落在某直线近旁，我们认为其近似为一线性函数，下面介绍求解步骤。

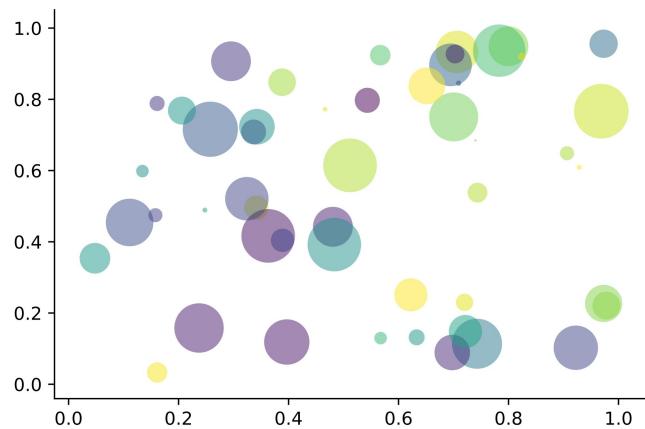


图 8.1: 散点图示例 $\hat{y} = a + bx$

以最简单的一元线性模型来解释最小二乘法。什么是一元线性模型呢？监督学习中，如果预测的变量是离散的，我们称其为分类（如决策树，支持向量机等），如果预测的变量是连续的，我们称其为回归。回归分析中，如果只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为多元线性回

归分析。对于二维空间线性是一条直线；对于三维空间线性是一个平面，对于多维空间线性是一个超平面。

性质 柯西列的性质

1. $\{x_k\}$ 是柯西列，则其子列 $\{x_k^i\}$ 也是柯西列。
2. $x_k \in \mathbb{R}^n$, $\rho(x, y)$ 是欧几里得空间，则柯西列收敛， (\mathbb{R}^n, ρ) 空间是完备的。

结论 回归分析 (regression analysis) 是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。运用十分广泛，回归分析按照涉及的变量的多少，分为一元回归和多元回归分析；按照因变量的多少，可分为简单回归分析和多重回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。

第8章 练习

1. 设 A 为数域 K 上的 n 级矩阵。证明：如果 K^n 中任意非零列向量都是 A 的特征向量，则 A 一定是数量矩阵。
2. 证明：不为零矩阵的幂零矩阵不能对角化。
3. 设 $A = (a_{ij})$ 是数域 K 上的一个 n 级上三角矩阵，证明：如果 $a_{11} = a_{22} = \dots = a_{nn}$ ，并且至少有一个 $a_{kl} \neq 0 (k < l)$ ，则 A 一定不能对角化。

第9章 常见问题集

我们根据用户社区反馈整理了下面一些常见的问题，用户在遇到问题时，应当首先查阅本手册和本部分的常见问题。

1. 有没有办法章节用“第一章，第一节，(一)”这种？

见前文介绍，可以使用 `scheme=chinese` 设置。

2. 大佬，我想把正文字体改为亮色，背景色改为黑灰色。

页面颜色可以使用 `\pagecolor` 命令设置，文本命令可以参考[这里](#)进行设置。

3. Package `ctex` Error: CTeX fontset ‘Mac’ is unavailable.

在 Mac 系统下，中文编译请使用 `XeLaTeX`。

4. ! LaTeX Error: Unknown option ‘scheme=plain’ for package ‘ctex’.

你用的 CTeX 套装吧？这个里面的 `ctex` 宏包已经是 10 年前的了，与本模板使用的 `ctex` 宏集有很大区别。不建议 CTeX 套装了，请卸载并安装 TeX Live 2021。

5. 我该使用什么版本？

请务必使用[最新正式发行版](#)，发行版间不定期可能会有更新（修复 bug 或者改进之类），如果你在使用过程中没有遇到问题，不需要每次更新[最新版](#)，但是在发行版更新之后，请尽可能使用最新版（发行版）！最新发行版可以在 GitHub 或者 TeX Live 2021 内获取。

6. 我该使用什么编辑器？

你可以使用 TeX Live 2021 自带的编辑器 TeXworks 或者使用 TeXstudio，TeXworks 的自动补全，你可以参考我们的总结 [TeXworks 自动补全](#)。推荐使用 TeX Live 2021 + TeXstudio。我自己用 VS Code 和 Sublime Text，相关的配置说明，请参考 [LaTeX 编译环境配置：Visual Studio Code 配置简介](#) 和 [Sublime Text 搭建 LaTeX 编写环境](#)。

7. 您好，我们想用您的 ElegantBook 模板写一本书。关于机器学习的教材，希望获得您的授权，谢谢您的宝贵时间。

模板的使用修改都是自由的，你们声明模板来源以及模板地址（GitHub 地址）即可，其他未尽事宜按照开源协议 LPPL-1.3c。做好之后，如果方便的话，可以给我们一个链接，我把你们的教材放在 ElegantLaTeX 用户作品集里。

8. 请问交叉引用是什么？

本群和本模板适合有一定 LaTeX 基础的用户使用，新手请先学习 LaTeX 的基础，理解各种概念，否则你将寸步难行。

9. 定义等环境中无法使用加粗命令么？

是这样的，默认中文并没有加粗命令，如果你想在定义等环境中使用加粗命令，请使用 `\heiti` 等字体命令，而不要使用 `\textbf`。或者，你可以将 `\textbf` 重新定义为 `\heiti`。英文模式不存在这个问题。

10. 代码高亮环境能用其他语言吗？

可以的，ElegantBook 模板用的是 `listings` 宏包，你可以在环境 (`lstlisting`) 之后加上语言（比如 Python 使用 `language=Python` 选项），全局语言修改请使用 `lset` 命令，更多信息请参考宏包文档。

11. 群主，什么时候出 Beamer 的模板（主题），ElegantSlide 或者 ElegantBeamer？

由于 Beamer 中有一个很优秀的主题 [Metropolis](#)。后续确定不会再出任何主题/模板，请大家根据需要修改已有主题。

第 10 章 版本更新历史

根据用户的反馈，我们不断修正和完善模板。截止到此次更新，ElegantBook 模板在 GitHub 上有将近 100 次提交，正式发行版本（release）有 17 次。由于 3.00 之前版本与现在版本差异非常大，在此不列出 3.00 之前的更新内容。

2021/05/02 更新：版本 4.1 正式发布。

- ① **重要改动：**由原先的 `BIBTEX` 改为 `biblatex` 编译方式（后端为 `biber`），请注意两者之间的差异；
- ② **重要改进：**修改对于定理写法兼容方式，提高数学公式代码的兼容性；
- ③ 页面设置改动，默认页面更宽；方便书写和阅读；
- ④ 支持目录文字以及页码跳转；
- ⑤ 不再维护 `pdflATEX` 中文支持方式，请务必使用 `XeLATEX` 编译中文文稿。
- ⑥ 增加多个语言选项，法语 `lang=fr`、荷兰语 `lang=nl`、匈牙利语 `lang=hu`、西班牙语 `lang=es`、蒙古语 `lang=mn` 等。

2020/04/12 更新：版本 3.11 正式发布，**此版本为 3.x 最后版本。**

- ① **重要修正：**修复因为 `gbt7714` 宏包更新导致的 `natbib` option `clash` 错误；
- ② 由于 `pgfornament` 宏包未被 TeX Live 2020 收录，因此删除 `base` 相关的内容；
- ③ 修复部分环境的空格问题；
- ④ 增加了意大利语言选项 `lang=it`。

2020/02/10 更新：版本 3.10 正式发布

- ① 增加数学字体选项 `math`，可选项为 `newtx` 和 `cm`。
重要提示：原先通过 `newtxmath` 宏包设置的数学字体改为 LATEX 默认数学字体，如果需要保持原来的字体，需要显式声明数学字体 (`math=newtx`)；
- ② 新增中文字体选项 `chinesefont`，可选项为 `ctexfont`、`founder` 和 `nofont`。
- ③ 将封面作者信息设置为可选，并且增加自定义信息命令 `\bioinfo`；
- ④ 在说明文档中增加版本历史，新增 `\datechange` 命令和 `change` 环境；
- ⑤ 增加汉化章节选项 `scheme`，可选项为汉化 `chinese`；
- ⑥ 由于 `\lvert` 问题已经修复，重新调整 `ctex` 宏包和 `amsmath` 宏包位置。
- ⑦ 修改页眉设置，去除了 `\lastpage` 以避免 page anchor 问题，加入 `\frontmatter`。
- ⑧ 修改参考文献选项 `cite`，可选项为数字 `numbers`、作者-年份 `authoryear` 以及上标 `super`。
- ⑨ 新增参考文献样式选项 `bibstyle`，并将英文模式下参考文献样式 `apalike` 设置为默认值，中文仍然使用 `gbt7714` 宏包设置。

2019/08/18 更新：版本 3.09 正式发布

- ① `\elegantpar` 存在 bug，删除 `\elegantpar` 命令，建议用户改用 `\marginnote` 和 `\marginpar` 旁注命令。
- ② 积分操作符统一更改为 `esint` 宏包设置；
- ③ 新增目录选项 `toc`，可选项为单栏 `onecol` 和双栏 `twocol`；
- ④ 手动增加参考文献选项 `cite`，可选项为上标形式 `super`；
- ⑤ 修正章节习题（`problemset`）环境。

2019/05/28 更新：版本 3.08 正式发布

- ① 修复 `\part` 命令。

-
- ② 引入 Note 模板中的 pad 选项 `device=pad`。
 - ③ 数学字体加入 `mtpro2` 可选项 `math=mtpro2`, 使用免费的 `lite` 子集。
 - ④ 将参考文献默认显示方式 `authoryear` 改为 `numbers`。
 - ⑤ 引入旁注命令 `\marginpar` (测试)。
 - ⑥ 新增章节摘要环境 `introduction`。
 - ⑦ 新增章节习题环境 `problemset`。
 - ⑧ 将 `\equation` 重命名为 `\extrainfo`。
 - ⑨ 完善说明文档, 增加致谢部分。
-

2019/04/15 更新: 版本 3.07 正式发布

- ① 删除中英文自定义字体总设置。
 - ② 新增颜色主题, 并将原绿色默认主题设置为蓝色 `color=blue`。
 - ③ 引入隐藏装饰图案选项 `base`, 可选项有显示 `show` 和隐藏 `hide`。
 - ④ 新增定理模式 `mode`, 可选项有简单模式 `simple` 和炫彩模式 `fancy`。
 - ⑤ 新增隐藏证明、答案等环境的选项 `result=noanswer`。
-

2019/02/25 更新: 版本 3.06 正式发布

- ① 删除水印。
 - ② 新封面, 新装饰图案。
 - ③ 添加引言使用说明。
 - ④ 修复双面 `twoside`。
 - ⑤ 美化列表环境。
 - ⑥ 增加 `\subsubsection` 的设置。
 - ⑦ 将模板拆分成中英文语言模式。
 - ⑧ 使用 `lstlisting` 添加代码高亮。
 - ⑨ 增加定理类环境使用说明。
-

2019/01/22 更新: 版本 3.05 正式发布

- ① 添加 `xeCJK` 宏包中文支持方案。
 - ② 修复模板之前对 `TikZ` 单位的改动。
 - ③ 更新 logo 图。
-

2019/01/15 更新: 版本 3.04 正式发布

- ① 格式化模板代码。
 - ② 增加 `\equation` 命令。
 - ③ 修改 `\date`。
-

2019/01/08 更新: 版本 3.03 正式发布

- ① 修复附录章节显示问题。
 - ② 小幅优化封面代码。
-

2018/12/31 更新: 版本 3.02 正式发布

- ① 修复名字系列命令自定义格式时出现的空格问题, 比如 `\listfigurename`。
- ② 英文定理类名字改为中文名。
- ③ 英文结构名改为中文。

2018/12/16 更新：版本 3.01 正式发布

- ① 调整 `ctex` 宏包。
 - ② 说明文档增加更新内容。
-

2018/12/06 更新：版本 3.00 正式发布

- ① 删除 `mathpazo` 数学字体选项。
- ② 添加邮箱命令 `\mailto`。
- ③ 修改英文字体为 `newtx` 系列，另外大型操作符号维持 `cm` 字体。
- ④ 中文字体改用 `ctex` 宏包自动设置。
- ⑤ 删除 `xeCJK` 字体设置，原因是不同系统字体不方便统一。
- ⑥ 定理换用 `tcolorbox` 宏包定义，并基本维持原有的定理样式，优化显示效果，支持跨页；定理类名字重命名，如 `etheorem` 改为 `theorem` 等等。
- ⑦ 删去自定义的缩进命令 `\Eindent`。
- ⑧ 添加参考文献宏包 `natbib`。
- ⑨ 颜色名字重命名。

附录 A 基本数学工具

本附录包括了计量经济学中用到的一些基本数学，我们扼要论述了求和算子的各种性质，研究了线性和某些非线性方程的性质，并复习了比例和百分数。我们还介绍了一些在应用计量经济学中常见的特殊函数，包括二次函数和自然对数，前 4 节只要求基本的代数技巧，第 5 节则对微分学进行了简要回顾；虽然要理解本书的大部分内容，微积分并非必需，但在一些章末附录和第 3 篇某些高深专题中，我们还是用到了微积分。

A.1 求和算子与描述统计量

求和算子是用以表达多个数求和运算的一个缩略符号，它在统计学和计量经济学分析中扮演着重要作用。如果 $\{x_i : i = 1, 2, \dots, n\}$ 表示 n 个数的一个序列，那么我们就把这 n 个数的和写为：

$$\sum_{i=1}^n x_i \equiv x_1 + x_2 + \dots + x_n \quad (\text{A.1})$$