

分类号: TN92

单位代码: 10335

密 级: 无

学 号: 22031098

# 浙江大学

## 硕士学位论文



中文论文题目: 基于惩罚对偶分解法的  
LDPC码译码技术研究

英文论文题目: Research on Decoder for LDPC Codes  
via Penalty Dual Decomposition Method

申请人姓名: 刘屹豪

指导教师: 赵民建

专业名称: 信息与通信工程

研究方向: 无线通信

所在学院: 信息与电子工程学院

论文提交日期 2023年3月23日



基于惩罚对偶分解法的  
LDPC码译码技术研究



论文作者签名: 刘屹豪

指导教师签名: 赵民建

论文评阅人1: 匿名

评阅人2: 匿名

评阅人3: 匿名

答辩委员会主席: 张宏纲 教授 之江实验室

委员1: 史治国 教授 浙江大学

委员2: 李旻 研究员 浙江大学

委员3: 赵明敏 副教授 浙江大学

委员4: 雷鸣 副研究员 浙江大学

委员5: \_\_\_\_\_

答辩日期: 2023年3月6日



## 浙江大学研究生学位论文独创性声明

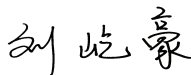

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：  签字日期： 2023 年 3 月 4 日

## 学位论文版权使用授权书

本学位论文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权浙江大学可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密的学位论文在解密后适用本授权书)

学位论文作者签名：  导师签名：   
签字日期： 2023 年 3 月 4 日 签字日期： 2023 年 3 月 4 日



## 致 谢

时光荏苒，流光过隙，两年半的硕士生涯已经走到了尾声，转眼间我已在浙江大学度过了六年半的时光。这是我人生中最美好的时光也是最宝贵的经历，美丽的校园，高质量的实验室平台，博学多识的老师，以及热心善良的同学都是一路支撑我前进的强大动力。回首在浙大求学的这些年，我满怀感激之情。

首先，我要感谢我的导师赵民建教授。初次见面时，我就被赵老师平易近人的性格和优雅的言谈举止所打动。赵老师对于整个通信领域有着非常深刻的见解，为我的工程项目和科研理论学习指明了方向和道路。当我在科研项目上遇到困难时，赵老师总能凭借其深厚的功底为我答疑解惑，引导我找到正确的方法，当我在学习和生活中感到迷茫时，赵老师也能给予我很多宝贵的建议，并耐心地开导我。

同时，我也要感谢赵明敏副教授，赵老师年轻有为，有着非常深厚的科研理论功底，对于学生非常负责友善，非常感谢赵老师在科研学术方面给我的启发，以及耐心地修改指导我的论文。感谢雷鸣老师在硕士期间对我的悉心指导，在工程实践，论文撰写等方面给予了很多宝贵的建议。感谢李立言老师在工程项目上给予的帮助和指导，李老师有着丰富的工程经验和深厚的功底，做事一丝不苟，指导我解决了很多工程上的难题，帮助我积累了不少的工程项目经验。感谢邵红霞老师，王婵老师，李旻老师，杨丽萍老师在学业和生活方面提供的关心和支持，使我的硕士生活更加丰富多彩。

感谢实验室编码组已毕业的丘耿鑫硕士和窦为龙硕士，感谢你们带我走入信道编码的大门，以及在学业生活上对我的指导和帮助，感谢同窗的陆昶硕士，以及余皓晨，赵子伟，邹文卓，蒋宇龙等师弟师妹，大家长期以来的学习交流让我获益匪浅。感谢实验室已毕业的韦逸博士，程谦儒硕士，黄钰鹏硕士，李绍标硕士，感谢还在攻读博士学位的胡棋昱博士，陈梓健博士，感谢你们在科研和项目方面给予的帮助和引导。感谢同窗的胡智祥博士，邵雨航硕士，邱伏韬硕士，罗群平硕士，段伟骏硕士，倪甫田硕士，感谢魏涵宇，丁悦晋，陈丹妮等师弟师妹，感谢你们让我体会到团队的力量，你们的陪伴和鼓励让我更好地进步。感谢本科室友龚来运，崔轲等人，感谢我的好友黄啸虎，陈畅，于炜鉴，高晨洵，方鹏，朱子涵等人，感谢你们在生活和学习上给予的关系和照顾，愿我们的友谊可以更长远。

感谢我的父母和家人数十年如一日对我的精心照顾和陪伴，你们在学习生活上给予的关怀和鼓励是我们前进的最大动力。感谢我的女朋友，在你的陪伴中我迈过了一道道的坎

坷，让我用更积极的态度去面对未来，为我的硕士生涯增添了最美妙的一抹色彩。

最后，向所有支持我陪伴我的人致以最诚挚的谢意，愿大家都有美好的光明的未来。

刘屹豪

2022年12月于求是园



## 摘 要

低密度奇偶校验 (Low Density Parity Check, LDPC) 码是一种非常逼近香农极限的纠错码, 在包括5G在内的各种通信领域都获得了广泛的应用。除了经典的置信传播 (Belief Propagation, BP) 算法, 基于最大似然 (Maximum Likelihood, ML) 问题的线性规划 (Linear Programming, LP) 译码算法凭借其较强的理论保证和优良的纠错性能得到了研究者的广泛关注, 如何降低其较高的复杂度以及进一步提升译码性能成为了近年来的研究热点。惩罚对偶分解 (Penalty Dual Decomposition, PDD) 算法是一种适用于求解大规模分布式优化问题的算法, 本文主要基于PDD算法框架对LDPC码的LP译码问题进行求解, 同时引入深度学习中的深度展开技术来提升译码性能, 并进一步地将LP译码的思想推广到码间干扰 (Inter-Symbol Interference, ISI) 信道下的LDPC码译码算法设计。

首先, 本文介绍了LDPC码的基本原理和经典的BP译码技术, 讲述了ISI信道下的Turbo均衡译码方案的基本原理, 对PDD算法框架进行了描述, 以及对深度学习技术进行了归纳整理, 并且对深度学习中基于模型驱动的深度展开技术进行了描述。

其次, 本文基于LDPC码的LP译码问题提出了一种双层迭代的PDD译码算法。具体地, 本文首先给出了ML译码问题的级联整数规划问题的描述, 并采用LP松弛技术处理离散约束, 引入罚函数加大对伪码字的惩罚。然后提出了基于级联整数规划问题的PDD译码算法, 同时采用超松弛技术来加快算法的收敛速度。进一步地, 本文通过深度展开技术将所提的PDD译码算法展开为一个模型驱动的神经网络, 即可学习PDD译码网络 (Learnable PDD Decoding Network, LPDN), 将所提PDD译码算法中可调整的参数转换成层间独立的可训练参数, 在网络训练阶段通过梯度下降的方法进行网络优化, 避免了手动调节译码参数并且进一步提升了译码性能。仿真结果表明所提的LPDN译码器相比PDD译码器可以提供更优的纠错性能和更低的计算复杂度。

最后, 将LP译码方法的思想推广到ISI信道, 本文进一步提出了ISI信道下的LDPC码的PDD译码算法。具体地, 本文首先给出了ISI信道下的LDPC码ML译码问题的二次规划 (Quadratic Programming, QP) 描述, 同样采用LP松弛技术和罚函数来简化约束条件。然后基于PDD算法框架提出了一种基于校验多面体约束的ISI-PDD译码算法。针对译码算法中的校验多面体投影步骤, 本文提出了一种基于迭代思想的快速校验多面体投影 (Fast Check Polytope Projection, FCPP) 算法, 其中引入了可动态调节的缩放因子来加快算法的收敛速度。仿真结果表明, FCPP在不损失译码性能的前提下可以大幅减少投影算法所需

的迭代次数，并且ISI信道下的PDD译码算法相比经典的Turbo均衡译码方案拥有更佳的纠错性能，并且其计算复杂度随着ISI信道记忆长度的增加呈线性增长。

**关键词：**低密度奇偶校验码，惩罚对偶分解，深度展开，码间干扰信道，校验多面体投影

## Abstract

Low density parity check (LDPC) codes have been widely used in various communication tasks due to their excellent performance approaching to Shannon capacity limits. Besides the classical belief propagation (BP) algorithm, the linear programming (LP) decoding algorithm based on maximum likelihood (ML) has drawn significant attention from worldwide researchers due to its strong theoretical guarantee and excellent performance. How to reduce the high complexity of LP decoding and further improve the decoding performance has become a research hotspot in recent years. The penalty dual decomposition (PDD) algorithm is suitable for solving large-scale distributed optimization problems. In this thesis, we solve the LP decoding problem of LDPC codes based on the PDD framework, and employ the deep unfolding technology to improve the performance of the decoder. Besides, the idea of LP decoding is extended to the LDPC decoding problem under the inter-symbol interference (ISI) channel.

Firstly, we introduce the theoretical basis of LDPC codes and classical BP decoding algorithms, and then describe the basic principles of Turbo equalization based on the ISI channel. Besides, we introduce the PDD framework, and summarize the existing deep learning technologies. In particular, we introduce the deep unfolding technique based on model-driven deep learning.

Secondly, we develop a double-loop iterative PDD decoding algorithm based on LP decoding of LDPC codes. Specifically, we first introduce the cascaded integer programming problem of the ML decoding, and utilize the LP relaxation to handle the discrete constraints and the penalty method to avoid pseudocodewords. Then, we propose the PDD decoding algorithm based on the cascaded integer programming problem and employ the over-relaxation method to improve convergence. Besides, to avoid manually finetuning the decoding parameters and to further improve the decoding performance, we unfold the proposed PDD decoding algorithm into a model-driven neural network, namely the learnable PDD decoding network (LPDN). We turn the tunable coefficients and parameters in the proposed PDD decoder into layer-dependent trainable parameters which can be optimized by gradient descent-based methods during network training. Simulation results demonstrate that the proposed LPDN with well-trained parameters is able to provide superior error-correction performance with much lower computational complexity as compared to the PDD decoder.

Finally, we extend the idea of LP decoding to ISI channel, and propose a PDD decoding algorithm for LDPC codes over ISI channel. Specifically, we first formulate the ML decoding for the LDPC codes over ISI channel as a Quadratic Programming (QP) problem, and utilize the LP relaxation and the penalty method to simplify the constraints. Then, we propose the ISI-PDD decoding algorithm based on check polytope constraints. Besides, to reduce the projection step of check polytope in the decoding algorithm, we propose an iterative fast check polytope projection (FCPP) algorithm, in which an adjustable scaling factor is introduced to improve convergence. Simulation results demonstrate that FCPP can reduce the iteration number of the projection algorithm without degrading the decoding performance. Besides, we show that the proposed PDD decoding algorithm based on ISI channel has better error-correction performance than the classical Turbo equalization decoding, and its computational complexity increases linearly with the memory length of ISI channel.

**Keywords:** Low density parity check codes, penalty dual decomposition, deep unfolding, inter-symbol interference channel, check polytope projection

## 插 图

2.1	系统码示意图.....	10
2.2	(7,4)线性分组码的Tanner图表示.....	10
2.3	AWGN信道传输模型.....	13
2.4	基于Turbo均衡的ISI信道传输模型.....	17
2.5	神经元模型.....	20
2.6	深度展开模型结构.....	23
3.1	度为6的校验节点分解过程.....	27
3.2	LPDN的网络架构（黄色圆圈表示训练参数，蓝色和橙色矩形分别表示内层和外层更新节点，红色实线框中描述了节点的具体结构）.....	34
3.3	$C_1$ 的FER和BER曲线.....	38
3.4	$C_2$ 的FER和BER曲线.....	39
3.5	$C_1$ 的平均迭代次数和译码运行时间曲线.....	40
3.6	$C_2$ 的平均迭代次数和译码运行时间曲线.....	41
4.1	ISI信道传输模型.....	44
4.2	校验多面体几何结构( $d_j = 5$ ).....	50
4.3	校验多面体投影说明.....	52
4.4	不同缩放因子的FCPP算法的收敛过程.....	57
4.5	LDPC码 $C_1$ 在EPR4和PR4信道下的FER曲线.....	59
4.6	LDPC码 $C_2$ 在EPR4和PR4信道下的FER曲线.....	60



## 表 格

3.1	单次迭代计算复杂度比较.....	34
4.1	校验多面体算法复杂度比较.....	54
4.2	ISI信道下的LDPC码译码算法复杂度比较.....	55
4.3	FCPP算法迭代次数对比 .....	57
4.4	针对码字 $C_1$ 的译码器参数设置 .....	58
4.5	针对码字 $C_2$ 的译码器参数设置 .....	60





## 缩写词列表

<b>3G</b>	The 3th Generation Mobile Communication System
<b>4G</b>	The 4th Generation Mobile Communication System
<b>5G</b>	The 5th Generation Mobile Communication System
<b>ACE</b>	Approximate Cycle Extrinsic message degree
<b>ADMM</b>	Alternating Direction Method of Multipliers
<b>AL</b>	Augmented Lagrangian
<b>AWGN</b>	Additive White Gaussian Noise
<b>BCD</b>	Block Coordinate Descent
<b>BF</b>	Bit Flipping
<b>BG</b>	Base Graph
<b>BP</b>	Belief Propagation
<b>BPSK</b>	Binary Phase Shift Keying
<b>CNN</b>	Convolutional Neural Network
<b>CPP</b>	Check Polytope Projection
<b>CTV</b>	Check-To-Variable
<b>DNN</b>	Deep Neural Network
<b>eMBB</b>	Enhanced Mobile Broadband
<b>EMD</b>	Extrinsic Message Degree
<b>FER</b>	Frame-Error-Rate
<b>FPGA</b>	Field Programmable Gate Array
<b>ICPP</b>	Iterative Check Polytope Projection
<b>ISI</b>	Inter-Symbol Interference
<b>LDPC</b>	Low Density Parity Check
<b>LLR</b>	Log-Likelihood Ratio
<b>LP</b>	Linear Programming
<b>LPDN</b>	Learnable PDD Decoding Network
<b>ML</b>	Maximum Likelihood
<b>mMTC</b>	Massive Machine Type Communication

---

<b>MSA</b>	Min-Sum Algorithm
<b>MSE</b>	Mean Squared Error
<b>NP-Hard</b>	Non-deterministic Polynomial-time Hard
<b>PDD</b>	Penalty Dual Decomposition
<b>PEG</b>	Progressive Edge Growth
<b>QC</b>	Quasi-Cyclic
<b>QP</b>	Quadratic Programming
<b>SGD</b>	Stochastic Gradient Descent
<b>SNND</b>	Sparse Neural Network Decoder
<b>SNR</b>	Signal-to-Noise Ratio
<b>SOVA</b>	Soft-Output Viterbi Algorithm
<b>SPA</b>	Sum-Product Algorithm
<b>uRLLC</b>	Ultra Reliable and Low Latency Communication
<b>VTC</b>	Variable-To-Check
<b>WBF</b>	Weighted Bit Flipping

# 目 次

致谢 .....	I
摘要 .....	III
Abstract .....	V
插图 .....	VII
表格 .....	IX
缩写词列表 .....	XI
目次 .....	
第一章 绪论 .....	1
1.1 研究背景与意义 .....	1
1.2 研究现状 .....	3
1.2.1 基于硬判决的LDPC码译码技术相关研究 .....	3
1.2.2 基于软判决的LDPC码译码技术相关研究 .....	4
1.2.3 ISI信道下的LDPC码译码技术相关研究 .....	5
1.2.4 基于深度学习的译码技术相关研究 .....	6
1.3 论文的主要内容与结构安排 .....	6
1.3.1 研究内容与贡献 .....	6
1.3.2 论文结构安排 .....	7
第二章 相关理论基础 .....	9
2.1 引言 .....	9
2.2 LDPC码的预备知识 .....	9
2.2.1 LDPC码的基本概念 .....	9
2.2.2 LDPC码的Tanner图表示 .....	10
2.2.3 LDPC码的构造 .....	11
2.2.4 LDPC码的编码 .....	11
2.2.5 LDPC码的BP译码技术 .....	12
2.3 Turbo均衡基本原理 .....	16
2.4 PDD算法基本原理 .....	17
2.5 深度学习技术简介 .....	20

2.5.1	神经网络的基本结构 .....	20
2.5.2	神经网络的训练方法 .....	21
2.5.3	深度展开的基本原理 .....	22
2.6	本章小结 .....	24
第三章	基于LP译码问题的LDPC码PDD译码技术 .....	25
3.1	引言 .....	25
3.2	优化问题描述 .....	25
3.2.1	ML译码问题 .....	25
3.2.2	LP问题的级联整数规划形式 .....	26
3.3	基于LP问题的PDD译码算法设计 .....	29
3.3.1	PDD译码算法 .....	29
3.3.2	复杂度分析 .....	32
3.4	基于深度学习的LPDN译码器 .....	33
3.5	仿真结果与性能分析 .....	37
3.6	本章小结 .....	40
第四章	ISI信道下的LDPC码的PDD译码技术 .....	43
4.1	引言 .....	43
4.2	优化问题描述 .....	43
4.3	基于QP问题的ISI-PDD译码算法设计 .....	45
4.4	FCPP算法设计 .....	48
4.4.1	校验多面体的几何结构 .....	49
4.4.2	FCPP算法 .....	50
4.5	算法复杂度分析 .....	53
4.6	仿真结果与性能分析 .....	56
4.6.1	FCPP算法仿真 .....	56
4.6.2	ISI-PDD译码算法仿真 .....	58
4.7	本章小结 .....	61
第五章	总结与展望 .....	63
5.1	本文工作总结 .....	63
5.2	下一步研究方向 .....	64
参考文献	.....	65
攻读硕士学位期间的研究成果	.....	69

# 第一章 绪论

## 1.1 研究背景与意义

从古代的飞鸽传书和烽火台到当代的互联网和无线通信技术，如何将信息高速可靠地从发送方传递给接收方一直是一个值得探索和研究的问题。现如今，得益于诸多通信研究者对于通信技术的深入研究和工程创新，无线通信技术实现了高速发展和广泛应用，并潜移默化地改变了人们的生活方式，成为人们日常生活工作中不可或缺的重要部分。经过几代通信系统的更迭，第五代移动通信技术（5G）已经在全球范围内进行部署。相比前几代通信系统，5G要实现高数据速率，高可靠，低延迟的数据传输和大规模设备连接，并且定义了三大应用场景：面向个人终端海量数据传输的增强移动宽带业务（Enhanced Mobile Broadband, eMBB），面向工业控制的高可靠低延迟通信业务（Ultra Reliable and Low Latency Communication, uRLLC）以及面向智能家居和智慧城市的规模物联网业务（Massive Machine Type Communication, mMTC）。

信道编码也称做纠错码，是通信系统中至关重要的一环，通过规则地增加冗余信息来确保信息的可靠性传输。与第四代移动通信技术（4G）相比，5G采用了新的信道编码方案并分别应用于数据信道和控制信道，以此来应对更高要求的通信场景。所以，在今后的通信系统中，如何进一步降低译码时延并提升纠错码的纠错性能来增强系统的可靠性是一个值得探索的问题。

1948年，Shannon发表了影响深远的开创性著作《通信的数学理论》<sup>[1]</sup>，首次提出了著名的信道编码定理，象征着信息论的问世。Shannon的信道编码定理指出，只要信息传输速率小于信道容量，则信息传输可以以任意小的错误概率进行，从而实现信息在有噪信道中的可靠性传输。而在之后的1950年，Hamming发表的著作《纠错和检测编码》标志着信道编码理论的创立<sup>[2]</sup>，提出了第一个实用的信道编码方案，即汉明码（Hamming Code），该码是一类重要的线性分组码，可以纠正全部一位错误。虽然汉明码的编码和译码都很简单，但是纠错性能距离香农限还有较大差距，所以在之后的若干年里涌现出许多更加先进的信道编码技术。

- 循环码：循环码的码字具有循环特性，其结构对称性使得编码和译码的复杂度低于其他线性码，其中最重要的两类循环码是BCH（Bose-Chuadhuri-Hocquen）码<sup>[3]</sup>和RS（Reed-Solomon）码<sup>[4]</sup>。BCH码可以纠正多个随机错误，编译码器结构简单，但是在

中长码情况下性能较差，吞吐率也不够理想。**RS**码是一类纠错性能较强的非二进制**BCH**码，在各类通信系统中获得了广泛的应用。

- **卷积码**：卷积码是不同于分组码的另一类编码<sup>[5]</sup>，其不同之处在于卷积码的编码器是有记忆的。由于卷积码的码元之间具有相关性，冗余也更高，所以纠错性能也更强。1967年，Viterbi提出了Viterbi算法<sup>[6]</sup>用于卷积码的译码，该算法被证明为卷积码的最大似然译码算法，使得卷积码广泛应用于各类通信场景中，比如第三代移动通信技术（3G）。
- **Turbo码**：Turbo码由若干个子码并行级联，并在子码间插入伪随机交织器得到，一般用卷积码作为子码<sup>[7]</sup>。Turbo码凭借其接近于香农极限的优良性能成为了一大研究热点，并被采纳为3G和4G中的信道编码方案。
- **极化码（Polar Code）**：2008年，Arikan教授基于信道极化的思想提出了Polar码<sup>[8]</sup>，该码是首个被证明可以达到香农极限的信道编码方案。近年来，人们对Polar码开展了大量的理论研究和工程实践工作，使得其最终被采纳为5G的eMBB场景中控制信道编码方案。
- **低密度奇偶校验（Low Density Parity Check, LDPC）码**：LDPC码最早在1962年由Gallager提出<sup>[9]</sup>，但在当时由于实现复杂度太高并未受到重视。直到20世纪90年代中期，受到Turbo码的迭代译码算法的启发，LDPC码被Mackay等人重新发现<sup>[10]</sup>。LDPC码凭借其线性复杂度的编译码算法和非常逼近香农极限的优良性质，被广泛应用于磁记录系统，深空通信等领域中，并被采纳为5G的eMBB场景中数据信道编码方案。

LDPC码的译码算法主要可以分成硬判决译码和软判决译码两大类。硬判决译码算法具有较低的实现复杂度，但由于没有充分考虑软信息，致使纠错性能不够理想，所以工程上应用更加广泛的还是软判决译码算法。经典的软判决译码算法是基于Tanner图信息传递的置信传播（Belief Propagation, BP）算法<sup>[11;12]</sup>，该算法充分利用了信道的软信息，将其用于迭代更新，译码性能可以逼近最大似然译码。然而由于短环和陷阱集（Trapping Sets）的存在，BP译码并不能确保收敛，并且在信噪比较高时可能会出现错误平层（Error Floor），不利于一些对可靠性要求极高的应用场景。

近年来，基于最大似然（Maximum Likelihood, ML）问题的线性规划（Linear Programming, LP）译码算法受到了研究者的广泛关注<sup>[13]</sup>。相比经典的BP译码算法，LP译码算法具有最大似然特性（ML Certificate），即译码器输出的整数解一定是最大似然解。此

外LP译码算法还提供了很强的理论保证，在高信噪比区域有更优的译码性能。但是LP译码算法在低信噪比区域的性能不佳，并且相比BP译码算法具有更高的译码复杂度。因此，如何进一步提升LP译码算法的译码性能，并且降低其译码复杂度成为了近年的研究热点。同时，一些研究将这类用优化方法来译码的思想推广到码间干扰（Inter-Symbol Interference, ISI）信道下的LDPC译码算法设计中，这些基于优化方法设计的算法相比经典的Turbo均衡技术有更低的计算复杂度和更优的纠错性能，为ISI信道下的LDPC码译码算法设计提供了新的思路。

目前，深度学习（Deep Learning）技术在计算机视觉和自然语言处理等领域大放异彩，同时在通信领域也受到了广泛关注。深度学习凭借其强大的特征提取能力、灵活的模块结构和数据处理等优势，通常被用于通信物理层中处理一些复杂的信号处理任务，比如信道估计和信道编码等<sup>[14;15]</sup>。应用于信道编码领域的神经网络结构主要可以分为两类：数据驱动（Data-Driven）和模型驱动（Model-Driven）。数据驱动的神经网络通常可以看作一个黑盒，训练过程中需要消耗大量的计算资源和时间，而且结构缺乏可解释性。模型驱动的深度学习方法将通信中的领域知识（Domain Knowledge）和深度学习相结合，代表性的有深度展开（Deep Unfolding）技术，这类方法不需要依赖大量标记数据来选择合适标准神经网络，从而可以大大减少对计算资源和训练时间的需求，并且其可以基于原算法寻找最优的参数分布来减少算法的迭代次数，有利于实现高吞吐率低延迟的设计目标，在纠错码的译码器设计中有着广阔的应用前景。

## 1.2 研究现状

LDPC码的译码算法主要可以分成硬判决译码和软判决译码两大类，而对于ISI信道下的LDPC码译码，除了经典的Turbo均衡方案外，近年来也有许多研究者提出基于优化理论来进行译码的方案。同时，近年来热门的深度学习技术也给通信物理层中的算法设计提供了新的思路，涌现出许多基于深度学习的译码技术相关研究。

### 1.2.1 基于硬判决的LDPC码译码技术相关研究

Gallager提出LDPC码的同时给出了一种基于硬判决的比特翻转（Bit Flipping, BF）译码算法，该算法的核心思想是通过特征校验（Syndrome Check）的结果来判断需要翻转的比特，实现比较简单，涉及的计算量也很小，但是译码性能并不理想。文献[16]提出了加权比特翻转（Weighted Bit Flipping, WBF）译码算法，当一个比特的不满足特征校验式的个数超过一个设定的阈值时，按照适当的概率对其进行翻转，该算法改善了译码性能

并减少了所需的迭代次数。文献[17]设计了基于多比特翻转的并行加权比特翻转（Parallel Weighted Bit Flipping, PWBF）译码算法，相比串行的比特翻转算法具有更高的收敛速度，并且几乎没有性能损失。

### 1.2.2 基于软判决的LDPC码译码技术相关研究

不同于硬判决译码算法，软判决译码算法在迭代过程中充分运用了信道的软信息，所以可以收获更多的性能增益。Mackay等人在重新发现LDPC码的时候提出了基于概率域的BP译码算法<sup>[10]</sup>，该算法的核心思想是译码节点结合自身信息和相关节点的信息来完成信息更新，并利用Tanner图的边来完成信息传递，当满足终止条件时停止迭代。由于概率域上BP算法表述不够直观，且乘法操作较为频繁，文献[18]将BP译码算法推广到对数域，提出了更为通用的和积译码算法（Sum-Product Algorithm, SPA），该算法将概率域上的乘法运算转换为对数域上的加法运算，进而降低了计算复杂度，今后的软判决译码算法研究大多也是基于对数域的表达方式。文献[19]用简单的比较和加法操作取代了BP译码算法中的双曲正切操作，提出了最小和算法（Min-Sum Algorithm, MSA），该算法大大降低了BP算法的复杂度，很好地做到了计算复杂度和译码性能的折中，有利于译码算法在硬件上的高效实现，为LDPC码的大规模工程部署奠定了基础。考虑到MSA中对双曲正切运算的近似操作引入的性能损失，文献[20]提出了归一化MSA和基于偏移项的MSA，两种算法分别通过引入归一化因子和偏移项来弥补近似操作带来的性能损失，相比普通的MSA均能获得更优的译码性能。

不同于传统的BP算法，Feldman提出的LP译码算法的核心思想是对ML译码问题进行LP松弛，然后引入优化理论的方法对松弛后的优化问题进行求解，为LDPC码的软判决译码算法提供了新的方向。虽然LP算法相比BP算法具有理论保证性更强，高信噪比区域误码率更低等优势，但是其较高的计算复杂度还是成为了工程应用上的一道阻碍。为了弥补LP算法存在的缺陷，众多研究人员对LP算法开展了深入研究。文献[21]提出了一种基于级联分解整数规划的LP译码算法，其通过引入辅助变量的方式将高度数的校验节点分解为若干低度数的校验节点，以此来简化原问题的校验约束。该算法的复杂度与校验约束呈线性关系，降低了原始LP算法的计算复杂度。

为了进一步降低LP译码算法的复杂度，文献[22]将交替方向乘子法（Alternating Direction Method of Multipliers, ADMM）应用于LP译码问题的求解，提出了一种基于ADMM的分布式LP译码算法，该算法充分利用了LP问题约束条件的几何结构，可以基于节点间信息传递进行迭代求解，有效提升了LP算法的译码性能并降低了计算复杂度。考虑



到ADMM译码算法中的校验多面体投影（Check Polytope Projection, CPP）操作复杂度较高，文献[23]提出了一种基于“割”搜索（Cut Search）的CPP算法，该算法只需要进行一次排序操作，有效地降低了CPP算法的复杂度。文献[24]进一步设计了无需排序操作的迭代校验多面体投影（Iterative Check Polytope Projection, ICPP）算法，该算法具有线性复杂度，显著地提高了译码效率。考虑到LP译码算法在低信噪比区域的性能不如BP译码算法，文献[25]通过在LP译码问题的目标函数中引入惩罚项的方式提出了ADMM罚函数算法，该算法中引入的罚函数使伪码字（Pseudocodewords）的代价更高，使译码结果更倾向于整数解，从而使LP译码算法的译码性能在不同信噪比下均优于BP译码算法。文献[26]将LP问题中的离散约束条件转换为一个有界约束和 $l_p$ 球面的交集，提出了无需罚函数的 $l_p$ -box ADMM译码算法，虽然相比罚函数的译码算法有更优的译码性能，但是 $l_p$ 球面的投影操作会引入额外的计算复杂度。文献[27]提出了一种基于级联分解整数规划的ADMM罚函数算法，避免了复杂的CPP操作，提高了译码效率。文献[28]提出了一种基于级联分解整数规划的双层循环迭代译码算法，该算法采用惩罚对偶分解（Penalty Dual Decomposition, PDD）框架对LP问题进行求解，可以确保生成的每个极限点都是ML问题的驻点，仿真结果显示该PDD译码算法在高信噪比和低信噪比下都有很好的译码性能。

### 1.2.3 ISI信道下的LDPC码译码技术相关研究

码间干扰（Inter-Symbol Interference, ISI）信道是通信系统和磁记录系统中的一种实用信道模型。在高数据速率的通信系统和高密度数据的存储系统中通常会存在码间干扰，一般采用均衡器和纠错码来联合抑制码间干扰。Turbo均衡是一种可以有效抑制ISI信道中码间干扰的技术<sup>[29]</sup>，其中信道均衡器和译码器像涡轮一样互相交换软信息进行迭代更新，其中均衡算法通常会采用软输出的BCJR算法<sup>[30]</sup>或者软输出的维特比（Soft-Output Viterbi Algorithm, SOVA）算法<sup>[31]</sup>，然而这两种均衡方法的计算复杂度均与信道记忆长度呈指数关系，当信道记忆长度较长时会引入较高的延迟。文献[32]将LP译码的思想推广到ISI信道中，将ISI信道中LDPC码的ML问题松弛为一个二次规划（Quadratic Programming, QP）问题，然后用内点法进行求解，相比传统的联合消息传递译码器具有更好的译码性能和更低的复杂度。文献[33]提出了ISI信道下联合信道检测问题的图形式，将QP问题线性化为等效的LP问题。文献[34]用基于LP的通信接收机统一框架来求解ISI信道中的LDPC码译码问题，并举例说明了在ISI信道下如何用LP接收机实现联合均衡和译码，证明了LP接收机的最大似然特性，同时解释了伪码字和伪距离（Pseudodistance）的通用概念。文献[35]提出了一种基于对偶理论的联合迭代LP译码算法，结构类似于Turbo均衡，复杂度较高。文献[36]将ADMM算法推广到ISI信道中实现了基于LDPC码的联合均衡和译码，提出了ISI信

道下的ADMM罚函数译码算法，该算法的计算复杂度与信道记忆长度呈线性关系，整体性能优于Turbo均衡的方案。文献[37]用 $l_2$ -box模型将QP问题的离散约束条件转换成一个有界连续的约束和一个 $l_2$ 球面的交集，提出了 $l_2$ -box ADMM译码算法，进一步提升了ISI信道下的ADMM译码算法的性能。

#### 1.2.4 基于深度学习的译码技术相关研究

数据驱动神经网络和模型驱动神经网络是两种适用于信道编码领域的最具代表性的神经网络结构。数据驱动神经网络的核心思想是把要学习的系统看作一个黑盒，通过深度神经网络（Deep Neural Network, DNN）和卷积神经网络（Convolutional Neural Network, CNN）等常用的神经网络结构来学习输入和输出之间的非线性映射关系。文献[38]将DNN应用于线性码的译码，训练得到的译码器可以接近ML性能，但是仅限于非常短的码长，同时他们发现结构化的码字相比随机码更容易学习。基于数据驱动的神经网络在训练时需要大量的训练样本，这会消耗大量的计算资源和时间，同时数据驱动神经网络的泛化性能并不理想，而且缺乏可解释性，所以其性能存在一定随机性。为了克服这些缺陷，更多关于信道编码的工作是基于模型驱动的神经网络。深度展开是一种非常重要的模型驱动技术<sup>[39-41]</sup>，其核心思想是利用现有的领域知识，将现有迭代算法展开成一个具有固定层数的类神经网络。基于深度展开的思想，文献[42]提出了一种BP-RNN译码器，他们给Tanner图的边赋上权重，并将权重视为可训练的参数，然后将BP算法展开为一个类似RNN的神经网络，训练后得到的BP-RNN译码器对于码长较短的BCH码可以获得接近最佳译码器的性能，同时降低了译码复杂度。文献[43]将Polar码的BP译码因子图转换成类似LDPC码的Tanner图，然后将Tanner图展开成为DNN的图形表示，得到了一种用于Polar码的稀疏神经网络译码器（Sparse Neural Network Decoder, SNND），在Polar码的短码下相比BP译码器获得了约0.5dB的增益，并且降低了60%的复杂度。文献[44]提出了一种基于ADMM的神经网络译码器，相比传统的ADMM译码器拥有更优的纠错性能和更低的计算复杂度。

### 1.3 论文的主要内容与结构安排

#### 1.3.1 研究内容与贡献

自从LDPC码被重新发现以来，经典的BP译码算法及其近似简化算法已经得到了充分的研究和工程实践，但是对于LP译码算法的研究还是相对匮乏。由于LP译码算法相

比BP算法存在一定的理论和性能优势，并且在译码性能和计算复杂度上都还有一定的提升空间，所以研究LDPC码的LP译码算法具有一定的实际价值。本文研究内容主要基于PDD算法框架，围绕LDPC码的LP译码算法设计，以及ISI信道下LDPC码的译码算法设计展开。

首先，针对LDPC码的译码，我们提出了一种基于级联分解整数规划问题和PDD算法框架的双层循环的译码算法，采用了LP松弛和引入惩罚项的方法来处理离散校验约束，同时使用超松弛（Over-relaxation）的方法来提升收敛速度。为了进一步提升译码性能，我们将所提的PDD译码算法展开成为一个模型驱动的神经网络，将原算法中可调整的系数和参数转换成逐层独立的训练参数，然后采用基于梯度下降的方法对网络进行训练，得到优化的神经网络译码器。

其次，我们将LP译码的思想推广到ISI信道下的LDPC码译码问题的求解，将ISI信道下LDPC码译码问题表示成一个QP问题，同样通过LP松弛和罚函数的方法处理校验约束，得到了ISI信道下基于校验多面体约束的PDD译码算法。针对其中基于校验多面体的校验约束条件，本文提出了一种基于迭代思想的快速校验多面体投影算法，引入可动态调节的缩放因子来加快算法收敛速度。

### 1.3.2 论文结构安排

本论文总共分为五章，每个章节的具体内容如下：

第一章为绪论，首先阐述了通信系统中信道编码的研究背景和意义，介绍了信道编码的起源和各类代表性纠错码的发展历程。然后分别介绍了基于硬判决和软判决的两大类LDPC码译码算法，以及ISI信道下的LDPC码译码技术的研究现状，并且对基于深度学习的译码技术相关的研究进行了阐述。最后对本文的主要研究内容和结构安排进行了说明。

第二章主要介绍了LDPC码的基本概念，Tanner图表示，常用的构造编码方式以及经典的BP译码算法原理。其次介绍了针对ISI信道的Turbo均衡技术的基本原理。然后对PDD算法的基本原理和框架构成，以及相比ADMM框架的一些优势进行了阐述。最后介绍了深度学习的基本原理，并着重阐述了基于模型驱动的深度展开技术的框架和应用方法。

第三章主要介绍了基于深度学习的PDD译码算法。首先给出了基于级联分解整数规划的LP问题描述，并介绍了LP松弛技术和罚函数法来处理离散校验约束，然后引入PDD框架进行求解得到了LDPC码的PDD译码算法，并对所提的PDD译码算法和其他译码算法进行了复杂度的分析和比较。接着基于深度展开的思想，将所提的PDD算法展开为一个模型

驱动的神经网络，将可调参数都转化成一系列层间独立的可训练参数，得到了一个可学习的PDD神经网络译码器。最后我们软件仿真分析和比较了所提的PDD译码器和其他对比译码器的译码性能。

第四章主要研究了ISI信道下的PDD译码算法设计。首先给出了ISI信道下LDPC码ML译码问题的描述，将其表述为一个QP问题，然后基于PDD算法框架提出了基于校验多面体约束的ISI-PDD译码算法。接着针对译码算法中的校验多面体投影步骤，分析了校验多面体的几何结构，然后提出了一种快速校验多面体投影算法来提升校验多面体投影的效率。最终对所提的算法进行复杂度分析和数值仿真，并和其他ISI信道下的LDPC码译码方案进行比较。

第五章对论文的主要研究内容进行总结概括，同时展望了该领域未来可行的研究方向。

## 第二章 相关理论基础

### 2.1 引言

本章主要对LDPC码，Turbo均衡，PDD算法以及深度学习的基本原理进行叙述。首先介绍了LDPC码的基本概念，图表示方法，常用的构造编码方式，以及经典的BP译码方法，包括最小和近似的算法。然后介绍了ISI信道下基于Turbo均衡技术的传输模型，以及Turbo均衡的算法框架。接着介绍了PDD算法的原理和基本框架。最后介绍了深度学习的基本原理，并对其中基于模型驱动的深度展开方法进行了具体阐述。

### 2.2 LDPC码的预备知识

LDPC码作为一种非常逼近香农极限的纠错码，自从20世纪90年代被重新发现以来得到了通信工作者们的广泛研究。优越的译码性能以及高效的编译码算法使得LDPC码在工程上获得了广泛的应用，并且成为了5G的核心技术之一。如何进一步提升LDPC码的译码性能，充分发挥LDPC码的优势，是当前LDPC码的研究重点。

#### 2.2.1 LDPC码的基本概念

LDPC码是一类非常重要的线性分组码，线性分组码的概念就是将发送的信息分成若干个码块进行编码，并且码字具有线性特征，可以由其校验矩阵或生成矩阵唯一确定。LDPC码一般由其校验矩阵来确定，顾名思义，LDPC码的校验矩阵需要满足稀疏矩阵的特性，即矩阵中大部分的元素都是“0”，只有少部分是“1”。按照校验矩阵的行和列中“1”的分布可以将LDPC码分为规则LDPC码和非规则LDPC码，以下给出了规则LDPC码的校验矩阵结构特性：

- (1) 每一行和每一列中分别有 $\rho$ 和 $\gamma$ 个“1”；
- (2)  $\rho$ 和 $\gamma$ 均远小于校验矩阵的列数和行数；
- (3) 任意两列或者两行具有共同“1”的个数不大于1。

特性(1)表示规则LDPC码的每行（列）包含“1”的个数是一样的，而非规则的LDPC码则是不同的，设计得当的非规则LDPC码的性能可以优于规则LDPC码；特性(2)遵循

了LDPC码的稀疏特性，一般密度只需要足够小以至可以有效地译码即可，并没有一个精确的界限；特性(3)是为了尽量避免短环的存在而影响译码性能<sup>[45;46]</sup>。

在LDPC码的实际应用中，一般都采用系统码字，如图2.1所示。一个由维度为 $M \times N$ 的校验矩阵 $\mathbf{H}$ 定义的LDPC码可以表示为 $(N, K)$  LDPC码的形式，其中 $K = N - M$ ， $K$ 个低位比特为系统位，对应的是消息序列，剩余 $M$ 个比特为编码后得到的校验位，该系统码的码率为 $R = K/N$ 。系统码在译码后可以直接得到系统位，而且误比特率性能更优。

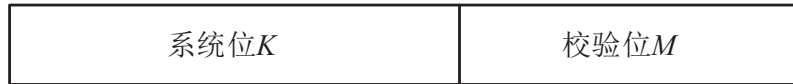


图 2.1 系统码示意图

### 2.2.2 LDPC码的Tanner图表示

为了更直观地表示LDPC码，Tanner首次给出了LDPC码的图形结构，使用二分图来表示LDPC码，这种图也被称作Tanner图<sup>[47]</sup>。Tanner图是一个由顶点和连接顶点的边组成的二分图，其顶点可以被分为变量节点和校验节点两部分。这里以一个 $(7, 4)$ 的线性分组码作为例子说明，以下是其维度为 $3 \times 7$ 的校验矩阵 $\mathbf{H}$ ：

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

令集合 $V = \{v_1, v_2, \dots, v_N\}$ 和 $C = \{c_1, c_2, \dots, c_M\}$ 分别表示变量节点和校验节点的集合，则该校验矩阵 $\mathbf{H}$ 可以用图2.2所示的Tanner图表示。

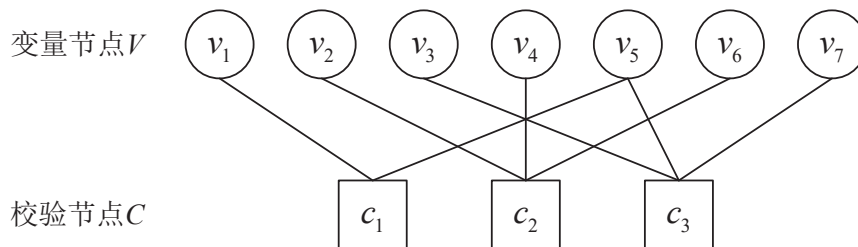


图 2.2  $(7,4)$ 线性分组码的Tanner图表示

在Tanner图中，连接变量节点和校验节点的边与校验矩阵中“1”的位置相对应，与节点相连的边数称为该节点的度 (Degree)。校验节点的度表示该校验节点所对应的校验方程中所包含比特的个数，变量节点的度表示包含该变量节点的校验方程个数。如果将

图2.2中的节点 $v_3$ 和 $c_1$ 也连上，则节点 $c_1$ 、 $v_3$ 、 $c_3$ 和 $v_5$ 之间形成了周期为4的短环，短环的存在会影响LDPC码的收敛速度，降低译码性能，所以在进行LDPC码的构造时要尽量避免出现短环。

### 2.2.3 LDPC码的构造

LDPC码的构造方式可以分为随机化构造和结构化构造两大类。随机化构造主要是指基于一定的设计准则，通过程序随机搜索的方式来得到符合约束规范的校验矩阵。典型的有基于最小环长设计和外部信息度（Extrinsic Message Degree, EMD）的渐进边增长（Progressive Edge Growth, PEG）算法<sup>[48;49]</sup>，可以提高环连通度的近似环外信息度（Approximate Cycle Extrinsic message degree, ACE）算法<sup>[50]</sup>，以及消除陷阱集的循环提升方法<sup>[51]</sup>。随机化构造的方法操作灵活，构造的LDPC码平均误码率低，但是其校验矩阵没有特定的结构，难以设计通用高效的编译码算法。

结构化构造可以根据有限域，有限几何等数学工具构造出一类非常重要的LDPC码型，即准循环（Quasi-Cyclic, QC）LDPC码<sup>[52]</sup>。QC-LDPC码是目前工程应用中最受欢迎的一类LDPC码，其被广泛应用于IEEE 802.11，DVB-S2等标准中，5G的标准协议中规定的两个基本图（Base Graph, BG）也是基于准循环的思想设计的<sup>[53]</sup>。虽然QC-LDPC码的平均误码率稍差于随机化构造的码型，但是其编码实现复杂度低，同时也有利于在硬件上实现高吞吐率的译码算法。下面通过举例来简单说明QC-LDPC码的构造：

$$\mathbf{H} = \begin{bmatrix} A_{0,0} & A_{0,1} & \dots & A_{0,k} & 0 & \dots & -1 & -1 \\ A_{1,0} & A_{1,1} & \dots & \dots & 0 & 0 & \dots & -1 \\ \dots & \dots & \dots & A_{i,k} & \dots & \dots & \dots & \dots \\ A_{m-2,0} & A_{m-2,1} & \dots & \dots & -1 & \dots & 0 & 0 \\ A_{m-1,0} & A_{m-1,1} & \dots & A_{m-1,k} & -1 & -1 & \dots & 0 \end{bmatrix}, \quad (2.2)$$

$\mathbf{H}$ 是一个具有双对角线结构的QC-LDPC码的母矩阵，维度为 $m \times n$ ，其中 $k = n - m$ 。母矩阵中的每个元素表示一个维度为 $z \times z$ 的子矩阵， $z$ 也被称为QC-LDPC码的提升因子，其中“-1”表示零矩阵，“0”表示单位矩阵，“A”表示由单位矩阵循环右移A位后得到的置换矩阵。

### 2.2.4 LDPC码的编码

LDPC码最直接的编码方式就是将信息序列和生成矩阵相乘来得到编码后的码字序列，然而由于LDPC码的生成矩阵并不是一个稀疏矩阵，这会导致编码的复杂度随着码长的增

加呈平方增长。本小节主要介绍基于QC-LDPC设计的递归编码算法，可以实现线性复杂度的编码。

基于(2.2)中的QC-LDPC码校验矩阵，为了方便编码过程，可以将 $\mathbf{H}$ 分成维度分别为 $m \times k$ 和 $m \times m$ 的信息块 $\mathbf{H}_s$ 和校验块 $\mathbf{H}_p$ 两部分，即 $\mathbf{H} = [\mathbf{H}_s \mid \mathbf{H}_p]$ 。接着，进一步对 $\mathbf{H}_p$ 的第一列向量设置以下约束条件：第一行的元素 $A_{0,k}$ 与最后一行元素 $A_{m-1,k}$ 均取“1”；这两个元素外的第三个元素 $A_{i,k}$ 为“0”，可以在中间任意位置；其余所有元素均为“-1”。将编码输出码字 $\mathbf{c}$ 分成系统位 $\mathbf{s}$ 和校验位 $\mathbf{p}$ ，即 $\mathbf{c} = [\mathbf{s} \mid \mathbf{p}]$ 。根据LDPC码的校验约束条件 $\mathbf{H} \cdot \mathbf{c}^T = \mathbf{0}$ ，可以得到

$$\mathbf{H}_s \cdot \mathbf{s} = \mathbf{H}_p \cdot \mathbf{p} = \mathbf{v}, \quad (2.3)$$

由于其中 $\mathbf{v}$ 是可以通过已知的 $\mathbf{H}_s$ 和 $\mathbf{s}$ 计算得到的，所以将 $\mathbf{H}_p \cdot \mathbf{p} = \mathbf{v}$ 展开后可以得到（以下均为模2运算）

$$\begin{aligned} p_0 &= \sum_{i=0}^{M-1} v_i, \\ p_1 &= v_0 + p_0, \\ &\dots \\ p_{i+1} &= (v_i + p_0) + p_i, \\ &\dots \\ p_{M-1} &= p_{M-2} + p_{M-3}. \end{aligned} \quad (2.4)$$

首先通过对 $\mathbf{v}$ 进行求和可以得到 $p_0$ ，剩下的所有校验位 $p_j$ （ $j = i + 1$ 除外）均可以通过前一次计算的 $p_{j-1}$ 和 $v_{j-1}$ 相加得到，而对于 $p_{i+1}$ 还需要加上 $p_0$ ，进而根据式(2.4)进行前向递归就可以得到所有的校验比特，从而完成QC-LDPC码的编码过程。

### 2.2.5 LDPC码的BP译码技术

LDPC码具有诸多高效的译码算法，可以适用于不同的应用场景，目前使用最为广泛的还是基于软判决的BP译码算法。BP译码算法可以分成基于概率域的BP算法和基于对数域的BP算法，考虑到概率域上的BP算法表述不够直观且乘法操作过于频繁，所以更常用的还是基于对数域的BP算法，也就是和积译码算法。和积译码算法通过对数似然比的操作将乘法运算转换成加法运算，从而降低了BP译码算法的复杂度。本小节将重点介绍更为常用的和积译码算法和近似简化的最小和算法。



(1) LDPC码的和积译码算法

本节考虑一个如图2.3所示的信道传输模型，采用 $(N, K)$  LDPC码，校验矩阵为 $\mathbf{H}$ 。将消息序列表示为 $\mathbf{m} \in \{0, 1\}^K$ ，编码后的发送序列为 $\mathbf{c} \in \{0, 1\}^N$ ；这里采用二进制相移键控（Binary Phase Shift Keying, BPSK）的调制方式，得到调制后的序列 $\mathbf{d} \in \{+1, -1\}^N$ ，其中 $d_i = 1 - 2c_i$ ；调制后的序列通过加性高斯白噪声信道（Additive White Gaussian Noise, AWGN）进行传输，在接收侧得到被噪声污染的接收序列 $\mathbf{y} = \mathbf{d} + \mathbf{e}$ ，其中 $\mathbf{e}$ 为噪声序列， $e_i$ 服从均值为0，方差为 $\sigma^2$ 的高斯分布。接收到的序列通过LDPC译码器恢复出消息序列 $\hat{\mathbf{m}} \in \{0, 1\}^K$ 。

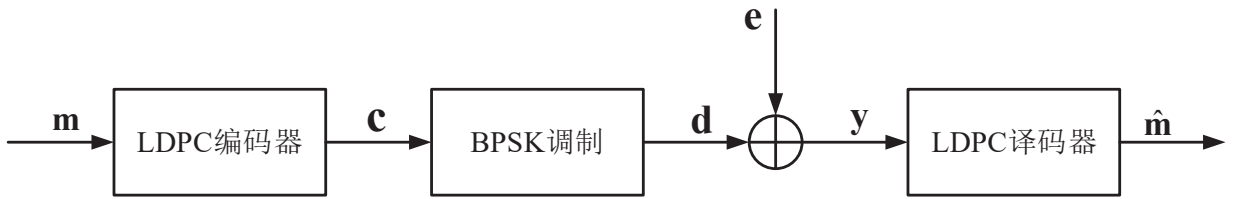


图 2.3 AWGN信道传输模型

首先根据AWGN信道的特性，可以得到信道转移概率

$$\Pr(y_i | d_i = +1) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - 1)^2}{2\sigma^2}\right), \quad (2.5a)$$

$$\Pr(y_i | d_i = -1) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i + 1)^2}{2\sigma^2}\right). \quad (2.5b)$$

假设发送码字的每个码元 $c_i$ 都是等概率发送的，则 $d_i$ 取+1和-1的概率也是相等的，所以根据贝叶斯（Bayes）公式对上式展开后可以得到

$$\frac{\Pr(d_i = +1 | y_i)}{\Pr(d_i = -1 | y_i)} = \frac{\Pr(y_i | d_i = +1)}{\Pr(y_i | d_i = -1)}. \quad (2.6)$$

基于式(2.5)和式(2.6)，可以知道发送信息等概分布时，对数后验概率比等于对数似然比，其中对数似然比（Log-Likelihood Ratio, LLR）可以定义为

$$L(v_i) \triangleq \log\left(\frac{\Pr(y_i | d_i = +1)}{\Pr(y_i | d_i = -1)}\right) = \frac{2y_i}{\sigma^2}, \quad (2.7)$$

当 $L(v_i) < 0$ 时，表示发送比特为1的概率更大，反之则表示发送比特为0的概率更大。LLR的绝对值大小表示该发送比特为0或是1的置信度，绝对值越大，则表明该发送比特为0或是1的概率更大。

在迭代译码过程中，对于发送信息估计的对数似然比 $L(\hat{\mathbf{d}})$ 可以由基于信道的对数似然比 $L(\mathbf{v})$ 和由译码校验关系提供的外部似然比 $L_e(\hat{\mathbf{d}})$ 组成：

$$L(\hat{\mathbf{d}}) = L(\mathbf{v}) + L_e(\hat{\mathbf{d}}), \quad (2.8)$$

其中 $L_e(\hat{\mathbf{d}})$ 由节点间信息传递更新得到， $L(\hat{\mathbf{d}})$ 在迭代期间用于硬判决来判断译码是否结束，所以也被称做伪后验概率信息。

在描述SPA的具体流程前，先给出以下定义：校验节点的索引集合表示为 $\mathcal{J} \triangleq \{1, 2, \dots, M\}$ ，变量节点的索引集合表示为 $\mathcal{I} \triangleq \{1, 2, \dots, N\}$ ；在Tanner图中与校验节点 $j$ 相连的邻居节点集合表示为 $\mathcal{N}_c(j)$ ，与变量节点 $i$ 相连的邻居节点集合表示为 $\mathcal{N}_v(i)$ ； $L(r_{ji})^k$ 表示在第 $k$ 次迭代中，由校验节点 $j$ 传递给变量节点 $i$ 的CTV（Check-To-Variable）消息， $L(q_{ij})^k$ 表示在第 $k$ 次迭代中，由变量节点 $i$ 传递给校验节点 $j$ 的VTC（Variable-To-Check）消息； $L(v_i)$ 表示变量节点 $i$ 对应的初始信道LLR， $L(Q_i)^k$ 表示第 $k$ 次迭代时的伪后验概率信息。下面给出SPA的具体步骤：

- 步骤一（初始化）：

设置迭代次数 $k = 0$ ，初始信道LLR为 $L(v_i) = 2y_i/\sigma^2$ ，将校验矩阵 $\mathbf{H}$ 中 $h_{ji} = 1$ 的 $L(q_{ij})^0$ 初始化为 $L(v_i)$ ， $L(r_{ji})^0$ 均取0。

- 步骤二（校验节点更新）：

$$L(r_{ji})^{k+1} = 2 \tanh^{-1} \left( \prod_{i' \in \mathcal{N}_c(j) \setminus i} \tanh(L(q_{i'j})^k/2) \right), \quad (2.9)$$

式(2.9)为第 $k$ 次迭代时，校验节点 $j$ 传递给变量节点 $i$ 的CTV消息更新过程，其中 $\mathcal{N}_c(j) \setminus i$ 表示除了节点 $i$ 外，所有与校验节点 $j$ 相邻的变量节点集合，双曲正切函数和反双曲正切函数定义为

$$\tanh(x/2) = \frac{e^x - 1}{e^x + 1}, \quad \tanh^{-1}(x) = \frac{1}{2} \ln \frac{1+x}{1-x}. \quad (2.10)$$

- 步骤三（变量节点更新）：

$$L(Q_i)^{k+1} = L(v_i) + \sum_{j \in \mathcal{N}_v(i)} L(r_{ji})^k, \quad (2.11a)$$

$$L(q_{ij})^{k+1} = L(Q_i)^{k+1} - L(r_{ji})^k, \quad (2.11b)$$

式(2.11a)表示第 $k$ 次迭代时变量节点 $i$ 对应的伪后验概率信息更新过程，式(2.11b)表示变量节点 $i$ 传递给校验节点 $j$ 的VTC消息，其只需要在伪后验概率信息更新的基础上减去来自校验节点 $j$ 的CTV消息。

- 步骤四（硬判决）：

$$\hat{v}_i = \begin{cases} 0 & L(Q_i)^k \geq 0, \\ 1 & L(Q_i)^k < 0, \end{cases} \quad (2.12)$$

式(2.12)表示在第 $k$ 次迭代时, 根据伪后验概率信息进行硬判决。如果硬判结果 $\mathbf{v}$ 满足特征校验式, 即 $\mathbf{H} \cdot \mathbf{v}^T = \mathbf{0}$ , 或者达到最大迭代次数 $k_{\max}$ , 则结束迭代译码过程, 将硬判结果输出得到消息序列 $\hat{\mathbf{m}}$ , 否则跳转至步骤二继续迭代。

## (2) LDPC码的归一化最小和译码算法

在SPA的变量节点更新中, 只有简单的加法运算, 但是在校验节点的更新中存在着复杂的双曲正切和反双曲正切运算, 这类运算在FPGA (Field Programmable Gate Array) 或者专用芯片上做硬件实现的时候需要消耗大量的逻辑存储资源, 所以在工程上更常用的是归一化的MSA。完整的归一化MSA如算法1所示。

---

### 算法 1 LDPC码的归一化MSA

---

输入: 信道LLR:  $L(\mathbf{v})$

- 1: 初始化 $L(q_{ij})$ 为 $L(v_i)$ ,  $L(r_{ji})$ 为0, 设置最大迭代次数 $k_{\max}$
- 2: **repeat**
- 3:     **for**  $j = 1, 2, \dots, M$  **do**
- 4:         **for**  $i \in \mathcal{N}_c(j)$  **do**
- 5:             基于式(2.13)更新校验节点信息 $L(r_{ji})$
- 6:         **end for**
- 7:     **end for**
- 8:     **for**  $i = 1, 2, \dots, N$  **do**
- 9:         基于式(2.11a)更新伪后验概率信息 $L(Q_i)$ , 同时基于式(2.12)完成硬判决
- 10:         **for**  $j \in \mathcal{N}_v(i)$  **do**
- 11:             基于式(2.11b)更新变量节点信息 $L(q_{ij})$
- 12:         **end for**
- 13:     **end for**
- 14:     **end for**
- 15: **until** 达到最大迭代次数 $k_{\max}$ 或者满足特征校验式 $\mathbf{H} \cdot \mathbf{v}^T = \mathbf{0}$

输出: 消息序列 $\hat{\mathbf{m}}$

---

MSA将SPA中校验节点的更新过程近似成一个取最小和的过程, 而归一化MSA则是在MSA的基础上引入了一个归一化因子。因为MSA的近似结果是偏大的, 所以通过引入一个小于1的归一化因子得到更好的近似结果, 从而提升MSA的性能。归一化MSA的校验

节点更新过程可以表示为以下形式：

$$L(r_{ji})^{k+1} = \alpha \times \left( \prod_{i' \in \mathcal{N}_c(j) \setminus i} \text{sign}(L(q_{i'j})^k) \right) \times \left( \min_{i' \in \mathcal{N}_c(j) \setminus i} (|L(q_{i'j})^k|) \right), \quad (2.13)$$

其中 $\alpha$ 为归一化因子， $\text{sign}(\cdot)$ 为符号函数。最小和操作是指除了节点 $i$ 以外，从校验节点 $j$ 的所有邻居节点的变量和信息中选取最小值。在硬件实现时，最小和操作可以通过并行排序算法来实现，只涉及比较运算，而归一化因子一般会取0.75，所以可以通过移位和加法的操作来实现归一化因子的乘法。

## 2.3 Turbo均衡基本原理

如果将纠错码编码后的码字通过ISI信道传输，ISI信道引入的码间干扰会使接收端的译码器产生严重的性能衰退，所以通常会引入均衡器或者检测器来减轻码间干扰的影响，使译码器可以更好地恢复出发送的数据。考虑到复杂度的因素，均衡问题和译码问题通常单独考虑，两个模块间的交互有限，所以性能并不理想。Turbo均衡则通过联合均衡和译码问题得到了更多的性能增益，其基本思想是将ISI信道视作串行级联系统中的内码，然后通过软信息迭代过程来均衡信道和译码。

Turbo码以及LDPC码都凭借迭代译码算法获得了很大的性能增益，这清楚地表明了软信息并不只能在一个方向上传递。当一个译码算法处理了软信息后，又可以生成自己的软信息，用于指示每个比特的似然信息。而Turbo均衡正是延续了这一思想，在均衡过程中通过交织并接收来自译码器的软信息，在均衡器和译码器之间创建一个反馈回路。在这个回路中，每个组成模块都接收来自对方的软信息，并生成自己对于某个比特的似然信息作为先验信息传递给对方，这一过程也可以称为置信度传播。需要注意的是，在反馈回路上需要避免产生直接反馈，所以在传递交互时只传递外部信息。

图2.4所示为基于Turbo均衡的ISI信道传输模型。其中 $\mathbf{a} \in \{0, 1\}^K$ 表示长度为 $K$ 的消息序列，通过LDPC编码器后得到码长为 $N$ 的码字序列 $\mathbf{b} \in \{0, 1\}^N$ 。交织器用于打乱数据块内的符号顺序，可以避免突发噪声，同时在Turbo均衡过程中可以有效抑制相邻符号间的相关性。交织后的序列 $\mathbf{c} \in \{0, 1\}^N$ 通过BPSK调制映射后得到发送序列 $\mathbf{d} \in \{-1, 1\}^N$ ，其中 $d_i = 1 - 2c_i$ 。记忆长度为 $T$ 的ISI信道可以建模为离散时间的有限脉冲响应序列 $(h_0, h_1, \dots, h_T) \in \mathbb{R}^{T+1}$ ，得到序列 $\mathbf{y}$ 可以表示为 $y_i = \sum_{t=0}^T h_t(1 - 2x_{i-t})$ ，噪声序列 $\mathbf{e}$ 满足均值为0，方差为 $\sigma^2$ 的高斯分布，接收到的序列 $\mathbf{r} \in \mathbb{R}^N$ 可以表示为 $r_i = y_i + e_i$ 。

对于均衡器和译码器相互独立的模型，均衡器没有任何可用的先验信息，只能依靠观测值。但是对于Turbo均衡而言，均衡器在得到观测值 $\mathbf{r}$ 的同时，可以将来自译码器的后验

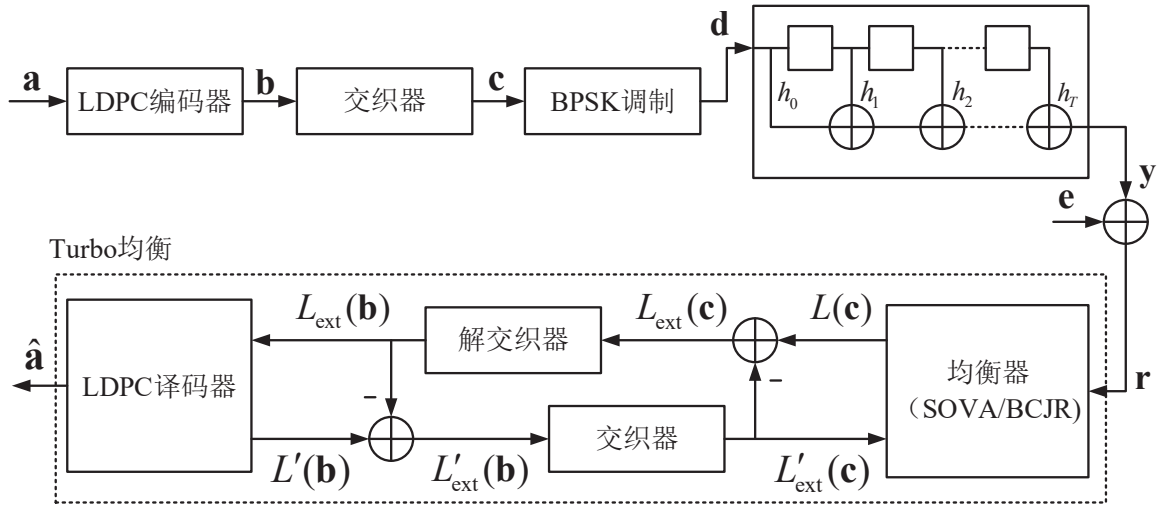


图 2.4 基于Turbo均衡的ISI信道传输模型

信息  $L'_{\text{ext}}(\mathbf{c})$  作为自己的先验信息输入，从而获得更多的性能增益。均衡算法通常可以选择软输出的 BCJR 算法<sup>[30]</sup> 或者 SOVA 算法<sup>[31]</sup>，得到的后验似然比信息可以表示为  $L(\mathbf{c})$ ，其可以分解为内部信息  $L'_{\text{ext}}(\mathbf{c})$  和外部信息  $L_{\text{ext}}(\mathbf{c}) = L(\mathbf{c}) - L'_{\text{ext}}(\mathbf{c})$ 。传递内部信息会产生直接的正反馈，从而偏离全局最优解，所以只反馈外部信息  $L_{\text{ext}}(\mathbf{c})$ 。交织器在反馈回路中可以进一步分散反馈效应，如果译码器基于均衡器提供的关于给定比特的软信息来生成自己关于该比特的软信息，那么均衡器就不能认为该信息独立于信道观察。引入交织器后，均衡器可以将接收信号中较远比特处收集的信息传递给译码器，反之译码器也是如此，从而可以很好地抑制相邻符号之间的相关性。解交织后得到的  $L_{\text{ext}}(\mathbf{b})$  作为先验信息送入 LDPC 译码器中，译码算法可以采用基于软判决的和积译码算法或者归一化最小和算法。输出后验概率的似然比信息  $L'(\mathbf{b})$ ，减去内部先验信息后得到外部信息  $L'_{\text{ext}}(\mathbf{b}) = L'(\mathbf{b}) - L_{\text{ext}}(\mathbf{b})$ ， $L'_{\text{ext}}(\mathbf{b})$  经过交织后得到的  $L'_{\text{ext}}(\mathbf{c})$  作为先验信息输入到均衡器中。

算法2中总结了 Turbo 均衡方案的具体过程，其中具体的算法过程表示为  $A = f(B)$  的形式， $A$  表示算法的输出， $f(\cdot)$  表示某一算法过程， $B$  表示算法的输入。

## 2.4 PDD 算法基本原理

在通信领域，很多问题都可以表述为一个优化问题，对于凸优化（Convex Optimization）问题，可以使用内点法或者其他凸优化的方法进行高效可靠地求解，但对于很多非凸非光滑（Nonconvex Nonsmooth）的优化问题，通常缺乏有效的算法来求解。在一些已有的 LDPC 码 LP 译码相关的工作中，都采用了 ADMM 算法对 LP 译码问题进行求解，ADMM 算法是增广拉格朗日（Augmented Lagrangian, AL）方法的一种重要变体，其调度

**算法 2 Turbo均衡****输入：**接收序列 $\mathbf{r}$ 

- 1: 初始化 $L'_{\text{ext}(\mathbf{c})}$ 为全零向量
- 2: **repeat**
- 3:    执行均衡算法:  $L(\mathbf{c}) = \text{Equalizer}(\mathbf{r}, L'_{\text{ext}(\mathbf{c})})$
- 4:     $L_{\text{ext}(\mathbf{c})} = L(\mathbf{c}) - L'_{\text{ext}(\mathbf{c})}$
- 5:    执行解交织:  $L_{\text{ext}(\mathbf{b})} = \text{Deinterleaver}(L_{\text{ext}(\mathbf{c})})$
- 6:    执行译码算法:  $[\hat{\mathbf{a}}, L'(\mathbf{b})] = \text{Decoder}(L_{\text{ext}(\mathbf{b})})$
- 7:     $L'_{\text{ext}(\mathbf{b})} = L'(\mathbf{b}) - L_{\text{ext}(\mathbf{b})}$
- 8:    执行交织:  $L'_{\text{ext}(\mathbf{c})} = \text{Interleaver}(L'_{\text{ext}(\mathbf{b})})$
- 9: **until** 达到最大迭代次数或者译码结果校验正确

**输出：**信息序列估计值 $\hat{\mathbf{a}}$ 

方式简单，适用于求解大规模的分布式优化（Distributed Optimization）问题，且对于凸问题的求解具有很强的收敛保证。但是在LP译码问题中引入罚函数后，其会转变为一个非线性非凸的问题，而ADMM算法在求解非凸的问题时并不一定收敛，无法保证取到全局最优解<sup>[25]</sup>。

PDD算法是一种用于求解非凸非光滑优化问题的双层迭代算法，其内层循环通过块坐标下降（Block Coordinate Descent, BCD）方法，通过求解由块坐标分解生成的一系列子问题，对非凸非平滑的AL问题进行近似求解，然后在外层循环中进行对偶变量和惩罚参数<sup>[54]</sup>的更新。文献[54]严格证明了PDD算法基于一定的约束条件下到KKT点的收敛性，给PDD算法的可行性提供了理论保证。以下给出PDD算法的具体描述，首先考虑包含多个变量的等式约束优化问题：

$$\min_{\mathbf{x}, \mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \quad (2.14a)$$

$$\text{s.t. } \mathbf{Ax} + \mathbf{Bz} = \mathbf{c}. \quad (2.14b)$$

其中 $\mathbf{x} \in \mathbb{R}^N$ ， $\mathbf{z} \in \mathbb{R}^M$ ， $\mathbf{A} \in \mathbb{R}^{P \times N}$ ， $\mathbf{B} \in \mathbb{R}^{P \times M}$ ， $\mathbf{c} \in \mathbb{R}^P$ ，在PDD算法框架中，函数 $f: \mathbb{R}^N \rightarrow \mathbb{R} \cup \{\pm\infty\}$ 和 $g: \mathbb{R}^M \rightarrow \mathbb{R} \cup \{\pm\infty\}$ 可以是凸函数，也可以是非凸函数。

接着，将问题(2.14)的目标函数和约束条件表述为一个AL函数：

$$L_{\mu}(\mathbf{x}, \mathbf{y}) \triangleq f(\mathbf{x}) + g(\mathbf{z}) + \frac{\mu}{2} \|\mathbf{Ax} + \mathbf{Bz} - \mathbf{c} + \frac{\lambda}{\mu}\|^2, \quad (2.15)$$

其中 $\lambda \in \mathbb{R}^P$ 为拉格朗日乘子，也称作对偶变量， $\mu > 0$ 为惩罚参数。AL方法将一个约束问题转换成一个无约束问题，相比普通的拉格朗日方法多了一个二次残差项，可以平滑对偶问题实现更快的收敛速度。

得到AL问题的表述后，可以采用PDD算法进行求解，首先在内层循环采用BCD方法将问题分解成若干个子问题，即分别求解 $\mathbf{x}$ 和 $\mathbf{y}$ ，每个子问题在第 $k$ 次内层迭代时更新过程如下：

$$\mathbf{x}^{k+1} = \arg \min_{\mathbf{x} \in \mathbb{R}^N} L_\mu(\mathbf{x}, \mathbf{z}^k), \quad (2.16a)$$

$$\mathbf{z}^{k+1} = \arg \min_{\mathbf{z} \in \mathbb{R}^N} L_\mu(\mathbf{x}^{k+1}, \mathbf{z}). \quad (2.16b)$$

这个过程也可以近似看作对偶上升（Dual Ascent）过程，当达到设定的最大迭代次数后，停止内层迭代，完整的BCD算法如算法3所示。

---

### 算法 3 BCD算法

---

- 1: 初始化 $\mathbf{x}^0$ ,  $\mathbf{y}^0$ ,  $\lambda$ , 设置合适的 $\mu$ ,  $k \leftarrow 0$
  - 2: **repeat**
  - 3:     根据式(2.16a)和(2.16b)相继更新 $\mathbf{x}^{k+1}$ 和 $\mathbf{z}^{k+1}$
  - 4:      $k \leftarrow k + 1$
  - 5: **until** 达到最大迭代次数 $k_{\max}$
- 输出:  $\mathbf{x} \leftarrow \mathbf{x}^{k+1}$ ,  $\mathbf{z} \leftarrow \mathbf{z}^{k+1}$
- 

PDD算法的外层循环进行对偶变量更新，其在第 $l$ 次外层迭代时更新过程如下：

$$\lambda^{l+1} = \lambda^l + \mu(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}). \quad (2.17)$$

同时，惩罚参数 $\mu$ 也需要进行更新，使其随着迭代次数的增加不断增大，可变的惩罚参数使得PDD算法可以自适应地在AL方法和惩罚方法间切换，从而有望找到一个合适的参数 $\mu$ 使得算法最终收敛。PDD算法的收敛条件为初始残差足够小，即：

$$\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2 \leq \varepsilon. \quad (2.18)$$

其中 $\varepsilon > 0$ 表示初始残差的容忍度，其选取条件根据具体问题具体情况讨论。

PDD算法的完整过程如算法4所示。PDD算法可以有效地求解无线通信，信道处理等领域中的各类优化问题，但是在实际应用上仍然存在一定的局限性，比如涉及大量高维矩阵的运算，以及双层循环引入的延迟。

---

**算法 4 PDD算法**

---

- 1: 初始化 $\mathbf{x}^0, \mathbf{y}^0, \lambda^0$ , 设置合适惩罚参数 $\mu, l \leftarrow 0$
- 2: **repeat**
- 3:     运行算法3来得到 $\mathbf{x}^{l+1}, \mathbf{y}^{l+1}$
- 4:     根据式(2.17)更新对偶变量 $\lambda^{l+1}$ , 同时增大惩罚参数 $\mu$
- 5:      $l \leftarrow l + 1$
- 6: **until** 达到最大迭代次数 $l_{\max}$ 或者达到收敛条件

输出:  $\mathbf{x} \leftarrow \mathbf{x}^{l+1}, \mathbf{z} \leftarrow \mathbf{z}^{l+1}$

---

## 2.5 深度学习技术简介

深度学习是机器学习中的一个重要分支，其核心出发点是通过一个多层的神经网络来模拟人脑的深层结构和认知机制，可以从海量数据中提取关键特征，学习某种行为模式，目前在计算机视觉，自然语言处理等领域得到了广泛应用，取得了丰富的成果。同时，凭借其强大的学习拟合能力，近年来深度学习也被应用于物理层通信领域，为无线通信和信号处理算法的设计提供了新的思路。

### 2.5.1 神经网络的基本结构

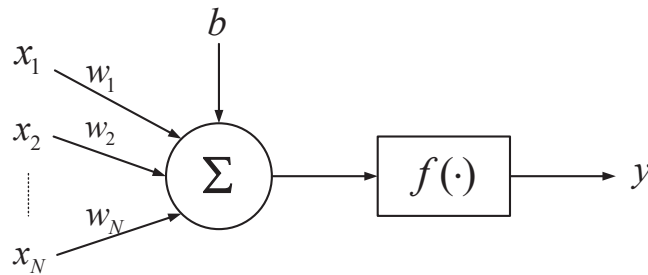


图 2.5 神经元模型

人工神经元 (Neuron) 作为神经网络的基本组成单元，可以看作一个多输入，单输出的非线性单元，其输出可以作为下一层神经元的输入，从而可以组成功能强大的神经网络。如图2.5所示是一个人工神经元的模型，其中 $\{x_1, x_2, \dots, x_N\}$ 表示神经元的 $N$ 个输入， $\{w_1, w_2, \dots, w_N\}$ 代表对应不同输入的权重，加权求和后的输入 $\sum_{i=1}^N w_i x_i$ 加上偏置项 $b$ 后通过一个非线性的激活函数 $f(\cdot)$ ，从而得到神经元的输出 $y$ ，具体过程可以表示为：

$$y = f\left(\sum_{i=1}^N w_i x_i + b\right), \tag{2.19}$$



激活函数决定了神经元的非线性特征，对于整个神经网络的拟合能力至关重要。常用的激活函数有以下几类：

- sigmoid函数：

$$y = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \in (0, 1), \quad (2.20)$$

sigmoid函数将输出映射到0和1之间，适用于预测概率的模型，连续可微，梯度比较平滑，但是sigmoid函数在训练时涉及的计算量大，且可能会存在梯度消失的情况。

- tanh函数：

$$y = \text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \in (-1, 1), \quad (2.21)$$

tanh函数的输出位于-1和1之间，以0为中心，负输入均被映射为负数，tanh函数的曲线与sigmoid函数相似，同样会出现梯度消失的问题。

- ReLU函数：

$$y = \text{ReLU}(x) = \max(0, x), \quad (2.22)$$

ReLU函数又称为修正线性单元（Rectified Linear Unit），是一种分段线性函数。ReLU函数可以有效解决tanh函数和sigmoid函数的梯度消失问题，并同时加快梯度下降的速度，且由于其只涉及线性计算，所以计算速度更快，但是当输入为负数的时候，ReLU函数将失去作用，在反向传播的时候梯度会完全变为0。

### 2.5.2 神经网络的训练方法

损失函数是神经网络训练过程中的关键一环，其用来度量模型的预测值和真实值之间的差异值，也被称为损失值。损失值可以通过后向传播（Back Propagation）来更新网络中的各项参数，从而降低真实值和预测值之间的差异程度，使得模型往正确的方向靠拢。损失函数主要可以分为基于距离度量的损失函数和基于概率分布度量的损失函数。

- 基于距离度量的损失函数将输入数据映射到欧氏空间等基于距离的特征空间中，代表性的有均方误差（Mean Squared Error, MSE）损失函数，具体表述如式(2.23)。MSE损失函数计算了模型输出和标签值之差的 $l_2$ 范数，常用于深度学习的回归任务中，收敛速度较快。

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2. \quad (2.23)$$

- 基于概率分布度量的损失函数通过样本的概率分布来度量其与标签分布之间的距离，代表性的有交叉熵（Cross-Entropy）损失函数，具体表述如式(2.24)。交叉熵最初是信息论中描述最小平均编码长度的一个概念，引入机器学习后常用于分类任务中，其参数更新速度与误差大小相关，可以有效避免梯度消失。

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^N [y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i)]. \quad (2.24)$$

神经网络的训练过程包括输入信号通过网络映射的前向传递和损失误差反向传播两个部分。其中，反向传播计算损失函数的梯度来调整网络中的权重参数，其中涉及到链式法则和梯度下降法，而梯度下降法的设计将直接影响网络训练的结果和收敛速度。常用的梯度下降法主要有以下四种：

- 随机梯度下降（Stochastic Gradient Descent, SGD）法<sup>[55]</sup>：SGD算法每次随机从样本集中抽取一个样本对权重进行更新，其不一定是向着最优的方向下降，所以对学习率的设置要求很谨慎，不一定收敛到全局最优。
- Momentum梯度下降法<sup>[56]</sup>：Momentum方法在原有的梯度下降法中引入了动量，这使得训练到达一个局部最优点的时候可能在动量的帮助下冲出这个局部最优点，并且可以减少许多震荡，更快到达全局最优点。
- RMSprop梯度下降法<sup>[57]</sup>：RMSprop法是针对自适应梯度下降的Adagrad的一个改进，Adagrad可以自适应改变学习率来加快收敛速度，而RMSprop则是在其基础上采用指数加权平均的思想，只对最近的梯度进行计算，从而避免出现提前停止的情况。
- Adam梯度下降法<sup>[58]</sup>：Adam算法不同于传统的随机梯度下降法，其结合了Adagrad和RMSprop的优势，可以对不同的参数设计自适应的学习率。Adam算法可以执行高效的计算任务，常被应用于求解大规模数据和参数的优化问题。

### 2.5.3 深度展开的基本原理

深度学习中主要有数据驱动和模型驱动两大类神经网络结构。其中数据驱动的神经网络主要由DNN, CNN等常用的神经网络结构组成，其可以看作学习输入和输出之间的映射关系的黑盒，这类方法的架构迁移性和学习能力都很强，应用非常广泛。而模型驱动的神经网络中会用到更多已知的领域知识，基于现有的算法模型来构建网络，在通信领域的应用中，模型驱动的神经网络相比于数据驱动的神经网络具有如下几个优势：可以基于通

信领域大量现有的无线通信和信号处理算法来设计神经网络结构，这些现有算法具有一定的性能保证，并且可以有效地进行硬件实现，不需要重新设计硬件架构；大多数作为基础模型的通信算法具有相对较少的可训练参数，可以简化训练的复杂度；网络结构更加直观，具有较强的可解释性。

深度展开是最具代表性的模型驱动神经网络设计方法，其整体思路可以归结如下：首先给出目标问题的一个迭代推理算法，这个算法是基于传统的领域知识设计的；接着将这个迭代算法逐层展开为一个类似神经网络的分层结构，每一层对应原算法的一次迭代过程，网络层数固定；然后将原算法的可调参数和系数逐层设置为可训练的参数，进而得到一个类似神经网络的架构，其中训练参数的选择和设置将直接影响网络的训练结果；在训练过程中使用基于梯度下降的方法来对可训练的参数进行调整，从而得到最优的网络参数配置。接下来举例说明深度展开的网络架构。

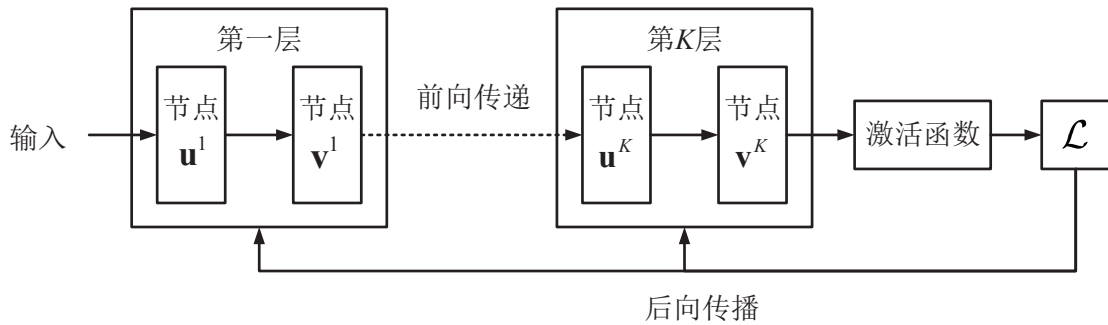


图 2.6 深度展开模型结构

图2.6是一个层数为 $K$ 的深度展开网络模型结构。第 $k$ 层的节点更新对应于所展开的推理算法中第 $k$ 次迭代中的变量更新过程，具体可以表示为

$$\begin{aligned} \mathbf{u}^k &= f(\mathbf{u}^{k-1}, \mathbf{v}^{k-1}; \eta_{k-1}, \theta_{k-1}), \\ \mathbf{v}^k &= g(\mathbf{u}^k, \mathbf{v}^{k-1}; \eta_{k-1}, \theta_{k-1}), \end{aligned} \tag{2.25}$$

其中 $\{\eta_k\}_{k=1}^K$ 和 $\{\theta_k\}_{k=1}^K$ 表示层间独立的可训练参数。网络通过 $K$ 层的前向传递过程后，选择合适的激活函数和损失函数来合理估计预测值和真实值之间的误差，然后通过误差的反向传播过程，调整可训练参数的值来降低预测误差，从而得到最优的网络参数配置。

深度展开结合了传统神经网络的推理能力以及推理算法的内部结构，可以在固定的层数中执行推理并获得最佳的性能。通信系统中的许多信号处理任务，比如检测和译码，都可以表述为优化问题，这些优化问题的求解通常需要用迭代的方式进行求解，但是在实际的高吞吐率低延迟的通信系统中，对于迭代次数的要求十分苛刻。而深度展开就给降低算

法迭代次数提供了解决方案，通过训练来寻找最优的参数，从而最大限度地减少算法达到指定增益所需的迭代次数。

## 2.6 本章小结

本章介绍了与论文相关的基本原理知识，给后续章节的工作奠定了理论基础。具体地，首先介绍了LDPC码相关的基本原理，常用的构造和编码方式，以及经典的和积译码算法和近似简化的归一化最小和译码算法。然后，介绍了Turbo均衡的基本原理和算法框架。接着，介绍了PDD算法的基本原理，与ADMM算法的区别，以及算法框架。最后，介绍了深度学习中神经网络的基本结构，一些常用的激活函数和损失函数，以及深度展开在通信领域应用的优势和设计方法。

## 第三章 基于LP译码问题的LDPC码PDD译码技术

### 3.1 引言

LDPC码作为一种具有诸多优良性质，且非常逼近香农极限的纠错码，已被广泛应用于各类通信系统中。基于软判决的BP译码算法可以获得逼近最大似然译码的译码性能，并且其各种改进简化算法已经在实际系统中得到广泛部署，但是BP算法并不能确保收敛，且在较高信噪比下可能出现错误平层，不利于一些高可靠性场景的应用。LP译码算法作为一种具有很强理论保证的算法，可以在高信噪比区域获得优于BP译码算法的性能，但是传统的LP译码算法在低信噪比区域性能不佳，且没有充分利用其数学结构，译码复杂度较高。如何进一步提升LP译码算法的性能，并降低其计算复杂度成为了近年来的研究热点。

受此启发，针对LDPC码的LP译码问题，本章研究了基于级联整数规划问题的PDD译码算法，并且引入了深度学习技术对PDD算法做进一步优化，旨在进一步提升PDD译码算法的性能并降低译码复杂度。

本章剩余部分安排如下：首先给出了ML译码及其级联整数规划问题的描述；然后，基于级联整数规划问题和PDD算法框架提出了PDD译码算法，并将所提的PDD译码算法和经典的SPA译码算法以及其他先进的LP译码算法进行复杂度分析和比较；进一步地，我们运用深度展开技术，将所提的PDD译码算法展开为一个模型驱动的神经网络；最后我们通过仿真对所提的译码器和其他先进的译码器进行了性能分析和比较。

### 3.2 优化问题描述

#### 3.2.1 ML译码问题

本章考虑码长为 $N$ ，码率为 $R$ 的LDPC码 $\mathcal{C}$ ，由维度为 $M \times N$ 的校验矩阵 $\mathbf{H}$ 定义。令集合 $\mathcal{I} \triangleq \{1, 2, \dots, N\}$ 和 $\mathcal{J} \triangleq \{1, 2, \dots, M\}$ 分别代表变量节点和校验节点的索引集合，令 $d_j$ ， $j \in \mathcal{J}$ 表示校验节点 $j$ 的度，表示在校验矩阵 $\mathbf{H}$ 的第 $j$ 行有 $d_j$ 个非零元素。假设 $\mathbf{x} \in \{0, 1\}^N$ 为通过无记忆二进制输入对称输出（Binary-Input Symmetric-Output）信道传输的码字，每个比特 $x_i$ 取0或1的概率相等， $\mathbf{y}$ 为加噪的接收信号。令 $\Pr(\mathbf{y}|\mathbf{x})$ 表示码字 $\mathbf{x}$ 的似然概率，基于

离散无记忆的特性有 $\Pr(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^N \Pr(y_i|x_i)$ ，接收对数似然比向量 $\mathbf{v} \in \mathbb{R}^{N \times 1}$ 定义为

$$v_i \triangleq \log \left( \frac{\Pr(y_i | x_i = 0)}{\Pr(y_i | x_i = 1)} \right), \forall i \in \mathcal{I}. \quad (3.1)$$

将式(3.1)展开可以得到

$$\log(\Pr(y_i|x_i)) = -v_i x_i + \log(\Pr(y_i|x_i = 0)). \quad (3.2)$$

ML译码问题旨在寻找使得似然概率最大的码字作为发送码字，可以表示为

$$\begin{aligned} \hat{\mathbf{x}} &= \arg \max_{\mathbf{x} \in \mathcal{C}} \Pr(\mathbf{y}|\mathbf{x}) \\ &= \arg \max_{\mathbf{x} \in \mathcal{C}} \prod_{i=1}^N \Pr(y_i|x_i). \end{aligned} \quad (3.3)$$

结合式(3.2)和式(3.3)，最大化 $\Pr(y_i|x_i)$ 等价于最小化 $v_i x_i$ ，所以ML译码问题可以表示成如下的线性规划问题<sup>[13]</sup>：

$$\min_{\mathbf{x}} \mathbf{v}^T \mathbf{x} \quad (3.4a)$$

$$\text{s.t.} \left[ \sum_{i=1}^N H_{ji} x_i \right]_2 = 0, \forall j \in \mathcal{J}, \quad (3.4b)$$

$$x_i \in \{0, 1\}, \forall i \in \mathcal{I}, \quad (3.4c)$$

其中 $[\cdot]_2$ 表示模2操作，约束项(3.4b)代表LDPC码的奇偶校验约束。特别地， $v_i$ 也可以理解为译码 $x_i = 1$ 的代价。

因为奇偶校验约束(3.4b)和离散约束(3.4c)使LP问题成为一个NP-Hard (Non-deterministic Polynomial-time Hard) 问题<sup>[59]</sup>，所以直接求解(3.4)的LP问题会存在困难，无法在确定的多项式时间内进行求解。在下一小节中，我们引入了级联度分解的方式来化简问题(3.4)，使其更容易进行求解。

### 3.2.2 LP问题的级联整数规划形式

级联整数规划<sup>[21]</sup>的核心思想是通过引入辅助变量的方式来将高度数的校验节点分解为若干低度数的校验节点，从而可以用简单的等式约束或者不等式约束来描述问题。

图3.1举例说明了一个度为6的奇偶校验约束分解过程，原校验式为

$$\left[ x_1^{(0)} + x_2^{(0)} + x_3^{(0)} + x_4^{(0)} + x_5^{(0)} + x_6^{(0)} \right]_2 = 0, \quad (3.5)$$

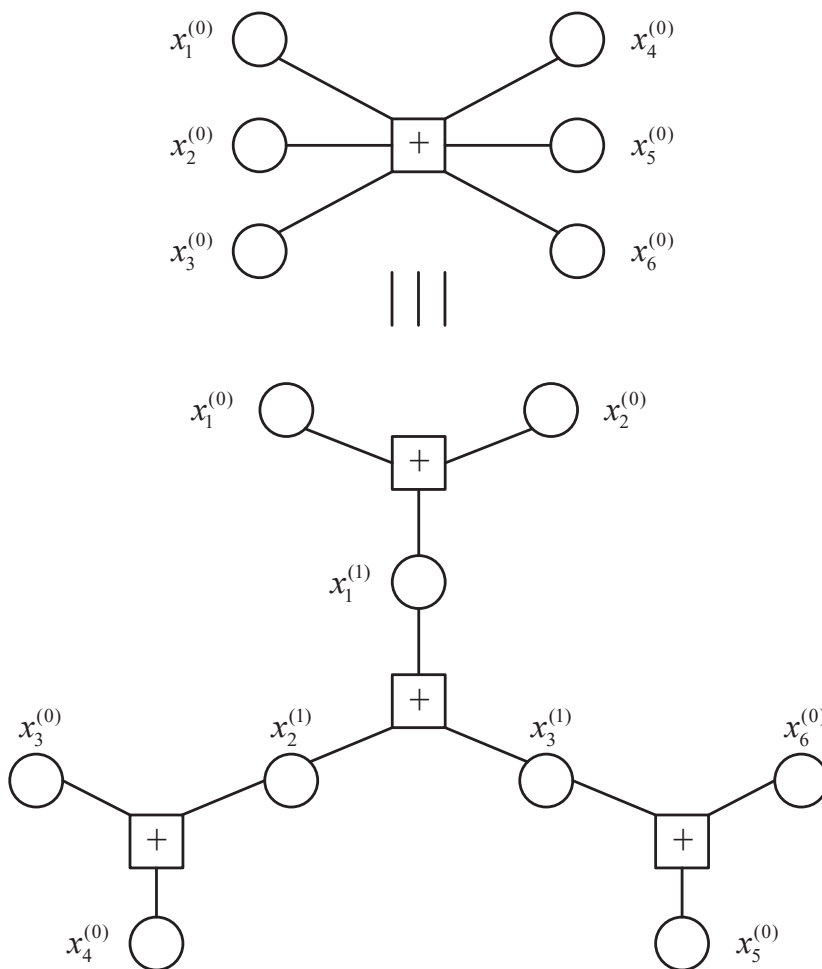


图 3.1 度为6的校验节点分解过程

其中  $x_i^{(0)} \in \{0, 1\}$  为原始变量节点。引入辅助变量  $x_1^{(1)}$ ,  $x_2^{(1)}$  和  $x_3^{(1)}$  后, 度为6的奇偶校验约束可以分解成为四个等价的三变量奇偶校验约束:

$$\begin{aligned} \left[ x_1^{(0)} + x_2^{(0)} + x_1^{(1)} \right]_2 &= 0, & \left[ x_3^{(0)} + x_4^{(0)} + x_2^{(1)} \right]_2 &= 0, \\ \left[ x_5^{(0)} + x_6^{(0)} + x_3^{(1)} \right]_2 &= 0, & \left[ x_1^{(1)} + x_2^{(1)} + x_3^{(1)} \right]_2 &= 0. \end{aligned} \tag{3.6}$$

下面给出一般化的校验分解过程, 校验矩阵  $\mathbf{H}$  中第  $j$  行的奇偶校验式分解可以写作:

- 如果  $l^{(t-1)} = 2l^{(t)} - 1$ :

$$\left[ x_{2k-1}^{(t-1)} + x_{2k}^{(t-1)} + x_k^{(t)} \right]_2 = 0, k = 1, \dots, l^{(t)} - 1, \tag{3.7a}$$

$$x_{2k-1}^{(t-1)} = x_k^{(t)}, k = l^{(t)}, \tag{3.7b}$$

- 如果  $l^{(t-1)} = 2l^{(t)}$ :

$$\left[ x_{2k-1}^{(t-1)} + x_{2k}^{(t-1)} + x_k^{(t)} \right]_2 = 0, k = 1, \dots, l^{(t)}, \quad (3.8a)$$

$$\left[ \sum_{k=1}^{l^{(t)}} x_k^{(t)} \right]_2 = 0, \quad (3.8b)$$

$$x_{2k-1}^{(t-1)}, x_{2k}^{(t-1)}, x_k^{(t)} \in \{0, 1\}, t = 1, \dots, \kappa_j, \quad (3.8c)$$

其中  $\kappa_j = \lceil \log_2(d_j/3) \rceil$  表示第  $j$  行校验式总共需要  $\kappa_j$  次分解,  $l^{(0)} = d_j$ ,  $l^{(t)} = \lceil l^{(t-1)}/2 \rceil$  表示第  $t$  次分解时引入的辅助变量数目,  $x_k^{(0)}$  表示原始变量节点, 即  $x_k^{(0)} = x_{\beta_k}$ ,  $[\beta_1, \dots, \beta_{d_j}]$  表示校验矩阵  $\mathbf{H}$  的第  $j$  行中“1”的索引集合, 而  $x_k^{(t)}$  则表示第  $t$  次分解时引入的辅助变量。递归完成上述分解过程后, 第  $j$  行的奇偶校验式最终可以被等价表示为若干个三变量的校验等式, 但是三变量的校验等式仍然是基于模2运算的表达形式, 这会给后续求解带来困难, 所以可以将每个三变量校验式转化为四个等价的不等式约束:

$$\begin{aligned} x_{2k-1}^{(t-1)} &\leq x_{2k}^{(t-1)} + x_k^{(t)}, & x_{2k}^{(t-1)} &\leq x_{2k-1}^{(t-1)} + x_k^{(t)}, \\ x_k^{(t)} &\leq x_{2k-1}^{(t-1)} + x_{2k}^{(t-1)}, & x_{2k-1}^{(t-1)} + x_{2k}^{(t-1)} + x_k^{(t)} &\leq 2. \end{aligned} \quad (3.9)$$

进一步地, 得到不等式约束(3.9)的矩阵形式  $\mathbf{T}\mathbf{f} \preceq \mathbf{a}$ , 其中:

$$\mathbf{f} = \left[ x_{2k-1}^{(t-1)}, x_{2k}^{(t-1)}, x_k^{(t)} \right]^T, \mathbf{a} = [0, 0, 0, 2]^T, \mathbf{T} = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \quad (3.10)$$

从以上的级联分解形式可以看出, 校验矩阵  $\mathbf{H}$  的每个校验约束需要额外引入  $d_j - 3$  个辅助变量和  $d_j - 2$  个三变量奇偶校验约束, 所以总共的辅助变量  $\Gamma_a$  和三变量校验约束  $\Gamma_c$  可以写作  $\Gamma_a = \sum_{j=1}^M (d_j - 3)$  和  $\Gamma_c = \sum_{j=1}^M (d_j - 2)$ 。为了进一步完整地表述问题的约束项, 给出以下定义: 拓展LLR为  $\mathbf{q} = [\mathbf{v}; \mathbf{0}_{\Gamma_a \times 1}]$ ,  $\mathbf{0}_{\Gamma_a \times 1}$  为全零向量; 拓展码字变量为  $\mathbf{u} = [\mathbf{x}; \tilde{\mathbf{x}}] \in \{0, 1\}^{(N+\Gamma_a) \times 1}$ , 其中  $\mathbf{x} \in \{0, 1\}^{N \times 1}$  为原始码字变量,  $\tilde{\mathbf{x}} \in \{0, 1\}^{\Gamma_a \times 1}$  为引入的辅助变量; 三变量约束矩阵  $\mathbf{A} = [\mathbf{TQ}_1; \dots; \mathbf{TQ}_\gamma; \dots; \mathbf{TQ}_{\Gamma_c}] \in \{0, \pm 1\}^{4\Gamma_c \times (N+\Gamma_a)}$ , 其中  $\mathbf{Q}_\gamma \in \{0, 1\}^{3 \times (N+\Gamma_a)}$  表示选择矩阵, 从  $\mathbf{u}$  中选择对应第  $\gamma$  个三变量方程的变量; 向量  $\mathbf{a}$  的拓展向量  $\mathbf{b} = \mathbf{1}_{\Gamma_c \times 1} \otimes \mathbf{a}$ ,  $\mathbf{1}_{\Gamma_c \times 1}$  表示维度为  $\Gamma_c \times 1$  的全一向量, 符号  $\otimes$  表示克罗内克 (Kronecker) 积。ML问题的原始码字空间可以看作一个基本多面体 (Fundamental Polytope), 文献[21]证明了这个基本多面体可以视作一组最小多面体的交集, 进而问



题(3.4)可以等价表示为以下基于级联分解形式的线性整数规划问题：

$$\min_{\mathbf{u}} \mathbf{q}^T \mathbf{u} \quad (3.11a)$$

$$\text{s.t. } \mathbf{A}\mathbf{u} - \mathbf{b} \preceq \mathbf{0}, \quad (3.11b)$$

$$\mathbf{u} \in \{0, 1\}^{(N+\Gamma_a) \times 1}. \quad (3.11c)$$

文章[21]中证明了基于级联分解的LP问题与原始LP问题<sup>[13]</sup>是等价的，并且两者的译码器具有相同的属性，比如最大似然特性和全零假设（All-zero Assumptions）。下一节中我们将采用PDD算法对基于级联分解的LP问题进行求解。

### 3.3 基于LP问题的PDD译码算法设计

本节我们提出了一种PDD译码算法来求解基于级联分解的LP问题(3.11)，其核心思想是利用LP松弛和罚函数的方法来处理问题中的离散约束，并且采用超松弛机制来加快译码器的收敛速度。PDD算法框架包含内外两层循环，内层循环对AL问题进行近似求解，外层循环进行对偶变量和惩罚参数的更新，可以对非凸问题进行有效地求解。

#### 3.3.1 PDD译码算法

为了契合PDD算法的框架，我们首先引入了辅助变量 $\mathbf{z} \in \mathbb{R}_+^{4\Gamma_c \times 1}$ 来把不等式约束(3.11b)变换成等式约束条件 $\mathbf{A}\mathbf{u} + \mathbf{z} = \mathbf{b}$ ；此外，由于离散约束(3.11c)的存在，问题(3.11)仍是一个NP-Hard问题，无法在多项式时间内求解，所以需要进一步地处理离散约束。本文中我们采用LP松弛和罚函数的方法来处理离散校验约束，除此之外还有一些其他的处理方法，但是会引入额外的复杂度：

- 文献[28]中通过引入额外的辅助变量和等式约束来处理离散约束：

$$u_i(\hat{u}_i - 1) = 0, \quad u_i = \hat{u}_i, \quad 0 \leq u_i \leq 1, \forall i. \quad (3.12)$$

这种方式虽然也可以取得不错的性能，但是因为引入了额外的辅助变量，所以在变量更新时会引入额外的计算复杂度，并且在进行深度展开时网络难以收敛。

- 文献[26]将离散约束转换成等价的有界约束和 $l_p$ 球面的交集：

$$\mathbf{u} \in \{0, 1\}^{N+\Gamma_a} \Leftrightarrow \mathbf{u} \in [0, 1]^{N+\Gamma_a} \cap \left\{ \mathbf{u} : \left\| \mathbf{u} - \frac{1}{2} \mathbf{1}_{N+\Gamma_a} \right\|_p^p = \frac{N+\Gamma_a}{2^p} \right\}. \quad (3.13)$$

其中 $\|\cdot\|_p$ 表示 $l_p$ 范数。利用这种 $l_p$ -box的方法虽然可以得到不错的性能增益，但是其多出的 $l_p$ 球面投影的操作会引入额外的计算复杂度。

首先，我们将式(3.11c)松弛为 $\mathbf{u} \in [0, 1]^{(N+\Gamma_a) \times 1}$ ，然后在目标函数(3.11a)中引入罚函数来增加伪码字的惩罚代价，从而我们可以得到问题(3.11)的惩罚松弛表示：

$$\min_{\mathbf{u}, \mathbf{z}} \mathbf{q}^T \mathbf{u} + \sum_{i \in \mathcal{I}} g(u_i) \quad (3.14a)$$

$$\text{s.t. } \mathbf{A}\mathbf{u} + \mathbf{z} = \mathbf{b}, \quad (3.14b)$$

$$\mathbf{u} \in [0, 1]^{(N+\Gamma_a) \times 1}, \mathbf{z} \in \mathbb{R}_+^{4\Gamma_c \times 1}, \quad (3.14c)$$

其中 $g: [0, 1] \rightarrow \mathbb{R} \cup \{\pm\infty\}$ 代表引入的罚函数，一般而言罚函数需要满足以下条件<sup>[25]</sup>：

- (1)  $g$ 在区间 $[0, 0.5]$ 上单调递增；
- (2)  $g$ 关于0.5对称，即对于 $x \in [0, 1]$ 有 $g(x) = g(1 - x)$ ；
- (3)  $g$ 在 $(0, 0.5)$ 上可微；
- (4)  $g$ 确保在PDD算法迭代时，变量 $u$ 的更新存在闭式解。

因为根据LP算法的最大似然特性，译码器输出的整数解一定是最大似然解，所以设置条件(1)来惩罚分数解，也就是避免伪码字的出现；条件(2)是为了满足全零假设，确保译码错误与发送码字无关；条件(3)和(4)是为了使该问题可以有效地通过分布式算法进行求解。本节中我们采用了 $l_2$ 罚函数，令 $g(u) = -\frac{\alpha}{2}(u - 0.5)^2$ ，其中 $\alpha$ 是罚函数系数。需要注意的是在引入罚函数后，原来的LP问题变成了一个非线性非凸的问题，而PDD算法可以对非凸优化问题进行高效可靠地求解<sup>[54]</sup>。

接下来，我们将问题(3.14)的目标函数和等式约束条件所对应的AL函数定义为

$$L_\mu(\mathbf{u}, \mathbf{z}) \triangleq \mathbf{q}^T \mathbf{u} + \sum_{i \in \mathcal{I}} g(u_i) + \frac{\mu}{2} \|\mathbf{A}\mathbf{u} + \mathbf{z} - \mathbf{b}\|_2^2, \quad (3.15)$$

其中 $\mu > 0$ 为惩罚参数， $\boldsymbol{\lambda} \in \mathbb{R}_+^{4\Gamma_c \times 1}$ 表示与(3.14b)相关的对偶变量，进而我们可以得到AL问题的表述：

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{z}} L_\mu(\mathbf{u}, \mathbf{z}) \\ \text{s.t. } \mathbf{u} \in [0, 1]^{(N+\Gamma_a) \times 1}, \mathbf{z} \in \mathbb{R}_+^{4\Gamma_c \times 1}. \end{aligned} \quad (3.16)$$

本节所提的PDD译码算法采用了双层循环的结构，其内层循环中使用BCD算法来对AL问题进行近似求解，外层循环完成对偶变量 $\boldsymbol{\lambda}$ 和惩罚参数 $\mu$ 的更新。将内层循环和外层循环的最大迭代次数分别表示为 $K$ 和 $L$ ，令 $k$ 和 $l$ 分别表示内层循环和外层循环的索引，对于内层循环，我们将问题分成 $\mathbf{u}$ 和 $\mathbf{z}$ 两部分进行求解，其迭代过程包括以下两个步骤：

(1) 在给定 $\mathbf{z}^k$ 的情况下更新 $\mathbf{u}^{k+1}$ :

此时 $\mathbf{z}^k$ 被看作一个常量，所以针对子问题 $\mathbf{u}$ 的求解可以将(3.16)化简为一个二次优化问题，同时保留 $\mathbf{u}$ 的约束条件，因为该二次优化问题关于每个 $u_i$ 是独立的，所以可以被分解为 $N + \Gamma_a$ 个子问题：

$$\begin{aligned} \min_{u_i} \theta_{i,l} u_i^2 + \omega_{i,l}^k u_i \\ \text{s.t. } 0 \leq u_i \leq 1, \end{aligned} \quad (3.17)$$

其中 $\theta_{i,l} = \frac{\mu}{2} e_i - \frac{\alpha}{2}$ ， $\omega_{i,l}^k = q_i + \mu_l \mathbf{a}_i^T (\mathbf{z}^k - \mathbf{b} + \frac{\boldsymbol{\lambda}^l}{\mu_l}) + \frac{\alpha}{2}$ ， $e_i$ 代表 $\mathbf{A}^T \mathbf{A}$ 的第 $i$ 个对角线元素， $\mathbf{a}_i$ 表示 $\mathbf{A}$ 的第 $i$ 个列向量。根据二次优化问题的一阶最优条件，问题(3.17)的最优解可以表示为

$$u_{i,l}^{k+1} = \Pi_{[0,1]}(-0.5 \frac{\omega_{i,l}^k}{\theta_{i,l}}), \quad (3.18)$$

其中 $\Pi_{[0,1]}(\cdot)$ 表示将元素映射到 $[0, 1]$ 上的欧几里得投影运算，即大于1的元素映射为1，小于0的元素映射为0，剩余元素不变。

(2) 在给定 $\mathbf{u}^{k+1}$ 的情况下更新 $\mathbf{z}^{k+1}$ :

由于此时 $\mathbf{u}^{k+1}$ 和 $\mathbf{q}$ 都是已知的常量，所以子问题 $\mathbf{z}$ 同样可以化简为一个二次优化问题，表示为以下形式：

$$\begin{aligned} \min_{\mathbf{z}} \|\mathbf{b} - \mathbf{A}\mathbf{u}^{k+1} - \frac{\boldsymbol{\lambda}^l}{\mu_l} - \mathbf{z}\|_2^2 \\ \text{s.t. } \mathbf{z} \in \mathbb{R}_+^{4\Gamma_c \times 1}. \end{aligned} \quad (3.19)$$

与第一步相似地，问题(3.19)的最优解可以表示为

$$\mathbf{z}^{k+1} = \Pi_{[0,\infty)}(\mathbf{b} - \mathbf{A}\mathbf{u}^{k+1} - \frac{\boldsymbol{\lambda}^l}{\mu_l}), \quad (3.20)$$

其中 $\Pi_{[0,\infty)}(\cdot)$ 表示将元素通过欧几里得投影映射到 $[0, \infty)$ 上。进一步地，我们采用了超松弛机制<sup>[60]</sup>，将式(3.20)中的 $\mathbf{A}\mathbf{u}^{k+1}$ 替换为 $\beta \mathbf{A}\mathbf{u}^{k+1} + (1 - \beta)(\mathbf{b} - \mathbf{z}^k)$ ，则子问题 $\mathbf{z}$ 的最优解可以表示为

$$\mathbf{z}^{k+1} = \Pi_{[0,\infty)}(\mathbf{b} - \beta \mathbf{A}\mathbf{u}^{k+1} - (1 - \beta)(\mathbf{b} - \mathbf{z}^k) - \frac{\boldsymbol{\lambda}^l}{\mu_l}), \quad (3.21)$$

其中 $\beta \in (1, 2)$ 表示超松弛参数。超松弛机制可以将上一次迭代的信息重新应用于本次迭代中，可以一定程度上加快算法的收敛速度，同时通过超松弛参数 $\beta$ 来控制松弛的程度。

在外层循环中，我们进行对偶变量 $\boldsymbol{\lambda}$ 的更新：

$$\boldsymbol{\lambda}^{l+1} = \boldsymbol{\lambda}^l + \mu_l (\mathbf{A}\mathbf{u}^K + \mathbf{z}^K - \mathbf{b}), \quad (3.22)$$

其中 $\mathbf{u}^K$ 和 $\mathbf{z}^K$ 表示内循环最终的输出。对偶变量可以解释为一个价值变量，所以对偶变量的更新也可以看作对偶上升（Dual Ascent）法中的价值上升（Price Ascent）步骤<sup>[61]</sup>。同时，根据 $\mu_{l+1} = c\mu_l$ 对惩罚参数 $\mu_l$ 进行更新，其中 $c > 1$ 为控制参数，使得 $\mu_l$ 在每次外层迭代中保持一定的增长率增长，从而保证对偶函数的上升。

设置PDD算法的收敛条件为初始残差和对偶残差足够小，即：

$$\begin{aligned} \|\mathbf{A}\mathbf{u}^{k+1} + \mathbf{z}^{k+1} - \mathbf{b}\|_2^2 &< \varepsilon_{ini}, \\ \|\mathbf{z}^{k+1} - \mathbf{z}^k\|_2^2 &< \varepsilon_{dual}, \end{aligned} \quad (3.23)$$

其中 $\varepsilon_{ini} > 0$ 和 $\varepsilon_{dual} > 0$ 分别代表算法对初始残差和对偶残差的容忍度。综上，本章所提的PDD译码算法的总体流程如算法5所示。

---

#### 算法 5 PDD 译码算法

---

输入：获取信道信息 $\mathbf{q}$

- 1: 初始化 $\mathbf{z}^0$ ,  $\boldsymbol{\lambda}^0$ 为零向量；设置合适的参数 $\alpha$ ,  $\beta$ ,  $\mu_0$ 和 $c$ ；令 $k = 0$ ,  
 $l = 0$
- 2: **repeat**
- 3:     **repeat**
- 4:         根据式(3.18)更新变量 $\mathbf{u}$
- 5:         根据式(3.21)更新变量 $\mathbf{z}$
- 6:          $k \leftarrow k + 1$
- 7:     **until** 达到最大迭代次数 $K$ 或者达到收敛条件(3.23)
- 8:     根据式(3.22)更新变量 $\boldsymbol{\lambda}$
- 9:      $\mu_{l+1} = c\mu_l$
- 10:      $\mathbf{u}^0 \leftarrow \mathbf{u}^K$ ,  $\mathbf{z}^0 \leftarrow \mathbf{z}^K$
- 11:      $l \leftarrow l + 1$ ,  $k \leftarrow 0$
- 12: **until** 达到最大迭代次数 $L$ 或者达到收敛条件(3.23)

输出：对 $\mathbf{u}$ 做判决得到译码输出码字 $\mathbf{x}$

---

### 3.3.2 复杂度分析

在本小节中，我们将所提的PDD译码算法和一些先进的LP译码算法进行复杂度的分析对比，同时也分析了经典的和积BP译码算法的复杂度。<sup>1</sup>为了方便讨论，本节考虑校验节

<sup>1</sup>本章所提到的BP译码算法均特指基于对数域的和积译码算法

点或者变量节点度都相同的规则LDPC码，并且设校验节点的度为 $d$ 。

- 对于所提的PDD译码算法，约束矩阵 $\mathbf{A}$ 中的所有可能值为0, -1和1，所以与 $\mathbf{A}$ 相关的所有矩阵乘法运算都可以通过简单的加法运算来执行。像 $\boldsymbol{\theta}$ 和 $\mathbf{q}$ 这些常向量的值都可以在迭代前计算得到，所以我们只需要考虑计算 $\boldsymbol{\omega}/\boldsymbol{\theta}$ 所涉及的乘法运算。除此之外，在外层循环中进行的对偶变量更新可以等价写作：

$$\frac{\boldsymbol{\lambda}^{l+1}}{\boldsymbol{\mu}_l} = \frac{\boldsymbol{\lambda}^l}{\boldsymbol{\mu}_l} + \mathbf{A}\mathbf{u} + \mathbf{z} - \mathbf{b}, \quad (3.24)$$

由于 $\boldsymbol{\lambda}^l/\boldsymbol{\mu}_l$ 和 $\mathbf{A}\mathbf{u}$ 已经在内层循环中计算过，在外层循环中不需要额外的乘法运算，所以只需要考虑内层循环的计算复杂度。进而所提的PDD译码算法的计算复杂度可以表示为 $\mathcal{O}(N + \Gamma_a)$ 。由于本节考虑规则LDPC码，根据 $\Gamma_a = \sum_{j=1}^M (d_j - 3)$ ，可以得到 $\Gamma_a = M(d - 3) = N(1 - R)(d - 3)$ ，根据LDPC的稀疏特性可以得到 $d \ll N$ ，所以 $\Gamma_a$ 与码长 $N$ 相当，PDD译码器的译码复杂度也可以近似表示为 $\mathcal{O}(N + M(d - 3))$ 。

- 文献[28]中提的PDD译码算法与本文所提的PDD译码算法复杂度相近。具体而言，文献[28]中的PDD译码算法的内层循环中进行了 $\{\mathbf{u}, \mathbf{z}, \hat{\mathbf{u}}\}$ 三组变量的更新，外层循环中进行了 $\{\mathbf{y}, \boldsymbol{\omega}, \boldsymbol{\eta}\}$ 三组对偶变量的更新。而在本文所提的PDD译码算法中，内层循环和外层循环分别更新了 $\{\mathbf{u}, \mathbf{z}\}$ 和 $\boldsymbol{\lambda}$ 。所以相对而言，本文所提的PDD译码算法需要更新的变量数目减少了一半，变量更新的执行过程会更简单，计算复杂度更低。
- 文献[25]中所提的ADMM L2译码算法中涉及到校验多面体映射操作 $\Pi_{\text{PP}_{d_j}}(\cdot)$ ，这也是整个算法中最耗时的一部分。如果采用文献[24]中所提的基于迭代的ICPP算法，ADMM L2译码器的复杂度可以表示为 $\mathcal{O}(N + Md)$ 。
- 对于经典的BP译码算法，根据式(2.9)和式(2.11)，其在变量节点的更新过程中只涉及到简单的加法运算，所以我们只考虑其校验节点更新中涉及到的双曲正切和反双曲正切运算。根据文献[18]，BP译码算法单次迭代的复杂度可以表示为 $\mathcal{O}(Md^2)$ 。

综上所述，各算法单次迭代的计算复杂度比较如表3.1所示。通过分析比较可以发现，所提的PDD译码算法的计算复杂度要低于文献[25]和[28]中设计的译码算法，和经典的BP译码算法相当。

### 3.4 基于深度学习的LPDN译码器

上一节详细介绍了本文所提的PDD译码算法，并和经典的BP算法以及其他先进的LP译码方案进行了复杂度的分析和比较。在所提的PDD译码算法中，选择合适的系数

表 3.1 单次迭代计算复杂度比较

译码算法	计算复杂度
BP译码算法 <sup>[18]</sup>	$\mathcal{O}(Md^2)$
ADMM L2译码算法 <sup>[25]</sup>	$\mathcal{O}(N + Md)$
PDD译码算法 <sup>[28]</sup>	$\mathcal{O}(N + M(d - 3))$
所提的PDD译码算法	$\mathcal{O}(N + M(d - 3))$

和参数是至关重要的。比如一个较大的惩罚参数 $\mu_0$ 可能会产生较大的初始残差，保证对偶函数的上升，并且加快收敛的速度，但同时其可能导致较差的译码性能，无法收敛到全局最优点。在传统的迭代算法中，寻找参数 $\{\alpha, \mu_0, c, \beta\}$ 的合适值通常需要通过大量的数值仿真，遍历参数的各种排列组合情况。这样的方式会消耗大量的计算资源和时间，并且可能无法找到最优的参数组合，从而导致译码器的性能并不能达到最优。所以在本节中我们基于所提的PDD译码算法，采用深度展开的技术，进一步提出了一种可学习的PDD译码网络(Learnable PDD Decoding Network, LPDN)，LPDN译码器可以进一步提升PDD译码器的译码性能并减少译码的平均迭代次数。

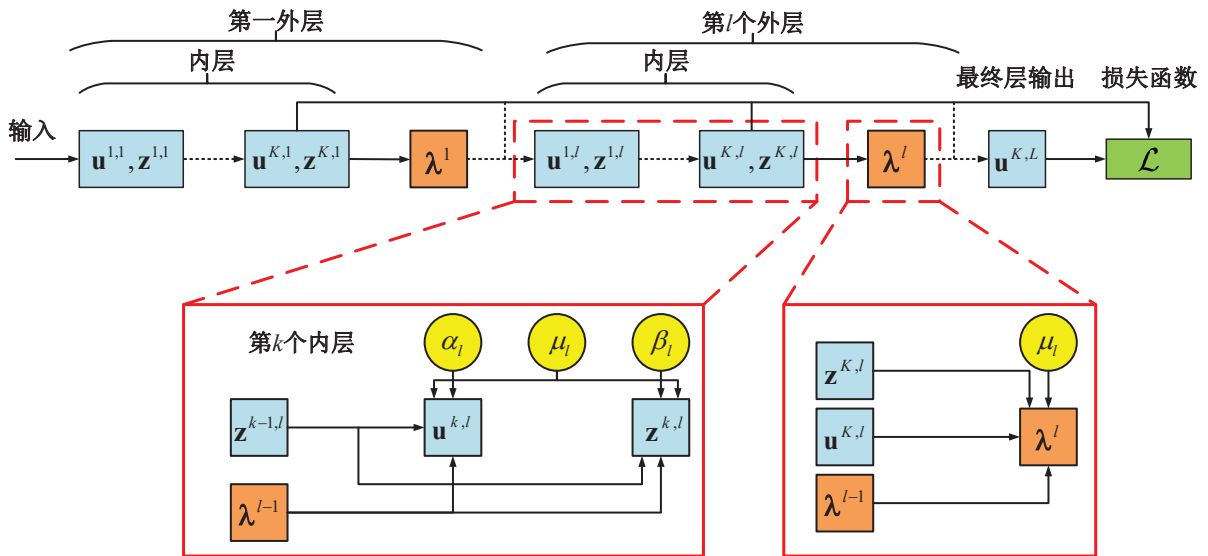


图 3.2 LPDN的网络架构（黄色圆圈表示训练参数，蓝色和橙色矩形分别表示内层和外层更新节点，红色实线框中描述了节点的具体结构）

将所提的PDD译码算法通过深度展开后得到的LPDN网络架构如图3.2所示。令 $L$ 表示外层的层数， $K$ 表示每个外层中包含的内层层数。我们将参数 $\mu$ ， $\alpha$ ，和 $\beta$ 看作一系

列分层独立的可训练参数，根据固定的层数 $L$ ，训练参数可以表示为 $\boldsymbol{\mu} = [\mu_1, \dots, \mu_L]$ ， $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_L]$ 和 $\boldsymbol{\beta} = [\beta_1, \dots, \beta_L]$ 。需要注意的是，相比经典的基于数据驱动的神经网络，我们所需要存储的训练参数在硬件实现的时候只需要消耗很少的存储空间。通过在网络的每个外层中设置独立的可训练参数 $\{\alpha_l, \beta_l, \mu_l\}$ ，可以使网络结构的调整更加灵活，并且可以减少译码性能对于初始参数的依赖。通过使用基于梯度下降的方法来优化本章所提的LPDN译码器，可训练参数 $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\mu}\}$ 可以在训练中自动地被优化校正，而优化后的训练参数可以很好地提升收敛速度以及译码器的译码性能。

在所提的LPDN中，每个内层包含两种更新节点，即节点 $\mathbf{u}$ 和节点 $\mathbf{z}$ 。除此之外，在每个外层中包含了一次节点 $\boldsymbol{\lambda}$ 的更新，节点 $\boldsymbol{\lambda}$ 的更新在 $K$ 个内层更新之后执行。令 $(\mathbf{u}^{k,l}, \mathbf{z}^{k,l})$ 表示在第 $l$ 个外层中第 $k$ 个内层的节点 $\mathbf{u}$ 和节点 $\mathbf{z}$ 的输出，令 $\boldsymbol{\lambda}^l$ 表示第 $l$ 个外层中节点 $\boldsymbol{\lambda}$ 的输出。内层和外层中节点的更新操作与所提的PDD译码算法中的节点更新过程相似，节点 $\mathbf{u}, \mathbf{z}, \boldsymbol{\lambda}$ 的更新过程可以用如下的数学公式表示：

$$\mathbf{u}^{k,l} = \boldsymbol{\eta} \odot \left( \mathbf{q} + \mu_l \mathbf{A}^T (\mathbf{z}^{k-1,l} - \mathbf{b} + \frac{\boldsymbol{\lambda}^{l-1}}{\mu_l}) + \frac{\alpha_l}{2} \right), \quad (3.25a)$$

$$\mathbf{z}^{k,l} = \text{ReLU}(\mathbf{b} - \beta_l \mathbf{A} \mathbf{u}^{k,l} + (1 - \beta_l)(\mathbf{b} - \mathbf{z}^{k-1,l}) - \frac{\boldsymbol{\lambda}^{l-1}}{\mu_l}), \quad (3.25b)$$

$$\boldsymbol{\lambda}^l = \boldsymbol{\lambda}^{l-1} + \mu_l (\mathbf{A} \mathbf{u}^{K,l} + \mathbf{z}^{K,l} - \mathbf{b}), \quad (3.25c)$$

其中 $\boldsymbol{\eta} = 1/(\alpha_l - \mu_l \text{diag}(\mathbf{A}^T \mathbf{A}))$ ，符号 $\odot$ 表示哈达玛（Hadamard）积。在这里使用激活函数 $\text{ReLU}(\cdot)$ 是因为其等价于 $\Pi_{[0, \infty)}(\cdot)$ 。需要注意的是， $\mathbf{z}^{K,l}$ 表示节点 $\mathbf{z}$ 在第 $l$ 个外层的输出，同时也可以看作是第 $l+1$ 个外层的输入 $\mathbf{z}^{0,l+1}$ 。

损失函数通常用于估计网络的输出值与标签值的差距。对于所提的LPDN，我们采用了sigmoid函数作为输出层的激活函数，交叉熵作为损失函数，从而可以将损失函数表示为

$$\mathcal{L} = - \sum_{i=1}^N [x_i \log(\sigma(-c_i)) + (1 - x_i) \log(1 - \sigma(-c_i))], \quad (3.26)$$

其中 $x_i$ 表示真实的标签值， $c_i = 1 - 2u_i$ 将最终层的节点 $\mathbf{u}$ 输出转化为双极型形式， $\sigma(\cdot)$ 表示sigmoid函数 $y = 1/(1 + e^{-x})$ ，将结果都映射到 $(0,1)$ 之间。除此之外，考虑到网络层数相对较多，我们引入了Multiloss机制<sup>[42]</sup>，将每一个外层的输出 $\mathbf{u}^{K,l}$ 都引入损失函数的计算。Multiloss机制可以提升前几层的梯度在误差反向传播时所占的比重，从而加快收敛速度，使结果更加准确。考虑Multiloss后网络最终输出的损失函数可以表示为

$$\mathcal{L}_{\text{Mul}} = - \sum_{l=1}^L \sum_{i=1}^N [x_i \log(\sigma(-c_i^l)) + (1 - x_i) \log(1 - \sigma(-c_i^l))], \quad (3.27)$$

其中 $c_l^i$ 对应第 $l$ 层的输出 $\mathbf{u}^{K,l}$ 的双极型形式，并且我们可以将LPDN网络的每个外层对应的损失函数表示为 $\mathcal{L}_l$ 。考虑损失函数在内的LPDN前向传递过程如算法6所示，需要注意的是我们只在训练阶段考虑损失函数。

---

**算法 6** LPDN前向传递过程
 

---

**输入:** 获取信道信息 $\mathbf{q}$ ，及相应的标签值 $\mathbf{x}$

- 1: 初始化 $\mathbf{z}^{0,1}$ ， $\boldsymbol{\lambda}^0$ 为零向量；初始化训练参数 $\boldsymbol{\alpha}$ ， $\boldsymbol{\beta}$ ， $\boldsymbol{\mu}$
- 2: **for**  $l = 1, 2, \dots, L$  **do**
- 3:     **for**  $k = 1, 2, \dots, K$  **do**
- 4:         根据式(3.25a)更新节点 $\mathbf{u}$
- 5:         根据式(3.25b)更新节点 $\mathbf{z}$
- 6:     **end for**
- 7:     根据式(3.25c)更新节点 $\boldsymbol{\lambda}$
- 8:      $\mathcal{L}_{\text{Mul}} \leftarrow \mathcal{L}_{\text{Mul}} + \mathcal{L}_l$
- 9:      $\mathbf{z}^{0,l+1} \leftarrow \mathbf{z}^{K,l}$
- 10: **end for**

**输出:**  $\mathcal{L}_{\text{Mul}}$

---

在由损失函数得到误差值之后，通过误差的反向传播过程来更新训练参数进而校正误差。反向传播过程主要有两部分组成，首先由链式法则来计算误差关于各参数的梯度值，然后由一定的梯度下降方法来向着梯度下降的方向更新训练参数。本文采用了Adam算法来执行参数更新过程<sup>[58]</sup>，在Adam算法中，动量被直接合并为梯度的一阶矩估计，其次，Adam对一阶矩和二阶矩的估计偏差进行了校正，具有较强的鲁棒性。以惩罚参数 $\boldsymbol{\mu}$ 为例，算法7给出了Adam算法具体的更新流程。在算法7中，学习率 $\epsilon$ 一般设置为0.001， $\rho_1$ 和 $\rho_2$ 分别设置为0.9和0.999， $\delta$ 默认设置为 $10^{-8}$ 。在训练过程中每次从训练集中随机抽取 $m$ 个样本作为一个最小批量（Mini-batch），每次以最小批量为单位计算平均梯度 $\mathbf{g}$ ，其中 $\mathcal{L}_{\text{Mul}}^{(i)}$ 表示一个最小批量中第 $i$ 个样本的损失值。

由于LPDN网络是基于PDD译码算法展开得到，其中每一层的节点的更新过程与PDD译码算法中每次迭代的变量更新过程基本相同。而且引入的层间独立的可训练参数也是由原有的可调参数转换而来，并未引入新的参数和更新过程。所以本节所提的LPDN译码器与所提的PDD译码算法有相同的计算复杂度。



**算法 7 Adam算法**

**输入:** 学习率 $\epsilon$ ; 矩估计的指数衰减率 $\rho_1$ 和 $\rho_2$ ; 用于数值稳定的常数 $\delta$

- 1: 初始化训练参数 $\mu$ ; 初始化一阶和二阶矩变量 $\mathbf{s} = \mathbf{0}$ ,  $\mathbf{r} = \mathbf{0}$ ; 初始化时间步长 $t = 0$
  - 2: **repeat**
  - 3: 计算梯度:  $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\mu} \sum_{i=1}^m \mathcal{L}_{\text{Mul}}^{(i)}$
  - 4:  $t \leftarrow t + 1$
  - 5: 更新有偏一阶矩估计:  $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1) \mathbf{g}$
  - 6: 更新有偏二阶矩估计:  $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2) \mathbf{g} \odot \mathbf{g}$
  - 7: 修正一阶矩的偏差:  $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1 - \rho_1^t}$
  - 8: 修正二阶矩的偏差:  $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1 - \rho_2^t}$
  - 9: 计算更新:  $\Delta \mu = -\epsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}} + \delta}}$
  - 10: 执行变量更新:  $\mu \leftarrow \mu + \Delta \mu$
  - 11: **until** 达到收敛条件
- 输出:** 更新后的变量 $\mu$

### 3.5 仿真结果与性能分析

针对本章所提的PDD译码器和LPDN译码器, 本节通过仿真与横向对比的方式来评估所提译码器的性能, 主要包括了译码器的纠错性能以及译码器的迭代次数。所提的LPDN神经网络在Python平台的Tensorflow框架下完成训练。译码器的仿真在Matlab软件上进行, 考虑AWGN信道, 并采用BPSK调制方式, 所考虑的LDPC码为(128,64) CCSDS LDPC码 $\mathcal{C}_1$ 和(256,128) CCSDS LDPC码 $\mathcal{C}_2$ <sup>[62]</sup>。除此之外, 我们引入了经典的BP译码算法<sup>[18]</sup>, 文献[25]中的ADMM L2译码算法, 以及文献[28]中的PDD译码算法进行译码性能的对比, 上述算法的复杂度已经在本章的3.3.2中进行了分析比较。

考虑到训练数据集的选取对于LPDN网络的训练起到关键作用, 我们要对数据集进行预筛选。首先考虑产生训练数据的信噪比条件, 如果信噪比过高, 那么几乎不存在译码错误的情况, 网络就难以从中有效地获取错误模式等有效信息; 如果信噪比过低, 过度严重的噪声导致很少的码字可以被正确译码, 网络也很难从中学习到有效的译码机制。所以我们将码字 $\mathcal{C}_1$ 和 $\mathcal{C}_2$ 的训练信噪比分别设置为3.5dB和2.5dB。除此之外, 我们筛去了一部分可以被BP译码器轻松译码的码字, 这部分码字可以在较少的迭代次数下完成译码, 所以对于网络的训练也没有很大的参考价值, 筛选后的训练集可以进一步提升网络的学习效率,

加快收敛速度。筛选后的数据被分为一个训练集和一个验证集，大小分别为 $4 \times 10^5$ 和 $10^5$ ，需要注意的是这里的验证集仅用于训练阶段判断网络是否收敛，在仿真测试阶段还是用随机生成的数据。在训练过程中，我们将Adam优化器前10个训练周期的学习率设置为 $10^{-3}$ ，然后在之后的训练周期减少到 $10^{-4}$ ，从而来减少过拟合现象的发生。此外，我们将LPDN网络在训练阶段的内层和外层数目分别设置为5和20。

在仿真过程中，经典的BP译码器和ADMM L2译码器的迭代次数均被设置为 $2 \times 10^4$ 。<sup>2</sup>对于文献[28]中的PDD译码器，相关参数均按照论文中推荐设置，初始惩罚参数 $\mu_0$ 设置为0.5，控制参数 $c$ 设置为1.001。对于本文所提的PDD译码器， $\mu_0$ 和 $c$ 同样分别设置为0.5和1.001，此外将罚函数系数 $\alpha$ 和超松弛参数 $\beta$ 分别设置为1.2和1.5。两种PDD译码器和LPDN译码器的最大内层迭代次数均设为5次。在前20个外层循环中，LPDN译码器加载训练好的参数集 $\{\alpha, \beta, \mu\}$ 进行译码，而在之后的循环中， $\alpha$ 和 $\beta$ 固定设置为1.2和1.5，而惩罚参数 $\mu$ 则参照PDD译码器的增长率继续增加，当译码器满足收敛条件或者达到最大迭代次数 $l_{\max} = 4000$ 时译码结束，如此总的内层迭代次数也等于 $2 \times 10^4$ 。

下面给出了我们的仿真结果，由于计算资源有限，每条曲线的最后两个信噪比下只收集了50个错误帧，在其余各个信噪比下都收集了至少100个错误帧，每一组曲线的信息序列和噪声都基于相同的随机种子生成。

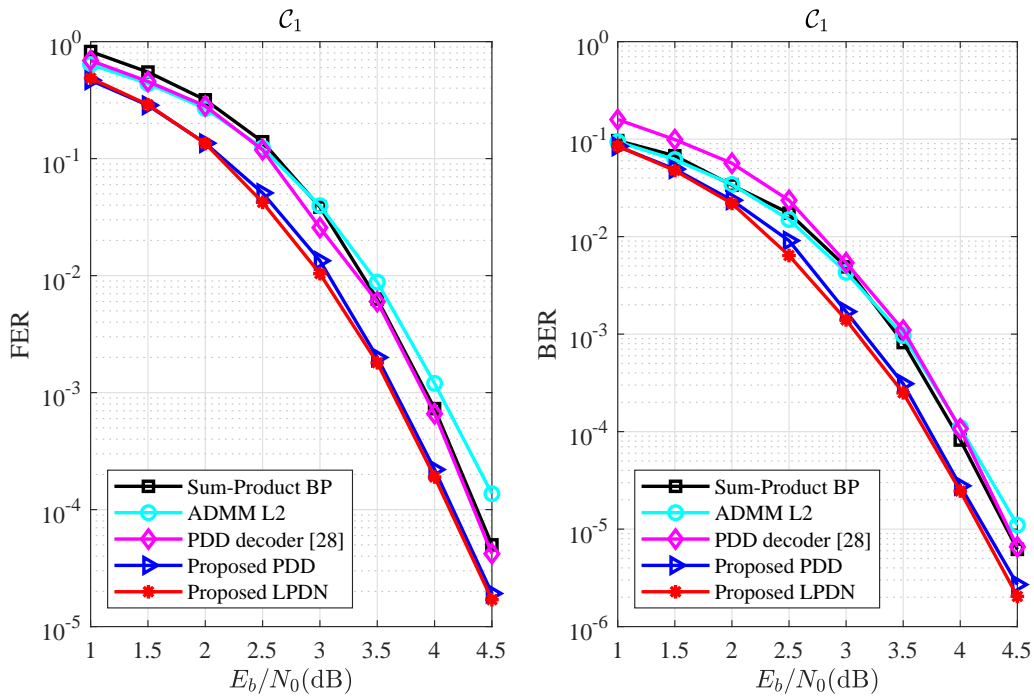


图 3.3  $C_1$ 的FER和BER曲线

<sup>2</sup>本文中，迭代次数被设置为较大的值来观察最终的收敛性能。

在图3.3中，我们提供了码字 $C_1$ 的误帧率（Frame-Error-Rate, FER）和误码率（Bit-Error-Rate, BER）性能曲线。从中可以看出LPDN译码器的性能在中高信噪比区域略好于所提的PDD译码器，两者的总体性能接近，这是因为LPDN译码器采用的模型本就是基于所提的PDD译码器，所以两者本质上是等效的，只是LPDN优化了参数配置。而我们所提的LPDN译码器和PDD译码器在各个信噪比的性能均优于其他几种译码器，当FER为 $10^{-3}$ 时，相比经典的BP译码器和文献[28]中的PDD译码器有大约0.2dB的增益，相比ADMM L2译码器有0.3dB左右的增益。

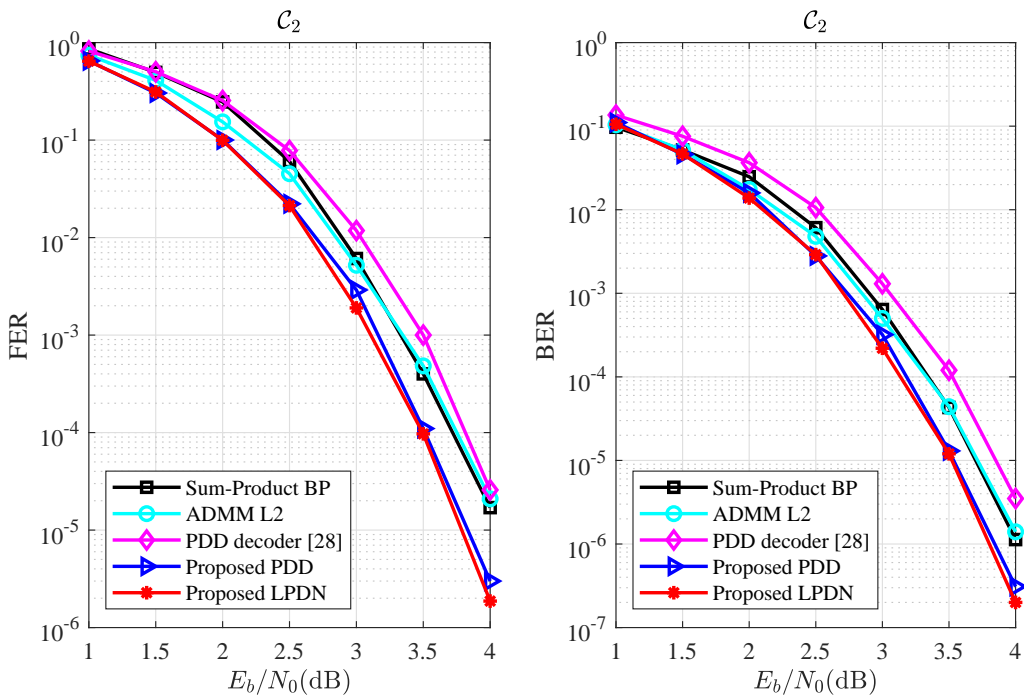


图 3.4  $C_2$ 的FER和BER曲线

在图3.4中，我们提供了码字 $C_2$ 的FER和BER性能曲线。此时所提的LPDN译码器和PDD译码器性能仍然相近，并且在各信噪比下要优于其他几种译码器，当FER为 $10^{-3}$ 时，相比经典的SPA译码器和ADMM L2译码器有大约0.2dB的增益，相比文献[28]中的PDD译码器有0.4dB左右的增益。

图3.5提供了码字 $C_1$ 下不同译码器达到图3.3中性能所需要的平均迭代次数和平均译码运行时间。由于在3.3.2中我们已经分析了这些译码器单次迭代的复杂度，并且这里比较的都是双层循环译码器，所以我们这里考虑内层迭代次数，译码运行时间考虑完成一帧译码所需的程序运行时间。从图中可以看出，平均迭代次数与平均运行时间具有相同的趋势，对于码字 $C_1$ ，本文所提的PDD译码器和文献[28]中所提的PDD译码器所需的迭代次数和运行时间相近，而LPDN译码器相比这两者可以减少大约50%的迭代次数和运行时间。

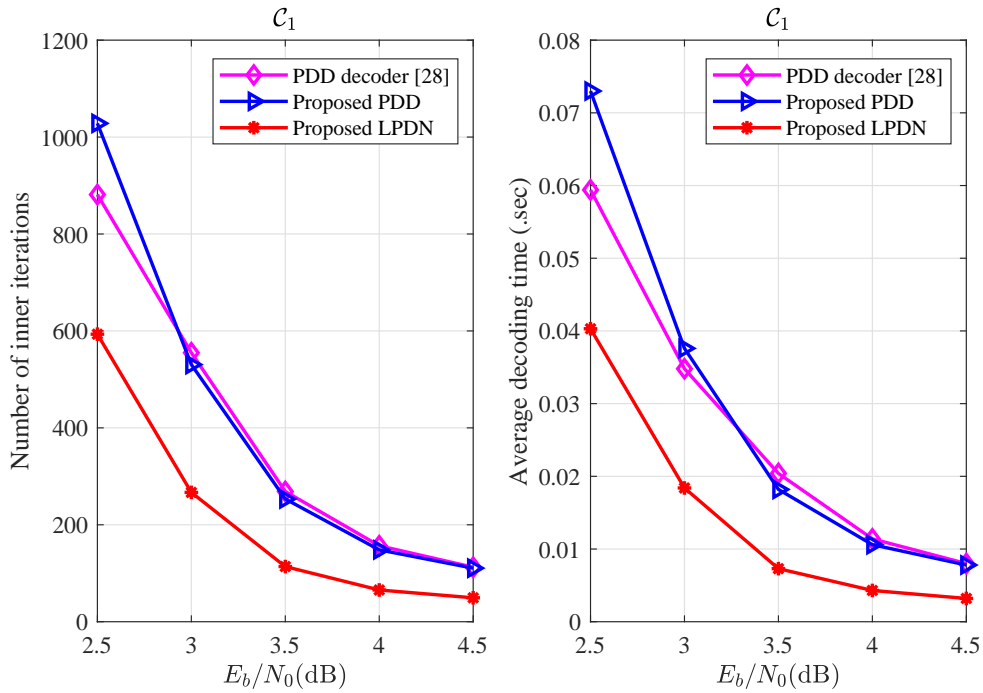


图 3.5  $C_1$ 的平均迭代次数和译码运行时间曲线

图3.6提供了码字 $C_2$ 下不同译码器达到图3.4中性能所需要的平均迭代次数和平均运行时间。对于码字 $C_2$ ，LPDN译码器相比本文所提的PDD译码器减少35%左右的迭代次数，相比文献[28]中所提的PDD译码器减少了将近70%的迭代次数。这说明本文所提的PDD译码器相比文献[28]中的PDD译码器译码效率更高，可以以更小的计算复杂度和更少的迭代次数获得更优的纠错性能，而本文所提的LPDN译码器在所提的PDD译码器的基础上进一步优化了网络参数，可以在减少迭代次数的同时获得更优的纠错性能。

综上所述，在相同的最大迭代次数条件下，本章所提的PDD译码器和LPDN译码器相比经典的BP译码器，ADMM L2译码器以及文献[28]中的PDD译码器均具有更低的误帧率，提升较为明显，而且由3.3.2可知本章所提的译码器在单次迭代中的复杂度也更有优势，所以译码效率也更高。尽管LPDN译码器在译码性能上相比所提的PDD译码器提升有限，但是可以很大程度上减少译码所需的平均迭代次数，说明模型驱动的神经网络确实可以优化参数分布，得到译码性能更优的译码器，从而进一步提升译码效率。

### 3.6 本章小结

本章主要研究了基于LP问题的PDD译码算法，其目的是弥补BP译码算法在错误平层和理论保证等方面存在的不足同时提升LP译码方法的性能。首先，本章基于级联整数规划

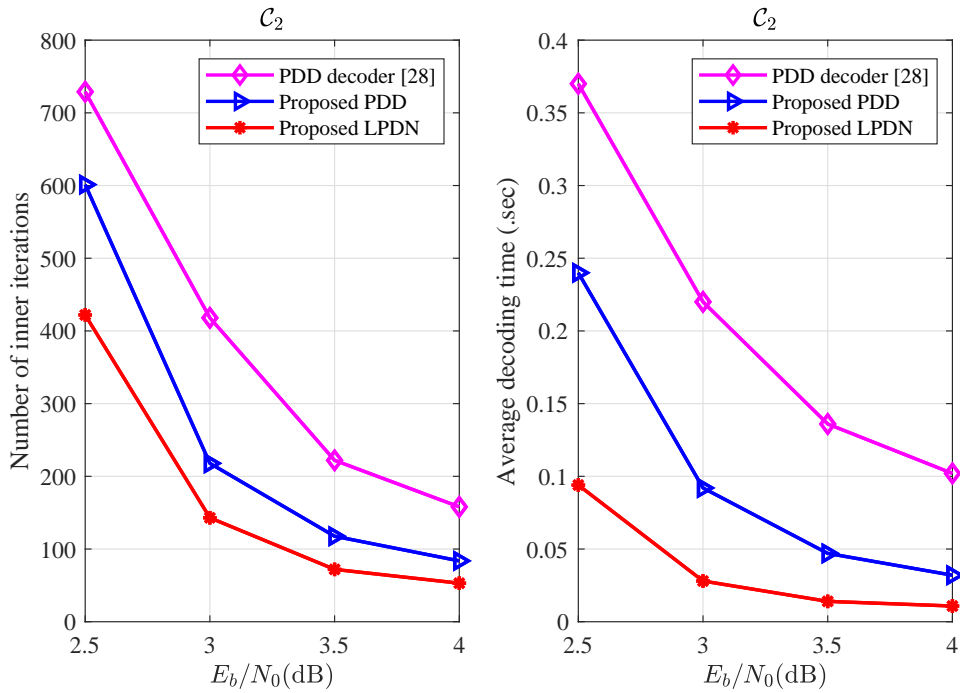


图 3.6  $C_2$ 的平均迭代次数和译码运行时间曲线

问题，提出了一种新的PDD译码算法，并且对所提的译码算法和其他先进的译码算法进行了复杂度分析与比较。此外，我们结合了深度学习中基于模型驱动的深度展开技术，进一步提出了LPDN译码器来提升PDD译码器的译码性能和效率。最后通过仿真结果验证了所提的PDD译码器和LPDN译码器可以提供更优的纠错性能和更低的计算复杂度，并且LPDN译码器在PDD译码器的基础上进一步提升了译码性能并降低了所需的迭代次数。

本章的相关研究成果已发表于SCI期刊IEEE COMMUNICATIONS LETTERS。



## 第四章 ISI信道下的LDPC码的PDD译码技术

### 4.1 引言

ISI信道是一种常用于通信系统和磁记录系统中的实用信道模型，通常采用均衡技术结合信道编码技术来联合抑制码间干扰，从而实现正确的检测译码。Turbo均衡是一种经典的迭代均衡和译码技术，其通过均衡算法和译码算法交换外部软信息进行迭代更新，可以很好地抑制码间干扰。常用的均衡算法有软输出的BCJR算法和SOVA算法，译码可以采用LDPC码中基于软判决的BP译码算法，然而BCJR算法和SOVA算法的复杂度均关于信道记忆长度呈指数增长，在信道记忆长度较长时可能引入较高的延迟。一些现有的研究将基于无记忆对称信道的LP译码方法的思想推广到ISI信道中，用优化理论的方法来实现ISI信道下LDPC码的高效译码，可以获得更优的纠错性能和更低的计算复杂度。

受此启发，针对ISI信道下的LDPC码QP译码问题，本章研究了基于校验多面体约束的ISI-PDD译码算法，并且对译码算法中的校验多面体投影算法进行了提升改进，引入了缩放因子来加速投影算法的收敛。

本章剩余部分安排如下：首先介绍了ISI信道传输模型，以及该信道下LDPC码ML译码问题的QP描述。然后基于PDD的算法框架提出了一种基于校验多面体约束的ISI-PDD译码算法，内层对AL问题近似求解，外层更新对偶变量和参数。同时我们提出了一种基于迭代的FCPP算法来提高校验多面体投影的效率，通过引入可动态调节的缩放因子来减少所需的迭代次数。最后对所提的算法和经典的方法进行了复杂度分析与比较，并基于不同的信道和码字对这些算法的仿真结果进行了性能分析和对比。

### 4.2 优化问题描述

本章考虑 $(N, K)$ 二进制LDPC码 $\mathcal{C}$ ，由维度为 $M \times N$ 的校验矩阵 $\mathbf{H}$ 定义。令集合 $\mathcal{I} \triangleq \{1, 2, \dots, N\}$ 和 $\mathcal{J} \triangleq \{1, 2, \dots, M\}$ 分别表示变量节点和校验节点的索引集合，令 $d_j$ ， $j \in \mathcal{J}$ 表示校验节点 $j$ 的度。考虑如图4.1所示的ISI信道模型，将消息序列表示为 $\mathbf{m} \in \{0, 1\}^K$ ，编码后的码字序列为 $\mathbf{x} \in \{0, 1\}^N$ ，经过BPSK调制后得到发送序列 $\mathbf{d}$ ，其中 $d_i = 1 - 2x_i \in \{-1, 1\}$ ；记忆长度为 $T$ 的ISI信道可以建模为一个具有加性高斯白噪声的离散时间有限脉冲

响应序列 $(h_0, h_1, \dots, h_T) \in \mathbb{R}^{T+1}$ ，在图中表示为抽头延迟电路，其中

$$y_i = \sum_{t=0}^T h_t(1 - 2x_{i-t}), \forall i \in \mathcal{I}. \quad (4.1)$$

噪声序列 $\mathbf{e}$ 满足均值为0，方差为 $\sigma^2$ 的高斯分布，接收到的序列 $\mathbf{r} \in \mathbb{R}^N$ 可以表示为 $r_i = y_i + e_i$ ， $i \in \mathcal{I}$ 。考虑到边界计算，我们设置码字 $\mathbf{x}$ 和接收符号 $\mathbf{r}$ 额外的边界值为

$$\begin{aligned} x_{-(T-1)} &= \dots = x_{-1} = x_0 = 0.5, \\ x_{N+1} &= x_{N+2} = \dots = x_{N+T} = 0.5, \\ r_{N+1} &= r_{N+2} = \dots = r_{N+T} = 0, \end{aligned} \quad (4.2)$$

从而涉及边界计算的 $r$ 和 $1 - 2x$ 均为0，消除了对整体结果的影响。

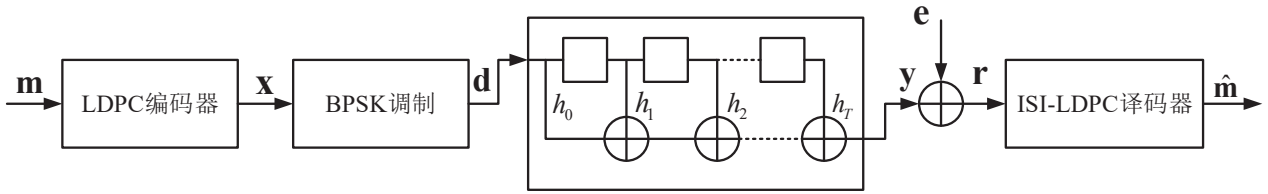


图 4.1 ISI信道传输模型

基于最大似然译码准则，ISI信道下的LDPC码译码问题可以等价表述成如下的最大似然译码问题<sup>[32]</sup>：

$$\begin{aligned} \min & \|\mathbf{r} - \mathbf{y}\|_2^2 \\ \text{s.t. } & \mathbf{x} \in \mathcal{C}. \end{aligned} \quad (4.3)$$

其中约束条件 $\mathbf{x} \in \mathcal{C}$ 可以等价表述为LDPC码的奇偶校验约束条件，即对码字 $\mathbf{x} \in \{0, 1\}^N$ ，有 $\mathbf{P}_j \mathbf{x} \in \mathbb{P}_{d_j}$ ， $j \in \mathcal{J}$ ，其中维度为 $d_j \times N$ 的二进制矩阵 $\mathbf{P}_j$ 是一个选择矩阵，用于选择码字 $\mathbf{x}$ 中参与到校验矩阵第 $j$ 行校验式中的 $d_j$ 个元素。 $\mathbf{P}_j$ 的每一行有且只有一个非零元素，非零元素的位置与校验节点 $j$ 的邻居变量节点的索引相对应，假设校验矩阵 $\mathbf{H}$ 的第 $j$ 行为 $(1, 0, 0, 1, 1, 0)$ ，则对应的 $\mathbf{P}_j$ 可以表示为

$$\mathbf{P}_j = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (4.4)$$

由于LDPC码的校验式均需要满足偶约束，则 $\mathbb{P}_{d_j}$ 可以定义为

$$\mathbb{P}_{d_j} \triangleq \{\mathbf{c} \in \{0, 1\}^{d_j} \mid \|\mathbf{c}\|_1 \text{ is even}\}, \quad (4.5)$$



$\mathbb{P}_{d_j}$ 也是长度为 $d_j$ 的单奇偶校验码 (Single Parity-Check Code) 的码书。文献[13]证明了求解原最大似然问题等价于在码字多面体上对问题进行求解, 从而可以将原奇偶约束条件 $\mathbf{P}_j \mathbf{x} \in \mathbb{P}_{d_j}$ 松弛为 $\mathbf{P}_j \mathbf{x} \in \mathbb{PP}_{d_j}$ , 其中

$$\mathbb{PP}_{d_j} \triangleq \text{conv}(\mathbb{P}_{d_j}) = \text{conv}(\{\mathbf{c} \in \{0, 1\}^{d_j} \mid \|\mathbf{c}\|_1 \text{ is even}\}), \quad (4.6)$$

$\mathbb{PP}_{d_j}$ 被称为校验多面体 (Check Polytope), 表示维度为 $d_j$ 的单奇偶校验码的码字凸包, 所以对于码书 $\mathcal{C}$ 中的任意码字,  $\mathbf{P}_j \mathbf{x}$ 都位于 $\mathbb{PP}_{d_j}$ 的顶点位置。

将ML问题(4.3)的约束条件替换为松弛后的校验多面体约束, 同时将式(4.1)带入ML问题的目标函数中并将其展开, 可以得到与ML问题(4.3)等价的QP问题:

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) &= \sum_{i=1}^N \left( r_i - \sum_{t=0}^T h_t (1 - 2x_{i-t}) \right)^2 \\ \text{s.t. } \mathbf{P}_j \mathbf{x} &\in \mathbb{PP}_{d_j}, \forall j \in \mathcal{J}. \end{aligned} \quad (4.7)$$

### 4.3 基于QP问题的ISI-PDD译码算法设计

本节我们提出了一种用于求解ISI信道下的LDPC码QP问题的PDD译码算法, 采用LP松弛和罚函数的方法来处理校验多面体的约束条件, 同时引入了超松弛的方法来提升译码器的收敛速度。ISI信道下的PDD译码算法同样包含内外两层循环, 内循环对AL问题进行分布式求解, 外循环进行对偶变量和惩罚参数的更新。

为了使QP问题(4.7)契合PDD算法的框架, 我们首先引入辅助变量 $\mathbf{z} \triangleq (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_M)$ , 其中对任意 $j \in \mathcal{J}$ 都有 $\mathbf{z}_j \in \mathbb{R}^{d_j}$ 。除此之外, 为了使问题可以在确定的多项式时间内求解, 我们还要对离散的码字约束进行处理。参考3.3.1的分析, 本章我们同样采用LP松弛和罚函数的方式, 将码字 $\mathbf{x}$ 松弛为有界的连续约束 $\mathbf{x} \in [0, 1]^N$ , 同时在目标函数中引入罚函数 $g(\mathbf{x})$ 来增加伪码字的代价, 进而我们可以得到QP问题(4.7)的惩罚松弛表示:

$$\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}) \quad (4.8a)$$

$$\text{s.t. } \mathbf{P}_j \mathbf{x} = \mathbf{z}_j, \forall j \in \mathcal{J}, \quad (4.8b)$$

$$\mathbf{z}_j \in \mathbb{PP}_{d_j}, \forall j \in \mathcal{J}, \quad (4.8c)$$

$$\mathbf{x} \in [0, 1]^N. \quad (4.8d)$$

其中罚函数 $g(\mathbf{x})$ 同样需要满足3.3.1中所列举的四项条件, 避免伪码字出现的同时确保变量的更新存在闭式解, 从而可以有效地用分布式算法对问题进行求解。在本章中, 我们选择 $l_2$ 形式的罚函数, 令 $g(\mathbf{x}) = -\alpha \|\mathbf{x} - 0.5\|_2^2 = \sum_{i \in \mathcal{I}} -\alpha (x_i - 0.5)^2$ , 其中 $\alpha$ 是罚函数系数。

接下来，我们采用增广拉格朗日方法，将问题(4.8)的目标函数和等式约束条件所对应的AL函数定义为

$$L_\mu(\mathbf{x}, \mathbf{z}) \triangleq f(\mathbf{x}) + g(\mathbf{x}) + \frac{\mu}{2} \sum_{j=1}^M \left\| \mathbf{P}_j \mathbf{x} - \mathbf{z}_j + \frac{\boldsymbol{\lambda}_j}{\mu} \right\|_2^2, \quad (4.9)$$

其中 $\mu \in \mathbb{R}^+$ 为拉格朗日函数的惩罚参数，与约束条件(4.8b)相关的对偶变量定义为 $\boldsymbol{\lambda} \triangleq (\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_M)$ ，对于任意 $j \in \mathcal{J}$ 都有 $\boldsymbol{\lambda}_j \in \mathbb{R}^{d_j}$ 。进一步地，我们可以将AL问题表示为

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}} L_\mu(\mathbf{x}, \mathbf{z}) \\ \text{s.t. } \mathbf{z}_j \in \mathbb{PP}_{d_j}, \forall j \in \mathcal{J}, \\ \mathbf{x} \in [0, 1]^N. \end{aligned} \quad (4.10)$$

本章所提的ISI信道下的PDD译码算法采用了双层循环的结构，内层循环使用BCD算法对AL问题进行近似求解，在外层循环中更新对偶变量 $\boldsymbol{\lambda}$ 和惩罚参数 $\mu$ 。令 $K$ 和 $L$ 分别表示内层循环和外层循环的最大迭代次数， $k$ 和 $l$ 分别表示内层循环和外层循环的索引，对于内层循环，我们将原始问题分成子问题 $\mathbf{x}$ 和子问题 $\mathbf{z}$ 两部分进行求解，其迭代过程包括以下两个步骤：

(1) 在给定 $\mathbf{z}^k$ 的情况下更新 $\mathbf{x}^{k+1}$ ：

此时 $\mathbf{z}^k$ 和 $\boldsymbol{\lambda}^l$ 可以看作常量，针对子问题 $\mathbf{x}$ 的求解可以看作一个二次优化问题：

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}) + \frac{\mu}{2} \sum_{j=1}^M \left\| \mathbf{P}_j \mathbf{x} - \mathbf{z}_j^k + \frac{\boldsymbol{\lambda}_j^l}{\mu} \right\|_2^2 \\ \text{s.t. } \mathbf{x} \in [0, 1]^N. \end{aligned} \quad (4.11)$$

根据二次优化问题的一阶最优条件，取问题(4.11)的目标函数的一阶偏导数并让其为零，可以得到如下的矩阵方程：

$$\frac{\partial f}{\partial \mathbf{x}} - 2\alpha \mathbf{x} + \alpha + \mu \sum_{j=1}^M \left( \mathbf{P}_j^T \mathbf{P}_j \mathbf{x} - \mathbf{P}_j^T \mathbf{z}_j^k + \frac{\mathbf{P}_j^T \boldsymbol{\lambda}_j^l}{\mu} \right) = \mathbf{0}_N, \quad (4.12)$$

其中 $\frac{\partial f}{\partial \mathbf{x}} = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_N} \right)^T$ 表示实值标量函数 $f(\mathbf{x})$ 的偏导向量，也称为梯度向量，根据问题(4.7)推导出梯度向量中的偏导表示：

$$\begin{aligned} \frac{\partial f}{\partial x_i} &= \frac{\partial}{\partial x_i} \sum_{i=1}^N \left( r_i - \sum_{t=0}^T h_t (1 - 2x_{i-t}) \right)^2 \\ &= 4 \sum_{n=i}^{i+T} h_{n-i} \left( r_n - \sum_{t=0}^T h_t (1 - 2x_{n-t}) \right) \\ &= \left( 8 \sum_{t=0}^T h_t^2 \right) x_i + 8 \sum_{m=0}^T \sum_{t \in \mathcal{T} \setminus m} h_m h_t x_{i+m-t} + \sigma_i, \forall i \in \mathcal{I}. \end{aligned} \quad (4.13)$$

其中 $\mathcal{T} \triangleq \{0, 1, \dots, T\}$ ,  $\mathcal{T} \setminus m$ 表示 $\mathcal{T}$ 中不包含 $m$ 的子集,  $\sigma_i$ 是一个不参与迭代更新的常量, 可以表示为

$$\sigma_i = 4 \sum_{m=0}^T h_m r_{i+m} - 4 \left( \sum_{t=0}^T h_t \right)^2. \quad (4.14)$$

令 $\mathbf{P} = \sum_{j=1}^M \mathbf{P}_j^T \mathbf{P}_j$ , 根据 $\mathbf{P}_j$ 的定义, 可以得到 $\mathbf{P}$ 是一个维度为 $N \times N$ 的对角矩阵, 用 $|\cdot|$ 表示集合中元素的数目, 则 $\mathbf{P}$ 中坐标为 $(i, i)$ 的元素为 $|\mathcal{N}_v(i)|$ , 也就是变量节点 $i$ 的度。  $\mathbf{P}_j^T \mathbf{z}_j^k$ 计算展开后可以得到一个长为 $N$ 的向量, 将 $\mathbf{z}_j^k$ 中的 $d_j$ 个元素分别放置于该向量中索引 $i \in \mathcal{N}_c(j)$ 的位置上, 对于 $i \notin \mathcal{N}_c(j)$ 的位置填0, 从而将 $\mathbf{P}_j^T \mathbf{z}_j^k$ 中的第 $i$ 个元素表示为 $z_j^{(i)}$ 。 同样的,  $\mathbf{P}_j^T \boldsymbol{\lambda}_j^l$ 中 $i \in \mathcal{N}_c(j)$ 的元素表示为 $\lambda_j^{(i)}$ , 对于 $i \notin \mathcal{N}_c(j)$ 的位置填0。 从而对于任意 $i \in \mathcal{I}$ , 方程(4.12)可以等价表示为

$$\frac{\partial f}{\partial x_i} - 2\alpha x_i + \alpha + \mu |\mathcal{N}_v(i)| x_i - \mu \sum_{j \in \mathcal{N}_v(i)} \left( z_j^{(i)} - \frac{\lambda_j^{(i)}}{\mu} \right) = 0. \quad (4.15)$$

根据(4.13)可以知道, 每一个 $x_i$ 对应的偏导数计算结果与其相邻的 $2T$ 个元素 $\{x_{i \pm t} \mid 0 \leq t \leq T\}$ 有关, 从而难以从(4.15)直接得到 $x_i$ 简洁的闭式解。 根据文献[36], 在第 $k+1$ 次迭代时的 $x_i$ 求解可以近似通过第 $k$ 次迭代中的 $x_{i \pm t}^k$ 来得到, 考虑到PDD算法随着迭代次数的增加而逐步收敛, 所以在当前迭代中使用上一次迭代的计算结果是合理的。 进而我们可以将(4.13)中 $\frac{\partial f}{\partial x_i}$ 在第 $k+1$ 次迭代中的表达式写作

$$\frac{\partial f}{\partial x_i} = \left( 8 \sum_{t=0}^T h_t^2 \right) x_i^{k+1} + \delta_i + \sigma_i, \quad (4.16)$$

其中 $\delta_i$ 是由上一次迭代的结果计算得到, 可以表示为

$$\delta_i = 8 \sum_{m=0}^T \sum_{t \in \mathcal{T} \setminus m} h_m h_t x_{i+m-t}^k. \quad (4.17)$$

综上, 将(4.16)代入(4.15)后可以得到问题(4.11)的最优解:

$$x_i^{k+1} = \Pi_{[0,1]} \left( \frac{\zeta_i - \alpha}{\omega_i - 2\alpha} \right), \quad \forall i \in \mathcal{I}, \quad (4.18)$$

其中 $\omega_i = \mu |\mathcal{N}_v(i)| + 8 \sum_{t=0}^T h_t^2$ ,  $\zeta_i = \mu \sum_{j \in \mathcal{N}_v(i)} \left( z_j^{(i)} - \lambda_j^{(i)} / \mu \right) - \delta_i - \sigma_i$ ,  $\delta_i$ 和 $\sigma_i$ 的定义分别与(4.17)和(4.14)中相同。

(2) 在给定 $\mathbf{x}^{k+1}$ 的情况下更新 $\mathbf{z}^{k+1}$ :

此时 $\mathbf{x}^{k+1}$ 和 $\boldsymbol{\lambda}^l$ 可以视为常量, 子问题 $\mathbf{z}$ 同样可以化简为一个二次优化问题, 表示为

$$\begin{aligned} \min_{\mathbf{z}} \sum_{j=1}^M \left\| \mathbf{P}_j \mathbf{x}^{k+1} - \mathbf{z}_j + \frac{\boldsymbol{\lambda}_j^l}{\mu} \right\|_2^2 \\ \text{s.t. } \mathbf{z}_j \in \mathbb{PP}_{d_j}, \quad \forall j \in \mathcal{J}. \end{aligned} \quad (4.19)$$

根据二次优化问题的一阶最优条件，问题(4.19)的最优解可以表示为

$$\mathbf{z}_j^{k+1} = \Pi_{\mathbb{PP}_{d_j}} \left( \mathbf{P}_j \mathbf{x}^{k+1} + \frac{\boldsymbol{\lambda}_j^l}{\mu} \right), \forall j \in \mathcal{J}, \quad (4.20)$$

其中 $\Pi_{\mathbb{PP}_{d_j}}(\cdot)$ 为校验多面体投影操作，表示括号中的向量在校验多面体 $\mathbb{PP}_{d_j}$ 上的投影，这也是在整个PDD译码算法中最耗时的部分，将在本章的下一节中详细介绍。跟第三章类似地，我们采用超松弛机制来加快PDD算法的收敛速度，将(4.20)中的 $\mathbf{P}_j \mathbf{x}^{k+1}$ 替换为 $\beta \mathbf{P}_j \mathbf{x}^{k+1} + (1 - \beta) \mathbf{z}_j^k$ ，从而子问题 $\mathbf{z}$ 的最优解又可以表示为

$$\mathbf{z}_j^{k+1} = \Pi_{\mathbb{PP}_{d_j}} \left( \beta \mathbf{P}_j \mathbf{x}^{k+1} + (1 - \beta) \mathbf{z}_j^k + \frac{\boldsymbol{\lambda}_j^l}{\mu} \right), \forall j \in \mathcal{J}, \quad (4.21)$$

其中 $\beta \in (1, 2)$ 为超松弛参数，可以控制松弛的程度，从而做到性能和收敛速度的折中。

在外层循环中，我们进行对偶变量 $\boldsymbol{\lambda}$ 和惩罚参数 $\mu$ 的更新。第 $l$ 次外层循环中的对偶变量更新过程可以表示为

$$\boldsymbol{\lambda}_j^{l+1} = \boldsymbol{\lambda}_j^l + \mu_l (\mathbf{P}_j \mathbf{x}^K - \mathbf{z}_j^K), \forall j \in \mathcal{J}, \quad (4.22)$$

其中 $\mathbf{x}^K$ 和 $\mathbf{z}_j^K$ 为内循环的输出。同时，惩罚参数根据 $\mu_{l+1} = c\mu_l$ 进行更新，其中 $c > 1$ 是一个控制参数，其可以使得 $\mu$ 在每次外层迭代中按照一定的增长率增长，从而保证对偶函数上升。

ISI信道下的PDD算法的收敛条件同样可以设置为初始残差和对偶残差足够小，即

$$\begin{aligned} \sum_{j=1}^M \|\mathbf{P}_j \mathbf{x}^{k+1} - \mathbf{z}_j^{k+1}\|_2^2 &< \varepsilon_{ini}, \\ \sum_{j=1}^M \|\mathbf{z}_j^{k+1} - \mathbf{z}_j^k\| &< \varepsilon_{dual}, \end{aligned} \quad (4.23)$$

其中 $\varepsilon_{ini} > 0$ 和 $\varepsilon_{dual} > 0$ 分别表示算法对初始残差和对偶残差的容忍度，一般设置为小于 $10^{-5}$ 的数。综上所述，本章所提的基于ISI信道下QP问题的ISI-PDD译码算法的总体流程如算法8所示。

#### 4.4 FCPP算法设计

校验多面体投影算法是整个ISI-PDD译码算法中最复杂耗时的部分，高效的投影算法可以有效降低译码算法的整体复杂度。本节首先回顾了校验多面体的定义，并介绍了校验多面体的几何结构，然后提出了一种基于迭代思想的快速校验多面体投影（Fast Check Polytope Projection, FCPP）算法。

**算法 8** 基于ISI信道下QP问题的ISI-PDD译码算法

输入：获取接收序列 $\mathbf{r}$

- 1: 初始化 $\mathbf{x}^0$ 为全零向量, 对任意 $j \in \mathcal{J}$ , 初始化 $\mathbf{z}_j^0$ 中的元素均为0.5,  $\lambda_j^0$ 为全零向量; 设置合适的参数 $\alpha, \beta, \mu_0$ 和 $c$ ; 令 $k = 0, l = 0$
- 2: **repeat**
- 3:     **repeat**
- 4:         根据式(4.18)更新变量 $\mathbf{x}$
- 5:         根据式(4.21)更新变量 $\mathbf{z}$
- 6:          $k \leftarrow k + 1$
- 7:     **until** 达到最大迭代次数 $K$ 或者达到收敛条件(4.23)
- 8:     根据式(4.22)更新变量 $\lambda$
- 9:      $\mu_{l+1} = c\mu_l$
- 10:      $\mathbf{x}^0 \leftarrow \mathbf{x}^K, \mathbf{z}^0 \leftarrow \mathbf{z}^K$
- 11:      $l \leftarrow l + 1, k \leftarrow 0$
- 12: **until** 达到最大迭代次数 $L$ 或者达到收敛条件(4.23)

输出：硬判后输出码字 $\mathbf{x} \in \{0, 1\}^N$

**4.4.1 校验多面体的几何结构**

在分析几何结构前首先回顾在4.3中给出的校验多面体的定义：

$$\begin{aligned} \mathbb{P}_{d_j} &\triangleq \{\mathbf{c} \in \{0, 1\}^{d_j} \mid \|\mathbf{c}\|_1 \text{ is even}\}, \\ \mathbb{PP}_{d_j} &\triangleq \text{conv}(\mathbb{P}_{d_j}) = \text{conv}(\{\mathbf{c} \in \{0, 1\}^{d_j} \mid \|\mathbf{c}\|_1 \text{ is even}\}), \end{aligned} \quad (4.24)$$

其中 $d_j$ 为校验节点 $j$ 的度。一种常规的判断向量 $\mathbf{v}$ 是否属于校验多面体的表达方式是令 $\sum_i \alpha_i = 1, \alpha_i \geq 0$ , 当且仅当 $\mathbf{c}_i \in \mathbb{P}_{d_j}$ 时, 向量 $\mathbf{v} = \sum_i \alpha_i \mathbf{c}_i \in \mathbb{PP}_{d_j}$ 。这种描述方式非常直观, 但是并没有涉及多面体的几何结构, 不利于校验多面体投影算法的设计。

文献[22]中清晰阐述了校验多面体的几何结构, 图4.2给出了 $d_j = 5$ 时的校验多面体几何结构作为示例。首先定义 $\mathbb{P}_{d_j}$ 中汉明重量为 $r$ 的子集 $\mathbb{P}_{d_j}^r$ 及其凸包 $\mathbb{PP}_{d_j}^r$ ：

$$\begin{aligned} \mathbb{P}_{d_j}^r &\triangleq \{\mathbf{c} \in \{0, 1\}^{d_j} \mid \|\mathbf{c}\|_1 = r\}, \\ \mathbb{PP}_{d_j}^r &\triangleq \text{conv}(\mathbb{P}_{d_j}^r), \end{aligned} \quad (4.25)$$

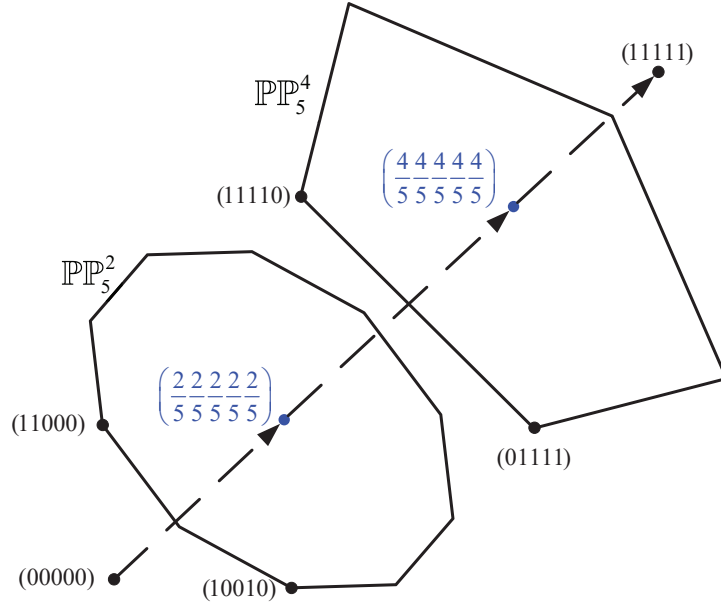


图 4.2 校验多面体几何结构( $d_j = 5$ )

其中 $\mathbb{P}\mathbb{P}_{d_j}^r$ 为单码字向量排列组合的凸包，所以又可以称为“排列多面体”，并且有 $\mathbb{P}\mathbb{P}_{d_j} = \text{conv}(\cup_{0 \leq r \leq d_j: r \text{ even}} \mathbb{P}\mathbb{P}_{d_j}^r)$ 。进一步地，定义所有元素和为 $r$ 的向量组成的仿射超平面为

$$\mathcal{H}_{d_j}^r \triangleq \{\mathbf{x} \in \mathbb{R}^{d_j} \mid \mathbf{1}_{d_j}^T \mathbf{x} = r\}, \quad (4.26)$$

其中 $\mathbf{1}_{d_j}$ 是长度为 $d_j$ 的全一向量，从而 $\mathbb{P}\mathbb{P}_{d_j}^r$ 又可以表示为仿射超平面与校验多面体 $\mathbb{P}\mathbb{P}_{d_j}$ 的交集：

$$\mathbb{P}\mathbb{P}_{d_j}^r = \mathbb{P}\mathbb{P}_{d_j} \cap \mathcal{H}_{d_j}^r, \quad (4.27)$$

其可以看作一个通过校验多面体的“切片”（Slice），所有切片 $\mathbb{P}\mathbb{P}_{d_j}^r$ 均与向量 $\mathbf{1}_{d_j}$ 正交。

进一步地，文献[22]提出了一种校验多面体中向量的“双切片”（Two-slice）表达方式，即任何给定的向量都可以表示为汉明重量为 $r$ 和 $r + 2$ 的两个二进制向量的凸组合，比如 $\mathbf{v} \in \mathbb{P}\mathbb{P}_{d_j}$ 可以表示为

$$\mathbf{v} = \alpha \mathbf{v}_r + (1 - \alpha) \mathbf{v}_{r+2}, \quad (4.28)$$

其中 $\mathbf{v}_r \in \mathbb{P}\mathbb{P}_{d_j}^r$ ， $\mathbf{v}_{r+2} \in \mathbb{P}\mathbb{P}_{d_j}^{r+2}$ ， $0 \leq \alpha \leq 1$ ， $r = \lfloor \|\mathbf{v}\|_1 \rfloor_{\text{even}}$ ， $\lfloor a \rfloor_{\text{even}}$ 表示小于等于 $a$ 的最大偶数。

#### 4.4.2 FCPP算法

上一小节给出了校验多面体的定义并介绍了其几何结构，可以知道校验多面体 $\mathbb{P}\mathbb{P}_{d_j}$ 被包含于超立方体 $[0, 1]^{d_j}$ 中，将一个向量投影到 $[0, 1]^{d_j}$ 中是非常简洁明了的，即

向量中大于1的元素设为1，小于0的元素设为0，其余元素保持不变。但是 $\mathbb{PP}_{d_j}$ 的投影从几何结构上分析并不直观，本小节将借助超立方体投影以及辅助平面来完成校验多面体的投影。对于向量 $\mathbf{v} \in \mathbb{R}^{d_j}$ ，超立方体投影和校验多面体投影可以分别表示为 $\mathbf{u} = \Pi_{[0,1]^{d_j}}(\mathbf{v})$ 和 $\mathbf{z} = \Pi_{\mathbb{PP}_{d_j}}(\mathbf{v})$ 。

基于文献[23]和[24]，校验多面体投影算法的首要步骤是先找到辅助超平面（Assistant Hyperplane）来判断向量是否在校验多面体内。定义辅助超平面 $\boldsymbol{\theta}^T \mathbf{x} = p$ ，对于超立方体内的点 $\mathbf{u} \in [0,1]^{d_j}$ ，如果 $\boldsymbol{\theta}^T \mathbf{u} > p$ ，则 $\mathbf{u} \notin \mathbb{PP}_{d_j}$ ，并且将这种情况称作 $\mathbf{u}$ 的“割”（Cut）；相反的，如果 $\boldsymbol{\theta}^T \mathbf{u} \leq p$ ，则说明 $\mathbf{u} \in \mathbb{PP}_{d_j}$ 。辅助超平面的获取可以参考文献[23]中的割搜索算法：首先初始化超平面的指示向量 $\boldsymbol{\theta}$ 为

$$\theta_i = \text{sgn}(u_i - 0.5) = \begin{cases} 1, & \text{if } u_i > 0.5, \\ -1, & \text{otherwise.} \end{cases} \quad (4.29)$$

然后判断 $\boldsymbol{\theta}$ 中1的个数 $t$ ，如果 $t$ 是偶数，则翻转最接近0.5的 $u_i$ 对应的 $\theta_i$ ，然后更新 $t$ ，从而确保 $\boldsymbol{\theta}$ 中1的个数为奇数，令 $p = t - 1$ 。值得注意的是根据超立方体投影和指示向量 $\boldsymbol{\theta}$ 的定义， $\boldsymbol{\theta}$ 也可以直接由 $\mathbf{v}$ 得到。

对于 $\boldsymbol{\theta}^T \mathbf{u} \leq p$ 的情况，超立方体投影等价于校验多面体投影，不需要额外的操作；而对于 $\boldsymbol{\theta}^T \mathbf{u} > p$ 的“割”情况，我们要找到一个偏移系数 $s$ ，使初始向量 $\mathbf{v}$ 沿着正交于辅助超平面的指示向量 $\boldsymbol{\theta}$ 的方向移动一段合适的距离得到 $\mathbf{v}'$ ，然后再通过超立方体投影映射到校验多面体内，即 $\mathbf{z} \in \mathbb{PP}_{d_j}$ 可以表示为

$$\mathbf{z} = \Pi_{[0,1]^{d_j}}(\mathbf{v}') = \Pi_{[0,1]^{d_j}}(\mathbf{v} - s\boldsymbol{\theta}), \quad (4.30)$$

其中 $s$ 取一个合适的标量值使得 $\boldsymbol{\theta}^T \mathbf{z} \leq p$ 。

偏移系数 $s$ 的计算是校验多面体投影算法中最复杂的部分，本节中我们基于迭代的思想来计算得到 $s$ 的估计量 $\hat{s}$ 。根据投影的几何关系，我们可以将超立方体上的 $\mathbf{u}$ 到辅助超平面的距离作为一个参照量，来确定向量 $\mathbf{v}$ 在每次迭代过程中的偏移距离。定义 $\mathbf{u} \in [0,1]^{d_j}$ 到辅助超平面 $\boldsymbol{\theta}^T \mathbf{x} = p$ 的距离 $\eta$ ，令 $\mathbf{x}' = \mathbf{u} - \eta\boldsymbol{\theta}$ 为 $\mathbf{u}$ 投影到辅助超平面上的点，求解方程组

$$\begin{cases} \mathbf{u} - \mathbf{x}' = \eta\boldsymbol{\theta} \\ \boldsymbol{\theta}^T \mathbf{x}' = p \end{cases} \quad (4.31)$$

可以得到 $\eta = (\boldsymbol{\theta}^T \mathbf{u} - p) / \boldsymbol{\theta}^T \boldsymbol{\theta} = (\boldsymbol{\theta}^T \mathbf{u} - p) / d_j$ 。定义 $\mathbf{v}_i$ 为 $\mathbf{v}$ 在第 $i$ 次迭代中偏移后得到的向量， $\mathbf{v}_0 = \mathbf{v}$ ， $\mathbf{u}_i = \Pi_{[0,1]^{d_j}}(\mathbf{v}_i)$ ， $\eta_i$ 为 $\mathbf{u}_{i-1}$ 到辅助超平面的距离。 $\gamma$ 是一个缩放因子，用来确

定每次迭代中 $\mathbf{v}_i$ 走过的步长大小， $\gamma$ 越大，则每次偏移的步长越大，所需的迭代次数就更少，从而我们可以将偏移系数 $s$ 的估计值表示为

$$\hat{s} = \gamma \sum_i \eta_i. \tag{4.32}$$

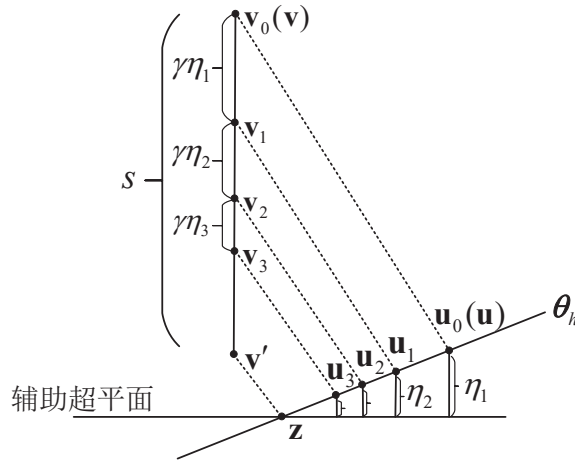


图 4.3 校验多面体投影说明

图4.3举例说明了校验多面体投影的过程，其中 $\theta_h$ 表示超立方体的一个边界。随着迭代次数的增加，向量 $\mathbf{v}$ 会越来越靠近要求的向量 $\mathbf{v}'$ 。首先讨论 $\gamma$ 为一个定值的情况。假设 $\gamma = 1$ ，则根据文献[24]的推论，向量 $\mathbf{v}$ 永远不会超过向量 $\mathbf{v}'$ ，即 $\eta_i \leq s - \sum_{j=1}^{i-1} \eta_j$ ，当 $\eta_i$ 小于一个很小的阈值 $\epsilon$ 时迭代停止，所以最终的向量 $\mathbf{v}$ 既可能到达 $\mathbf{v}'$ ，也可能是收敛到 $\mathbf{v}'$ 。当缩放因子 $\gamma > 1$ 时，向量 $\mathbf{v}$ 每次迭代中的偏移量会更大，从而所需的迭代次数会更少。但是在最后一次迭代时，因为缩放因子的存在可能会超过 $\mathbf{v}'$ ，由于我们选取的阈值非常小，通常 $\epsilon < 10^{-4}$ ，所以误差也会在这个数量级，如果 $\gamma$ 取值合适的话不会对整个译码算法的性能造成影响，后续我们通过仿真实实验证实了这一说法。

由于偏移系数 $s$ 是固定的，所以 $\gamma$ 的取值越大，则所需的迭代次数肯定会更少。但是当 $\gamma$ 较大时，最后超出 $\mathbf{v}'$ 的值也会更大，可能会引起译码性能的退化，所以我们考虑引入一个控制系数 $\rho < 1$ 来动态调节 $\gamma$ 的大小。当 $\gamma$ 的初值较大时，在每次迭代过程中通过 $\gamma_{i+1} = \rho\gamma_i$ 对 $\gamma$ 的值进行更新，使得 $\gamma$ 在算法收敛时减小到一个不会引起译码性能退化的值<sup>1</sup>，从而最大限度减少迭代次数的同时确保译码性能不会衰退。其中 $\rho$ 的大小可以通过结合分析 $\gamma$ 的初值和降到门限值所需的迭代次数得到，本文中 $\rho$ 取0.93。结合上述分析，基于迭代思想的FCPP算法具体步骤如算法9所示。

<sup>1</sup>在后续仿真中，我们发现固定 $\gamma = 2$ 时还不会引起译码性能的退化，所以将 $\gamma$ 动态调节的最小值门限值设为2。



**算法 9 FCPP算法**

输入: 向量  $\mathbf{v} \in \mathbb{R}^{d_j}$

- 1: 初始化  $\varepsilon$ ,  $t = 0$
- 2: 对  $i = 1, \dots, d_j$ , 令  $\theta_i = \text{sgn}(u_i - 0.5)$
- 3: **if** 集合  $\{i : \theta_i = 1\}$  的模  $t$  为偶数 **then**
- 4:      $i^* \leftarrow \arg \min |0.5 - v_i|$
- 5:      $\theta_{i^*} \leftarrow -\theta_{i^*}$
- 6: **end if**
- 7:  $p = t - 1$
- 8:  $\mathbf{u} = \Pi_{[0,1]^{d_j}}(\mathbf{v})$
- 9: **if**  $\boldsymbol{\theta}^T \mathbf{u} > p$  **then**
- 10:     **repeat**
- 11:          $\eta = (\boldsymbol{\theta}^T \mathbf{u} - p)/d_j$
- 12:          $\mathbf{v} \leftarrow \mathbf{v} - \gamma \eta \boldsymbol{\theta}$
- 13:          $\mathbf{u} = \Pi_{[0,1]^{d_j}}(\mathbf{v})$
- 14:         **if**  $\gamma > 2$  **then**
- 15:              $\gamma \leftarrow \rho \gamma$
- 16:         **end if**
- 17:     **until**  $\eta < \varepsilon$
- 18: **end if**
- 19:  $\mathbf{z} = \mathbf{u}$

输出:  $\mathbf{v}$  的校验多面体投影  $\mathbf{z} \in \text{PP}_{d_j}$

## 4.5 算法复杂度分析

考虑到校验多面体投影算法是基于校验多面体约束的译码算法中非常重要的一环, 在本节中, 首先对所提的FCPP算法和其他的校验多面体投影算法进行复杂度的分析与比较, 然后再对所提的ISI信道下的ISI-PDD译码算法与其他ISI信道下的LDPC码译码方法进行复杂度的讨论与对比。为了方便讨论, 本节考虑所有校验节点或者变量节点的度都相同的规则LDPC码, 令校验节点的度为  $d$ 。

首先, 将所提的FCPP算法与文献[22]、[23]、[63]和[24]中的校验多面体投影算法进行比较。

- 文献[22]中最早提到了校验多面体投影算法，其复杂度主要体现在两次排序操作中，而文献[23]所提的基于“割搜索”的校验多面体投影算法将排序操作减少到一次，所以文献[22]和[23]的复杂度为 $\mathcal{O}(d \log d)$ 。
- 文献[63]将校验多面体投影用单纯形（Simplex）投影替代，仍然存在部分排序操作，其平均复杂度可以达到线性复杂度 $\mathcal{O}(d)$ ，最坏的情况为 $\mathcal{O}(d \log d)$ 。
- 文献[24]提出了一种基于迭代思想的校验多面体投影算法，其中不涉及任何排序操作。在设置最大迭代次数 $I_{\max}$ 为定值的情况下，相比[22]和[23]的超线性复杂度，[24]在最坏情况下具有 $d$ 的线性复杂度，可以表示为 $\mathcal{O}(d)$ 。
- 本文所提的FCPP算法同样也是基于迭代思想，复杂度可以表示为 $\mathcal{O}(d)$ ，但是由于在迭代算法的基础上引入了缩放因子，所以达到相同收敛条件所需的迭代次数相比文献[24]会更少，整体的计算复杂度会更低。

结合上述分析，各校验多面体投影算法的复杂度如表4.1所示。在下一节中，我们通过仿真分析了所提的FCPP算法相比文献[24]中的算法在收敛速度上的提升。

表 4.1 校验多面体算法复杂度比较

校验多面体算法	计算复杂度
文献[22], [23], [63]中的CPP算法	$\mathcal{O}(d \log d)$
文献[24]中的CPP算法, 所提的FCPP算法	$\mathcal{O}(d)$

进一步地，对本章所提的ISI信道下的ISI-PDD译码算法，文献[36]所提的ADMM L2译码算法，以及Turbo均衡的方案进行了复杂度分析与比较。

- 考虑到本章所提的ISI-PDD译码算法由内层循环和外层循环两部分组成，其中外层循环中的对偶变量更新过程(4.22)可以等价写作

$$\frac{\lambda_j^{l+1}}{\mu_l} = \frac{\lambda_j^l}{\mu_l} + \mathbf{P}_j \mathbf{x}^K - \mathbf{z}_j^K, \forall j \in \mathcal{J}. \quad (4.33)$$

考虑到其中 $\frac{\lambda_j^l}{\mu_l}$ ， $\mathbf{P}_j \mathbf{x}^K$ 和 $\mathbf{z}_j^K$ 都已经在内层循环中计算过，在外层循环中不需要再进行额外的计算，所以只需要考虑内层循环的计算复杂度。对于内层循环需要分别考虑子问题 $\mathbf{x}$ 和子问题 $\mathbf{z}$ 的计算复杂度：子问题 $\mathbf{x}$ 中的 $\omega_i$ 是一个与迭代无关的常量，可以

在迭代前计算得到, 对于所有  $i \in \mathcal{I}$  中的  $\zeta_i$ , 计算  $N$  次  $\mu \sum_{j \in \mathcal{N}_v(i)} (z_j^{(i)} - \lambda_j^{(i)}) / \mu$  的复杂度为  $\mathcal{O}(N|\mathcal{N}_v(i)|)$ , 也可以表示为  $\mathcal{O}(Md)$ , 计算  $N$  次  $\delta_i$  的复杂度为  $\mathcal{O}(NT)$ ,  $\sigma$  是以一个与迭代无关的常量, 从而对于变量  $\mathbf{x}$  的更新所需的计算复杂度为  $\mathcal{O}(NT + Md)$ ; 子问题  $\mathbf{z}$  中复杂度最高的是FCPP算法, 当最大迭代次数  $I_{\max}$  为定值时, 子问题  $\mathbf{z}$  的计算复杂度可以写作  $\mathcal{O}(d)$ 。综上, 本章所提的ISI-PDD译码算法的计算复杂度可以表示为  $\mathcal{O}(NT + Md)$ 。

- 对于文献[36]中所提的ADMM L2译码算法, 其单次迭代中主要涉及变量  $\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}$  的更新, 更新变量  $\mathbf{x}$  的计算复杂度为  $\mathcal{O}(NT + Md)$ , 由于[36]中采用了[63]中的单纯形投影算法, 所以更新变量  $\mathbf{z}$  的最坏复杂度为  $\mathcal{O}(d \log d)$ 。从而[36]的ADMM译码算法的计算复杂度可以表示为  $\mathcal{O}(NT + Md \log d)$ 。
- 对于Turbo均衡的方案主要考虑均衡算法和译码算法的复杂度: 均衡算法采用软输出的SOVA算法, SOVA算法是一种基于网格的算法, 其复杂度由网格状态的数目决定, 随着部分响应信道的记忆长度呈指数增长, 所以对记忆长度为  $T$  的ISI信道以及码长为  $N$  的二进制LDPC码, 其均衡算法的计算复杂度可以表示为  $\mathcal{O}(N2^T)$ ; Turbo均衡中的译码算法采用基于软判决的和积BP译码算法, 其中计算复杂度最高的是校验节点更新中涉及的双曲正切和反双曲正切运算, 从而BP译码算法单次迭代的复杂度可以表示为  $\mathcal{O}(Md^2)$ 。结合上述分析, Turbo均衡方案的计算复杂度可以表示为  $\mathcal{O}(N2^T + Md^2)$ 。

综上所述, 各译码算法单次迭代的计算复杂度比较如表4.2所示。由于文献[36]中的ADMM译码算法采用了[63]中的单纯形投影算法, 仍然存在部分排序操作, 其最坏的情况为  $\mathcal{O}(d \log d)$ , 而本文所提的ISI-PDD译码算法中采用了所提的FCPP译码算法, 最坏情况的复杂度为  $\mathcal{O}(d)$ , 所以总体的计算复杂度还是优于[36]中的ADMM译码算法。对于Turbo均衡方案, 如果ISI信道的记忆长度很长, 则其复杂度呈指数增长, 复杂度较高。

表 4.2 ISI信道下的LDPC码译码算法复杂度比较

译码算法	计算复杂度
Turbo均衡	$\mathcal{O}(N2^T + Md^2)$
ISI信道下的ADMM L2译码算法 <sup>[36]</sup>	$\mathcal{O}(NT + Md \log d)$
所提的ISI-PDD译码算法	$\mathcal{O}(NT + Md)$

## 4.6 仿真结果与性能分析

针对本章所提的ISI信道下的ISI-PDD译码算法以及其中关键的FCPP算法，本节通过软件仿真和横向对比的方式来评估所提算法的性能，首先评估了FCPP算法的缩放因子对迭代次数的影响，然后对ISI信道下的ISI-PDD译码算法以及其他译码算法在不同信道和码长下的纠错性能进行了评估和比较，并且评估了不同缩放因子的FCPP算法对于译码算法整体性能的影响。本节的仿真都在Matlab软件平台上进行，采用BPSK的调制映射方式，考虑的ISI信道有EPR4信道，其中 $T = 3$ ， $\{h_0, h_1, h_2, h_3\} = \{0.5, 0.5, -0.5, -0.5\}$ ，以及PR4信道，其中 $T = 2$ ， $\{h_0, h_1, h_2\} = \{1/\sqrt{2}, 0, -1/\sqrt{2}\}$ 。考虑的ISI信道采用均值为0，方差为 $\sigma^2$ 的高斯白噪声加噪，其信噪比（Signal-to-Noise Ratio, SNR）可以定义为 $\text{SNR} \triangleq \sum_{i=0}^T h_i^2 / \sigma^2$ 。考虑的LDPC码字为CCSDS的(256,128) LDPC码 $C_1$ 和Wimax的(1152,576) LDPC码 $C_2$ <sup>[62]</sup>。

### 4.6.1 FCPP算法仿真

首先我们讨论校验多面体算法的收敛过程，图4.4基于码字 $C_1$ 和EPR4信道举例说明了不同的缩放因子 $\gamma$ 对于FCPP算法收敛速度的影响，其中横坐标表示算法的迭代次数 $k$ ，纵坐标表示累积的偏移系数 $\hat{s}^k = \sum_{i=0}^k \eta_i$ ，为了表述更加清晰，我们选择了所需迭代次数较高的样本，图中的红线表示标准的偏移系数 $s$ 。对于 $\gamma = 1.5$ 和2的情况， $\gamma$ 在迭代过程中保持不变，对于 $\gamma = 4$ ， $\rho = 0.93$ 的情况， $\gamma$ 在迭代过程中通过 $\rho$ 动态调整到一个门限附近的值。从图中可以看出，没有引入缩放因子的ICPP算法<sup>[24]</sup>收敛速度最慢，所需的迭代次数最多，在FCPP算法中引入缩放因子后，每一次迭代有了更大的步长，收敛速度有了明显的提升，最后达到标准偏移系数所需的迭代次数也更少，且随着缩放因子的增大，收敛速度更快。

进一步地，我们通过仿真分析了FCPP算法的缩放因子 $\gamma$ 对于收敛速度的提升效果。考虑到校验多面体投影算法为ISI-PDD译码算法中的子算法，算法的输入向量 $\mathbf{v} \in \mathbb{R}^{d_j}$ 由式(4.20)中的 $\mathbf{P}_j \mathbf{x}^{k+1} + \frac{\lambda_j^k}{\mu}$ 计算得到，所以在分析收敛行为时不需要考虑不同信道的影响，只考虑不同的校验节点度带来的影响。

表4.3的仿真结果基于EPR4信道，分别采用(256,128) LDPC码 $C_1$ 和(1152,576) LDPC码 $C_2$ 来获取向量 $\mathbf{v}$ 的仿真样本，其中码字 $C_1$ 的校验节点度均为8，码字 $C_2$ 的校验节点度为6或者7。考虑到高信噪比下迭代次数普遍偏小，所以基于 $C_1$ 和 $C_2$ 的统计样本分别在SNR为3.5dB和2dB时生成，每个码字对应的统计样本收集了 $5 \times 10^4$ 组向量 $\mathbf{v}$ 用于统计不同缩放因子对应的迭代次数，校验多面体投影算法中用于判断收敛的阈值设置为 $\varepsilon = 10^{-5}$ 。

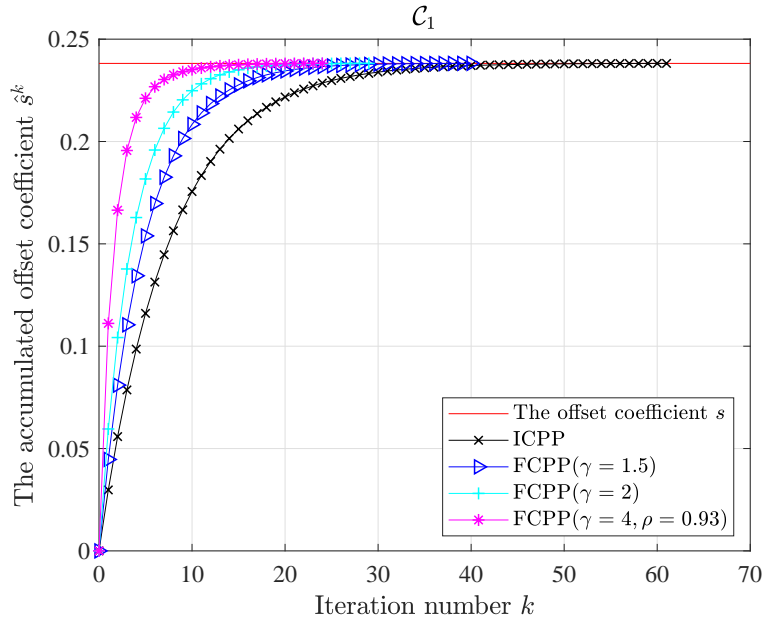


图 4.4 不同缩放因子的FCPP算法的收敛过程

表 4.3 FCPP算法迭代次数对比

	$C_1 (d = 8)$		$C_2 (d = 6, 7)$	
	平均情况	最坏情况	平均情况	最坏情况
ICPP <sup>[24]</sup>	23.4098	71	21.5050	60
FCPP( $\gamma = 1.5$ )	14.4781	46	13.1716	39
FCPP( $\gamma = 2$ )	9.7715	34	8.7714	28
FCPP( $\gamma = 4, \rho = 0.93$ )	5.0876	29	4.0161	23

表中我们统计了不同算法和缩放因子对于同一组统计样本所需的平均迭代次数和最坏情况所需的迭代次数，可以看出引入缩放因子的FCPP算法相比ICPP算法可以同时减少平均迭代次数和最大迭代次数，且可以根据缩放因子的大小来控制收敛速度。其中 $\gamma = 1.5$ 的FCPP算法相比ICPP算法在码字 $C_1$ 和 $C_2$ 的条件下分别减少了38%和39%的迭代次数， $\gamma = 2$ 的FCPP算法相比ICPP算法在码字 $C_1$ 和 $C_2$ 的条件下分别减少了58%和60%的迭代次数， $\gamma = 4, \rho = 0.93$ 的FCPP算法相比ICPP算法在码字 $C_1$ 和 $C_2$ 的条件小分别减少了78%和81%的迭代次数。需要注意的是， $\gamma = 4, \rho = 0.93$ 的FCPP算法中 $\gamma$ 在10次迭代后取到门限附近的最小值，所以对所需迭代次数较少的样本提升较为明显，对于所需迭代次

数较多的最坏情况提升较少。

### 4.6.2 ISI-PDD译码算法仿真

本小节中我们将所提的ISI-PDD译码算法与其他ISI信道下的译码算法进行了仿真比较，其中Turbo均衡算法中的均衡算法采用了SOVA算法，译码算法采用标准的和积BP译码算法。考虑到Turbo均衡为双层迭代算法，其内层最大迭代次数 $k_{\max}$ 和外层最大迭代次数 $l_{\max}$ 可以表示为 $(k_{\max}, l_{\max})$ 。对于所有仿真曲线，由于计算资源有限，每条曲线的最后两个信噪比下只收集了50个错误帧，在其余每个信噪比点都至少收集了100个错误帧，且每一组曲线的信息序列和噪声都基于相同的随机种子生成。

#### (1) LDPC码 $C_1$

首先考虑码率为0.5的(256,128) CCSDS LDPC码 $C_1$ 分别在EPR4信道和PR4信道下的译码性能。将Turbo均衡算法的内层最大迭代次数和外层最大迭代次数在括号内给出，并且将SOVA-BP(30,30)的Turbo均衡方案作为比较的基准，即总的内层迭代次数为900。为了进行公平的比较，将ISI-PDD的内层最大迭代次数和外层迭代次数分别设置为4和225，从而保持总的内层迭代次数也为900，由于ISI信道下的ADMM L2译码算法<sup>[36]</sup>为单层迭代算法，所以将ADMM译码器的最大迭代次数设置为900。同时我们通过遍历搜索的方式来选择误码率最低的参数作为译码算法的参数，列举在表4.4中。进一步地，为了分析FCPP算法中的缩放因子是否会引起ISI-PDD译码算法的性能衰退，我们对不同缩放因子的ISI-PDD算法进行了仿真。

表 4.4 针对码字 $C_1$ 的译码器参数设置

	ISI-PDD				ADMM <sup>[36]</sup>	
	$\alpha$	$\mu$	$\beta$	$c$	$\alpha$	$\mu$
$C_1$ EPR4	3.8	2.1	1.5	1.003	3.8	1.8
$C_1$ PR4	4.6	3.4	1.5	1.003	4.6	2.6

在图4.5中，我们提供了LDPC码 $C_1$ 分别在EPR4信道和PR4信道下的误帧率（Frame-Error-Rate, FER）性能曲线。在EPR4信道下，本文所提的ISI-PDD译码器在各信噪比下的FER性能均优于作为基准的SOVA-BP(30,30)的Turbo均衡方案，在低信噪比区域

与ADMM L2译码器的性能相近。当FER为 $10^{-3}$ 时，所提的ISI-PDD译码器相比Turbo均衡方案有0.5dB左右的增益，相比ADMM L2译码器有0.2dB左右的增益。在PR4信道下，本文所提的ISI-PDD译码器在各信噪比下的FER性能同样均要优于作为基准的SOVA-BP(30,30)的Turbo均衡方案，且在高信噪比区域优势较为明显，在低信噪比区域与ADMM L2译码器的性能相近。当FER为 $10^{-4}$ 时，所提的ISI-PDD译码器相比基准的Turbo均衡方案有0.3dB的增益，相比ADMM L2译码器有0.2dB左右的增益。从图中我们可以发现，相比于采用ICPP算法的ISI-PDD译码算法，采用带缩放因子的FCPP算法的ISI-PDD译码算法性能并没有衰退，不同缩放因子的ISI-PDD算法性能非常接近，说明引入可以动态调整的缩放因子可以在确保译码性能不衰退的前提下，大大降低CPP算法所需的迭代次数。

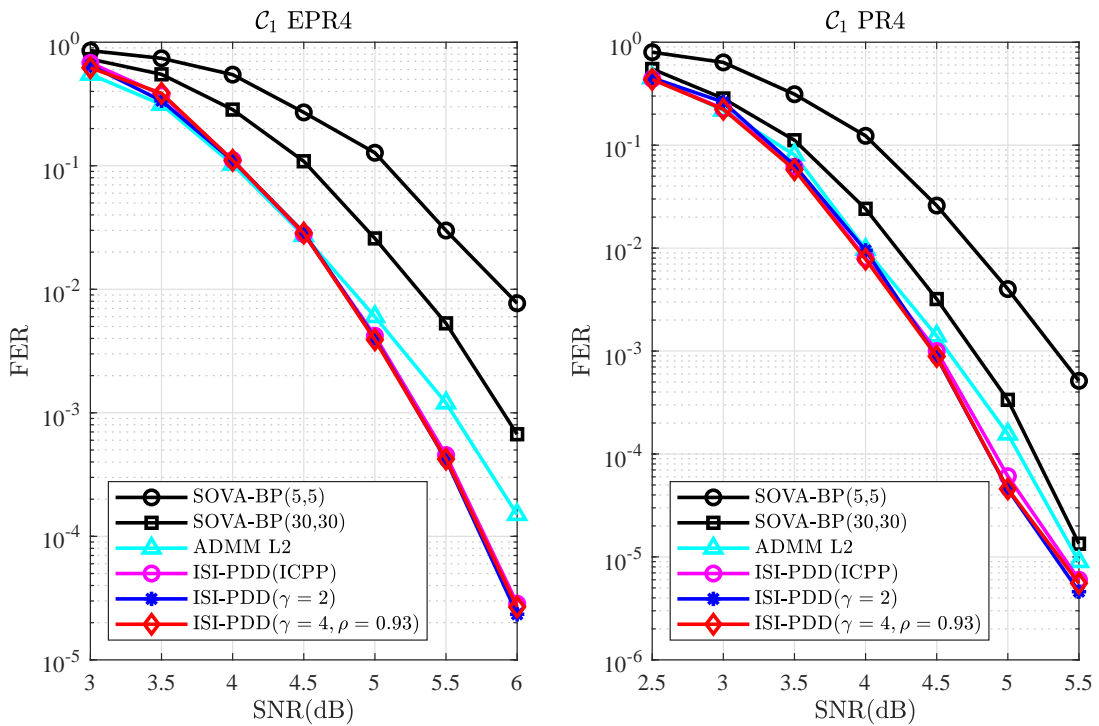


图 4.5 LDPC码 $C_1$ 在EPR4和PR4信道下的FER曲线

(2) LDPC码 $C_2$

接着考虑Wimax的(1152,576) LDPC码 $C_2$ 分别在EPR4信道和PR4信道下的译码性能。同样将SOVA-BP(30,30)的Turbo均衡方案作为比较的基准，令总的内层迭代次数为900。为了公平的比较，将ISI-PDD的内层最大迭代次数和外层迭代次数分别设置为4和225，控制总的内层迭代次数为900，并且将ISI信道下的ADMM L2译码算法的最大迭代次数设置为900。考虑到对码字 $C_1$ 的仿真已经证明了合适的缩放因子 $\gamma$ 并不会导致译码算法性能的衰

退，所以对码字 $C_2$ 的仿真中采用 $\gamma = 4$ ， $\rho = 0.93$ 的FCPP算法作为ISI-PDD算法的校验多面体算法。表4.5中列举了译码算法性能最佳的参数。

表 4.5 针对码字 $C_2$ 的译码器参数设置

	ISI-PDD				ADMM <sup>[36]</sup>	
	$\alpha$	$\mu$	$\beta$	$c$	$\alpha$	$\mu$
$C_1$ EPR4	4.2	3.7	1.5	1.003	3.8	2.6
$C_1$ PR4	4.8	5.4	1.5	1.003	4.8	2.9

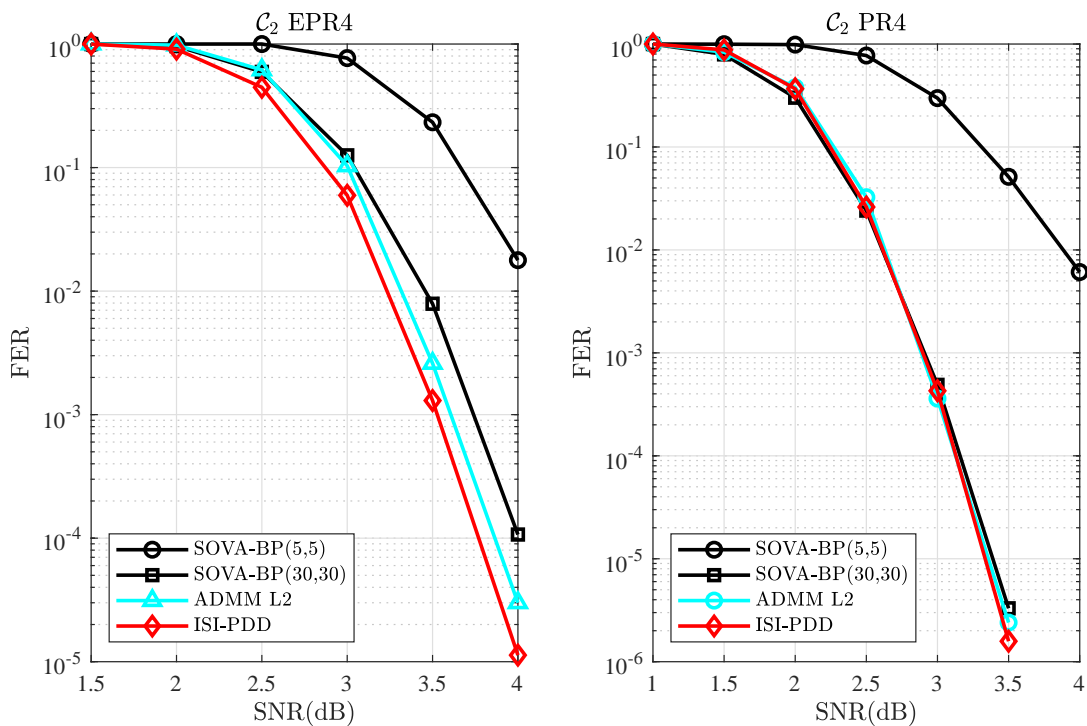


图 4.6 LDPC码 $C_2$ 在EPR4和PR4信道下的FER曲线

在图4.6中，我们提供了LDPC码 $C_2$ 分别在EPR4信道和PR4信道下的FER性能曲线。在EPR4信道下，本文所提的ISI-PDD译码器在各信噪比下的FER性能均优于SOVA-BP(30,30)的Turbo均衡方案，在高信噪比区域优势较为明显。当FER为 $10^{-4}$ 时，所提的ISI-PDD译码器相比Turbo均衡方案有0.2dB左右的增益，相比ADMM L2译码器有0.1dB的增益。对于PR4信道下码字 $C_2$ 的译码，Turbo均衡提升迭代次数所带来的增益较大，SOVA-BP(30,30)相比SOVA-BP(5,5)在FER为 $10^{-2}$ 时有1dB以上的增益，本文所提的ISI-PDD译码器



与SOVA-BP(30,30)的Turbo均衡方案和ADMM L2译码器性能接近，在高信噪比区域性能略好于Turbo均衡方案。

## 4.7 本章小结

本章主要研究了ISI信道下的PDD译码算法及其关键的校验多面体投影算法，其目的是弥补Turbo均衡方案存在的不足，改善ISI信道下LDPC译码器的性能，同时提升PDD译码算法中校验多面体投影算法的效率。首先，我们介绍了ISI信道传输模型，以及该信道下的LDPC码ML译码问题的QP描述，然后提出了一种基于校验多面体约束的ISI-PDD译码算法。进一步地，针对ISI-PDD译码算法中的校验多面体投影步骤，提出了一种新的FCPP算法，通过引入可动态调节的缩放因子来减少投影算法所需的迭代次数，大大提升了译码算法的效率。接着我们对本章所提的算法进行了复杂度的分析，并且与经典算法进行了比较。最后，我们对所提的FCPP算法和ISI-PDD译码算法进行了软件仿真，并和其他经典算法进行对比分析，仿真结果表明了FCPP算法可以在确保译码性能的前提下大大减少CPP步骤所需的迭代次数，以及ISI-PDD译码器相比Turbo均衡的译码方案具有更优的译码性能。



## 第五章 总结与展望

### 5.1 本文工作总结

纠错码是提高通信系统可靠性的关键技术，而LDPC码作为一种接近香农极限的纠错码，在学术界和工业界都获得了广泛的研究和应用，如今已被采纳为5G NR中eMBB场景下的数据信道编码方案。考虑到经典的BP译码算法存在理论保证弱，高信噪比存在错误平层等缺陷，基于ML问题的LP译码算法受到了编码工作者的广泛关注。虽然LP译码算法具有较强的理论保证，并且可以有很低的错误平层，但是其实现复杂度较高，限制了其在工程上的应用。近年来的一些研究提出用大规模分布式优化算法来求解LP问题，并且获得了不错的效果，而PDD算法是一种基于BCD且可以求解大规模非凸非平滑优化问题的迭代算法。受此启发，本文基于PDD算法框架来求解LDPC码的LP译码问题，将深度学习技术作为辅助，并进一步将LP译码的思想推广到ISI信道下LDPC码的译码。现将本文的主要研究成果总结如下：

- (1) 针对LDPC码的LP译码问题，本文提出了一种基于级联度分解的PDD译码算法。首先介绍了ML问题的级联整数规划形式，并采用LP松弛法来处理离散的校验约束，目的是使原问题可以在多项式时间内求解，然后引入罚函数来避免伪码字的出现，并使用超松弛机制来加快算法的收敛速度。进一步地，本文利用深度展开技术将所提的PDD译码算法展开为一个模型驱动的神经网络LPDN，将可调参数设置为训练参数，并采用适当的梯度下降法对网络进行训练，得到最优的网络参数配置。仿真结果表明，所提的PDD译码器和LPDN译码器相比其他先进的译码器具有更优的纠错性能，并且所提的LPDN译码器相比所提的PDD译码器可以大大减少迭代次数，提高译码效率。
- (2) 将LP译码的思想推广到ISI信道中，本文进一步研究了基于ISI信道的LDPC码PDD译码算法。首先本文将ISI信道下的LDPC码ML译码问题描述为一个QP问题，用校验多面体的形式来描述校验约束条件，同样采用LP松弛法和罚函数法来简化约束条件，引入超松弛机制加快收敛速度，然后基于PDD的算法框架提出了ISI-PDD译码算法。考虑到译码算法中的校验多面体投影步骤最为复杂，耗时最多，本文提出了一种基于迭代思想的FCPP算法，通过引入可动态调节的缩放因子来减少校验多面体投影算法所需的迭代次数，从而提升译码器整体的译码效率。仿真结果首

先证明了FCPP算法的有效性，可以通过适当改变缩放因子的大小来调整算法的收敛速度，并且不会造成译码器的性能衰退，其次所提的ISI-PDD译码算法相比经典的Turbo均衡译码方案具有更好的纠错性能，而且其计算复杂度随着ISI信道的记忆长度呈线性增长。

## 5.2 下一步研究方向

本文基于PDD算法框架对LDPC码的LP译码问题开展了深入研究，同时引入了深度学习技术辅助并将LP译码的思想推广到ISI信道中，目前取得了一些研究成果，但是仍存在如下一些工作值得进一步探索研究：

- (1) LDPC码的和积BP译码算法凭借简化的最小和算法已经在工业界获得了广泛的应用，而基于LP译码问题的PDD译码算法虽然在性能上获得了不错的提升，并且相比初始的LP译码算法大大降低了复杂度，但是算法的硬件实现仍然比较复杂。考虑能否寻找一种硬件简化策略，来近似PDD译码算法中的乘法和矩阵运算，对于PDD译码算法的硬件实现和工程应用有着重要意义。
- (2) PDD算法作为一种适用于求解大规模非凸非平滑优化问题的迭代算法，在求解LDPC码的LP译码问题方面获得了不错的性能表现，但是在其他纠错码中的应用尚待发掘，将LP译码的思想推广到BCH码，Polar码等线性纠错码中，开发易于分析求解且性能优良的译码算法同样是一个值得挖掘和探索的方向。
- (3) 定义在 $GF(2^q)$ 的伽罗瓦域上的非二进制LDPC码在高数据速率系统和存储系统中有着潜在应用价值，相比二进制的LDPC码可以更好消除短环，并且有着较强的抗突发干扰能力。所以将现有的PDD译码算法推广到非二进制的LDPC码也是一个有价值的议题。

## 参考文献

- [1] C. E. Shannon. A mathematical theory of communication[J]. The Bell System Technical Journal, 1948. 27(3):379–423.
- [2] R. W. Hamming. Error detecting and error correcting codes[J]. The Bell System Technical Journal, 1950. 29(2):147–160.
- [3] E. Berlekamp. On decoding binary Bose-Chadhuri-Hocquenghem codes[J]. IEEE Transactions on Information Theory, Oct. 1965. 11(4):577–579.
- [4] I. S. Reed, G. Solomon. Polynomial codes over certain finite fields[J]. Journal of the Society for Industrial and Applied Mathematics, 1960. 8(2):300–304.
- [5] P. Elias. Coding for noisy channels[J]. IRE Convention Record, 1955. (4):37–46.
- [6] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm[J]. IEEE Transactions on Information Theory, 1967. 13(2):260–269.
- [7] C. Berrou, A. Glavieux. Near optimum error correcting coding and decoding: Turbo-codes[J]. IEEE Transactions on Communications, 1996. 44(10):1261–1271.
- [8] E. Arikan. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels[J]. IEEE Transactions on Information Theory, Jul. 2009. 55(7):3051–3073.
- [9] R. Gallager. Low-density parity-check codes[J]. IRE Transactions on Information Theory, 1962. 8(1):21–28.
- [10] D. J. C. MacKay. Good error-correcting codes based on very sparse matrices[J]. IEEE Transactions on Information Theory, 1999. 45(2):399–431.
- [11] J. Chen, M. P. C. Fossorier. Near optimum universal belief propagation based decoding of low-density parity check codes[J]. IEEE Transactions on Communications, Mar. 2002. 50(3):406–414.
- [12] 王新梅. 纠错码-原理与方法 (修订版) [M]. 西安:西安电子科技大学出版社, 2009.
- [13] J. Feldman, M. Wainwright, D. Karger. Using linear programming to decode binary linear codes[J]. IEEE Transactions on Information Theory, Mar. 2005. 51(3):954–972.
- [14] T. O’ Shea, J. Hoydis. An introduction to deep learning for the physical layer[J]. IEEE Transactions on Cognitive Communications and Networking, 2017. 3(4):563–575.
- [15] 韦逸. 基于深度学习的物理层通信技术研究[D]. 杭州:浙江大学, 2022.
- [16] N. Miladinovic, M. P. C. Fossorier. Improved bit-flipping decoding of low-density parity-check codes[J]. IEEE Transactions on Information Theory, 2005. 51(4):1594–1606.
- [17] X. Wu, C. Zhao, X. You. Parallel weighted bit-flipping decoding[J]. IEEE Communications Letters, 2007.

- 11(8):671–673.
- [18] F. R. Kschischang, B. J. Frey, H.-A. Loeliger. Factor graphs and the sum-product algorithm[J]. IEEE Transactions on Information Theory, 2001. 47(2):498–519.
- [19] M. P. C. Fossorier, M. Mihaljevic, H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation[J]. IEEE Transactions on Communications, 1999. 47(5):673–680.
- [20] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, X. Hu. Reduced-complexity decoding of LDPC codes[J]. IEEE Transactions on Communications, 2005. 53(8):1288–1299.
- [21] K. Yang, X. Wang, J. Feldman. A new linear programming approach to decoding linear block codes[J]. IEEE Transactions on Information Theory, 2008. 54(3):1061–1072.
- [22] S. Barman, X. Liu, S. C. Draper, B. Recht. Decomposition methods for large scale LP decoding[J]. IEEE Transactions on Information Theory, 2013. 59(12):7870–7886.
- [23] X. Zhang, P. H. Siegel. Efficient iterative LP decoding of ldpc codes with alternating direction method of multipliers[C]//IEEE International Symposium on Information Theory (ISIT). 2013:1501–1505.
- [24] H. Wei, A. H. Banihashemi. An iterative check polytope projection algorithm for ADMM-based LP decoding of LDPC codes[J]. IEEE Communications Letters, 2018. 22(1):29–32.
- [25] X. Liu, S.C. Draper. The ADMM penalized decoder for LDPC codes[J]. IEEE Transactions on Information Theory, 2016. 62(6):2966–2984.
- [26] Q. Wu, F. Zhang, H. Wang, J. Lin, Y. Liu. Parameter-free  $l_p$ -box decoding of LDPC codes[J]. IEEE Communications Letters, 2018. 22(7):1318–1321.
- [27] J. Bai, Y. Wang, Q. Shi. Efficient QP-ADMM decoder for binary LDPC codes and its performance analysis[J]. IEEE Transactions on Signal Processing, 2020. 68:503–518.
- [28] M. M. Zhao, Q. Shi, Y. Cai, M. J. Zhao, Q. Yu. Decoding binary linear codes using penalty dual decomposition method[J]. IEEE Communications Letters, 2019. 23(6):958–962.
- [29] M. Tuchler, A. C. Singer. Turbo equalization: An overview[J]. IEEE Transactions on Information Theory, 2011. 57(2):920–952.
- [30] L. Bahl, J. Cocke, F. Jelinek, J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate[J]. IEEE Transactions on Information Theory, 1974. 20(2):284–287.
- [31] J. Hagenauer, P. Hoeher. A viterbi algorithm with soft-decision outputs and its applications[C]//IEEE GLOBECOM,Dallas,TX. volume 3. 1989:1680–1686.
- [32] T. Wadayama. Interior point decoding for linear vector channels based on convex optimization[J]. IEEE Transactions on Information Theory, 2010. 56(10):4905–4921.
- [33] M. H. Taghavi, P. H. Siegel. Graph-based decoding in the presence of ISI[J]. IEEE Transactions on Information Theory, 2011. 57(4):2188–2202.
- [34] M. F. Flanagan. A unified framework for linear-programming based communication receivers[J]. IEEE

- Transactions on Communications, 2011. 59(12):3375–3387.
- [35] B. Kim, H. D. Pfister. Joint decoding of LDPC codes and finite-state channels via linear-programming[J]. IEEE Journal of Selected Topics in Signal Process., 2011. 5(8):1563–1576.
- [36] X. Jiao, J. Mu, Y. He, W. Xu. Linear-complexity admm updates for decoding LDPC codes in partial response channels[J]. IEEE Communications Letters, 2019. 23(12):2200–2204.
- [37] X. Jiao, H. Liu, J. Mu, Y. He.  $l_2$ -box ADMM decoding for LDPC codes over ISI channels[J]. IEEE Transactions on Vehicular Technology, 2021. 70(4):3966–3971.
- [38] T. Gruber, S. Cammerer, J. Hoydis, S. Brink. On deep learning-based channel decoding[C]//Annual Conference on Information Sciences and Systems (CISS). 2017:1–6.
- [39] J. R. Hershey, J. L. Roux, F. Wenginger. Deep unfolding: Model-based inspiration of novel deep architectures[J]. ArXiv:1409.2574, 2014.
- [40] A. Balatsoukas-Stimming, C. Studer. Deep unfolding for communications systems: A survey and some new directions[C]//IEEE International Workshop on Signal Processing Systems (SiPS). 2019:266–271.
- [41] N. Shlezinger, Y. C. Eldar, S. P. Boyd. Model-based deep learning: On the intersection of deep learning and optimization[J]. IEEE Access, 2022:1–1.
- [42] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, Y. Be’ery. Deep learning methods for improved decoding of linear codes[J]. IEEE Journal of Selected Topics in Signal Process., 2018. 12(1):119–131.
- [43] W. Xu, X. You, C. Zhang, Y. Be’ery. Polar decoding on sparse graphs with deep learning[C]//Asilomar Conference on Signals, Systems, and Computers. 2018:599–603.
- [44] Y. Wei, M. M. Zhao, M. J. Zhao, M. Lei. ADMM-based decoder for binary linear codes aided by deep learning[J]. IEEE Communications Letters, 2020. 24(5):1028–1032.
- [45] W. E. Ryan, S. Lin. Channel Codes: Classical and Modern[M]. Cambridge:Cambridge University Press, 2009.
- [46] 仇佩亮, 张朝阳, 谢磊, 余官定. 信息论与编码[M]. 北京:高等教育出版社, 2011.
- [47] R. Tanner. A recursive approach to low complexity codes[J]. IEEE Transactions on Information Theory, 1981. 27(5):533–547.
- [48] X. Y. Hu, E. Eleftheriou, D. M. Arnold. Regular and irregular progressive edge-growth tanner graphs[J]. IEEE Transactions on Information Theory, 2005. 51(1):386–398.
- [49] X. Hua, A. H. Banihashemi. Improved progressive-edge-growth (PEG) construction of irregular LDPC codes[J]. IEEE Communications Letters, 2004. 8(12):715–717.
- [50] T. Tao, C. R. Jones, J. D. Villasenor, R. D. Wesel. Selective avoidance of cycles in irregular LDPC code construction[J]. IEEE Transactions on Communications, 2004. 52(8):1242–1247.
- [51] R. Asvadi, A. H. Banihashemi, M. Ahmadian-Attari. Lowering the error floor of LDPC codes using cyclic

- liftings[J]. *IEEE Transactions on Information Theory*, 2011. 57(4):2213–2224.
- [52] Y. Kou, S. Lin, M. P. C. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results[J]. *IEEE Transactions on Information Theory*, 2001. 47(7):2711–2736.
- [53] 徐俊, 袁弋飞. 5G-NR信道编码[M]. 北京:人民邮电出版社出版社, 2018.
- [54] Q. Shi, M. Hong. Penalty dual decomposition method for nonsmooth nonconvex optimization — part i: Algorithms and convergence analysis[J]. *IEEE Transactions on Signal Processing*, 2020. 68:4108–4122.
- [55] L. Bottou. Large-scale machine learning with stochastic gradient descent[M]//*Proceedings of COMPSTAT*. 2010. 177–186.
- [56] N. Qian. On the momentum term in gradient descent learning algorithms[J]. *Neural networks*, Aug. 1999. 12(1):145–151.
- [57] M. C. Mukkamala, M. Hein. Variants of rmsprop and adagrad with logarithmic regret bounds[C]//*ICML*. 2017:2545–2553.
- [58] D. P. Kingma, J. Ba. Adam: a method for stochastic optimization[J]. *ArXiv:1412.6980*, 2015.
- [59] D. Micciancio. The hardness of the closest vector problem with preprocessing[J]. *IEEE Transactions on Information Theory*, Mar. 2001. 47(3):1212–1215.
- [60] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers[J]. *Foundations and Trends in Machine Learning*, Jan. 2011. 3(1):1–123.
- [61] 张贤达. 矩阵分析与应用[M]. 北京:清华大学出版社, 2013.
- [62] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, N. Wehn. Database of channel codes and ML simulation results. [www.uni-kl.de/channel-codes](http://www.uni-kl.de/channel-codes), 2017.
- [63] G. Zhang, R. Heusdens, W. B. Kleijn. Large scale LP decoding with low complexity[J]. *IEEE Communications Letters*, 2013. 17(11):2152–2155.



## 攻读硕士学位期间的研究成果

### 发表论文（第一作者）

1. Yihao Liu, Ming-Min Zhao, Chan Wang, Ming Lei, Min-Jian Zhao, "PDD-Based Decoder for LDPC Codes with Model-Driven Neural Networks," *IEEE Communications Letters*, vol. 26, no. 11, pp. 2532-2536, Nov. 2022. (SCI, 正文第三章)

### 发明专利（第一申请人）

1. 刘屹豪,赵明敏,雷鸣,赵民建,袁辉,"一种基于惩罚对偶分解法的深度展开信道译码方法及装置,"申请号: ZL202210718915.0 (已受理)

### 参与的科研项目

1. "XX低截获动态协作网络物理层设计与实现"

负责高吞吐率的信道编码方案设计,完成了7/8码率的QC-LDPC码的编译码器的Matlab仿真与FPGA硬件实现,基于并行分层译码架构(PLDA)将译码器吞吐率提升至Gbps量级。

2. "XX多速率系统设计与实现"

参与多速率档下的多码长多码率LDPC码的码型结构方案设计以及编译码的Matlab仿真与FPGA硬件实现;设计了译码器硬件架构的复用方案,用单译码器实现多码长多码率LDPC码的译码。

3. "5G LDPC编译码器设计与实现"

研究基于5G标准的LDPC编译码器,负责设计能够覆盖5G标准中码长码率的QC-LDPC部分并行译码架构。

### 在校获得荣誉

1. 2021-2022年度,浙江大学优秀研究生,圣邦微电子奖学金