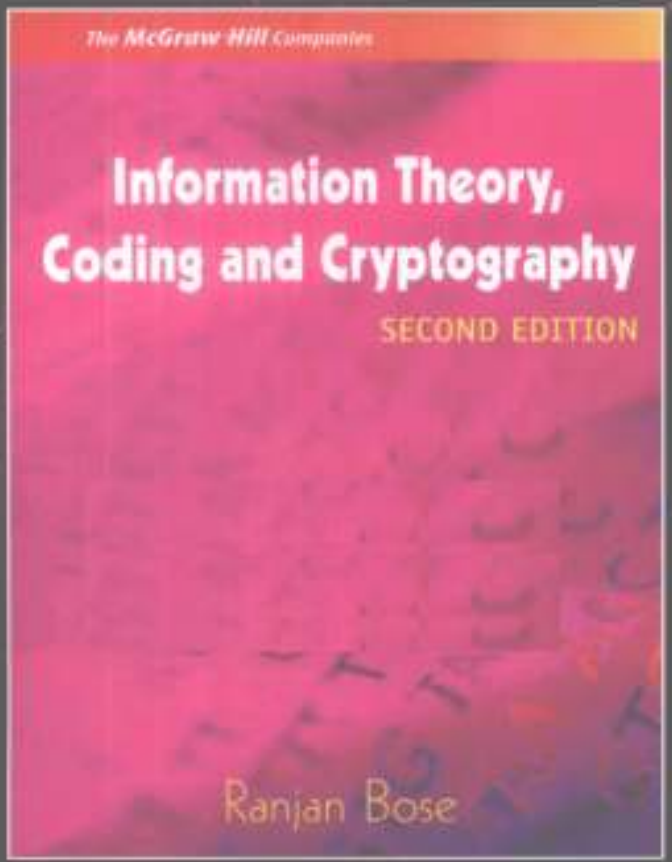
计 算 机 科 学 丛 书

第2版

Education

信息论、编码与密码学

(印) Ranjan Bose 著 武传坤 李徽 译



Information Theory, Coding and Cryptography

Second Edition

机械工业出版社
China Machine Press

信息论、编码与密码学（第2版）

大多数介绍信息论、编码和密码学的书，不是太过数学化，就是太注重基础。本书避免了上述缺点，既考虑到数学的严谨性，又充分考虑了易读性。在第2版中我们收录了许多正在产生的、成熟的或者已经在工业标准和应用中使用的新概念和新想法。

本书特点：

- 数学论证后面都接有信源编码的直观解释。
- 涵盖更完整的知识面，包括线性分组码、循环码、BCH码和RS码等。
- 完整覆盖卷积码、Turbo码和网格编码调制（TCM）。
- 全面介绍密码学基础，公钥和私钥加密，当前的加密标准及密码学最新趋势。

本版更新内容：

- 新增关于空时分组码和空时网格码的介绍。
- 在Turbo码和网格编码方面进行了内容扩展。
- 新增内容有
  - 算术编码
  - MIMO信道
  - 低密度奇偶校验（LDPC）码
  - 最小距离的界
  - Turbo码的交织器设计
  - 椭圆曲线密码学
  - 量子密码学
  - 生物加密

作者简介

Ranjan Bose 在美国宾夕法尼亚大学获电气工程博士学位，曾在位于美国加利福尼亚州圣何塞的Alliance半导体公司任高级设计工程师，现任印度理工学院教授。2003年荣获印度国家工程师学会（INAE）颁发的“杰出青年工程师奖”。

客服热线：(010) 88378991, 88361066
购书热线：(010) 68326294, 88379649, 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

华章网站 http://www.hzbook.com

网上购书 www.china-pub.com

Mc Graw Hill Education
www.mheducation.com

ISBN 978-7-111-30888-1

9 787111 308881

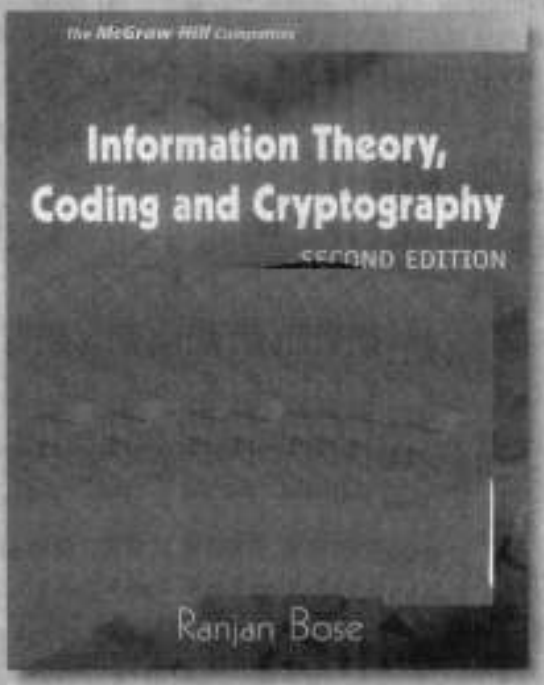
定价：39.00元

计 算 机 科 学 丛 书

第2版

信息论、编码与密码学

(印) Ranjan Bose 著 武传坤 李徽 译



Information Theory, Coding and Cryptography

Second Edition

机械工业出版社
China Machine Press

信息论、错误控制编码和密码学是现代数字通信系统中的三大支柱，本书用有限的篇幅将三者中所有重要的概念有机地结合起来，涉及信息论、信源编码、信道编码和密码学等方面的知识，不仅内容丰富，而且技术深度适当。

本书适合作为高等院校信息安全、电子工程及相关专业信息论和编码课程的教材，从事相关工作的专业技术人员也能从中受益。

本书版权登记号：图字：01-2010-1514

图书在版编目（CIP）数据

信息论、编码与密码学 第2版 /（印）博斯（Bose, R.）著；武传坤，李徽译. —北京：机械工业出版社，2010.7

（计算机科学丛书）

书名原文：Information Theory, Coding and Cryptography, Second Edition

ISBN 978-7-111-30888-1

Ⅰ. 信… Ⅱ. ① 博… ② 武… ③ 李… Ⅲ. ① 信息论 ② 信源编码-编码理论 ③ 密码-理论 Ⅳ. ① TN911.2 ② TN918.1

中国版本图书馆CIP数据核字（2010）第102762号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：秦 健

北京市荣盛彩色印刷有限公司印刷

2010年9月第2版第1次印刷

184mm × 260mm · 15.25印张

标准书号：ISBN 978-7-111-30888-1

定价：39.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自1998年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：(010) 88379604
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

华章教育

华章科技图书出版中心

译 者 序

自从Shannon在1948年发表了一篇关于通信的数学理论的论文之后，人们开始了对信息理论的系统研究。Shannon在第二年（即1949年）又发表了一篇关于安全系统的通信理论的论文，于是又引发了对信息安全的系统研究。现代信息论除了一般的信息理论部分外，它的重要组成部分还包括信源编码、信道编码和密码学。这些部分既有信息理论上的描述，又有它们自己独特的设计技术和方法。在研究上，信源编码、信道编码和密码学都相对较独立，但又有不少将它们相结合的研究。因此，有必要对这些内容进行全面了解。

目前在市场上可以见到大量关于信息论和编码方面的书，也有许多专门研究密码和信息安全的书，但将它们融为一体的书却不多见。有些书的内容过于庞杂，对需要了解这方面知识的读者来说不够简洁易懂。Bose的这本书用较短的篇幅覆盖了信息论、信源编码、信道编码和密码学部分，不仅覆盖面超出了许多大部头的书，而且也有一定的技术深度，即使这方面的专家读起来也不乏味。这种精湛的概括和有机结合是本书的主要特色，使它成为一本很好的简明参考书。因此我们组织翻译出版了本书的第1版。

随着信息科学持续高速的发展，作为信息科学领域的一些专著文献也很快表现出与最新科技发展的差距。一本负责任的科技著作应该随所涉及领域的科学技术发展而及时更新其内容。本书作者本着对读者负责的态度，及时对本书内容进行了更新，因此出版了第2版。为了让读者阅读到最新的内容，我们也相应地对新版进行了翻译。

新版的翻译工作是在第1版翻译工作的基础上进行的。新版新添加内容和更新内容的翻译工作主要由李徽完成，武传坤做了一些校对工作。武传坤将校对时发现的问题反馈给李徽，李徽进行修改后再传给武传坤，这样的校对工作反复进行了4次，其中第5章更是进行了多次修改。即便如此，我们对翻译的准确性仍没有完全的把握，欢迎读者批评指正。

最后感谢机械工业出版社对我们工作的信任，希望读者喜欢本书。

译 者

第2版前言

从2002年本书英文第1版问世以及随后的9次重印中，我收到了许多热情读者的来信以及在各种会议上的信息反馈。他们给了我很大的鼓励和很多的赞美之词。信息论、编码和密码学领域正在不断扩张，并且在过去的六年里，有很多新的思想产生、成熟，然后在行业标准和应用中采用。在第2版中，我们收录了许多新的知识，这能帮助那些从工科学院毕业的学生们和一些工程师们更快更有效地学习这些知识。

新版共添加了**14个小节、23个新例子以及48个新练习问题**。增加这些内容的目的有两个：

(1) 可以更加完整地介绍知识。这样的内容有：Shannon-Fano-Elias编码、算术编码、随机过程的熵率、马尔可夫链的熵率、最小距离的界、截短循环码、Reed-Solomon编码器详细的硬件实现、Diffie-Hellman密钥协商协议等。

(2) 可以向读者介绍当前研究趋势——那些在实际系统中应用的理论。这样的内容有：多输入多输出（MIMO）信道、MIMO系统的信道容量、低密度奇偶校验（LDPC）码、空时码、Turbo码的交织器设计、量子密码学、生物加密等。

本书的读者是电机工程系和计算机科学系的低年级研究生和高年级本科生。同样，本书也可以为那些希望开阔知识面并且掌握新技术的工程师们提供便利。

本书的结构非常适合作为研究生阶段的一门完整课。如果针对高年级本科生，下面一些章节可以跳过：1.11，1.13，2.7，2.8，3.14，3.15，4.11，5.8，5.9，5.10，6.13，7.7，7.8，7.9，8.13，8.14，8.15，8.16和8.17。上面所选择的章节同样可以作为研究机构为业界人士提供短期培训的内容。

下面是新版中增加的内容：

第1章：Shannon-Fano-Elias编码、算术编码、随机过程的熵率和马尔可夫链的熵率。

第2章：多输入多输出（MIMO）信道、MIMO系统的信道容量。

第3章：低密度奇偶校验（LDPC）码、最小距离的界、空时分组码。

第4章：准循环码和截短循环码。

第5章：Reed-Solomon编码器详细的硬件实现，实信道上RS码性能。

第6章：Turbo码的交织器设计。

第7章：空时格码。

第8章：素数、复杂性类、椭圆曲线密码学、Diffie-Hellman密钥协商协议、量子密码学、生物加密。

所有建议、意见和问题都欢迎发信给作者，E-mail地址是rbose@ee.iitd.ac.in。

祝阅读愉快！

第1版前言

信息论、错误控制编码和密码学是现代数字通信系统中的三大支柱。这三个课题都很大，而且针对其中的任何一个课题，都有很多好书加以讨论。本书试图用有限的篇幅将信息论、错误控制编码和密码学中所有重要的概念有机地结合起来，而不需要将书写得很厚。本书的意图就是使之成为一本简洁而生动的书。

本书是我在印度理工学院（Indian Institute of Technology, IIT）教授有关信息论和编码的不同课题的成果。在写本书的时候，我必须决定数学在本书中应占的分量。引用Richard W. Hamming的话："数学就是一种有趣的智力运动，但它不应该挡住获取物理过程中合理信息的路。"一本书若是太数学化就有吓倒缺乏强大数学功底的学生的危险。另一方面，如果需要把信息论和错误控制编码中的概念学到一定深度，那么数学的应用也不能无限度地减少。这样一来，就要掌握好分寸。我在本书中努力达到极好的折中：只有在非用不可的时候才用到数学。在可能的情况下都用直观的解释。我也相信借助实例来教学是很有效的方法，因此，当引入一个新概念时，我总试图给出至少一个例子。

如何阅读本书

本书不但是对信息论、编码和密码学这一令人着迷的领域的生动介绍，而且还涉及相当有深度的详细内容。全书共分三个逻辑部分：

第一部分：信息论和信源编码

第二部分：错误控制编码（信道编码）

第三部分：安全通信编码

第一部分包括两章——第1章讨论信息的概念及其有效的表示方式。信息的有效表示引起数据压缩。本章还介绍了游程编码的概念、率失真函数和优化量化器的设计。本章最后简单介绍了图像压缩。

第2章讨论通信信道和信道容量的概念。本章试图回答这样的问题：给定一个已知带宽和信噪比的信道，该信道每秒可传递多少比特的信息呢？这同时也提出了错误控制编码的必要性。

第二部分包括五章内容，都是关于错误控制编码的——第3章介绍线性分组码。线性分组码是很有实用价值、指导性强而且简单的一类码。我们将讨论这类码的编码和译码策略，同时还将介绍完备码、最优线性码和最大距离可分（MDS）码的概念。

第4章讨论的是循环码，这是线性分组码的子类。循环码对纠正突发性错误特别有用。Fire码、Golay码和循环冗余校验（CRC）码都是特殊类型的循环码，本章对它们也进行了讨论。本章以循环码的电路实现结束。

第5章将读者带到BCH（Bose-Chaudhuri Hocquenghem）码的世界，这是一类可纠正多个错误的功能极强的码。本章还讨论了Reed-Solomon码——BCH码的子类。

第6章讨论的是卷积码，这是一类本质上带记忆的码。本章将介绍网格码的概念并详细讨论维特比译码技术。还探讨了一些已知的好卷积码。最后介绍的是Trubo码，这是一类还不太旧的码。

第7章讨论网格编码调制（TCM），这是一种将编码和调制相结合的方案。本章将讨论TCM的编码和译码方法。读者将学到如何为加性高斯白噪声信道（additive white Gaussian noise channel）及衰退信道设计TCM方案。

第三部分仅包含关于密码学的一章——第8章，将介绍编码的另一种用法，即在安全通信方面的编码。本章将通过实例分别讨论保密密钥和公钥加密技术。还将讨论单向散列和应用浑沌函数进行加密等其他技术。本章在结尾时给出了关于密码学政治因素的一个注解。

我试图在所有需要的地方引入实例。每章在结束时都有一个结论性的评论，包含描述重要结果和贡献来源的简单历史性评注。每章最后还有一个简单总结，可作为概括性参考或对某一特殊公式或定义的快速查找工具，也可直接为读者考试前的准备增加信心。每章后面的练习题能帮助读者将文中讨论的概念具体化。每章后面还加进了基于计算机的练习题，建议将这些练习题变成学习本课程的一部分。

我尽了最大努力使书中没有错误，遗憾的是没有一种简单易行的错误控制技术。我试图包括所有与本领域有关的重要的、实际的和有趣的概念。欢迎读者将发现的错误、遗漏及其他建设性建议发送到rbose@ee.iitd.ac.in。

最后，我引用Blaise Pascal的话作结，他说："一个人在写书过程中了解到的最后一件事是把什么内容作为开始。"

致谢

我想感谢印度理工学院电机工程系提供的令人振奋的学术环境，特别感谢S. C. Dutta Roy教授、Surendra Prasad教授、H. M. Gupta教授、V. K. Jain教授、Vinod Chandra教授、Santanu Chaudhury教授、S. D. Joshi教授、Sheel Aditya教授、Devi Chadha教授、D. Nagchoudri教授、G. S. Visweswaran教授、R. K. Patney教授、V. C. Prasad教授、S. S. Jamuar教授和R. K. P. Bhatt教授。对与Subrat Kar博士、Ranjan K. Mallik博士和Shankar Prakriya博士的友好讨论也很感激。我很庆幸有几批杰出的学生，他们的反馈对改善本书内容很有帮助。每章后面的许多练习题都曾作为学生作业或考试题使用过。

我从内心感激宾夕法尼亚大学的Bernard D. Steinberg教授，他一直引领着我，是我的良师益友，也是我博士论文的指导教师。对每当我请求帮助时总是给予支持和建议的Tel Aviv大学的Avraham Freedman教授我也心存感激。感谢德国Darmstadt理工大学的Rolf Jakoby教授就各种主题所做的令人鼓舞的讨论。我想感谢印度科学学院（Indian Institute of Science）电子通信工程小组的B. Sundar Rajan教授，我们曾就撰写此书做了初步的讨论。

K. Vasudevan	IIT电机工程系，Kanpur
Pankaj Joshi	Jodhpur工程学院及研究中心电子与通信工程系，Jodhpur
Tanveer Hassan	阿里格尔穆斯林大学工学院，Aligarh
S. L. Maskara	Jaypee科技信息研究所，Noida
Azizar Rahaman	未来信息职业技术学院计算机工程与科学系，Kolkata
Indrajit Das	梅格纳德萨哈技术学院计算机工程与科学系，Kolkata
Anindya Jyoti Pal	Heritage技术研究所，Kolkata
B. P. Pati	ETCE部门马哈拉施特拉邦工程院院士，Pune
M. Murugan	VIIT ETCE部门，Pune
J. N. Sarvaiya	SVNIT电机工程系，Surat

Vikram Gadre	IIT电机工程系，Mumbai
M. Padmini	MVJ工程学院，Bangalore
Suresh Kuri	Gogte技术研究所，Belgaum
Mandi V. Mahalinga	安贝德卡技术研究所，Bangalore
Prakasa Rao	政府工程学院ECE部门，Warangal
C. R. Nataraj	Sri Jayachamarajendra技术学院（VTU），Mysore

我深深感激我的父母，因为他们给了我一生的爱和精神支持。我也感谢我的祖父母给我的祝福，和我弟弟Shantanu关于某些题目的无尽的讨论。

最后，我想感谢我的妻子也是最好的朋友，Aloka，她在我写本书的每一阶段都给予鼓励。她那建设性的意见和恰当的批评使本书更具有可读性。是她那无限的耐心、永久的支持、理解、爱、牺牲精神和幽默感才使我写这本书的梦想得以实现。

作者简介

Ranjan Bose 印度理工学院（IIT）电机工程系的教授。他在IIT（Kanpur分校）的电机工程系获得工学学士学位，在美国宾夕法尼亚大学电机工程系获得硕士和博士学位。之后在Alliance半导体公司任高级设计工程师。自1997年11月，他成为印度理工学院的教员。他在1999年获得URSI青年科学家奖，在2000年7月获得Humboldt研究奖金，在2003年获得印度国家科学院青年工程师奖，在2004年获得青年教师AICTE事业奖和2005年由印度科技部颁发的BOYSCAST奖金。

目　录

第三部分 安全通信编码

第一部分 信息论和信源编码

第1章 信源编码

并不是我们注意到的每一件事都重要，也不是每一件重要的事我们都注意到了。

爱因斯坦（1879—1955）

1.1 信息论简介

今天，我们生活在信息时代。因特网（Internet）已经成为我们生活中不可缺少的一部分，这使得太阳系第三大行星成为一个地球村。人们通过手机交谈已经是一件很平常的事。电影可以以DVD碟片的形式租回家欣赏。名片上印上电子邮箱和网址也很正常。许多人宁愿给朋友发送电子邮件和电子贺卡而不去发普通信件。股票行情也可以通过手机来查看。

信息已成为成功的关键（它一直是成功的关键之一，但在今天的世界上它是最关键的）。在所有这些信息的背后，信息的交换却依靠小小的1和0（即无所不在的比特），通过它们一个接一个地排在一起来表达信息。我们今天所生活的信息时代的存在主要归功于发表于1948年的一篇精辟的论文，这篇论文为奇妙的信息论奠定了基础。信息论的创始人是美国电子工程师Claude E. Shannon，他在发表于*Bell System Technical Journal*（1948）的论文"The Mathematical Theory of Communication"（通信的数学理论）中阐述了自己的思想。广义地说，信息包括一切标准通信媒体的内容，如电报、电话、无线电、电视以及来自电子计算机、伺服机械装置系统和其他数据处理器件的信号。该理论甚至可应用于人体和其他动物神经网络的信号。

信息论最关注的是发现能描述为通信和处理信息而设计的控制系统的数学定律。它建立量化指标来度量信息以及不同系统在传输、储存和处理信息时的容量。有些要解决的问题与发现最好的使用各种已有通信系统的方法相关，也和最好的将有用的信息或信号同无用的信息或噪声分开的方法有关。另一个问题就是对给定的信息载体（通常称为信道）给出容量上界。尽管主要是通信工程师对那些结果感兴趣，但有些概念已被诸如心理学和语言学等领域采用并发现它们很有用。

信息论的界限非常模糊。这种理论与通信理论有很大的重叠部分，但它主要面向信息处理和通信方面的基本限制，而较少涉及所用元器件的详细运作情况。

本章，我们将首先阐述对信息的直观理解。然后用数学模型来描述信息源以及对信息源所发出的信息的量化度量。然后我们将陈述并证明信源编码定理。有了基本的数学框架之后，我们将介绍四种信源编码技术，即**Huffman编码**、**Shannon-Fano-Elias编码**、**算术**（Arithmetic）**编码**和**Lempel-Ziv编码**。本章还将讨论**游程编码**（Run Length Encoding）、**率失真函数**（Rate Distortion Function）和**优化量化器**（Optimum Quantizer）。为了说明相关的随机变量，我们将学习随机过程的**熵率**。本章后面介绍图像压缩，它是信源编码的一个重要应用领域。特别是，我们将简单讨论JPEG（Joint Photographic Experts Group）标准。

1.2 不确定性和信息

任何信源，不管是模拟的还是数字的，都产生本质上随机的输出。假若不是随机，即我们能准确地知道其输出，那么就没必要传输它！信源存在**模拟信源**和**离散信源**两种。我们生活在一个模拟世界里，多数信源都是模拟信源，例如语音、温度波动等。离散信源都是人造的信源，例如从有限字母集中产生一连串字母（如，写电子邮件）的信源（如，人）。

在进一步介绍信息的数学度量之前，让我们先找一下直观感觉。请阅读下面的句子：

（1）明天太阳将从东边升起。

（2）在一小时后电话会响。

（3）今年冬天德里将下雪。

这三个句子带有不同量的信息。事实上，第一个句子几乎带不来任何信息，因为每个人都知道太阳从东边升起，也就是说这件事再次发生的概率几乎为1（N. Bohr说道："作预言是有风险的，特别是在涉及将来的时候。"）。第二个句子看来比第一个句子带来更多的信息，因为电话可能响，也可能不响。电话在一小时后响的概率是有限的（除非维修人员又在工作！）。最后一个句子你可能要读两遍，因为德里从来没下过雪，因此德里下雪的概率非常低。有趣的是，上述句子所携带的信息量与句子中所陈述的事件发生的概率有关，而且我们也观察到相反的关系。第一个句子讲的是发生概率几乎为1的事件，它几乎没带什么信息。第三个句子发生的概率很低，看来带来了不少的信息。（我们应读上两遍以确定没看错！）另外可以注意到句子的长度与所携带的信息量无关。事实上，第一个句子最长但所携带的信息量最少。

现在我们将建立信息的数学度量。

定义1.1 考虑可能输出为x_i，$i=1,2,\cdots,n$的离散随机变量X。则事件$X=x_i$的**自信息**（self-information）定义为

$$I(x_i)=\log\left(\frac{1}{P(x_i)}\right)=-\log P(x_i) \tag{1-1}$$

我们注意到高概率事件没有低概率事件所携带的信息多。对于$P(x_i)=1$的事件，有$I(x_i)=0$。因为低概率事件意味着高度的不确定性（反之亦然），具有高度不确定性的随机变量带有更多的信息。我们在本章中将一直用不确定性和信息量的这种相关性作自然解释。

$I(x_i)$的单位取决于取对数的底数，通常取为2或e。当以2为底数时，单位为**比特**（bit），而当以e为底数时，单位为**奈特**（nat，自然单位）。由于$0\leqslant P(x_i)\leqslant 1$，故$I(x_i)\geqslant 0$，即自信息是非负的。下面的两个例子说明为什么用对数来度量信息是合适的。

例1.1 考虑一个掷硬币的二元信源：若正面（H）出现则输出1，若反面（T）出现则输出0。对于该信源，有$P(1)=P(0)=0.5$。来自该信源的每一个输出所含的信息量为

$$\begin{aligned} I(x_i)&=-\log_2 P(x_i)\\ &=-\log_2 P(0.5)=1\ \text{(bit)} \end{aligned} \tag{1-2}$$

事实上，我们只能用1比特来表示来自这个二元信源的输出（例如，我们用1代表H，用0代表T）。

现在假定从该二元信源连续输出是统计独立的，即信源是无记忆的。考虑一个m bit的数组，则共有2^m个可能的m比特组，每一个都等可能地具有概率2^{-m}。

一个m比特组的自信息为

$$I(x_i) = -\log_2 P(x_i)$$
$$= -\log_2 2^{-m} = m \text{ (bit)} \tag{1-3}$$

同时，我们也注意到确实需要m个比特来表示可能的m比特组。

因此，当信源的一些输出被看做一个组的时候，这种信息的对数度量具有所期望的可加性。

例1.2 考虑一个离散无记忆信源（DMS）（信源C），它的输出为每次两个比特。该信源由例1.1中提到的两个二元信源（信源A和B）构成，每个信源贡献一个比特。信源C中的这两个二元信源是独立的。从直觉上看，组合信源（信源C）的信息量应该是组成该信源的两个独立信源的输出所含信息之和。让我们分析一下信源C的输出所含的信息量。有四种可能的情况{00，01，10，11}，每种情况都具有概率$P(C)=P(A)P(B)=(0.5)\times(0.5)=0.25$，这是因为信源$A$和$B$是独立的。信源$C$的每个输出所含的信息量为：

$$I(C) = -\log_2 P(x_i)$$
$$= -\log_2 P(0.25) = 2 \text{ (bit)} \tag{1-4}$$

我们不得不用两比特来表示这个组合二元信源的输出。

因此，信息的对数度量对独立事件具有可加性。

下面考虑两个离散随机变量X和Y，其可能的输出分别为x_i，$i=1, 2, \cdots, n$和y_j，$j=1, 2, \cdots, m$。假设我们观察到输出$Y=y_j$，想由此确定该事件所提供的关于事件$X=x_i$，$i=1, 2, \cdots, n$的信息量，也就是说我们想用数学方式表示出它们的互信息。我们注意到两种极端的情况：

（1）X和Y独立，在此情况下，$Y=y_j$的出现不提供任何$X=x_i$的信息。

（2）X和Y为完全依赖的事件，在此情况下，$Y=y_j$的出现决定了$X=x_i$的出现。

满足此条件的一种合适的度量是对条件概率

$$P(X=x_i|Y=y_j) = P(x_i|y_j) \tag{1-5}$$

除以概率

$$P(X=x_i) = P(x_i) \tag{1-6}$$

所得的商取对数。

定义1.2 定义x_i, y_j之间的**互信息**$I(x_i; y_j)$为

$$I(x_i; y_j) = \log\left(\frac{P(x_i|y_j)}{P(x_i)}\right) \tag{1-7}$$

同前面一样，$I(x)$的单位取决于取对数的底数，通常为2或e。当底数为2时，单位为比特。注意

$$\frac{P(x_i|y_j)}{P(x_i)} = \frac{P(x_i|y_j)P(y_i)}{P(x_i)P(y_i)} = \frac{P(x_i, y_j)}{P(x_i)P(y_j)} = \frac{P(y_j|x_i)}{P(y_j)} \tag{1-8}$$

因此

$$I(x_i; y_j) = \log\left(\frac{P(x_i|y_j)}{P(x_i)}\right) = \log\left(\frac{P(y_j|x_i)}{P(y_j)}\right) = I(y_j; x_i) \tag{1-9}$$

对$I(x_i; y_j)=I(y_j; x_i)$的自然解释如下：事件$Y=y_j$的出现所提供的关于$X=x_i$的信息完全等同于事件$X=x_i$的出现所提供的关于$Y=y_j$的信息。

现在让我们查看两种极端情况：

(1) 当随机变量X与Y统计独立时，有$P(x_i|y_j)=P(x_i)$，它导致$I(x_i; y_j)=0$。

(2) 当事件$Y=y_j$的出现惟一决定事件$X=x_i$的出现时，有$P(x_i|y_j)=1$，从而互信息为

$$I(x_i; y_j)=\log\left(\frac{1}{P(x_i)}\right)=-\log P(x_i) \tag{1-10}$$

这就是事件$X=x_i$的自信息。

因此，互信息的对数定义确认了我们的直觉。

例1.3 考虑图1-1所示的**二元对称信道**（BSC）。这是一种从发射端（Tx）到接收端（Rx）传递许多1和0的信道。它偶尔以概率p发生错误。一个BSC会把1变为0，也以同样概率把0变为1。令随机变量X和Y分别表示该BSC的输入和输出，并设输入符号等可能地出现，而且输出符号根据下面的信道转移概率依赖于输入：

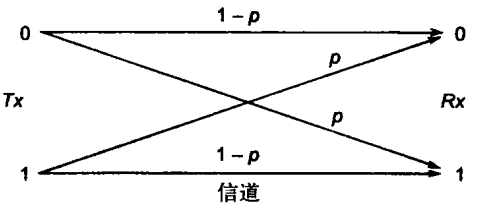


图1-1　一个二元对称信道

$P(Y=0|X=0)=1-p$

$P(Y=0|X=1)=p$

$P(Y=1|X=1)=1-p$

$P(Y=1|X=0)=p$

这说明在该BSC中传输时，数字发生转变（即出错）的概率为p。从这些信道转移概率中我们得到

$$\begin{aligned}P(Y=0)&=P(X=0)\times P(Y=0\mid X=0)+P(X=1)\times P(Y=0\mid X=1)\\&=0.5(1-p)+0.5(p)=0.5\\P(Y=1)&=P(X=0)\times P(Y=1\mid X=0)+P(X=1)\times P(Y=1\mid X=1)\\&=0.5(p)+0.5(1-p)=0.5\end{aligned}$$

假定我们在接收端根据接收到的信号想确定从发送端发送的是什么。在给定$Y=0$的情况下，关于事件$X=0$发生的互信息为

$$I(x_0; y_0)=I(0;0)=\log_2\left(\frac{P(Y=0\mid X=0)}{P(Y=0)}\right)=\log_2\left(\frac{1-p}{0.5}\right)=\log_2 2(1-p)$$

类似地，

$$I(x_1; y_0)=I(1;0)=\log_2\left(\frac{P(Y=0\mid X=1)}{P(Y=0)}\right)=\log_2\left(\frac{p}{0.5}\right)=\log_2 2p$$

下面我们考虑一些特殊情况。

假定$p=0$，即为理想信道（无噪声），则

$$I(x_0; y_0)=I(0;0)=\log_2 2(1-p)=1\ \text{(bit)}$$

因此，从输出我们可以确定所传送的是什么。回想一下事件$X=x_0$的自信息也是1bit。

但是，若$p=0.5$，我们得到

$$I(x_0; y_0)=I(0;0)=\log_2 2(1-p)=\log_2 2(0.5)=0$$

很明显，从输出我们得不到关于发送的是什么的任何信息。因此，这是个无用信道。对于

这样的一个信道，我们可以在接收端用掷硬币的方法来确定发送的是什么。

假设我们有一个$p=0.1$的信道，则

$$I(x_0; y_0)=I(0; 0)=\log_2 2(1-p)=\log_2 2(0.9)=0.848 \text{ (bit)}$$

例1.4 令X和Y为表示图1-2所示的二元信道的输入和输出的二元随机变量。假定输入的符号等可能地选取，而且输出符号根据下面的信道转移概率依赖于输出：

$P(Y=0|X=0)=1-p_0$

$P(Y=0|X=1)=p_1$

$P(Y=1|X=1)=1-p_1$

$P(Y=1|X=0)=p_0$

由信道转移概率我们得到

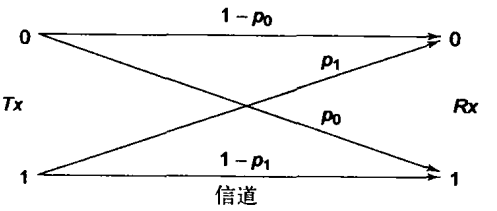


图1-2 一个非对称概率的二元信道

$$\begin{aligned}P(Y=0) &= P(X=0).P(Y=0\mid X=0)+P(X=1).P(Y=0\mid X=1)\\ &= 0.5(1-p_0)+0.5(p_1)=0.5(1-p_0+p_1)\\ P(Y=1) &= P(X=0).P(Y=1\mid X=0)+P(X=1).P(Y=1\mid X=1)\\ &= 0.5(p_0)+0.5(1-p_1)=0.5(1-p_1+p_0)\end{aligned}$$

假设我们在接收端想根据所收到的信号来确定在发送端发送的是什么。在给定$Y=0$的情况下，关于事件$X=0$发生的互信息为

$$I(x_0; y_0)=I(0; 0)=\log_2\left(\frac{P(Y=0|X=0)}{P(Y=0)}\right)=\log_2\left(\frac{1-p_0}{0.5(1-p_0+p_1)}\right)=\log_2\left(\frac{2(1-p_0)}{1-p_0+p_1}\right)$$

类似地，

$$I(x_1; y_0)=I(1; 0)=\log_2\left(\frac{P(Y=0|X=1)}{P(Y=0)}\right)=\log_2\left(\frac{2p_1}{1-p_0+p_1}\right)$$

定义1.3 定义在给定$Y=y_j$的情况下，事件$X=x_i$的**条件自信息** (conditional self information)为

$$I(x_i\mid y_j)=\log\left(\frac{1}{P(x_i|y_j)}\right)=-\log P(x_i\mid y_j) \tag{1-11}$$

由此，我们可以写为

$$I(x_i; y_j)=I(x_i)-I(x_i\mid y_j) \tag{1-12}$$

条件自信息可解释为在事件$Y=y_j$的基础上关于事件$X=x_i$的自信息。回顾上述结论$I(x_i)\geqslant 0$及$I(x_i|y_j)\geqslant 0$，因此，当$I(x_i)<I(x_i|y_j)$时，有$I(x_i; y_j)<0$，而当$I(x_i)>I(x_i|y_j)$时，有$I(x_i; y_j)>0$。因此，互信息可为正、负或0。

例1.5 考虑例1.3中的BSC。互信息$I(x_0; y_0)$关于错误概率p的对应平面图如图1-3所示。

从图1-3可以看出，当$p>0.5$时，$I(x_0; y_0)$为负值。对此的自然解释为：一个负值互信息表明我们在观察到$Y=y_0$时，应避免选取$X=x_0$作为发送的比特。

对$p=0.1$，

$$I(x_0; y_1)=I(0; 1)=\log_2 2(p)=\log_2 2(0.1)=-2.322 \text{ (bit)}$$

这表明对$p=0.1$，事件$X=x_0$和$Y=y_1$之间的互信息为负值。对$p=1$时的极端情况，我们有

$$I(x_0; y_1)=I(0; 1)=\log_2 2(p)=\log_2 2(1)=1 \ \text{(bit)}$$

信道总是把0变成1，反过来也如此（因为$p=1$）。这表明若在接收端观察到y_1，则可以下结论说x_0是实际被发送的。这实际是个有着100%比特错误率的有用信道！我们仅需把收到的比特反过来就行了。

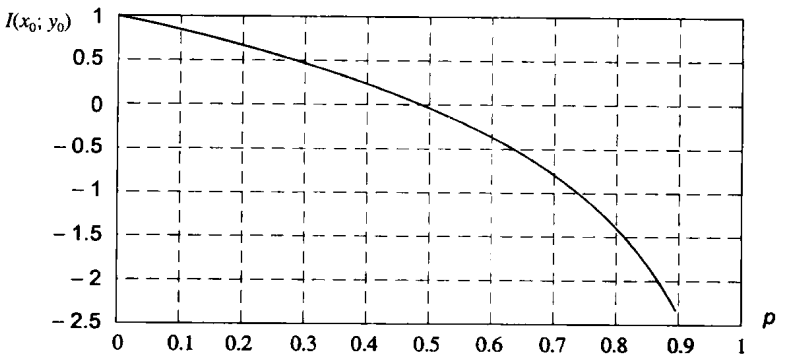


图1-3　互信息$I(x_0; y_0)$关于错误概率p的对应平面图

1.3　平均互信息和熵

到现在为止，我们已经学习了与事件对x_i和y_j相联系的互信息，它们是两个随机变量X和Y的可能输出。我们现在想找到这两个随机变量之间的平均互信息。这可以通过对$I(x_i; y_j)$以相联事件的出现概率加权，然后对所有这些相联事件求和得到。

定义1.4　两个随机变量X和Y的**平均互信息**为

$$\begin{aligned} I(X;Y)&=\sum_{i=1}^{n}\sum_{j=1}^{m}P(x_i,y_j)I(x_i;y_j)=\sum_{i=1}^{n}\sum_{j=1}^{m}P(x_i,y_j)\log\frac{P(x_i,y_j)}{P(x_i)P(y_j)} \\ &=\sum_{j=1}^{n}\sum_{i=1}^{m}P(x_i)P(y_j\mid x_i)\log\frac{P(y_j\mid x_i)}{P(y_j)} \\ &=\sum_{j=1}^{n}\sum_{i=1}^{m}P(y_j)P(x_i\mid y_j)\log\frac{P(x_i\mid y_j)}{P(x_i)} \end{aligned} \tag{1-13}$$

对于X与Y统计独立的情况，有$I(X; Y)=0$，即X和Y之间没有平均互信息。平均互信息的一个重要性质是$I(X; Y)\geqslant 0$，其中当且仅当X与Y统计独立时，等号成立。

定义1.5　随机变量X的**平均自信息**定义为

$$H(X)=\sum_{i=1}^{n}P(x_i)I(x_i)=-\sum_{i=1}^{n}P(x_i)\log P(x_i) \tag{1-14}$$

其中X代表信源可能输出的字母符号，$H(X)$代表每个信源字母的平均信息。在此情况下，我们称$H(X)$为**熵**（entropy）。X的熵可以表示为$\log\left(\dfrac{1}{P(X)}\right)$的期望值。术语熵取自统计力学，在该学

科中它用来表示一个系统的无序程度。

我们可以看出，$0 \leqslant P(x_i) \leqslant 1$，$\log\left(\frac{1}{P(X)}\right) \geqslant 0$，那么$H(X) \geqslant 0$。

例1.6 考虑一个产生统计独立的符号序列的离散二元信源，其输出为0的概率为p，输出为1的概率为$1-p$。该二元信源的熵为：

$$H(X) = -\sum_{i=0}^{1} P(x_i)\log P(x_i) = -p\log_2(p) - (1-p)\log_2(1-p) \tag{1-15}$$

该**二元熵函数**相对于p的平面图如图1-4所示。

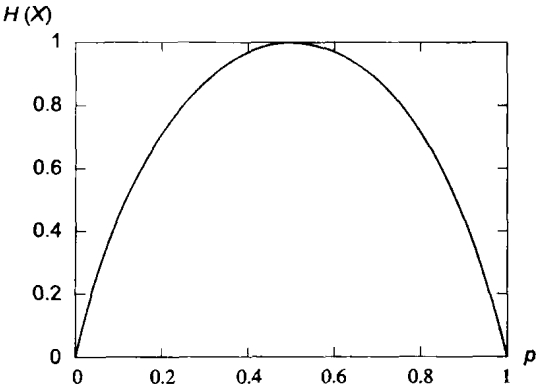


图1-4 二元熵函数$H(X) = -p\log_2(p) - (1-p)\log_2(1-p)$

从图中我们观察到该二元熵函数在$p = 0.5$时，即当0和1出现机率相同时达到最大值。一般情况下可以证明，一个离散信源在其字母出现概率相同时，熵的值达到最大。

例1.7 考虑使用字母{A, B, …, Z}的英文语言。如果每一个字母出现的概率一样，并且相互独立，则每一个字母的熵是$\log_2 26 = 4.70$。这是一个绝对上界。但是我们知道所有字母并不是以相同概率出现的。如果我们考虑不同字母出现概率不同的情况（正常的字母出现频率），每一个字母的熵H_L是

$$H_L \leqslant H(X) \approx 4.14 \text{ (bit)} \tag{1-16}$$

这仍然是一个高估值，因为字母的出现概率不是相互独立的。例如，Q后面经常跟着U，双字母组TH经常出现。因此，通过把双字母组出现的情况考虑进来，我们可以得到对熵的一个更好的估计值。如果X^2表示英语中双字母组的随机变量，则H_L的上界可以精确为

$$H_L \leqslant \frac{H(X^2)}{2} \approx 3.56 \text{ (bit)} \tag{1-17}$$

这种方法可以扩展到n字母组的情况。因此，英文语言的熵可以定义为

$$H_L = \lim_{n\to\infty} \frac{H(X^n)}{n} \tag{1-18}$$

虽然很难确定H_L的准确值，但统计分析表明，对英语来说，

$$1 \leqslant H_L \leqslant 1.5 \text{ (bit)} \tag{1-19}$$

因此，每个英语字母最多只有1.5bit的信息。我们假设$H_L \approx 1.25$bit。因此英语的冗余度是

$$R_{Eng} = 1 - \frac{H_L}{\log_2 26} = 1 - \frac{1.25}{4.70} = 0.75 \tag{1-20}$$

这就表明，理论上，英文文章可以无损耗地压缩为其原来长度的四分之一。有趣的是，世界上大部分自然语言都有相似的冗余度。

我们简要地分析英语口语的冗余度。假设一个平常人平均每分钟说60个单词，每个单词平均长度是6个字母。每秒所说的字母平均是6个。假设每个字母带有1.25bit的信息，那么一个平常人说话的信息率是7.5bit/s。如果每个字母用5bit来表示，那么正常人说话的比特率是30bit/s。但是，一般说话所要求的数据率是32Kbit/s。

定义1.6　平均条件自信息又称作**条件熵**，定义为

$$H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j)\log\frac{1}{P(x_i|y_j)} \tag{1-21}$$

对该定义的自然解释如下：$H(X|Y)$ 是在观察到Y的情况下，X中的信息（或不确定性）。根据$H(X|Y)$ 和$H(Y|X)$ 的定义，我们可以写为

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{1-22}$$

我们观察到下面的结论：

(1) 因为$I(X; Y)\geqslant 0$，这表明$H(X)\geqslant H(X|Y)$。

(2) $I(X; Y)=0$的情况表明$H(X)=H(X|Y)$，并且当且仅当X和Y统计独立时这种情况才可能发生。

(3) 由于$H(X|Y)$ 是在Y给定的情况下关于X的平均不确定性（信息），并且$H(X)$ 是X的平均不确定性（自信息），因此$I(X; Y)$ 是在观察到Y的情况下关于X的平均不确定性（互信息）。

(4) 因为$H(X)\geqslant H(X|Y)$，对Y的观察结果不增加熵（不确定性），只能降低熵。也就是说，观察Y不会降低关于X的信息，它只能增加这种信息。

定义1.7　一对联合概率分布是$p(x, y)$的离散随机变量(X, Y)的**联合熵**定义为：

$$H(X; Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_i)\log p(x_i, y_j) \tag{1-23}$$

通过使用$H(X)$、$H(X, Y)$和$H(X|Y)$的数学定义，我们有下面的规则：

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y) \tag{1-24}$$

从(1-22)和 (1-24)可以得出

$$I(X; Y) = H(X) + H(Y) - H(X, Y) \tag{1-25}$$

最后，我们注意到

$$I(X; X) = H(X) - H(X|X) = H(X) \tag{1-26}$$

这就解释了为什么在定义1.5中平均自信息也被称为熵。熵和互信息之间的关系可以用Venn图来表示，如图1-5所示。

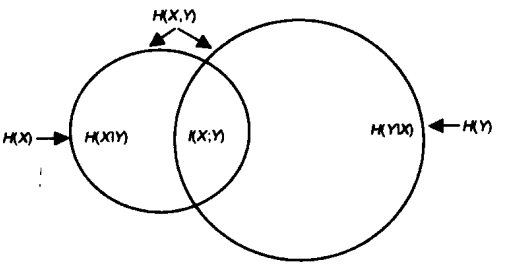


图1-5　熵与互信息之间的关系

例1.8　考虑例1.3中讨论的BSC。假设输入符号是“0”的概率为q，是“1”的概率为$1-q$，如图1-6所示。

该二元信源的熵为

$$H(X) = -\sum_{i=0}^{1} P(x_i)\log P(x_i) = -q\log_2(q) - (1-q)\log_2(1-q)$$

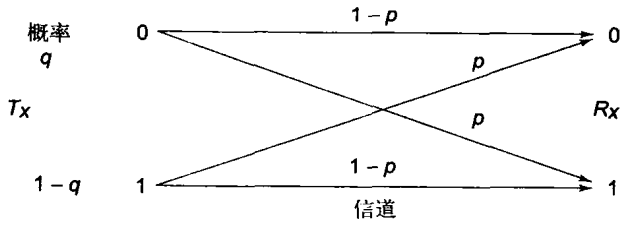


图1-6 一个输入符号概率为q和$1-q$的二元对称信道（BSC）

条件熵则由下式给出

$$H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j)\log\frac{1}{p(x_i|y_j)} \tag{1-27}$$

为了计算$H(X|Y)$的值，我们用到下列等式

$$P(x_i, y_j) = P(x_i \mid y_j)\,P(y_j) = P(y_j \mid x_i)\,P(x_i) \tag{1-28}$$

$H(X|Y)$相对q的平面图由图1-7给出，其中p是个参数。

平均互信息$I(X; Y)$在图1-8中给出。从平面图中可以看出，当参数p的值从0增加到0.5时，$I(X; Y)$在递减。这直观地表明，当信道的可靠性下降（增加$p \leqslant 0.5$的值）时，随机变量X（在发送端）与随机变量Y（在接收端）的互信息减少。

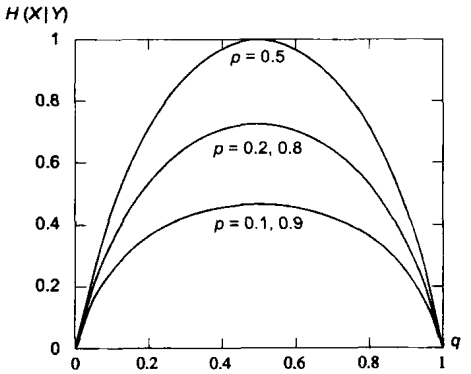


图1-7 条件熵$H(X|Y)$相对q的平面图

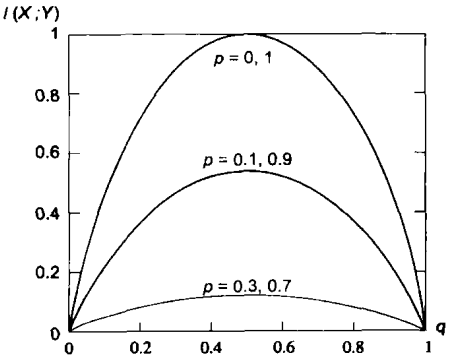


图1-8 平均互信息$I(X; Y)$相对q的平面图

1.4 连续随机变量的信息度量

对离散随机变量的互信息的定义可以直接推广到连续随机变量。设X和Y是两个随机变量，它们的联合概率密度函数（pdf）为$p(x, y)$，边界pdf为$p(x)$和$p(y)$。X和Y的平均互信息由下列定义给出。

定义1.8 *两个连续随机变量X和Y之间的*平均互信息*定义为*

$$I(X;Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(x)p(y|x)\log\frac{p(y|x)p(x)}{p(x)p(y)}\,\mathrm{d}x\mathrm{d}y \tag{1-29}$$

应该指出的是，平均互信息的定义可以从离散随机变量到连续随机变量，但其概念和自然解释却不能。原因是连续随机变量所带的信息量实际上是无限的，我们需要用无限比特才能精确表示一个连续随机变量。自信息和熵因此也是无限的。为了解决这个问题，我们定义一个叫微分熵（differential entropy）的量。

定义1.9　连续随机变量X和Y的**微分熵**定义为：

$$h(X)=-\int_{-\infty}^{\infty}p(x)\log p(x)\,\mathrm{d}x \tag{1-30}$$

再一次强调，上述定义的量没有自然意义。我们将进一步推广这个定义。

定义1.10　连续随机变量X和Y的**平均条件熵**定义为

$$h(X\mid Y)=-\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}p(x,y)\log p(x\mid y)\,\mathrm{d}x\mathrm{d}y \tag{1-31}$$

平均互信息可以表示为

$$I(X;Y)=h(X)-h(X\mid Y)=h(Y)-h(Y\mid X) \tag{1-32}$$

1.5　信源编码定理

在本节中我们将探索一个信源所产生的符号的有效表示（有效编码），基本目的就是通过对符号的有效表示进行数据压缩。假设一个离散无记忆信源（DMS）每t秒输出一个符号，每个符号都是从有限符号集合x_i，$i=1,2,\cdots,L$以概率$P(x_i)$，$i=1,2,\cdots,L$出现。该DMS每个符号的熵为

$$H(X)=-\sum_{j=1}^{L}P(x_i)\log_2 P(x_i)\leqslant\log_2 L \tag{1-33}$$

当所有符号等可能出现时，上述等号成立。这说明平均每个信源符号的比特数是$H(X)$，因而信源输出率为$H(X)/t$ bit/s。

现在让我们用比特表示英语字母表中的26个字母。我们注意到$2^5=32>26$，因此每个字母可以惟一地用5bit来表示。这就是**定长编码**（FLC）的一个例子。每个字母对应一个5bit长的**码字**。

定义1.11　一个**编码**就是所有**码字**的集合。

假设一个DMS信源发出的符号是从一个有限符号集合x_i，$i=1,2,\cdots,L$中选取的。当L是2的幂时，惟一编码需要的比特数R为

$$R=\log_2 L \tag{1-34}$$

而当L不是2的幂时，有

$$R=\lfloor\log_2 L\rfloor+1 \tag{1-35}$$

正如我们前面所观察到的，要对英语字母表中的字母进行编码，需要$R=\lfloor\log_2 26\rfloor+1=5$ bit。对英语字母表的FLC说明字母表中的每个字母都（可能）同等重要，因此，每一个都需要5bit来表示。但是，我们知道某些字母没那么常用（如x、q、z等），而另外一些则被更频繁地用到（如s、t、e等）。看来用同样的比特数来表示经常用到的字母和不常用的字母，这并不是一种有效的表示（编码）方法。直观上，我们应该用少量的比特数来表示经常用到的字母，用多一些的比特

数来表示不经常用到的字母。用这种方式，如果我们要对一整页文字进行编码，可能会在总体上用到较少的比特数。当信源符号不是等可能地出现时，更有效的方法是**变长编码**（VLC）。

例1.9 假设在我们的词汇表中只有英语字母表中的前8个字母（A～H）。对这个字母集合的**定长编码**（FLC）为

字 母	码 字	字 母	码 字
A	000	E	100
B	001	F	101
C	010	G	110
D	011	H	111

定长编码

对同样字母集合的变长编码（VLC）可以是

字 母	码 字	字 母	码 字
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

变长编码1

假设我们要对一系列字母“A BAD CAB”进行编码，则对这个句子的定长和变长表示为

定长编码	000 001 000 011 010 000 001	总比特数＝21
变长编码	00 010 00 100 011 00 010	总比特数＝18

可注意到变长编码用了较少的比特数，只是因为在句子中出现频率较高的字母用了较少的比特数来表示。

我们再来看一下对英语字母表中前8个字母的另一种VLC：

字 母	码 字	字 母	码 字
A	0	E	10
B	1	F	11
C	00	G	000
D	01	H	111

变长编码2

这第二种变长编码在字母的表示方面表现出更高的效率。

变长编码1	00 010 00 100 011 00 010	总比特数＝18
变长编码2	0 1001 0001	总比特数＝9

但VLC2有一个问题。考虑这里的比特序列0 1001 0001，它是用来表示“A BAD CAB”的。我们可以将这些比特用另一种方式重新分组为[0] [10][0][1] [0][0][01]，它被翻译成A EAB AAD，或[0] [1][0][0][1] [0][0][0][1]，它代表A BAAB AAAB！显然惟一译码有问题。因为码字的长度是变化的，我们不知道哪里是一个码字（符号）的结束，哪里是下一个码字（符号）的开始。但VLC1不存在这个问题，这里没有一个码字构成另一个码字的前缀。这叫

做**前缀条件**（prefix condition）。一旦检测到一个比特序列对应可能码字中的某一个，我们就做译码判决。这种码称为**即时码**（instantaneous code），并不造成译码延迟。如果允许译码延迟，我们可以使用译码字符只能由一个可能输入字符产生的**惟一可译码**（uniquely decodable code）。在这个例子中，VLC2不是惟一可译码的，因此无实用价值。VLC1是惟一可译码的，尽管在每个符号所占的比特数方面不是很经济。

定义1.12　前缀码是一种没有一个码字构成另一码字的前缀的码。这种码又称为**即时码**。

我们现在设计一种系统方法来构造惟一可译码的变长码，使其在每个信源字母平均所占的比特数方面是高效的。设信源的输出符号都来自于一个有限符号集合$x_i, i=1, 2, \cdots, L$，其出现的概率为$P(x_i), i=1, 2, \cdots, L$。每个信源字母平均所占用的比特数为

$$\overline{R} = \sum_{i=1}^{1} n_i P(x_i) \tag{1-36}$$

其中n_i是符号x_i所对应码字的长度（比特数）。

定理1.1　（Kraft不等式）存在码字长度为$n_1 \leqslant n_2 \leqslant \cdots \leqslant n_L$的二元码的充分必要条件是满足前缀条件

$$\sum_{k=1}^{L} 2^{-n_k} \leqslant 1 \tag{1-37}$$

证明　首先证明充分性。考虑阶数（深度）为$n=n_L$的二叉树。这个树有2^n个端节点，如图1-9所示。让我们选任意的阶数为n_1的码作为第一个码字c_1。因为任何一个码字都不能成为其他码字的前缀（前缀条件），所以这种选择排除了2^{n-n_1}个端节点。继续这个过程直到最后一个码字被赋予端节点$n=n_L$。考虑阶数为$j<L$的节点，被排除的端节点的个数占总节点的个数的比例为

$$\sum_{k=1}^{j} 2^{-n_k} < \sum_{k=1}^{L} 2^{-n_k} \leqslant 1 \tag{1-38}$$

因此，我们可以构造一个嵌入到有n_L个节点的全树中的前缀码。被排除的节点在图中用一个指向它们的带虚线的箭头标出。

现在证明必要性。我们注意到在阶数为$n=n_L$的码树中，从总数为2^n的端节点中排除的端节点的个数为

$$\sum_{k=1}^{L} 2^{n-n_k} \leqslant 2^n \tag{1-39}$$

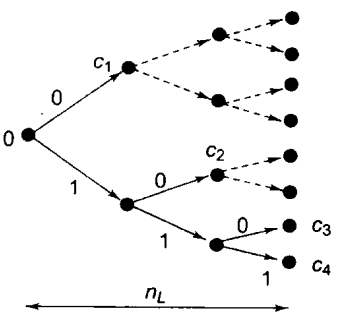


图1-9　一个阶数为n_L的二叉树

这导致

$$\sum_{k=1}^{L} 2^{-n_k} \leqslant 1 \tag{1-40}$$

证明可以很容易地扩展到一个规模为M的字母表对应的前缀码上。相应地，我们只需要将二叉树变成M叉树即可。

对应的不变式变成

$$\sum_{k=1}^{L} M^{-n_k} \leqslant 1 \tag{1-41}$$

例1.10 考虑利用二叉树构造前缀码。

我们从二叉树的母节点开始进行构造（如图1-10所示）。将母节点标记为‘0’（也可以标记为‘1’）。每个节点发出两个分支（二叉树）。让我们用‘0’标记上面的分支，用‘1’标记下面的分支（这种标记可以相互交换）。首先我们从母节点往上面的分支走，这样我们得到在节点n_{00}处结束的第一码字$c_1=0$。因为我们想构造一个前缀码，它的任何码字都不是任何其他码字的前缀，我们必须放弃所有由标记为c_1的节点所产生的子节点。

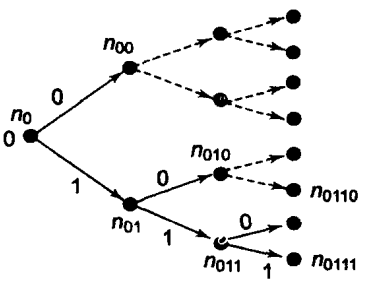


图1-10 利用二叉树构造二元前缀码

接下来，我们从母节点往下面的分支走，一直到节点n_{01}处。然后我们先从上面的分支到达节点n_{010}，我们把这个码字记为$c_2=10$（这个分支的标记是指从母节点过来的）。从节点n_{01}下面的分支，我们最终到达端节点n_{0110}和n_{0111}，它们分别对应码字$c_3=110$和$c_4=111$。

因此这棵二叉树给了我们四个前缀码字：{0，10，110，111}。从构造过程可知道这是一个前缀码。对于这个码，下式成立。

$$\sum_{k=1}^{L} 2^{-n_k} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 0.5 + 0.25 + 0.125 + 0.125 = 1$$

因此，它满足Kraft不等式。

我们现在叙述并证明无噪声信源编码定理，它适用于满足前缀条件的码。

定理1.2 （**信源编码定理**）设X为离散无记忆信源（DMS）的字母集合，有着有限熵$H(X)$，而且其输出符号x_k，$k=1, 2, \cdots, L$出现的概率为$P(x_k)$，$k=1, 2, \cdots, L$。给出这些参数后，则可以构造满足前缀条件的码，其平均码长R满足不等式

$$H(X) \leqslant \overline{R} < H(X) + 1 \tag{1-42}$$

证明 首先考虑该不等式的下界。对于长度为n_k，$1 \leqslant k \leqslant L$的码字，差$H(X)-\overline{R}$可以表示为

$$H(X)-\overline{R} = \sum_{k=1}^{L} P(x_k)\log_2\frac{1}{P(x_k)} - \sum_{k=1}^{L} P(x_k)n_k = \sum_{k=1}^{L} P(x_k)\log_2\frac{2^{-n_k}}{P(x_k)}$$

我们现在利用不等式$\ln x \leqslant x-1$来得到

$$H(x)-\overline{R} \leqslant (\log_2 \mathrm{e}) \sum_{k=1}^{L} P(x_k)\left(\frac{2^{-n_k}}{p(x_k)} - 1\right) \leqslant (\log_2 \mathrm{e})\left(\sum_{k=1}^{L} 2^{-n_k} - 1\right) \leqslant 0$$

最后一个不等式是由Kraft不等式得到的。当且仅当对所有$1 \leqslant k \leqslant L$都满足$P(x_k)=2^{-n_k}$时，上式中的等号成立。因此下界得到证明。

下面我们证明上界。让我们选取码字长度n_k使其满足$2^{-n_k} \leqslant P(x_k) < 2^{-n_k+1}$。首先考虑$2^{-n_k} \leqslant P(x_k)$。将$1 \leqslant k \leqslant L$对两边求和，我们得到

$$\sum_{k=1}^{L} 2^{-n_k} \leq \sum_{k=1}^{L} P(x_k) = 1$$

这就是Kraft不等式，对此存在满足前缀条件的码。

下面考虑$P(x_k) < 2^{-n_k+1}$。对两边取对数得到

$$\log_2 P(x_k) < -n_k + 1$$

或者

$$n_k < 1 - \log_2 P(x_k)$$

两边乘$P(x_k)$，之后对$1 \leqslant k \leqslant L$求和，可得

$$\sum_{k=1}^{L} P(x_k) n_k < \sum_{k=1}^{L} P(x_k) + \left(-\sum_{k=1}^{L} P(x_k) \log_2 P(x_k) \right)$$

或者

$$\bar{R} < H(X) + 1$$

因此上界得证。

信源编码定理告诉我们，对于任意用来表示一个信源中符号的前缀码，用于表示信源符号的**最小**比特数平均必须至少等于该信源的熵。如果对某个信源X，我们找到了一种满足$\bar{R}=H(X)$的前缀码，则必须放弃进一步搜索，因为不可能找到更好的。该定理还告诉我们，一个有高熵（不确定性）的信源用前缀码平均需要更多的比特数来表示信源的符号。

定义1.13　前缀码的**效率**定义为

$$\eta = \frac{H(x)}{\bar{R}} \tag{1-43}$$

由信源编码定理可知前缀码的效率满足$\eta \leqslant 1$。对符号的有效表示导致对数据的**压缩**。信源编码主要用于数据（语音、文本、图像、视频等）压缩。

例1.11　考虑一个以概率$P(x_1)=0.5$, $P(x_2)=0.3$, $P(x_3)=0.1$, $P(x_4)=0.1$产生四个符号的信源X。该信源的熵为

$$H(X) = -\sum_{k=1}^{4} P(x_k) \log_2 P(x_k) = 1.685 \text{ (bit)}$$

假定我们使用例1.10中的前缀码{0，10，110，111}，则平均码字长度$\bar{R}$为

$$\bar{R} = \sum_{k=1}^{4} n(x_k) P(x_k) = 1(0.5) + 2(0.3) + 3(0.1) + 3(0.1) = 1.700 \text{ (bit)}$$

因此，我们得到

$$H(X) \leqslant \bar{R} \leqslant H(X) + 1$$

该码的效率为$\eta=(1.685/1.700)=0.9912$。假若信源符号的概率为$P(x_k)=2^{-n_k}$，即$P(x_1)=2^{-1}=$

0.5，$P(x_2)=2^{-2}=0.25$，$P(x_3)=2^{-3}=0.125$，$P(x_4)=2^{-3}=0.125$，那么平均码字长度则为，$\bar{R}=$1.750比特$=H(X)$。在这种情况下，$\eta=1$。

1.6 霍夫曼编码

我们现在学习一种有效的构造DMS信源码的算法，信源符号并不是等可能地出现。根据信源符号的出现概率$P(x_i), i=1,2,\cdots,L$，霍夫曼（Huffman）在1952年给出了一种变长编码算法。该算法在信源符号表示平均所需要的比特数方面是最优的，而且也满足前缀条件。下面是霍夫曼编码算法的步骤：

(1) 将信源符号按概率递减的次序排列。

(2) 将图1-11中最下面的两个符号连在一起。将这两个符号的概率之和写在它们的结合节点上。将这两个分支分别标记为“1”和“0”，如图1-11所示。

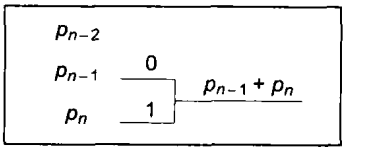


图1-11 霍夫曼编码中将概率结合

(3) 将这个概率和看做一个新符号的概率。重新选取最小的两个概率，将它们绑定在一起构成一个新的概率。每一次我们把两个符号结合在一起时，符号总数减少1。每当把两个概率（节点）结合在一起时，我们总是把两个分支标记为“1”和“0”。

(4) 将此过程继续下去直到只剩下一个概率（如果你加对了，它应该是1！）。这就完成了霍夫曼树的构造。

(5) 要找出任何符号的前缀码字，需要找到从最后节点到该符号的一个路径，在沿反向追踪路径时读出分支的标号，这就是该符号的码字。

通过下面的例子可以很容易理解上述算法。

例1.12 考虑有7个符号$x_i, i=1,2,\cdots,7$，它们的概率分别为$P(x_1)=0.37$、$P(x_2)=0.33$、$P(x_3)=0.16$、$P(x_4)=0.07$、$P(x_5)=0.04$、$P(x_6)=0.02$和$P(x_7)=0.01$的离散无记忆信源。我们首先将概率按递减次序排列，然后构造霍夫曼树，如图1-12所示。

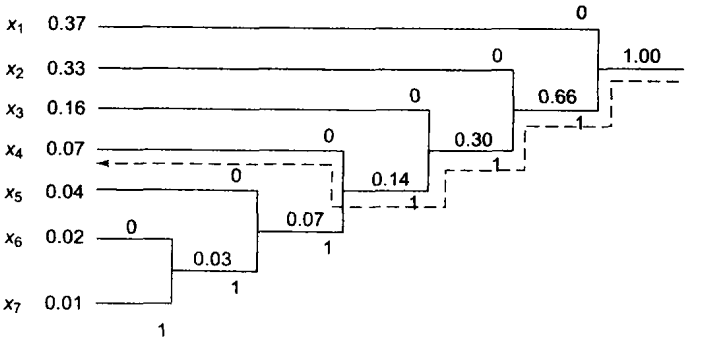


图1-12 例1.12中的霍夫曼编码

要找出任何特别符号的码字，我们仅需要找到从最后的节点到该符号的路径。为了便于演示，我们在图中用虚线标出了概率为0.07的符号x_4的路径。我们在沿路径走的时候读出各分支的标记，便得到码字1110。

符　号	概　率	自 信 息	码　字
x_1	0.37	1.4344	0
x_2	0.33	1.5995	10
x_3	0.16	2.6439	110
x_4	0.07	3.8365	1110
x_5	0.04	4.6439	11110
x_6	0.02	5.6439	111110
x_7	0.01	6.6439	111111

可以计算出该信源的熵为

$$H(X) = -\sum_{k=1}^{7} P(x_k)\log_2 P(x_k) = 2.1152 \text{ (bit)}$$

从而可计算出平均每个符号的二元比特数为

$$\begin{aligned}\overline{R} &= \sum_{k=1}^{7} n_k P(x_k) \\ &= 1(0.37) + 2(0.33) + 3(0.16) + 4(0.07) + 5(0.04) + 6(0.02) + 6(0.01) \\ &= 2.1700 \text{ (bit)}\end{aligned}$$

该码的效率为$\eta = (2.1152/2.1700) = 0.9747$。

例1.13　本例子将证明霍夫曼编码不惟一。考虑有7个符号$x_i, i = 1, 2, \cdots, 7$，概率分别为$P(x_1) = 0.46$、$P(x_2) = 0.30$、$P(x_3) = 0.12$、$P(x_4) = 0.06$、$P(x_5) = 0.03$、$P(x_6) = 0.02$和$P(x_7) = 0.01$的DMS。

符　号	概　率	自 信 息	码　字	码　长
x_1	0.46	1.1203	1	1
x_2	0.30	1.7370	00	2
x_3	0.12	3.0589	010	3
x_4	0.06	4.0589	0110	4
x_5	0.03	5.0589	01110	5
x_6	0.02	5.6439	011110	6
x_7	0.01	6.6439	011111	7

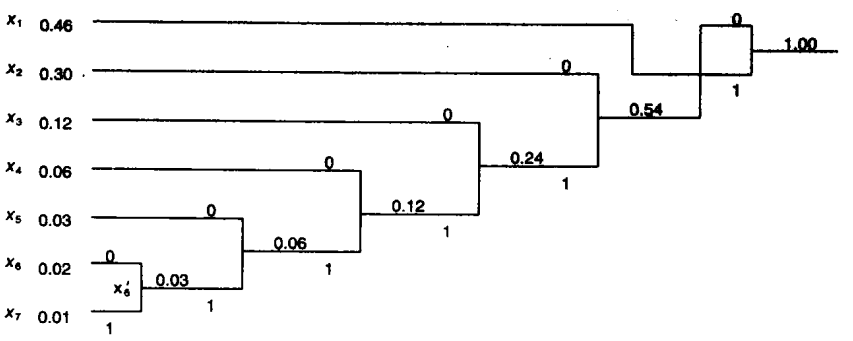


图1-13　例1.13中的霍夫曼编码

可以计算出该信源的熵为

$$H(X) = -\sum_{k=1}^{7} P(x_k)\log_2 P(x_k) = 1.9781 \text{(bit)}$$

从而可以计算出每个符号的平均二元比特数为

$$\begin{aligned}\bar{R} &= \sum_{k=1}^{7} n_k P(x_k) \\ &= 1(0.46) + 2(0.30) + 3(0.12) + 4(0.06) + 5(0.03) + 6(0.02) + 6(0.01) \\ &= 1.9900 \text{ (bit)}\end{aligned}$$

该码的效率为$\eta = (1.9781/1.9900) = 0.9940$。

我们现在将看到霍夫曼编码不是惟一的。第一步，考虑最小两个概率的组合（符号x_6和x_7），它们的和为0.03，等于与符号x_5对应的下一个较高的概率。第二步，我们可以选择使这个组合概率（比如说属于符号x'_6）高于或低于符号x_5。假定我们把组合概率放在下面。继续进行，又将发现x'_6和x_5组合后的概率为0.6，等于符号x_4的概率。我们又一次可以选择使组合概率高于或低于x_4。每次我们做出一种选择（掷硬币）时，最后都会导致符号的码字的改变。在图1-14中，每次在有两个相同概率的情况下要做出一种选择时，我们把组合符号的概率放在上面。

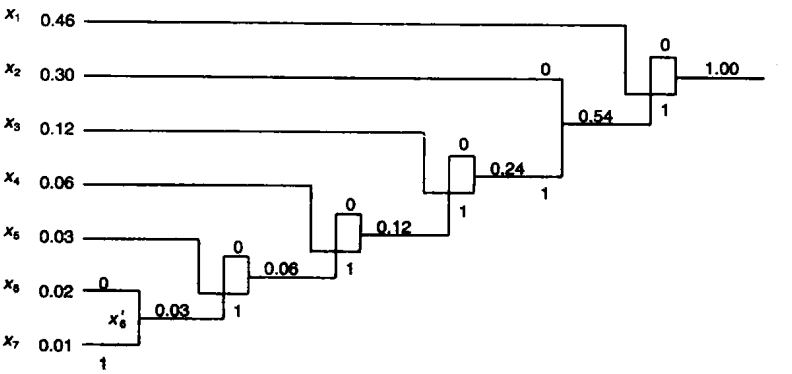


图1-14 例1.13中的另一种霍夫曼编码方法导致一个不同的码

符 号	概 率	自 信 息	码 字	码 长
x_1	0.46	1.1203	1	1
x_2	0.30	1.7370	00	2
x_3	0.12	3.0589	011	3
x_4	0.06	4.0589	0101	4
x_5	0.03	5.0589	01001	5
x_6	0.02	5.6439	010000	6
x_7	0.01	6.6439	010001	6

该信源的熵为

$$H(X) = -\sum_{k=1}^{7} P(x_k) \log_2 P(x_k) = 1.9781 \text{ (bit)}$$

而平均每个符号的比特数为

$$\begin{aligned}\bar{R} &= \sum_{k=1}^{7} n_k P(x_k) \\ &= 1(0.46) + 2(0.30) + 3(0.12) + 4(0.06) + 5(0.03) + 6(0.02) + 6(0.01) \\ &= 1.9900 \text{(bit)}\end{aligned}$$

该码的效率为$\eta=(1.9781/1.9900)=0.9940$。因此两种码是同样地有效。这就说明，给定一组概率，相应的霍夫曼编码并不是惟一的。

假设我们稍微改变一下概率，新概率为$P(x_1)=0.42$，$P(x_2)=0.30$，$P(x_3)=0.15$，$P(x_4)=0.07$，$P(x_5)=0.03$，$P(x_6)=0.02$，$P(x_7)=0.01$。此信息源的熵为$H(X)=-\sum_{k=1}^{7}P(x_k)\log_2P(x_k)=2.0569(\text{bit})$。霍夫曼编码如下：

符　号	概　率	自 信 息	码　字	码　长
x_1	0.42	1.2515	1	1
x_2	0.30	1.7370	00	2
x_3	0.15	2.7370	010	3
x_4	0.07	3.8365	0110	4
x_5	0.03	5.0589	01110	5
x_6	0.02	5.6439	011110	6
x_7	0.01	6.6439	011111	6

我们观察到，虽然概率值发生变化，但霍夫曼编码仍然和图1-13相同。平均码长$\overline{R}=\sum_{k=1}^{7}n_kP(x_k)=2.0800(\text{bit})$。此码的效率是$\eta=(2.0569/2.0800)=0.9889$。直观上，我们可以说码长（一个整数）不能因为要反映输入信号的概率（以及自信息）的变化而变化。

我们进行另外一个类似的实验，符号概率集合如下：

符　号	概　率	自 信 息	码　字	码　长
x_1	1/2	1	1	1
x_2	$1/2^2$	2	00	2
x_3	$1/2^3$	3	010	3
x_4	$1/2^4$	4	0110	4
x_5	$1/2^5$	5	01110	5
x_6	$1/2^6$	6	011110	6
x_7	$1/2^7$	6	011111	6

称一个概率分布是关于D的D-adic 分布，如果每一个概率等于D^{-n}（n为某个整数）。根据定义，上表的分布是D-adic的。此信息源的熵是

$$H(X)=-\sum_{k=1}^{7}P(x_k)\log_2P(x_k)=1.9688(\text{bit})$$

平均每个符号的二元比特数为

$$\overline{R}=\sum_{k=1}^{7}n_kP(x_k)=1.9688(\text{bit})$$

因此，此码的效率是$\eta=1$。霍夫曼编码把码长和符号的概率（以及自信息）准确地匹配起来。例如，如果x_5的自信息是5bit，霍夫曼编码将为此符号分配码长为5的码字。很明显，为了达到最优（$\eta=1$），符号的自信息必须是整数，这就意味着概率必须是2的负整数次方。而在现实中并不是这样的。一个更加有效的用来把码长和符号概率进行匹配的机制将在本章稍后介绍，即算术编码（Arithmetic Code）。

在上述例子中，编码是通过一个个符号完成的。一种更为有效的方法是一次给B个符号的

一个组一起编码。在此情况下信源编码定理的界变为

$$BH(X) \leqslant \overline{R_B} < BH(X) + 1 \tag{1-44}$$

这只是因为B-符号组的熵为$BH(X)$，而$\overline{R}_B$是每个B-符号组的平均比特数。我们可以将上述的界重新写为

$$H(X) \leqslant \frac{\overline{R_B}}{B} < H(X) + \frac{1}{B} \tag{1-45}$$

其中 $\overline{R_B}/B \equiv \overline{R}$ 是每个信源符号的平均比特数。因此只要选取充分大的组B，就可以使$\overline{R}_B$成为任何接近$H(X)$的值。

例1.14 考虑下表中列出的信源符号和它们相应的概率。

符 号	概 率	自 信 息	码 字
x_1	0.40	1.3219	1
x_2	0.35	1.5146	00
x_3	0.25	2.0000	01

对于这个码，信源的熵为

$$H(X) = -\sum_{k=1}^{3} P(x_k)\log_2 P(x_k) = 1.5589\,(\text{bit})$$

每个符号的平均二元字节数为

$$\begin{aligned}\overline{R} &= \sum_{k=1}^{3} n_k P(x_k) \\ &= 1(0.40) + 2(0.35) + 2(0.25) = 1.60\ (\text{bit})\end{aligned}$$

而该码的效率为$\eta = (1.5589/1.6000) = 0.9743$。

我们现在把符号每两个分一组，再重新应用霍夫曼编码算法。符号对的概率按递减次序排列如下表。

符 号 对	概 率	自 信 息	码 字
x_1x_1	0.1600	2.6439	10
x_1x_2	0.1400	2.8365	001
x_2x_1	0.1400	2.8365	010
x_2x_2	0.1225	3.0291	011
x_1x_3	0.1000	3.3219	111
x_3x_1	0.1000	3.3219	0000
x_2x_3	0.0875	3.5146	0001
x_3x_2	0.0875	3.5146	1100
x_3x_3	0.0625	4.0000	1101

对于这个码，其熵为

$$\begin{aligned}2H(X) &= -\sum_{k=1}^{9} P(x_k)\log_2 P(x_k) = 3.1177\ (\text{bit}) \\ \Rightarrow H(X) &= 1.5589\ \text{bit}(\text{bit})\end{aligned}$$

注意信源的熵没有改变！每个组（符号对）的平均比特数为

$$\begin{aligned}\overline{R_B} &= \sum_{k=1}^{9} n_k P(x_k) \\ &= 2(0.1600) + 3(0.1400) + 3(0.1400) + 3(0.1225) \\ &\quad + 3(0.1000) + 4(0.1000) + 4(0.0875) + 4(0.0875) + 4(0.0625) \\ &= 3.1775 \text{ (bit/符号对)} \\ \Rightarrow \overline{R} &= 3.1775/2 = 1.5888 \text{ (bit/符号)}\end{aligned}$$

该码的效率为$\eta=(1.5589/1.5888)=0.9812$。因此我们看到将两个字母组成一个符号提高了编码效率。

例1.15　考虑下表所列的信源符号和它们相应的概率。

符　号	概　率	自 信 息	码　字
x_1	0.50	1.0000	1
x_2	0.30	1.7370	00
x_3	0.20	2.3219	01

对于这个码，信源的熵为

$$H(X) = -\sum_{k=1}^{3} P(x_k)\log_2 P(x_k) = 1.4855 \text{ (bit)}$$

每个符号的平均二元比特数为

$$\begin{aligned}\overline{R} &= \sum_{k=1}^{3} n_k P(x_k) \\ &= 1(0.50) + 2(0.30) + 2(0.20) \\ &= 1.50 \text{ (bit)}\end{aligned}$$

而该码的效率为$\eta=(1.4855/1.5000)=0.9903$。

我们现在把符号每两个分一组，再重新应用霍夫曼编码算法。符号对的概率按递减次序排列如下表。

符 号 对	概　率	自 信 息	码　字
x_1x_1	0.25	2.0000	00
x_1x_2	0.15	2.7370	010
x_2x_1	0.15	2.7370	011
x_1x_3	0.10	3.3219	100
x_3x_1	0.10	3.3219	110
x_2x_2	0.09	3.4739	1010
x_2x_3	0.06	4.0589	1011
x_3x_2	0.06	4.0589	1110
x_3x_3	0.04	4.6439	1111

对于这个码，其熵为

$$\begin{aligned}2H(X) &= -\sum_{k=1}^{9} P(x_k)\log_2 P(x_k) = 2.9710 \text{ (bit)}, \\ \Rightarrow H(X) &= 1.4855 \text{ (bit)}\end{aligned}$$

每个组（符号对）的平均比特数为

$$\begin{aligned}\overline{R_B} &= \sum_{k=1}^{9} n_k P(x_k) \\ &= 2(0.25)+3(0.15)+3(0.15)+3(0.10)+3(0.10)+4(0.09) \\ &\quad +4(0.06)+4(0.06)+4(0.04)=3.00(\text{bit/符号对}) \\ &\Rightarrow \overline{R}=3.00/2=1.5000(\text{bit/符号})\end{aligned}$$

该码的效率为$\eta_2=(1.4855/1.5000)=0.9903$。在这种情况下，将两个字母组合在一起并没有提高编码效率！然而，我们每次把3个字母组合在一起（三元组），然后应用霍夫曼编码，可得到编码效率$\eta_3=0.9932$。在每次把4个字母组合在一起的情况下，我们看到进一步的改进（$\eta_4=0.9946$）。

1.7 Shannon-Fano-Elias编码

使用码长为 $l(x)=\left\lceil \log\frac{1}{p(x)} \right\rceil$ 的编码，我们称之为**Shannon码**。Shannon码的码长满足Kraft不等式，因此可以用来构造惟一的可译码。在本节中，我们将要学习另外一种基于Shannon-Fano-Elias编码技术来构造惟一的可译码的方法。此方法使用**累积分布函数**（Cumulative Distribution Function）来分配码字。此累积分布函数定义为：

$$F(x)=\sum_{z\leqslant x} P(z) \tag{1-46}$$

其中，$P(z)$是符号的出现概率。如图1-15所示，累积分布函数是由大小为$P(x)$的多步累积而成。我们继续定义一个修正的累积分布函数：

$$\overline{F}(x)=\sum_{z<x} P(z)+\frac{1}{2}P(x) \tag{1-47}$$

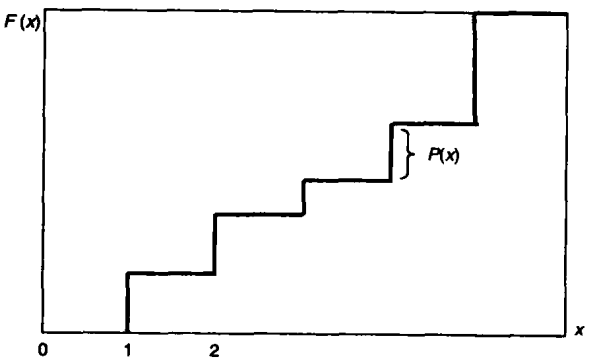


图1-15 累积分布函数

其中，$\overline{F}(x)$表示所有$z<x$的概率之和加上x的概率的一半。函数 $\overline{F}(x)$的值是累积分布函数$F(x)$在x处对应区间的中间值。由于概率是正的，所以如果$x\neq z$，则$F(x)\neq F(z)$。因此，当给定 $\overline{F}(x)$时，可以仅仅通过观察累积分布函数的图来确定x，从而 $\overline{F}(x)$的值可以用来对x进行编码。

一般来说，$\overline{F}(x)$是一个实数。这就意味着我们需要无穷多个比特来表示 $\overline{F}(x)$，而这将导致编码效率非常低。假设我们对 $\overline{F}(x)$进行如下修整：只使用前面的$l(x)$个比特，并记为 $\left\lfloor \overline{F}(x) \right\rfloor_{l(x)}$。通过修整的定义，我们有：

$$\bar{F}(x)-\left\lfloor \bar{F}(x)\right\rfloor_{l(x)}<\frac{1}{2^{l(x)}} \tag{1-48}$$

如果 $l(x)=\left\lceil \log\frac{1}{p(x)}\right\rceil+1$，则

$$\frac{1}{2^{l(x)}}<\frac{P(x)}{2}=\bar{F}(x)-F(x-1) \tag{1-49}$$

这就暗示了值 $\left\lfloor \bar{F}(x)\right\rfloor_{l(x)}$ 位于累积分布函数$F(x)$在x处对应区间中，并且用$l(x)$个比特来表示x是足够的。任意码字的间隔的长度是$2^{-l(x)}$。从式(1-48)中我们可以看出此间隔小于$F(x)$在x处对应区间高度的一半。由于我们使用 $l(x)=\left\lceil \log\frac{1}{p(x)}\right\rceil+1$ 个比特来表示x，则此编码长度的期望值是

$$\bar{R}=\sum_x P(x)l(x)=\sum_x P(x)\left(\left\lceil \log\frac{1}{P(x)}\right\rceil+1\right)<H(X)+2 \tag{1-50}$$

因此Shannon-Fano-Elias编码机制能够实现码长在比熵多两比特之内。

例1.16　按给定的下表考虑D-adic分布

符号	概率	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$（二进制）	$l(x)=\left\lceil \log\frac{1}{p(x)}\right\rceil+1$	码字
x_1	1/2	0.5	0.25	0.01	2	01
x_2	$1/2^2$	0.75	0.625	0.101	3	101
x_3	$1/2^3$	0.875	0.8125	0.1101	4	1101
x_4	$1/2^3$	1	0.9375	0.1111	4	1111

此分布的熵是1.75bit。但是Shannon-Fano-Elias编码机制的平均码长是2.75bit。可以很容易地看出，如果删除所有码字的最后一位，我们就得到了最优码（霍夫曼编码）。值得注意的是，不同于霍夫曼编码方式，我们不用首先对概率进行降序排序。接下来我们重新做一次试验：

符号	概率	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$（二进制）	$l(x)=\left\lceil \log\frac{1}{p(x)}\right\rceil+1$	码字
x_1	$1/2^2$	0.25	0.125	0.001	3	001
x_2	1/2	0.75	0.625	0.10	2	10
x_3	$1/2^3$	0.875	0.8125	0.1101	4	1101
x_4	$1/2^3$	1	0.9375	0.1111	4	1111

我们可以观察到通过Shannon-Fano-Elias编码机制得到的码字不是惟一的。平均码长是2.75bit。但是，这次我们发现仅仅删除所有码字的最后一位，得不到最优码，因为如此处理，编码就不再是前缀码（prefix code）。Shannon-Fano-Elias编码的基本思想已经在算术编码中使用，此种编码利用计算上高效的算法来进行编码和解码。

1.8　算术编码

从例1.13可知，如果符号的概率是2的负整数次方，那么霍夫曼编码是惟一最优编码，这是因为所有的前缀码是在比特层面上进行编码的。我们可以从下面两方面来理解：

（1）前缀码试图使用码长为整数的码字去匹配符号的自信息。这种长度匹配使得码字可

能比自信息长，也可能短。而且当且仅当自信息是整数的时候，两者长度是完全相同的。

（2）如果我们使用二叉树产生前缀码，无论树枝的概率是0.5/0.5还是0.9/0.1，树枝的决定也要占据一个比特。在后者的情况下，理论上需要用0.15bit($-\log_2(0.9)$)选择第一个树枝，用3.32bit($-\log_2(0.1)$)选择第二个树枝。所以平均码长是0.467bit($0.9\times0.15+0.1\times3.32$)。霍夫曼编码还需要为每一次决定使用一个比特。

算术（Arithmetic）编码没有这种限制。它的工作方式是把整个文件编码成0和1之间的某个实数的区间。根据某个符号的出现概率，消息中连续出现的符号降低了此区间。

例1.17 假设我们的字母表只由三个符号A、B、C组成，它们出现的概率分布是$P(A)=0.5$，$P(B)=0.25$，$P(C)=0.25$。我们首先把[0, 1]区间根据字母的出现概率分成三个子区间，如同图1-16的步骤1一样。因此，A对应[0, 0.5)，B对应[0.5, 0.75)，C对应[0.75, 1.0)。注意，子区间的长度正比于它们的出现概率。下一步，假设输入消息流是B A C A…，我们首先对B进行编码。首先选择对应区间[0.5, 0.75)，然后此子空间又根据符号出现概率的比例被分成三个子空间，因此，如图1-16步骤2所示，A对应[0.5, 0.625)，B对应[0.6255, 0.6875)，C对应[0.6875, 1.0)。由于接下来的符号是A，我们选择A对应的区间[0.5, 0.625)。此区间又根据符号出现概率比例被分成三个子空间，A对应[0.5, 0.625)，B对应[0.5625, 0.59375)，C对应[0.59375, 0.625)。接下来的符号是C，其对应的区间是[0.59375, 0.625)，它又同样被分成三个区间……依次类推，在对A完成编码后，剩下的区间是[0.59375, 0.609375)。BACA的算术编码为此区间中任意的一个数。我们可以说对BACA的算术编码为0.59375。

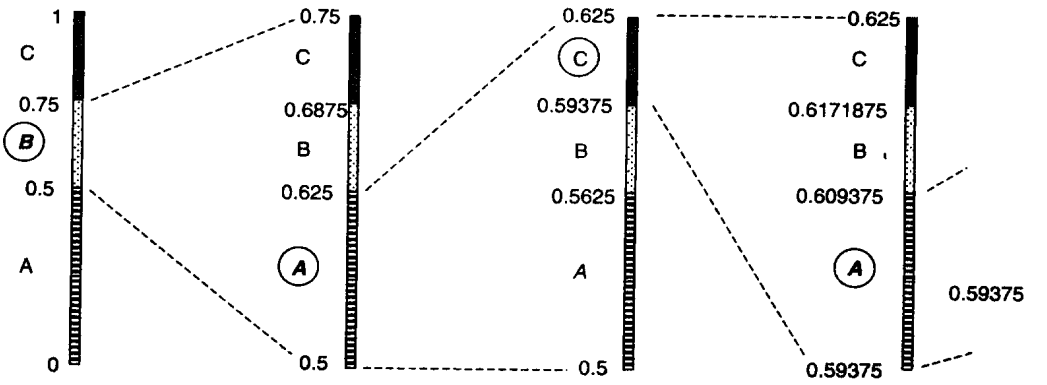


图1-16 对B A C A进行算术编码的例子

1.9 Lempel-Ziv算法

霍夫曼编码需要符号的概率。但许多日常生活现象并不预先提供符号的概率（如信源的统计特性未知）。原则上通过对信源的输出进行长时间观察，可以估计出符号的概率。但在实用中这是不现实的。另外，霍夫曼编码对DMS，也就是当一个符号的出现不会影响到后续符号的出现概率时是最优的，但对有记忆信源这就不是最好的选择。例如考虑书写文字的压缩问题。我们知道许多字母成对或成组出现，如“q-u”、“t-h”、“i-n-g”等。用字母间的统计互相关性连同字母各自的出现概率应该会更有效。这就是Lempel和Ziv在1977年提出的方案。他们的信源编码算法不需要信源的统计特性。这是一种变长到定长信源编码算法，属于广义信源编码算法类。

Lempel-Ziv广义编码隐含的逻辑如下：通过对由一连串的0和1组成的预先串（前缀串），另加一个新的比特进行编码，对任意比特序列的压缩是可能的。然后，由原来的前缀串添加一个新的比特所形成的新串又作为未来串的前缀串。这些变长的组叫做**短语**。这些短语列在一个字典里，其中记录了已存在的短语和它们的位置。当对一个新的短语编码时，我们指明已有的短语在字典中的位置并附加新字母。我们可以通过下面的例子来更好地理解Lempel-Ziv算法是怎么工作的。

例1.18　假定我们想对串101011011010101011编码。我们首先做语法分析，将它变成一些用逗号分开的短语，它们代表的字符串可以用原来的一个字符串作前缀另加一个比特表示。

第一个比特是1，它前面没东西，因此它有一个空前缀串而且另外的1比特就是它自己：

1，01011011010101011

对0也是一样，因为它不能用仅存的前缀来表示：

1，0，1011011010101011

到目前为止，我们的字典包含字符串“1”和“0”。接下来会遇到一个1，但它已经在我们的字典中存在了，因此我们继续往下进行。接下来的10显然是由前缀1和一个0构成，于是我们现在得到：

1，0，10，11011010101011

如此继续下去，我们最终将整个字符串分拆如下：

1，0，10，11，01，101，010，1011

现在有8个短语，我们将用3比特来标记空短语和前7个短语，它们一共有8个编过号的短语。接下来我们用构成一个新短语所需的前缀短语的数字加新的比特来表示字符串。为了直观，我们将首先用括号和逗号将它们分开。这8个短语可描述为：

(000,1)，(000,0)，(001,0)，(001,1)，(010,1)，(011,1)，(101,0)，(110,1)

可以这样读：(在位置0，1的码字)，(在位置0，0的码字)，(在位置1，0的码字)，(在位置1，1的码字)，(在位置2，1的码字)，(在位置3，1的码字)，等等。因此上述字符串编码之后变成了：

0001000000100011010101111010101101

这个例子中的字典由表1-1给出。在这种情况下，我们没得到任何压缩，编码后的字符串实际上更长了！但是，最初的字符串越大，随着编码的进程我们会有更大的节余，因为很大的前缀可用很小的数字索引来表示。事实上，Ziv证明了对于长文件，文件的压缩可以达到由文件信息量所决定的能获得的最优值。

下一个问题是这个表应该多长。在实际应用中，不管表的长度如何，最终总是要溢出。这个问题可通过预先确定一个充分大的字典来解决。编码器和译码器可以通过周期性地用比较常用的短语取代字典中不常用的短语来更新字典。Lempel-Ziv算法已在实际中广泛应用。UNIX操作系统中的压缩和解压应用程序用的就是这种算法的修订版。用于压缩二元文件的标准算法用12bit的字和传递另外1bit来表示一个新序列。使用这样的码，Lempel-Ziv算法可以将英文文本的传递压缩大约55%左右，而霍夫曼码只能压缩43%。

表1-1 Lempel-Ziv 算法的字典

字典位置	字典内容	定长码字
001	1	0001
010	0	0000
011	10	0010
100	11	0011
101	01	0101
110	101	0111
111	010	1010
–	1011	1101

在下一节中我们将学习另外一种编码方案，它对传真件传递和图像压缩极其有用。

1.10 游程编码和PCX格式

游程编码（Run Length Encoding, RLE）是一种用来缩减重复字符串大小的技术。这个重复字符串称为一个**游程**（run）。典型的RLE把一个符号的游程编码为两个字节：一个是计数，另一个是符号。RLE对任何种类的数据都能压缩，不管它的信息量如何，但被压缩数据的信息量影响到压缩率。RLE相对其他压缩方法来说得不到很高的压缩率，但容易实现也能快速执行。多数位图文件格式，如TIFF、BMP和PCX都支持RLE。

例1.19 考虑下列比特流：

$$S=11111111111111100000000000000000001111 \tag{1-51}$$

它可以表示为：十五个1、十九个0、四个1，即（15, 1）、（19, 0）、（4, 1）。因为最大重复次数为19，这个数可以用5bit来表示，因此我们可以将上述比特流编码为（01111, 1）、（10011, 0）、（00100, 1）。在这种情况下，压缩率为18:38＝1:2.11。

RLE非常适用于典型办公文件的传真图像。这些双色图像（黑和白）以白色为主。如果我们对这些图像采样后转换成数字数据，就会发现许多水平线整个都是白的（长的0游程）。另外，若某个给定的像素是黑的或白的，则很可能下一个像素也是相同的。传真机的编码实际上是游程编码和霍夫曼编码的结合。一个游程码把游程长度映射到码字，而且码本也被分成两部分：第一部分包含游程长度为64倍数的那些符号；第二部分则由有0～63个像素的游程构成。例如一个有205个像素的游程将通过一个长为192（3×64）的游程的码字加上长为13的游程的码字来发送。通过这种方法，用来表示该游程所需要的比特数会显著地减少。另外，某些已知的出现概率高的游程被编码成短的码字，这进一步减少了需要传递的比特数。用这类编码方法，典型的对传真传输件的压缩将达到4：1～8：1。连接到高速调制解调器上，这种压缩将传输一页文件的时间减少到不足一分钟。

游程编码还应用于对PCX格式图像的压缩。PCX格式是作为ZSoft公司出售的个人计算机上的画图及编辑用的绘图刷系列的一部分引进的。目前，PCX格式实际上是能辨别许多图像压缩方法中用的哪一种的总名称。我们这里将只把注意力放在其中一种用于256色图像的方法上。我们将只关注那些实际包含编了码的图像的PCX数据流，而不是用于储存调色板和图像信息，比如有多少行、每行的像素数、文件和编码方法的那部分。

基本方案如下：如果一个像素串在颜色值上都是等同的，那么可以把它们编码为特殊旗

标字节（flag byte），它包含一个计数后面连接一个表示这个重复像素的值的字节。如果像素没有重复，就直接把它编码为它本身的字节。这样的一个简单方案在实际中经常变得很复杂。考虑在上面的方案中，如果一个图像的调色板中用到了所有256种颜色，那么我们需要一个字节的256个值来表示那些颜色。因此，如果我们只用字节作为码的基本单位，那么将不再有任何可能的用过的字节值可作为旗标/计数字节。另一方面，如果我们对任一已编码像素都使用两个字节，这样可以留有余地用作旗标/计数的组合，我们可能不仅没把图像的大小压缩，而是放大了两倍。

对PCX格式的折中是基于它的设计者的理念：许多用户制作的图画（这正是他们的软件的基本输出）将不会用到所有的256种颜色。于是，他们对只有192种颜色的情况优化了压缩方案。具有更多颜色的图像也可能得到好的压缩，但用此方案可能不会得到很好的结果。

例1.20 PCX压缩将一种颜色（也就是说，不属于具有同一种颜色的游程的像素）编码为0～191，直接用它的数值的二元字节表示。考虑表1-2。

表1-2　PCX编码例

像素颜色值	十六进制码	二进制码
0	00	00000000
1	01	00000001
2	02	00000010
3	03	00000011
⋮	⋮	⋮
190	BE	10111110
191	BF	10111111

对颜色192（连同所有高于192的颜色），码字等于一个字节，其中最高位比特位（MSB）都设为1。我们将用这些码字来表示旗标和计数字节。如果两个MSB都等于1，我们说它们增加了旗标的一个计数。在旗标/计数字节中的剩余6bit将被解释为该计数的6bit二进制数（从0～63）。紧接这个字节的就是表示颜色的字节。事实上，如果我们有一种颜色的一个像素游程具有大于191的调色板码，我们仍然很容易对此进行游程编码，因为在游程编码字节对里，上面的两个比特并没有在第二个颜色码的字节中保留。

如果一个像素游程的长度超过63，我们就用这个码字本身来表示游程中的前63个像素，然后对游程中多余的像素编码，直到穷尽游程中的所有像素。接下来的问题是：当没有游程时我们怎么处理剩余的几乎整个调色板图像的颜色编码？我们仍然把这些当做游程来编码，只是让游程长度为1即可。这就是说，对64种颜色出现在图像的单一像素而它又不是游程的一部分的情况，我们把数据增加一倍。幸运的是，这种情况极少发生！

下一节我们将学习模拟信源的编码。回顾一下理想状态下我们需要无穷多的比特数才能精确地表示模拟信源，任何有限的比特数都只能近似地表示。我们可以选择用越来越少的比特数来给出对原来信号的越来越差的近似表示。因此，对采样信号振幅的量化的结果就是对数据进行了压缩。我们也要研究在信息源的采样量化后所引入的失真概念。

1.11　率失真函数

尽管我们生活在一个模拟的世界里，多数通信却都是以数字形式进行的。因为多数自然

信源（如语音、影像等）都是模拟的，我们首先对它们进行采样、量化，然后再处理。考虑一个波形为$x(t)$的模拟消息，它是随机过程$X(t)$的一个采样波形。假定$X(t)$是个有限带宽的平稳过程，则它可以用一个以Nyquist速率获得的正态采样序列来表示。这些采样先对振幅量化然后再编码成一个比特序列。一个简单的编码策略可以是定义L级，然后对每一个采样依照下述编码：

$$\begin{aligned}&\text{当}L\text{是2的幂时，}R=\log_2 L\text{，或}\\&\text{当}L\text{不是2的幂时，}R=\lfloor\log_2 L\rfloor+1\end{aligned}\tag{1-52}$$

如果所有级并不等可能，我们可以用熵编码以达到更有效的表示。为了更精确地表示模拟波形，我们需要更多的级数，这也就意味着每个采样需要更多的比特数。理论上每个采样需要无穷多个比特才能完全表示一个模拟信源。对振幅的量化导致数据压缩，但这将以信号的完整性为代价。这是一种有损数据压缩形式，其中从失真情况可以得到一些实际信源采样$\{x_k\}$和对应的量化值$\{\tilde{x}_k\}$之间的区别的度量。

定义1.14 定义**平方误差失真**（squared-error distortion）为

$$d(x_k,\tilde{x}_k)=(x_k-\tilde{x}_k)^2\tag{1-53}$$

一般情况下，一种失真度量可以表示为

$$d(x_k,\tilde{x}_k)=|x_k,\tilde{x}_k|^p\tag{1-54}$$

考虑有n个采样的序列$\boldsymbol{X}_n$和它对应的n个量化值$\tilde{\boldsymbol{X}}_n$。令$d(x_k,\tilde{x}_k)$为每个采样（字母）的失真度量，则原序列和量化值序列之间的失真度量就是它们在n个信源输出采样的平均值，即

$$d(\boldsymbol{X}_n,\tilde{\boldsymbol{X}}_n)=\frac{1}{n}\sum_{k=1}^{n}d(x_k,\tilde{x}_k)\tag{1-55}$$

我们观察到信源就是个随机过程，因此$\boldsymbol{X}_n$以及$d(\boldsymbol{X}_n,\tilde{\boldsymbol{X}}_n)$都是随机变量。我们现在定义失真如下：

定义1.15 定义n个采样的序列$\boldsymbol{X}_n$和它对应的n个量化值$\tilde{\boldsymbol{X}}_n$之间的**失真**为

$$D=E[d(\boldsymbol{X}_n,\tilde{\boldsymbol{X}}_n)]=\frac{1}{n}\sum_{k=1}^{n}E[d(x_k,\tilde{x}_k)]=E[d(x_k,\tilde{x}_k)]\tag{1-56}$$

这里假定随机过程是平稳的。

下面假设一个无记忆信源具有连续输出$\boldsymbol{X}$和量化输出字母表$\tilde{\boldsymbol{X}}$。假设这个连续振幅的概率密度函数为$p(x)$，而每个字母失真度量为$d(x,\tilde{x})$，其中$x\in\boldsymbol{X}$并且$\tilde{x}\in\tilde{\boldsymbol{X}}$。我们下面引进率失真函数，它给出在预先给定允许失真度的情况下要表示信源输出符号的时候，每个采样所需的最少比特数。

定义1.16 在失真率小于或等于D的情况下用于表示无记忆信源输出X的最小速率（以比特数/信源输出为单位）称为**率失真函数**$R(D)$，定义为

$$R(D)=\min_{p(\tilde{x}|x):E[d(X,\tilde{X})]\leq D}I(X;\tilde{X})\tag{1-57}$$

其中$I(X;\tilde{X})$是X和$\tilde{X}$之间的平均互信息。

我们现在将陈述（没有证明）与率失真函数有关的两个定理。

定理1.3　基于每个符号的平均平方误差失真度量，用于表示一个方差为σ_x^2的离散时间连续振幅无记忆高斯信源的最小信息率为

$$R_g(D)=\begin{cases}\frac{1}{2}\log_2(\sigma_x^2/D) & 0\leqslant D\leqslant\sigma_x^2\\ 0 & D>\sigma_x^2\end{cases} \tag{1-58}$$

考虑下面两种情况：

(1) $D\geqslant\sigma_x^2$：对于这种情况没有必要传递任何信息，因为我们可以用统计独立、均值为零、方差满足$D=\sigma_x^2$的高斯噪声采样来复原采样（失真率大于或等于方差）。

(2) $D<\sigma_x^2$：对于这种情况随着D的增加，每个输出符号的比特数单调递减。图1-17给出了率失真函数的平面图。

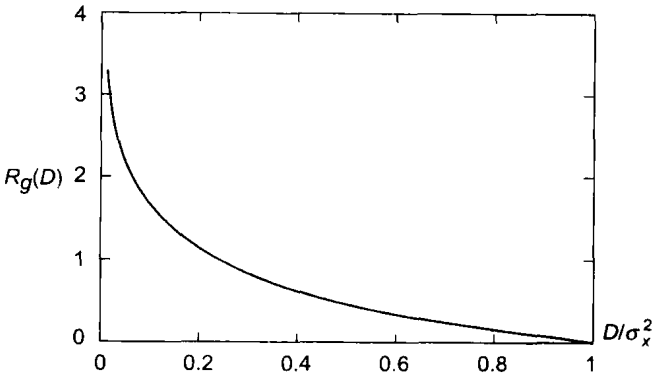


图1-17　$R_g(D)$ 相对D/σ_x^2的平面图

定理1.4　能够将具有平均失真率的信源输出复原到任意接近D的一个充分条件是存在这样一种编码方案，它把信源输出映射到码字，使对任意失真率D的最小速率为每采样$R(D)$比特。

因此，任意信源的失真函数给出了对任意可能失真程度的信源速率的下界。

定义1.17　离散时间无记忆高斯信源的**失真率函数**（distortion rate function）定义为

$$D_g(R)=2^{-2R}\sigma_x^2 \tag{1-59}$$

例1.21　对一个离散时间无记忆的高斯信源，失真率（单位为dB）作为它的方差的函数可表示为

$$10\log_{10}D_g(R)=-6R+10\log_{10}\sigma_x^2 \tag{1-60}$$

因此平均平方失真率以每比特6dB的速度递减。

一个离散时间、无记忆、具有连续振幅、均值为零、方差为σ_x^2的信源的率失真函数关于平均平方误差失真度量D的上界为

$$R(D)\leqslant\frac{1}{2}\log_2(\sigma_x^2/D)\quad 0\leqslant D\leqslant\sigma_x^2 \tag{1-61}$$

这个上界可以直观地理解为：我们知道对于一个给定的方差，零均值高斯随机变量展示

了任意随机变量可以达到的最大微分熵。因此，对于一个给定的失真率，每个采样所需要的最小比特数以此高斯随机变量为上界。

下面一个显然的问题是：对量化器的设计什么样是好的？有没有一种不用太多比特就能使失真达到最小的量化器的构造方法？在下一节中我们可以找到这些问题的答案。

1.12 优化量化器的设计

在本节中，我们来看一下优化量化器的设计问题。现在考虑一个连续振幅信号，其振幅不是均匀分布，但其方差符合某个概率密度函数$p(x)$。我们想设计优化常量量化器，它能使量化误差$q=\tilde{x}-x$的某些函数达到最小值，其中$\tilde{x}$是x的量化值。由量化造成的失真率可表示为

$$D=\int_{-\infty}^{\infty} f(\tilde{x}-x)\,p(x)\mathrm{d}x \tag{1-62}$$

其中$f(\tilde{x}-x)$是误差的期望函数。优化量化器就是能通过优化选取输出水平和每个输出水平所对应的输入区间使D达到最小值的函数。最终的优化量化器称为**Lloyd-Max量化器**。对一个L级量化器，它造成的失真可表示为

$$D=\sum_{k=1}^{L}\int_{x_{k-1}}^{x_k} f(\tilde{x}_k-x)p(x)\,\mathrm{d}x \tag{1-63}$$

将D对$\{x_k\}$和$\{\tilde{x}_k\}$求微分可得到失真最小化的必要条件。下面的方程组就是微分过程的一个结果

$$\begin{aligned} &f(\tilde{x}_k-x_k)=f(\tilde{x}_{k+1}-x_k),\ k=1,2,\cdots,L-1\\ &\int_{x_{k-1}}^{x_k} f'(\tilde{x}_{k+1}-x)\,p(x)\mathrm{d}x,k=1,2,\cdots,L \end{aligned} \tag{1-64}$$

对$f(x)=x^2$，即失真率的平均平方值，上述方程组简化为

$$\begin{aligned} &x_k=\frac{1}{2}(\tilde{x}_k+\tilde{x}_{k+1}),\ k=1,2,\cdots,L-1\\ &\int_{x_{k-1}}^{x_k} f'(\tilde{x}_k-x)\,p(x)\mathrm{d}x=0,\ k=1,2,\cdots,L \end{aligned} \tag{1-65}$$

非均匀量化器在失真方面得到了优化。但是，每个量化后的采样都用了相同的比特数来表示（比如R比特/采样）。要得到更有效的VLC也是可能的。由量化得到的离散信源的输出用一组概率p_k来描述。这些概率又用于设计有效的VLC（信源编码）。为了比较不同非均匀量化器的性能，我们首先把失真率D固定，然后比较每个采样所需要的平均比特数。

例1.22 考虑高斯随机变量有八个级的量化器。这个问题由Max首先在1960年解决。该随机变量均值为零，方差为1。对于平均平方误差的最小化，x_k和$\tilde{x}_k$的值如表1-3。

对于表中的这些值，$D=0.0345$，即-14.62dB。

这个八级优化量化器每采样的比特数为$R=3$。当实行霍夫曼编码时，每个采样需要的平均比特数为$R_H=2.88$bit/采样。理论限为$H(X)=2.82$bit/采样。

表1-3　优化量化和霍夫曼编码

级　数	x_k	$\tilde{x}_k$	p_k	霍夫曼码
1	−1.748	−2.152	0.040	0010
2	−1.050	−1.344	0.107	011
3	−0.500	−0.756	0.162	010
4	0	−0.245	0.191	10
5	0.500	0.245	0.191	11
6	1.050	0.756	0.162	001
7	1.748	1.344	0.107	0000
8	∞	2.152	0.040	0011

1.13　随机过程的熵率

通过对信源编码定理的扩展，我们知道使用平均$nH(X)$比特就可以表示n个独立同分布随机变量，每一个随机变量的熵是$H(X)$。但是实际中，我们经常遇到非独立分布的随机变量。如果随机变量形成了一个平稳随机过程，情况又如何呢？

定义1.18　一个随机过程被称为是**平稳的**（Stationary），如果对于任意时刻的任意随机变量序列的子集的联合分布不变，也就是，

$$P(X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n) = P(X_{1+m} = x_1, X_{2+m} = x_2, \cdots, X_{n+m} = x_n) \tag{1-66}$$

对于任意的时间平移m，以及任意的$x_1, x_2, \cdots, x_n \in X$，上式成立。

定义1.19　一个离散的随机过程$X_1, X_2, \cdots$被称为**马尔可夫链**，或者**马尔可夫过程**，如果，对于$n=1, 2, \cdots$

$$P(X_{n+1} = x_{n+1} \mid X_n = x_n, X_{n-1} = x_{n-1}, \cdots, X_1 = x_1) = P(X_{n+1} = x_{n+1} \mid X_n = x_n) \tag{1-67}$$

对于任意的$x_1, x_2, \cdots, x_n, x_{n+1} \in X$，上式成立。

马尔可夫过程的概率密度函数可以写成

$$p(x_1, x_2, \cdots, x_n) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1}) \tag{1-68}$$

如果给定一串n个随机变量，序列熵的增长与n之间的关系是一个非常值得探讨的问题。为了说明这种增长率，我们定义了熵率这个概念。

定义1.20　随机过程X的**熵率**是：

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \cdots, X_n) \tag{1-69}$$

如果此极限存在的话。

例1.23　若$X_1, X_2, X_3, \cdots$是独立同分布的随机变量，则相应的熵率是

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \cdots, X_n) = \lim_{n\to\infty} \frac{nH(X_1)}{n} = H(X_1)$$

但是如果$X_1, X_2, X_3, \cdots$是独立但非同分布的随机变量时，则

$$H(X_1, X_2, \cdots, X_n) = \sum_{i=1}^{n} H(X_i)$$

因此，熵率是

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}H(X_i)$$

如果$H(X_i)$不相等，则很有可能通过选择X_1, X_2, X_3,…上的分布序列使得$\frac{1}{n}\sum_{i=1}^{n}H(X_i)$的极限不存在。

定义1.21 对于平稳的**马尔可夫链**而言，其熵率是：

$$H(X)=\lim_{n\to\infty}H(X_n\mid X_{n-1},\cdots,X_1)=\lim_{n\to\infty}H(X_n\mid X_{n-1})=H(X_2\mid X_1). \tag{1-70}$$

例1.24 考虑这样的两状态马尔可夫链，其概率转换矩阵为

$$\boldsymbol{P}=\begin{bmatrix}1-p_1 & p_1\\ p_2 & 1-p_2\end{bmatrix} \tag{1-71}$$

如图1-18所示，对于平稳分布，状态转换图中任意割集的净概率分布（net probability distribution）是0。我们用α和β表示两个状态的平稳概率。因此，平稳分布为

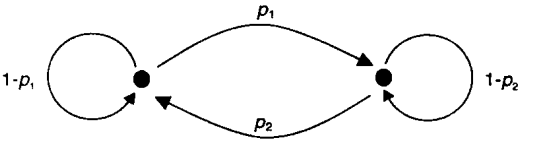


图1-18 两状态马尔可夫链的状态转换图

$$\alpha=\frac{p_2}{p_1+p_2}\text{ 和 }\beta=\frac{p_1}{p_1+p_2} \tag{1-72}$$

注意到$\alpha+\beta=1$。在时间点n上状态X_n的熵是

$$H(X_n)=H\left(\frac{p_2}{p_1+p_2},\frac{p_1}{p_1+p_2}\right)=H(\alpha,\beta) \tag{1-73}$$

此两状态马尔可夫链的熵率是

$$H(X)=H(X_2\mid X_1)=\frac{p_2}{p_1+p_2}H(p_1)+\frac{p_1}{p_1+p_2}H(p_2) \tag{1-74}$$

1.14 图像压缩简介

在本章前面部分，我们讨论了用于压缩的数据集的编码。应用这些技术我们可以用少于源数据的比特数来储存或传输一个数据串所有的信息内容。传送源数据所有信息所必需的最小比特数决定于信源的熵度量。对充分大的源数据组，通过熵编码器和通用编码器可以得到好的压缩率。在本节中我们介绍用于图像储存和传输的压缩技术。

图像可以采样和量化到充分精细，使用二元数据流表示源数据时，可以达到令人满意的程度。因为我们可以用从1 000～100万字节的数据来表示一幅图，因此应该能够用前面学过的技术直接对那些数据进行压缩，以便储存和传输。首先，我们考虑以下几点：

(1) 高质量的图像要用很大的数据集表示。一个摄影质量的图像可能需要400～1000万比特来表示。为了使这些大文件的储存和传输（特别是对电影）实际可行，这带来了对极高压缩率的需求。

(2) 为了做到在视觉上接受，涉及图像的应用如电视、电影、计算机图形用户界面以及万维网（World Wide Web）需要快速执行和跨分布网络的传输，特别是当它们涉及运动图像时。

(3) 我们认为图像有比其他类型的数据高得多的冗余度。例如图像中一对相邻的水平线几

乎完全一样，而书中相邻的两行字一般来说不相同。

前两点表明对图像数据的运动和储存需要有最高的压缩技术。第三点表明高压缩率是可以达到的。第三点还说明某些特别的压缩技术可以将图像数据的结构和性质利用起来。可以对图像中相邻像素的紧密关系充分利用来改善压缩率。这也受到实时应用图像数据编码和译码的重要影响。

另外值得注意的一点是人们的眼睛对图像近似误差有很高的容忍度。因此压缩图像数据时将某些（对于人的眼睛）不很重要的信息忽略掉也是可能的。也就是说在图像质量方面做一下折中，可以获得对数据大小的很大缩减。相对于前面讨论的**无损压缩**（lossless compression）技术，这种技术叫做**有损压缩**（lossy compression）。这种自由性不可以用于财务或文本等类型的数据！有损压缩只能用于图像和声音等类型数据，它们的缺陷可以被人们在视觉和听觉方面的包容性所弥补。

1.15　无损压缩的JPEG标准

联合图像专家组（Joint Photographic Experts Group, JPEG）由两个标准机构——欧洲电信标准组织（The European Telecommunication Standards Organisation, CCIT）和国际标准化组织（ISO）联合组成。我们现在先考虑JPEG**图像压缩标准**中的无损压缩选项，它是对29种不同的图像压缩编码系统的描述。为什么有这么多方法呢？因为不同的用户对质量相对压缩和压缩相对计算时间的要求有很大的差异，于是委员会决定提供很多方法供选择。我们这里将只简要讨论两种应用熵编码的方法。

这里讨论的两种无损JPEG压缩选项只在用于数据的熵码形式上有所不同。用户可选择使用**霍夫曼码**或**算术码**。我们对算数码将不讨论详细内容，但总结它的主要特征如下：

算术码就像霍夫曼码一样，通过利用数据的概率特征，使信息在传输或储存时使用了比源数据流更少的比特数。它优于霍夫曼码的基本优势是当数据只用到较小的字母集时，它更接近于对数据流压缩的Shannon熵的界。原因是当符号出现的概率可表示为2的幂的分数时，霍夫曼码最可行（最大压缩率）。算数码的构造并不像霍夫曼码一样与这些特定值密切相关。算术码的编码和译码在计算量上比霍夫曼码要大。算数码通常比霍夫曼码能减小文件5%～10%。

如果我们能利用前一个像素预测下一个像素，则可以得到一些压缩。这样做我们只需要传输预测相关系数（数值的不同）即可，而不需要传输整个像素。用于无损JPEG编码方案来得到新数据的预测过程也是可变的，但是在此情况下变化并不依赖于用户的选择，而是以一个图像的行为基础的。选择是根据预测方法能对整个行给出最好的预测来做出的。

在JPEG编码标准中有8种不同的预测方法，其中之一（无预测选项）不是用作无损编码选项的，这就是我们在这里讨论的。另外7种可以分为下面几类：

(1) 预测下一个像素与同一行中的前面的有相同的值。

(2) 预测下一个像素与前一行中相同位置（即上面）的像素有相同的值。

(3) 预测下一个像素的值与前面的、上面的以及上面的前面的像素的值的组合有关。这种组合之一就是取三个值的平均值。

用于JPEG标准的微分编码由实际图像像素值和预测值之间的差组成。作为多数图像的光滑度和冗余度的结果，这些差导致了较小的代表预测中典型误差的正数和负数。因此，对小的更新值，这些值的概率很大，而对大的更新值它们却很小。这恰好是利用熵码可以很好地压缩的一类数据流。

对自然图像的典型无损压缩率为2:1。这种压缩已经很可观了，但它仍不能解决在处理高质量影像时所遇到的对大量图像序列的储存和移动问题。

1.16 有损压缩的JPEG标准

通过研究人们能接受的图像失真程度，JPEG标准包括了一些复杂的有损压缩选项。JPEG有损压缩算法包括一个图像简化步骤，它去掉图像的复杂性，代价是失去一些逼真程度，紧接着是无损压缩步骤，它基于预测过滤以及霍夫曼编码或算数编码。

有损图像简化步骤也称作图像缩减，它是建立在一种人们熟知的**离散余弦变换**（DCT）运算的基础上的，这种变换的定义如下：

$$Y(k,l)=\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}4y(i,j)\cos\left(\frac{\pi k}{2N}(2i+1)\right)\cos\left(\frac{\pi l}{2M}(2j+1)\right) \tag{1-75}$$

其中输入图像有N行M列个像素，$y(i,j)$是第i行第j列的像素的像素亮度，$Y(k,l)$是DCT矩阵中第k行第l列上的DCT系数。所有DCT的乘法都是按实数进行的，这样一来比起离散傅里叶变换（DFT）需要少一些的乘法运算。对多数图像来说，信号能量大都集中在低频率部分，出现在DCT的左上角。右下角的值表示高频信号，因此具有很小的值（通常小到即使忽略，最多也只会造成很小的视觉失真）。

在JPEG图像缩减过程中，将DCT应用到图像的8×8的像素块中。因此如果图像有256×256的像素，我们把它分为32×32的方块，每一块都有8×8的像素，它们将被独立地处理。每一块的64个像素值经DCT变换成一组新的64个值，这64个称为DCT系数的新值给出了一种全新的方式来表示图像。DCT系数表示图像子块的空间频率。DCT矩阵的左上角是低频率部分，右下角是高频率部分（见图1-19）。最左上角的系数称为DC系数，它的值比起8乘8列像素块的平均值只是几分之一。其余系数称为AC系数。

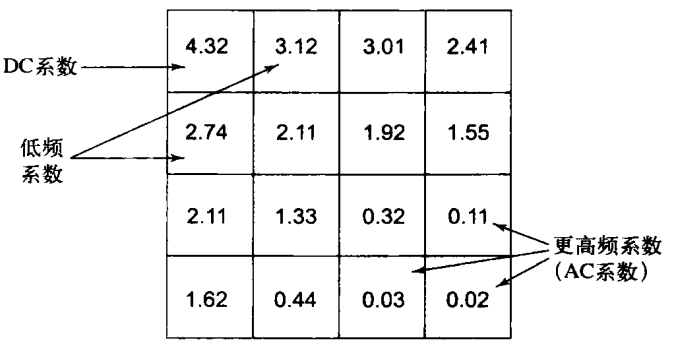


图1-19 一些典型离散余弦变换（DCT）值

到现在为止，由DCT我们还没有得到任何缩减。但是，自然图像的本质决定了最大能量（信息）落在低频率部分而不是高频率部分。我们可以粗略地表示高频率部分，或把它们都丢掉，这样并不会严重影响到复原的图像的质量。这导致了大量的压缩（有损）。JPEG有损压缩算法包括下列运算：

(1) 首先把重量最低的变为零。

(2) 然后对剩余的重量量化（也就是用一个离散码中某个最接近的值取代），有些量化比其他的更粗略些，这要根据观察者对这些退化的敏感程度而定。

对所有图像块，许多无损压缩步骤用于由上述DCT和量化过程得到的权重数据。我们观察到表示平均图像亮度的DC系数从一个8×8像素块到下一个这样的块时变化很慢。因此从周围的块来预测这个值通常较准确。我们只需要传送一个DC系数以及接下来的块的DC系数的差别即可。这种差别也可以用信源编码。

下面我们看一下AC系数。我们首先对它们进行量化，把大多数高频率的系数变为零。然后用一种**锯齿形**编码方法，如图1-20所示。锯齿形编码的目的是逐渐地从低频率到高频率，避免在数值上的突然跳跃。锯齿形编码将产生长的0游程，这些长的0游程对RLE后跟霍夫曼或算术编码的方法很理想。

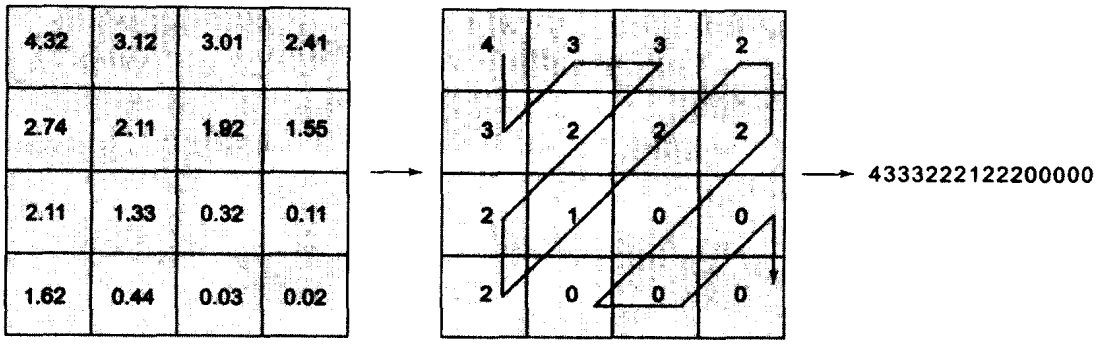


图1-20　一个量化后用锯齿形编码的例子

JPEG典型的性能表现是，自然景观摄影质量的图像可以以约为20:1或25:1的压缩比率来保存。可用的质量（即不用于关键目的）可以用比率为200:1～230:1的压缩得到。

1.17　评注

1948年，Shannon发表了他的里程碑式的著作“通信的数学理论”。他在这篇关于信息论的先驱性论文的开始便观察到，**通信的基本问题是在某一点准确地或大概地重现在另一点选取的消息**。然后他全面地建立了信息论的基础，他的框架和术语也成为标准。Shannon的理论直接带来了通信工程师的成功，这也刺激了今天信息技术的发展。Shannon在不同领域发表了很多启发性的、具有重大影响的论文。他的硕士论文“接替和开关电路的符号分析”用布尔代数建立了数字电路的理论基础。这项工作有着广泛的重要意义，因为数字电路是现代计算机和通信系统运转的基础。

Shannon因为他的兴趣和能力而著称。一个人所共知的故事就是他骑着独轮车从贝尔实验室大厅出来的时候手里还玩着杂耍（像杂技运动员一样扔着好几个球）。他设计并建造了下象棋、走迷宫、杂耍和测谎机器。这些活动都证明了他自己强调的动力多来自好奇心而不是用途。用他自己的话说就是“我只是对事物是如何构成的感到好奇”。

霍夫曼码是美国科学家D. A. Huffman于1952年创造的。修改后的霍夫曼编码在今天应用于联合图像专家小组（JPEG）和动态图像专家小组（Moving Picture Experts Group, MPEG）的标准中。

1970年，以色列人Abraham Lempel和Jacob Ziv开发了一种不需知道信源元素发生概率的信源编码技术。UNIX操作系统的压缩和解压应用软件就用了这种算法的修订版本。由CompuServe开发的图像的GIF格式（图形交换格式）仅涉及Lempel-Ziv-Welch(LZW)广义编码算法在图像数据中的一种应用。

最后，作为本章的总结，我们需要提一下，信息论之父Shannon在2001年2月24日去世了。专家们在发表于《纽约时报》上的讣告中写道：

Claude Shannon，数学家，终年84岁

GEORGE JOHNSON报道

Claude Elwood Shannon博士，美国数学家和计算机科学家，他的理论奠定了给世界带来辉煌的电子通信网络的基础，于星期六病逝于Medford, Mass……

1.18 小结

- 事件$X=x_i$的自信息为 $I(x_i)=\log\left(\frac{1}{P(x_i)}\right)=-\log\ P(x_i)$
- x_i和y_j之间的互信息为 $I(x_i;y_j)=\log\left(\frac{P(x_i\mid y_j)}{P(x_i)}\right)$
- 在给定$Y=y_j$的条件下，事件$X=x_i$的条件自信息为

$$I(x_i\mid y_j)=\log\left(\frac{1}{P(x_i\mid y_j)}\right)=-\log\ P(x_i\mid y_j)$$

- 两个随机变量X和Y的平均互信息为

$$I(X;Y)=\sum_{i=1}^{n}\sum_{j=1}^{m}P(x_i,y_j)\log\frac{P(x_i,y_j)}{P(x_i)P(y_j)}$$

当X和Y统计独立时，有$I(X;Y)=0$。平均互信息满足$I(X;Y)\geqslant 0$，当且仅当X和Y统计独立时等号成立。

- 随机变量X的平均自信息由公式

$$H(X)=\sum_{i=1}^{n}P(x_i)I(x_i)=-\sum_{i=1}^{n}P(x_i)\log\ P(x_i)$$

给出，$H(X)$ 又称为熵。

- **平均条件自信息**又称为**条件熵**，它可以表示为

$$H(X\mid Y)=\sum_{i=1}^{n}\sum_{j=1}^{m}P(x_i,y_j)\log\frac{1}{P(x_i\mid y_j)}$$

- 我们知道有$I(x_i;y_j)=I(x_i)-I(x_i\mid y_j)$ 和$I(X;Y)=H(X)-H(X\mid Y)=H(Y)-H(Y\mid X)$。因为$I(X;Y)\geqslant 0$，这表明$H(X)\geqslant H(X\mid Y)$。
- 具有联合距离$P(x,y)$的离散随机变量对(X,Y)的联合熵定义为

$$H(X\mid Y)=-\sum_{i=1}^{n}\sum_{j=1}^{m}P(x_i,y_j)\log P(x_i,y_j)$$

- $H(X)$、$H(X,Y)$与$H(X\mid Y)$之间的链式法则关系是

$$H(X,Y)=H(X)+H(Y\mid X)$$

- 熵与互信息之间的关系可表示为

$$I(X;Y)=H(X)-H(X\mid Y)=H(Y)-H(Y\mid X)$$

- 互信息与联合熵之间的关系可表示为

$$I(X;Y)=H(X)+H(Y)-H(X,Y)$$

- 两个连续随机变量X和Y之间的**平均互信息**可表示为

$$I(X;Y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}p(x)p(y|x)\log\frac{p(y|x)p(x)}{p(x)p(y)}\mathrm{d}x\mathrm{d}y$$

- 连续随机变量X和Y的**微分熵**表示为

$$H(X)=-\int_{-\infty}^{\infty}p(x)\log\ p(x)\mathrm{d}x$$

- 连续随机变量X和Y的**平均条件熵**表示为

$$H(X|Y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}p(x,y)\log\ p(x|y)\mathrm{d}x\mathrm{d}y$$

- 存在码字长度为$n_1\leqslant n_2\leqslant\cdots\leqslant n_L$的二元码的充分必要条件是满足前缀条件$\sum_{k=1}^{L}2^{-n_k}\leqslant 1$。前缀码的效率为

$$\eta=\frac{H(x)}{\overline{R}}$$

- 设X为一个DMS的全部字母的集合，它具有有限熵$H(X)$。**信源编码定理**说明可以构造满足前缀条件的码，使它的平均码长$\overline{R}$满足$H(X)\leqslant\overline{R}\leqslant H(X)+1$。对符号的有效表示导致了数据压缩。
- 霍夫曼编码的期望码字长度的范围是$\overline{R}\leqslant H(X)+1$
- Shannon-Fano-Elias编码的期望码字长度的范围是$\overline{R}\leqslant H(X)+2$
- Shannon-Fano-Elias码的基本概念用于算术码的编码和译码的高效算法中。算术码需要把文件编码成0和1之间的实数区间。
- **霍夫曼编码**和Lempel-Ziv编码是两种最流行的信源编码技术。不同于霍夫曼编码的是，Lempel-Ziv技术不依赖于信源的统计特性。Lempel-Ziv技术产生定长码，而霍夫曼码是一种变长码。
- **游程编码**（RLE）是一种用于缩减重复字符串大小的技术。这种重复的串叫做一个游程。多数位图文件格式如TIFF、BMP及PCX都支持游程编码。
- **失真**表明在实际信源采样$\{x_k\}$和它们对应的量化值$\{\tilde{x}_k\}$之间差别有一种度量。平方误差**失真**可表示为$d(x_k,\tilde{x}_k)=(x_k-\tilde{x}_k)^2$。一般情况下，一种失真度量可以表示为$d(x_k,\tilde{x}_k)=|x_k-x_k|^p$。
- 表示一个失真率小于或等于D的无记忆信源输出X所需的最小速率（以比特数/信源输出为单位）称为率失真函数$R(D)$，定义为$R(D)=\min_{p(\tilde{x}|x):E[d(X,\tilde{X})\leqslant D]}I(X;\tilde{X})$；其中$I(X;\tilde{X})$是$X$和$\tilde{X}$之间的平均互信息。
- 由量化导致的失真可表示为$D=\int_{-\infty}^{\infty}f(\tilde{x}-x)p(x)dx$，其中$f(\tilde{x}-x)$是误差的期望函数。优化量化器能通过优化地选取输出水平及对应每一个输出水平的一个输入范围来使D最小化。最终的优化量化器称为Lloyd-Max量化器。
- 随机过程$X_1,X_2,\cdots$又称为马尔可夫链或马尔可夫过程。

$$P(X_{n+1}=x_{n+1}|X_n=x_n,X_{n-1}=x_{n-1},\cdots,X_1=x_1)=P(X_{n+1}=x_{n+1}|X_n=x_n)$$

其中$n=1,2,\cdots,x_1,x_2,\cdots,x_n,x_{n+1}\in X$。

- 随机过程X的熵率为

$$H(X)=\lim_{n\to\infty}\frac{1}{n}H(X_1,X_2,\cdots,X_n)$$

的极限存在。

- 稳定的马尔可夫链的熵率为

$$H(X)=\lim_{n\to\infty}H(X_n\mid X_{n-1},\cdots,X_1)=\lim_{n\to\infty}H(X_n\mid X_{n-1})=H(X_2\mid X_1)$$

- 量化和信源编码技术（霍夫曼编码、算数编码及游程编码）在对图像压缩的JPEG标准中得到应用。

障碍是当你将眼睛从目标上移开时所看到的那些可怕的东西。

——亨利·福特（1863—1947）

习题

1.1 考虑一个信源概率为{0.30，0.25，0.20，0.15，0.10}的DMS。求信源熵$H(X)$。

1.2 证明一个离散信源在它的输出符号等概率的情况下其熵达到最大值。

1.3 证明不等式$\ln x\leqslant x-1$。画出曲线$y_1=\ln x$和$y_2=x-1$的平面图以表明上述不等式的正确性。

1.4 证明$I(X;Y)\geqslant 0$。在什么条件下等号成立？

1.5 有一个信源X，它有无穷多个可能的输出，它们出现的概率为$P(x_i)=2^{i-1}$，$i=1,2,3,\cdots$，这个信源的平均自信息$H(X)$是什么？

1.6 考虑另一个几何分布的随机变量X，满足$P(x_i)=p(1-p)^{i-1}$，$i=1,2,3,\cdots$，这个信源的平均自信息$H(X)$是什么？

1.7 X表示抛掷一个硬币直到第一次出现背面的抛掷次数。

(1) 假设抛币是公平的，求熵$H_f(X)$。

(2) 假设抛币是不公平的，一次抛币中，背面向上的概率为p。求熵$H_u(X)$。

1.8 考虑一个满足后缀条件的编码：任意一个码字不是其他码字的后缀。

(1) 这种码字是否惟一可译码的？为什么？

(2) 你是否能把后缀码的最小平均码长和使用相同随机变量的霍夫曼编码的平均码长联系起来？请做深入解释。

1.9 两个概率质量函数（probability mass function）$p(x)$和$q(x)$的相对熵或者Kullback-Leibler距离定义为：

$$D(p\parallel q)=\sum_{x\in X}p(x)\log\left(\frac{p(x)}{q(x)}\right)\tag{1-76}$$

(1) 证明$D(p\parallel q)$是非负的。

(2) 证明Kullback-Leibler距离不满足距离的另外两个性质，即对称性和三角不等式。

(3) 证明$I(X;Y)=D(p(x,y)\parallel p(x)\,q(y))$。这样也能很自然地证明出互信息是非负的。

1.10 假设一个信源产生独立的符号（符号均属于字符表$\{a_1,a_2,a_3\}$），对应的概率分别是$p_1=0.4999999$，$p_2=0.4999999$，$p_3=0.0000002$。

(1) 计算此信源的熵$H(X)$。

(2) 找到此信源的最优编码，并且计算码长的期望值。

(3) 找到此信源的第二种扩展（也就是包含两个符号的分组）的最优编码，计算码长的

期望值，以及码长除以2的期望值。

(4) 证明：如果对此信源的长度为N的分组进行编码，为了压缩到1%的熵，N的值最小为5。注意，如果使用式(1-45)，你能得到一个非常宽松的上界。

1.11 我们希望对一个字符序列进行编码，此字符序列来自一个有$d+3$个字符的字母表。我们想要使用3比特长的码字来分别对符号a_1, a_2, a_3进行编码，而使用8比特长的码字来分别对符号$a_4, a_5, \cdots, a_{d+3}$进行编码。如果需要使得此种编码能惟一解码，则$d$的最大值是多少？

1.12 考虑一个只取整数值的随机变量X，满足 $P(X=n)=\dfrac{1}{An\log^2 n}$，其中 $A=\sum_{n=2}^{\infty}\dfrac{1}{n\log^2 n}$，$n=2,3,\cdots,\infty$。求熵$H(X)$。

1.13 计算概率分布函数为

$$p(x)=\begin{cases}a^{-1} & 0\leqslant x\leqslant a\\ 0 & (\text{否则})\end{cases} \tag{1-77}$$

的均匀分布随机变量X的微分熵$H(X)$。画出$H(X)$相对于参数a（$0.1<a<10$）的平面图，并对结果进行评论。

1.14 考虑一个信源概率为{0.35，0.25，0.20，0.15，0.05}的DMS。

(1) 给出此信源的霍夫曼码。

(2) 计算这些码字的平均码长$\overline{R}$。

(3) 这个码的效率η是多少？

1.15 再次考虑信源概率为{0.35, 0.25, 0.20, 0.15, 0.05}的DMS。

(1) 使用符号1、2、3给出此信源的三元霍夫曼编码。

(2) 计算这些码字的平均码长$\overline{R}$，并把它和三元情况下的平均码长作比较。

1.16 考虑一个信源概率为{0.20，0.20，0.15，0.15，0.10，0.10，0.05，0.05}的DMS。

(1) 给出此信源的一种有效定长码。

(2) 给出此信源的霍夫曼码。

(3) 比较这两种码并给出评论。

1.17 一个DMS只有三个输出符号，它们的概率为{0.5，0.4，0.1}。

(1) 给出此信源的霍夫曼码并确定编码效率η。

(2) 每次考虑两个符号时给出此信源的霍夫曼码并确定编码效率η。

(3) 每次考虑三个符号时给出此信源的霍夫曼码并确定编码效率η。

1.18 对一个熵为$H(X)$的信源，证明B－符号组的熵为$BH(X)$。

1.19 设X和Y是取值分别为$x_1, x_2, \cdots, x_r$和$y_1, y_2, \cdots, y_s$的随机变量。令$Z=X+Y$。

(1) 证明$H(Z|X)=H(Y|X)$。

(2) 如果X和Y独立，解释$H(Y)\leqslant H(Z)$和$H(X)\leqslant H(Z)$。并对此做出评论。

(3) 在什么条件下$H(Z)=H(X)+H(Y)$？

1.20 确定下列比特流的Lempel-Ziv码：0100111110010100000101010110011000从码字流中恢复原来的序列。

1.21 考虑对四进制数据（符号：0，1，2，3）的Lempel-Ziv编码。

(1) 对下面的四进制数据进行编码：1 3 3 0 0 2 0 2 1 1 1 3 0 0 0 0 2 2 1 2 2 2 3 3。压缩率是多少？压缩率定义为编码后的比特数除以编码前的比特数。

(2) 接下来，为每一个四进制符号用相等的二进制形式表现出来（0→00, 1→01, 2→10,

3→11)。使用二进制数进行Lempel-Ziv编码，并计算压缩率。

(3) 比较（1）和（2）的结果，并进行评价说明。

1.22 考虑由信源A产生的比特流。游程在比特流中出现的概率为：

$$p_r = \begin{cases} 2^{-r} & 1 \leqslant r < n \\ 2^{-(n-1)} & r = n \end{cases} \tag{1-78}$$

其中r是游程的长度，n是游程可能的最大长度。注意这里说的是可能的游程，它可以是1的重复，也可以是0的重复。

(1) 设计一个游程编码，找出它所提供的压缩。

(2) 假设我们希望对游程用霍夫曼编码进行编码。给出此比特流的编码。此编码提供的压缩是多少？

1.23 我们接下来尝试游程编码的另一种方案。我们不断重复游程编码过程，直到不存在更多可能的压缩为止。使用这种方案，如果我们希望对某一个游程，比如n个1进行编码，最大可能的压缩是多少？

1.24 求满足$p=0.5$的符合Bernoulli分布的随机变量X的率失真函数$R(D)=\min I(X;\tilde{X})$，其中失真由下列函数给出：

$$d(x,\tilde{x}) = \begin{cases} 0 & x = \tilde{x} \\ 1 & x = 1,\ \tilde{x} = 0 \\ \infty & x = 0,\ \tilde{x} = 1 \end{cases} \tag{1-79}$$

1.25 考虑一个在$\{1, 2, \cdots, m\}$上均匀分布的信源X。求这个信源由下式定义的汉明失真（Hamming distortion）的率失真函数：

$$d(x,\tilde{x}) = \begin{cases} 0 & x = \tilde{x} \\ 1 & x \neq \tilde{x} \end{cases} \tag{1-80}$$

上机习题

1.26 写一个程序能在给定概率的条件下实现霍夫曼编码。它应该产生码字并算出编码效率。

1.27 修改上述程序使它能将n个信源符号分为一组，然后产生霍夫曼码。对下述信源符号的概率（0.55, 0.25, 0.20）画出编码效率η相对n的平面图。对什么样的n值，编码效率可以比0.9999还要好？对下述信源符号概率重复上述练习：{0.45, 0.25, 0.15, 0.10, 0.05}。

1.28 写一个执行Lempel-Ziv算法的程序。该程序的输入可以是英文字母。它应该将字母转化为它们的ASCII码然后进行压缩。它应该输出压缩结果。用这个程序求对下列的字符串所得到的压缩：

(1) The Lempel Ziv algorithm can compress the English text by about fifty five percent.

(2) The cat cannot sit on the canopy of the car.

1.29 写一个程序对比特序列执行游程编码（RLE），并给出编码后的输出以及压缩率。若下列序列是该程序的输入，它的输出是什么：

1100000000111100000111111111111111111100000110000000

1.30 假设对于某一种通信，我们只使用三个字母A、B、C。它们出现的概率分别是{0.5, 0.3, 0.2}。但是，使用这些字符不是相互独立的。双字母出现概率分别是$P(AB)=0.25$,

$P(BC)=0.05$, $P(CA)=0.35$, $P(BA)=0.15$, $P(CB)=0.05$, $P(AC)=0.15$。我们称考虑双字符的霍夫曼编码为伪马尔可夫霍夫曼编码（PMH）。

(1) 设计一个针对此通信场景（考虑双字符）的PMH编码。编写一个计算机程序，输入一些如上的概率，输出一个对应的PMH编码。

(2) 编写一个对应此种情况的算术编码，并和PMH编码进行比较。

1.31　写一个程序，它可以接受2^n级灰度的图像（每个像素n比特），并执行下列运算：

(1) 把它分为8×8的像素块。

(2) 在每个块上执行DCT变换。

(3) 通过只保留m个最高比特位（MSB）将DCT系数进行量化，其中$m\leqslant n$。

(4) 执行锯齿形编码然后执行游程编码。

(5) 对上述得到的比特流执行霍夫曼编码（考虑一种合理的计算符号概率的方法）。

(6) 计算压缩率。

(7) 执行解压（即第5步返回到第1步的逆操作）。用该程序对不同的m值实行图像压缩。当m达到什么样的值时，原图像和压缩图像间的差别不能被察觉到？

第2章 信道容量和编码

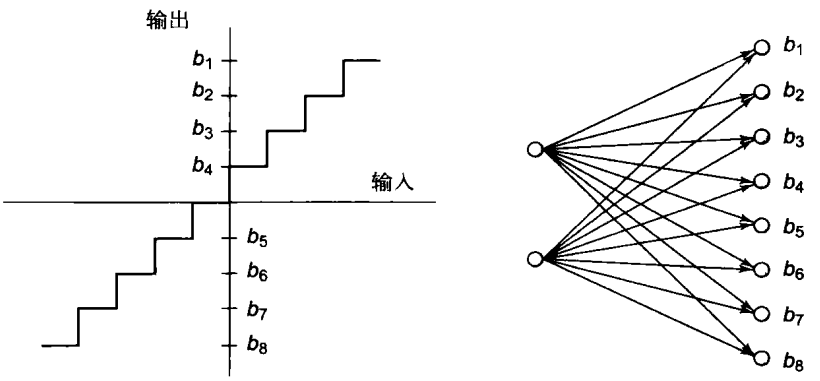
实验者们认为那是一个数学定理，而数学家们则相信那是个实验事实。

——Lippman, Gabriel (1845—1921)

2.1 引言

在第1章中我们看到多数自然信源都有固有的冗余度，因此通过不同信源编码技术将这些冗余去掉以实现对数据的压缩是可能的。当信源符号可以用尽量少的比特数来有效地表示以后，我们通过信道（如电话线、光纤等）来传输这些比特流。这些比特可能被原样传输（基带通信），或经过调制后传输（通带通信）。令人遗憾的是，现实生活中的所有信道都有噪声。噪声这个术语是指那些对传输和处理通信系统中的有用信号造成干扰的无用波。噪声的来源可能是外界的（如大气噪声、人工噪声等），也可能是内在的（如热噪声、放射噪声等）。结果是在接收端得到的比特流很可能与原来传输的不一样。在通带通信中，解调器处理的是信道弄损的波形，这些波形再化简为一个矢量或向量来近似表示传输的数据符号。检波器在解调器之后判决原来传输的比特是0还是1。这种方法叫**硬判决译码**（hard decision decoding）。在译码器上的这种判决过程与两个等级的二元量化很相似。如果在量化中有多于两个的等级，则检波器实施的方法称为**软判决译码**（soft decision decoding）。

使用硬判决译码会在接收端造成不可挽回的信息丢失。假设调制器发送的只是二元信号，而解调器接收到Q个符号的字符，再假定量化器的使用如图2-1a所示，这里我们有$Q=8$。这样的信道称为**二元输入Q元输出离散无记忆信道**（discrete memoryless channel，DMC）。这种对应的信道如图2-1b所示。译码器的性能取决于量化器的表示级数的位置，而量化器的表示级数又取决于信号水平和噪声功率。相应地，为了实现一个多级量化器，解调器必须采用自动增益控制技术。很明显构造这样的译码器比硬判决译码器要复杂得多。但是软判决译码比硬判决译码有着明显的改善。



a) 多级量化器传输特征　　b) 信道转移概率图

图 2-1

数字通信工程师们必须考虑三个因素：传输信号的功率、信道带宽以及通信系统的可靠性（在比特误差率方面）。信道编码允许我们牺牲上面的一种因素（信号功率、带宽或可靠性）。在本章中，我们将学习在有噪声的情况下如何得到可靠的通信。我们将问自己这样的问题：在一个给定带宽和信噪比（SNR）的信道上每秒钟能传输多少比特？为此我们首先介绍几个信道模型。

2.2 信道模型

我们已经看到了最简单的信道模型——**二元对称信道**（Dinary ymmetric hannel，BSC）。若调制器采用的是二元波形，而检波器用硬判决，那么信道可以看做是在发送端输入一个二元比特流，在接收端输出另一个比特流。图2-2表示出了这种模型。

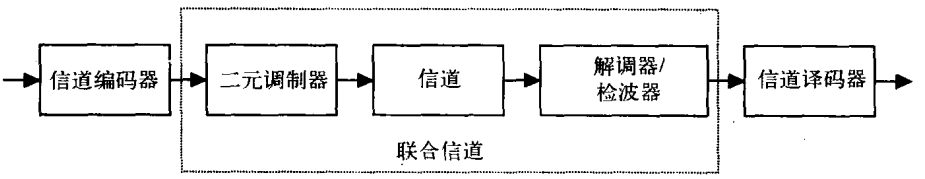


图2-2　一种联合的离散输入离散输出信道（由调制器和解调器/检波器组成）

联合的**离散输入离散输出信道**（discrete-input, discrete-output channel）可描述为一个可能的输入集$X=\{0, 1\}$、一个可能的输出集$Y=\{0, 1\}$，以及将可能输出与将可能输入相关联的条件概率。假设信道中的噪声以错误概率p在二元序列的传输中产生独立错误

$$\begin{aligned} P(Y=0|X=1)&=P(Y=1|X=0)=p \\ P(Y=1|X=1)&=P(Y=0|X=0)=1-p \end{aligned} \tag{2-1}$$

图2-3所示的就是一个BSC。

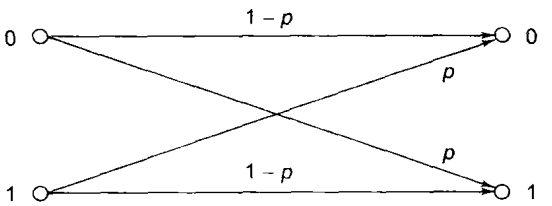


图2-3　一个二元对称信道（BSC）

BSC是离散输入离散输出信道的一种特殊情况。设信道输入是q元符号，即$X=\{x_0, x_1, \cdots, x_{q-1}\}$，在接收端检波器的输出为$Q$元符号，即$Y=\{y_0, y_1, \cdots, y_{Q-1}\}$。我们假定信道和调制过程都是无记忆的。那么输入输出之间可以由qQ个条件概率相联系：

$$P(Y=y_i|X=x_j)=P(y_i|x_j) \tag{2-2}$$

其中$i=0, 1, \cdots, Q-1$且$j=0, 1, \cdots, q-1$。这种信道称为**离散无记忆信道**（DMC），如图2-4所示。

定义2.1　*定义条件概率$P(y_i|x_j)$为**信道转移概率**，并记为p_{ji}。*

定义2.2　*描述一个DMC的条件概率$\{P(y_i|x_j)\}$可以用矩阵形式$\boldsymbol{P}=[p_{ji}]$给出。$\boldsymbol{P}$称为信道的**概率转移矩阵**（probability transition matrix）。*

到目前为止，我们讨论了具有离散输入和离散输出的单个信道。同样可以在发射器和接

收器之间实现多信道。通过在发射器端和接收器端使用多个天线，这种信道可以在无线通信环境中实现。下面是四种显著的组合。

(1) **单个输入单个输出**（SISO）：这种是常见的无线配置，即在发射端和接收端均使用一个天线。

(2) **单个输入多个输出**（SIMO）：这种在发射端使用单个天线，但在接收端使用多个天线。

(3) **多个输入单个输出**（MISO）：这种在发射端使用多个天线，但在接收端使用单个天线。

(4) **多个输入多个输出**（MIMO）：这种在发射端和接收端均使用多个天线。

考虑MIMO系统，其中发射端天线数目是M_T，接收端天线数目是M_R。我们用$h_{ij}(\tau, t)$表示第$j(j=1, 2, \cdots, M_T)$个发射天线和第$i(i=1, 2, \cdots, M_R)$个接收天线之间的脉冲响应。注意我们考虑的是波形通道，调制器/解调器不属于波形通信部分。MIMO信道能够用$M_R \times M_T$矩阵表示如下：

$$\boldsymbol{H}(\tau, t) = \begin{bmatrix} h_{1,1}(\tau,t) & h_{1,2}(\tau,t) & \cdots & h_{1,M_T}(\tau,t) \\ h_{2,1}(\tau,t) & h_{2,2}(\tau,t) & \cdots & h_{2,M_T}(\tau,t) \\ \vdots & \vdots & \cdots & \vdots \\ h_{M_R,1}(\tau,t) & h_{M_R,2}(\tau,t) & \cdots & h_{M_R,M_T}(\tau,t) \end{bmatrix} \tag{2-3}$$

变量τ用来表示信道变化的时间。如果第j个发射天线发送信号$s_j(t)$，则第i个接收天线所收到的信号是

$$y_i(t) = \sum_{j=1}^{M_T} h_{i,j}(\tau,t)s_j(t),\ i = 1,2, \cdots, M_R \tag{2-4}$$

MIMO信道的输入输出关系可以用矩阵符号表示为

$$\boldsymbol{y}(t) = \boldsymbol{H}(\tau, t)\boldsymbol{s}(t) \tag{2-5}$$

其中，$\boldsymbol{s}(t) = [s_1(t), s_2(t), \cdots, S_{M_T}(t)]^{\mathrm{T}}$，$\boldsymbol{y}(t) = [y_1(t), y_2(t), \cdots, y_{M_T}(t)]^{\mathrm{T}}$。

每一对发射器和接收器之间的每一条链路都可以独立地用离散无记忆信道表示。

2.3 信道容量

考虑一个DMC，它的输入字符集为$X = \{x_0, x_1, \cdots, x_{q-1}\}$，输出字符集为$Y = \{y_0, y_1, \cdots, y_{r-1}\}$。让我们用$P(y_i|x_j)$表示信道转移概率的集合。输出字符集集合$Y$关于输入字符集$X$的平均互信息为（见1.2节）

$$I(X;Y) = \sum_{j=0}^{q-1}\sum_{i=0}^{r-1} P(x_j)P(y_i|x_j)\log\frac{P(y_i|x_j)}{P(y_i)} \tag{2-6}$$

信道转移概率$P(y_i|x_j)$决定于信道的特征（特别是信道中的噪声）。但是输入符号的概率$P(x_j)$受离散信道编码器的控制。平均互信息$I(X; Y)$的值在输入符号概率集上达到的最大值是仅仅依赖于信道转移概率$P(y_i|x_j)$的一个量，这个量称为**信道容量**。

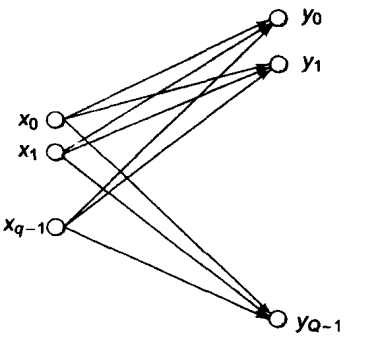


图2-4 一个具有q-元输入和Q-元输出的离散无记忆信道（DMC）

定义2.3 一个DMC的**容量**定义为每一次使用该信道时的最大平均互信息，其中最大值是在所有可能的输入概率上求得的，即

$$C = \max_{P(x_j)} I(X; Y)$$
$$= \max_{P(x_j)} \sum_{j=0}^{q-1} \sum_{i=0}^{r-1} P(x_j)P(y_i|x_j) \log \frac{P(y_i|x_j)}{P(y_i)} \tag{2-7}$$

$I(X; Y)$ 的最大值是在下列约束条件下求得的

$$P(x_j) \geqslant 0, \text{并且} \sum_{j=0}^{q-1} P(x_j) = 1$$

信道容量的单位是每次信道使用的比特数（只要对数的底数为2）。

例2.1 考虑一个BSC，它的信道转移概率为

$$P(0|1) = p = P(1|0)$$

由对称性可知，信道容量 $C = \max_{P(x_j)} I(X; Y)$ 在 $p = 0.5$ 时可取得最大值。由式(2-7) 可得到BSC的信道容量为

$$C = 1 + p \log_2 p + (1 - p) \log_2 (1 - p)$$

定义熵函数为

$$H(p) = -p \log_2 p - (1 - p) \log_2 (1 - p)$$

则可以将二元对称信道的容量写为

$$C = 1 - H(p)$$

图2-5给出了信道容量相对 p 的平面图。从该平面图中可以观察到下述结果：

(1) 当 $p = 0$ 时（即无噪声信道），信道容量为所期望的每次使用1比特。每次我们用此信道时，可以成功传输1比特信息。

(2) 当 $p = 0.5$ 时，信道容量为0，即从输出结果得不到任何关于输入的信息。它等同于信道断开时的情况，我们可以干脆放弃信道而用掷硬币的方法来确定发送的是什么。

(3) 当 $0.5 < p < 1$ 时，信道容量随 p 的增加而增加。在此情况下，我们仅在BSC的输出端将0和1互换就行了。

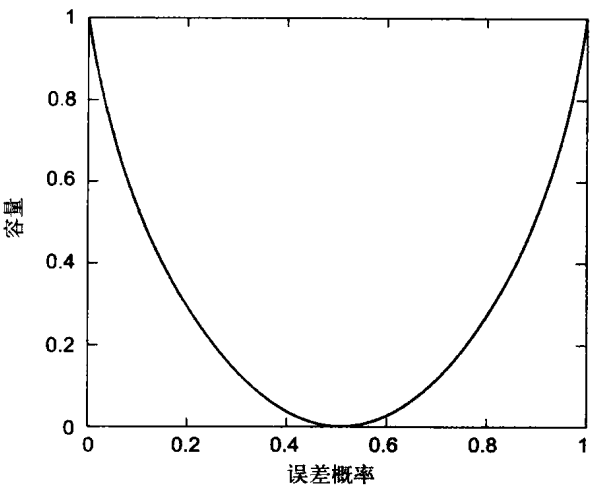


图2-5　BSC的信道容量

(4) 当 $p = 1$ 时（即每个比特都被信道改变了），信道容量为所期望的每次使用1比特。在此情况下，只需要做与信道效果相反的事就行，即把每个输出比特反过来。

(5) 由于 p 是关于信噪比（SNR）的单调递减函数，BSC容量则是SNR的单调递增函数。

有了信道容量的概念后，我们现在需要把它同信道的可靠通信联系起来。到目前为止，

我们只讲了信道在每次使用时能传输的比特数。但每秒钟所能传输的比特数又是多少呢？为了回答这个问题，我们引入**信道编码**的概念。

2.4 信道编码

所有现实生活中的信道都受到噪声的影响。噪声造成数字通信系统中输入和输出数据序列间的不符合（错误）。一个典型噪声信道的比特错误概率可以高达10^{-2}。这表明在这个信道上平均每传输100比特，就有一比特被改变了。对大多数应用来说，这样的可靠程度远不能满足要求。不同应用对可靠程度（服务质量的一项内容）的要求也不同。表2-1列出了不同应用可接受的比特错误率。

表2-1 不同应用可接受的比特错误率

应 用	错误概率
语音电话	10^{-4}
声乐数据	10^{-6}
电子邮件、电子报纸	10^{-6}
因特网访问	10^{-6}
可视电话、高速计算	10^{-7}

为了能得到如此高的可靠性，我们可以借助**信道格式编码**。信道编码的基本目标是增加数字通信系统的抗噪声能力。这可以通过可控制地在传输的数据流中加入冗余度来实现。

在信道编码中，我们把进来的数据序列映射到信道输入序列。这一编码过程由**信道编码器**完成。然后将编码后的序列通过有噪信道传输。在接收端信道输出序列被反映射到输出数据序列，这称为译码过程，由**信道译码器**完成。编码器和译码器都受设计者控制。

正如已经提到的，编码器以预先设计好的方式引进冗余度，译码器利用这个冗余度以便尽可能精确地恢复原来信源序列。因此信道编码使在不可靠（有噪声）的信道中实现可靠通信成为可能。信道编码又叫做**错误控制编码**（error control coding），我们将可互换地使用这些术语。可以注意到信源编码通过减少冗余度以提高效率，而信道编码则在控制下添加冗余度来提高可靠性。

我们首先看到的一类信道码叫做**分组码**（block code）。在这类码中，进入的消息序列首先分成一系列的组，每个组的长度为k比特。每个k比特长信息组再由信道编码器对应到一个n比特组，其中$n>k$。这表明对任意k比特的信息，有$(n-k)$冗余的比特添加进来。比率

$$r=\frac{k}{n} \tag{2-8}$$

称为**码率**（code rate）。任何编码方案的码率自然地要小于1。小的码率意味着每个组有更多的冗余比特，相应地有更高的编码负载。这样可能会减轻噪声的影响，但也同时降低通信率，因为我们最终要传输更多的冗余比特和更少的信息比特。我们所面临的问题是是否存在一种编码方案使消息比特发生错误的概率可以任意小，而码率又不会太小呢？答案是肯定的，它首先由Shannon在他的信道容量的第二个定理中给出。我们很快就将学到这个定理。

让我们在讨论中引入时间的概念。我们希望研究这样的问题：比如通过一个有噪信道一秒钟可以传输多少比特使比特错误率达到任意低？假设DMS有信源字符集X、熵为每信源符号$H(X)$ bit，而且信源每T_s秒产生一个符号。那么信源的平均信息率为每秒$H(X)/T_s$ bit。让我们假设信道可以每T_c秒使用一次，而信道容量为每次信道使用C比特。那么单位时间的信道容量

为每秒钟C/T_c bit。我们现在阐述Shannon的第二定理**有噪编码定理**，又叫做**信道编码定理**。

定理2.1　信道编码定理（有噪编码定理）

(1) 设一个DMS信源字符集X的熵为每信源符号$H(X)$比特，而且信源每T_s秒产生一个符号。设一个离散无记忆信道的容量为C，且每隔T_c秒使用一次。若

$$\frac{H(X)}{T_s} \leqslant \frac{C}{T_c} \tag{2-9}$$

则存在一个编码方案，使信源输出可以通过该有噪信道传输并能以任意低的错误概率恢复。

(2) 相反地，若

$$\frac{H(X)}{T_s} > \frac{C}{T_c} \tag{2-10}$$

则通过该信道传输的信息不可能以任意小的错误概率恢复。

参数$\frac{C}{T_c}$称为**临界率**（critical rate）。

信道编码定理是信息论中的一个非常重要的结果。该定理说明了信道容量C是通过有噪DMS信道实现可靠通信的基本界限。应该注意到信道编码定理告诉我们存在一些在有噪环境下实现可靠通信的码，但遗憾的是它没给出构造这些码的方法。因此信道编码仍然是一个很活跃的研究领域，因为人们仍然在搜寻着越来越好的码。从下一章开始我们将学习一些较好的信道码。

例2.2　考虑一个DMS信源，它每T_s秒等可能地（$p=0.5$）产生一个二元符号。该二元信源的熵为

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p) = 1 \text{ (bit)}$$

该信源的信息率为

$$\frac{H(X)}{T_s} = \frac{1}{T_s} \text{ (bit/s)}$$

假设我们想从一个有噪信道传输该信源符号。信源序列将被用于码率为r的信道编码器。该信道编码器每T_c秒使用一次该信道来传送编过码的序列。我们想得到可靠的通信（出错概率小到所期望的值）。根据信道编码定理，若

$$\frac{1}{T_s} \leqslant \frac{C}{T_c} \tag{2-11}$$

通过适当选取一种信道编码方案，我们可以使错误概率小到所期望的值，从而得到可靠的通信。该编码器的码率可以表示为

$$r = \frac{T_c}{T_s} \tag{2-12}$$

因此，实现可靠通信的条件可以写为

$$r \leqslant C \tag{2-13}$$

因此，对一个BSC来说，不管信道的噪声有多大，总可以找到一种信道编码方案使码率$r \leqslant C$，从而实现可靠通信！当然我们可以说至少存在这样的一种码，但要找到这种码不一定轻

而易举。我们以后将看到，信道中噪声的程度将在信道容量或码率上表现出来。

例2.3 考虑一个转移概率$p=10^{-2}$的BSC。这样的错误率在无线信道中很典型。在例2.1中，我们看到一个BSC的信道容量由下述公式给出：

$$C = 1 + p\log_2 p + (1-p)\log_2(1-p)$$

将$p=10^{-2}$的值代入，得到信道容量$C=0.919$。从前面的例子我们可以肯定至少存在一种码率$r\leqslant 0.919$的编码方案保证（非零）出错概率如预期般小。

例2.4 考虑每个消息比特都重复n次的**重复码**，其中n是一个奇数。例如当$n=3$时，我们有如下的对应关系：

$$0\rightarrow 000;\quad 1\rightarrow 111$$

类似地，当$n=5$时，我们有下列的对应关系：

$$0\rightarrow 00000;\quad 1\rightarrow 11111$$

注意组长为n的重复码的码率为

$$r=\frac{1}{n} \tag{2-14}$$

译码策略为：若在一个收到的长为n的组中0的个数比1的个数多，就判决为0，反之亦然。这种方法又叫做**最大似然译码**。这也回答了重复码中的n为什么是奇数的问题。

设$n=2m+1$，其中m是一个正整数。当有多于m个比特发生错误时，这种译码策略就会出错，因为在那种情况下，若0被编码后传送，则收到的码字中将有更多的1。我们假定0和1的先前概率相等，则平均错误概率为

$$P_e=\sum_{i=m+1}^{n}\binom{n}{i}p^i(1-p)^{n-i} \tag{2-15}$$

其中p是信道转移概率。表2-2给出了在不同码率下重复码的平均错误概率。

表2-2 重复码的平均出错概率

码率r	平均出错率P_e
1	10^{-2}
1/3	3×10^{-4}
1/5	10^{-6}
1/7	4×10^{-7}
1/9	10^{-8}
1/11	5×10^{-10}

从表2-2中我们看到随着码率的降低，平均错误概率急剧降低。P_e降低的速度远比码率r降低的速度要快得多。但是对重复码，当我们想要越来越小的P_e时，码率将趋近于零。因此重复码用码率换取消息的可靠性，但信道编码定理说要得到任意小的错误概率，码率不一定要趋于零。定理仅要求码率r小于信道容量C。所以一定存在码率$r\approx 0.9$的码（不是重复码）可以达到任意低的错误概率。这样的编码方案将只需在9个信息比特中添加1个校验比特（或者在90个信息比特中添加10个额外的比特），便会给我们想要的任意小的P_e（比如说10^{-20}）！困难部分是如何找到这样的一个码。

2.5 信息容量定理

到目前为止，我们学习的是信息在信道中可靠地传输所达到的最大速率的界限，该界限描述为信道容量。在这一节中我们将给出有限带宽、有限功率的高斯信道的信息容量定理的公式。考虑一个均值为零、带宽限为W Hz的平稳随机过程$X(t)$。设X_k，$k=1, 2, \cdots, K$表示通过对过程$X(t)$以每秒取$2W$个采样的奈奎斯特（Nyquist）速率均匀采样得到的连续随机变量。这些符号通过一个带宽限也为W Hz的有噪信道传输。信道输出遭到均值为零、功率谱密度（psd）为$N_0/2$的加性白高斯噪声（AWGN）的损坏。由于信道的局限性，噪声的带宽限也为W Hz。令$Y_k(k=1, 2, \cdots, K)$表示接收到的信号的采样。于是有

$$Y_k = X_k + N_k,\quad k=1, 2, \cdots, K \tag{2-16}$$

其中N_k是均值为零、方差为$\sigma^2 = N_0 W$的噪声采样。这里假定$Y_k\ (k=1, 2, \cdots, K)$是统计独立的。由于发射器通常具有有限功率，让我们在X_k的平均功率上加一限制条件：

$$E[X_k^2] = P,\ k=1, 2, \cdots, K \tag{2-17}$$

该有限带宽、有限功率信道的信息容量是信道输入X_k和信道输出Y_k之间互信息的最大值，最大化是在满足功率限制等式(2-17)输入X_k的所有分布上的取值。因此信道的信息容量（与信道容量相同）可表示为

$$C = \max_{f_{X_k}(x)} \{I(X; Y) \mid E[X_k^2] = P\} \tag{2-18}$$

其中$f_{X_k}(x)$是X_k的概率密度函数。

由第1章的式(1-32)得到

$$I(X_k; Y_k) = h(Y_k) - h(Y_k \mid X_k) \tag{2-19}$$

注意X_k和Y_k是独立的随机变量，因此，在X_k已知时Y_k的条件微分熵就等于N_k的微分熵。直观上这是由于在X_k给定的条件下，Y_k的不确定性的增加完全取决于N_k，即

$$h(Y_k \mid X_k) = h(N_k) \tag{2-20}$$

于是我们可将式(2-19)写为

$$I(X_k; Y_k) = h(Y_k) - h(N_k) \tag{2-21}$$

因为$h(N_k)$独立于X_k，求$I(X_k; Y_k)$的最大值的问题可以转化为求$h(Y_k)$的最大值的问题。可以证明，要想使$h(Y_k)$达到最大值，Y_k必须为高斯随机变量（见习题2.10）。若我们假定Y_k为高斯变量，而N_k根据定义也为高斯变量，那么X_k也是高斯变量。这是因为两个高斯随机变量的和（或差）仍是一个高斯变量。因此为了能使信道输入X_k和信道输出Y_k之间的互信息最大化，传输的信号也应为高斯的。因此我们可以将式(2-18)重新写为

$$C = I(X; Y) \mid E[X_k^2] = P \text{ 且 } X_k \text{ 为高斯分布} \tag{2-22}$$

我们知道若将两个独立的高斯随机变量相加，所得的高斯随机变量的方差为原来两个方差的和。因此收到的采样Y_k的方差等于$P + N_0 W$。可以证明方差为σ^2的高斯随机变量的微分熵为$1/2 \log_2(2\pi e\sigma^2)$（见习题2.10）。因此有

$$h(Y_k) = \frac{1}{2}\log_2[2\pi e(P + N_0 W)] \tag{2-23}$$

或

$$h(N_k)=\frac{1}{2}\log_2[2\pi e(N_0W)] \tag{2-24}$$

将这些值代入Y_k和N_k的微分熵中可得

$$C=\frac{1}{2}\log_2\left(1+\frac{P}{N_0W}\right)\text{(bit/信道使用)} \tag{2-25}$$

我们在每秒钟传输$2W$个采样，即在一秒钟内使用$2W$次信道。因此信息容量又可表示为

$$C=W\log_2\left(1+\frac{P}{N_0W}\right)\text{(bit/s)} \tag{2-26}$$

对于有限带宽、AWGN波型信道，当输入是有限带宽、平均有限功率时，信息容量的基本公式首先是Shannon在1948年建立的。它被称为Shannon第三定理，又称为**信息容量定理**。

定理2.2 （信息容量定理） *带宽为W Hz的连续信道的信息容量，当受功率谱密度为$N_0/2$、带宽限为W赫兹的加性白高斯噪声干扰时，可以表示为*

$$C=W\log_2\left(1+\frac{P}{N_0W}\right)\text{(bit/s)}$$

*其中P是平均传输功率。该定理又称为**信道容量定理**。*

信息容量定理是信息论中重要的结果之一。在一个公式中我们可以看到信道带宽、平均传输功率及噪声功率谱密度这三者的折中。给定信道带宽和SNR，就可以算出信道容量(bit/s)。这个信道容量是在有限功率、有限带宽的高斯信道上实现可靠通信的基本速率界限。应该记住，为了接近这一界限，传送的信号必须满足本质为高斯（Gaussian）的统计特性。注意我们在可互换地使用信道容量和信息容量的术语。

我们现在用更直观的方法来导出同样的结果。假定我们有一种可以接受的低出错概率的编码方案。设这种编码方案把k比特的信息编码成n比特长的码字。总的码字数为$M=2^k$。设每比特的平均功率为P，于是传输整个码字所需要的平均功率为nP。假设这些码字要通过一个噪声方差为σ^2的高斯信道传输，则收到的n比特向量也是高斯的，其均值与被传输的码字相同，方差为$n\sigma^2$。因为这是一个好码（可接受的出错率），向量在以传输的码字为中心、以$\sqrt{n\sigma^2}$为半径的球内。这个球又包含在半径为$\sqrt{n(P+\sigma^2)}$的大球内，这里$n(P+\sigma^2)$是接收到的向量的平均功率。

这一概念可以通过图2-6说明。有一个半径为$\sqrt{n(P+\sigma^2)}$的球，里面装有M个半径为$\sqrt{n\sigma^2}$的小球，这里$M=2^k$是码字的总数。每个这样的小球中心都是一个码字，它们称为**译码球**（decoding sphere）。任何收到的字都译码为它所在的球中心的码字。假设一个码字通过一个有噪信道传输，那么接收到的向量将以很高的概率落到正确译码球中（因为这是个相当好的码）。这就出现了一个问题：在这个大球中可以装进多少个不相交的小球？装进去的小球数越多，该码在码率方面就越有效。这称为**装球问题**（sphere packing problem）。

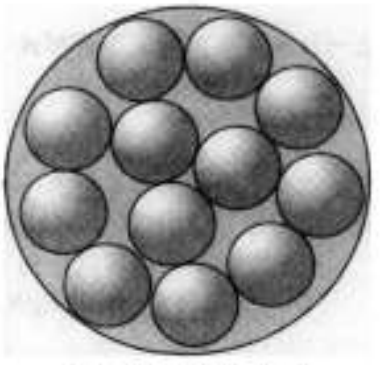


图2-6 装球问题的图示

半径为r的n维球的容积可以表示为

$$V=A_n r^n \tag{2-27}$$

其中A_n是尺度因子。因此大球（所有可能收到的向量组成的球）的容积可写为

$$V_{\text{all}}=A_n[n(P+\sigma^2)]^{n/2} \tag{2-28}$$

而译码球的容积为

$$V_{ds}=A_n[n\sigma^2]^{n/2} \tag{2-29}$$

能够装进由所有可能收到的向量组成的大球的不相交译码球的最大个数为

$$M=\frac{A_n[n(P+\sigma^2)]^{n/2}}{A_n[n\sigma^2]^{n/2}}=\left(1+\frac{P}{\sigma^2}\right)=2^{(n/2)\log_2(1+P/\sigma^2)} \tag{2-30}$$

将等式两边取以2为底的对数，得到

$$\log_2 M=\frac{n}{2}\log_2\left(1+\frac{P}{\sigma^2}\right) \tag{2-31}$$

观察到$k=\log_2 M$，可得

$$\frac{k}{n}=\frac{1}{2}\log_2\left(1+\frac{P}{\sigma^2}\right) \tag{2-32}$$

注意每次我们使用信道时实际传输了k/n比特。因此信道的每次使用所能传输的最大比特数，在低错误概率的情况下，为$1/2\ \log_2(1+P/\sigma^2)$，正如式（2-25）所示。注意$\sigma^2$表示噪声功率，在AWGN功率谱密度函数为$N_0/2$和带宽限为$W$的情况下，它等于$N_0W$。

2.6 Shannon限

考虑一个有限功率和有限带宽的高斯信道。我们希望利用在这些约束条件下一个通信系统的限度。首先定义一种理想的系统，它能以等于信道容量C的比特速率R_b传输数据，即$R_b=C$。假定每比特的能量为E_b。则平均传输功率为

$$P=E_bR_b=E_bC \tag{2-33}$$

因此，该理想系统的信道容量定理可写为

$$\frac{C}{W}=\log_2\left(1+\frac{E_b}{N_0}\frac{C}{W}\right) \tag{2-34}$$

这一公式可重新写成下列形式

$$\frac{E_b}{N_0}=\frac{2^{C/W}-1}{C/W} \tag{2-35}$$

带宽效率R_b/W相对于E_b/N_0的平面图称为**带宽效率图**（bandwidth efficiency diagram），如图2-7所示。理想系统由直线$R_b=C$表示。

从带宽效率图中可以得到下列结论：

(1) 对无限带宽，比率E_b/N_0趋向于极限值

$$\left.\frac{E_b}{N_0}\right|_{W\to\infty}=\ln 2=0.693=-1.6\ \text{dB} \tag{2-36}$$

这个值称为Shannon限。值得注意的是Shannon限是一个分数。这说明对很大的带宽来说，即使信号功率小于噪声功率，实现可靠通信也是可能的！对应于这个限定值的信道容量为

$$C|_{W\to\infty}=\frac{P}{N_0}\log_2 e \tag{2-37}$$

因此，对无限带宽，信道容量由SNR决定。

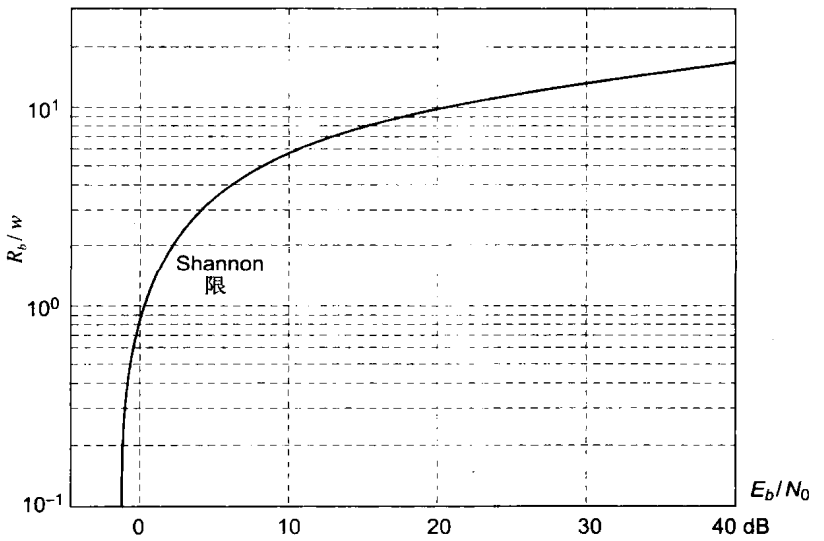


图2-7 带宽效率图

(2) 临界率$R_b=C$的曲线称为**容量边界**（capacity boundary）。当$R_b>C$时，不能保证可靠通信。但当$R_b<C$时，存在某些可以使错误概率任意低的编码方案。

(3) 带宽效率图说明了几个量值R_b/W、E_b/N_0和错误概率P_e之间的折中。注意在设计任何通信系统时，能得到的最基本的设计参数为带宽、SNR和比特错误率（BER）。BER由具体应用及预期的服务质量（QoS）决定，而带宽和功率可以相互折中提供理想的BER。

(4) 带宽效率图上的每一个点对应与一组SNR、带宽效率和BER值相当的一个操作点。

信息容量定理预测对给定的SNR，通过一个给定带宽所能传输的最大信息量。从图2-7中可以看到即使对低的SNR，只要有充足的带宽，也可以得到能接受的容量。当信号有点像噪声而最小SNR维持在接收端时，一个给定的带宽可以得到最优使用。这一原理是任何扩频通信系统，如**码分多址**（CDMA）的核心。

2.7 MIMO系统的信道容量

我们使用一个$M_R\times M_T$维的矩阵$\boldsymbol{H}$来对式(2-3)中描述的MIMO信道进行建模。假设平均传输符号能量是E_S，则取样的信号模型可以表示为：

$$\boldsymbol{y}[k]=\sqrt{\frac{E_s}{M_T}}\boldsymbol{H}_s[k]+\boldsymbol{n}[k] \tag{2-38}$$

其中，$\boldsymbol{y}[k]$是所收到的$M_R\times 1$维信号向量，$\boldsymbol{s}[k]$是所传送的$M_T\times 1$维信号向量，$\boldsymbol{\eta}[k]$是$M_R\times 1$维时空的零均值复高斯白噪音向量，并且每一维的方差是N_0。这种信号模型适合**频率平坦衰弱**（frequency flat fading）信道。为更清楚地表示，当上式中去掉时间指数k后，就变成了如下的

形式：

$$y = \sqrt{\frac{E_s}{M_T}} H_s + n \tag{2-39}$$

s的协方差矩阵是：

$$R_{ss} = \mathrm{E}\{ss^{\mathrm{H}}\}. \tag{2-40}$$

上标'H'表示**厄米特操作**（Hermitian operation）。我们假设信道H是确定信道，并且被接收方知道。则使用控制信号或者训练信号，接收方可以计算出**信道状态信息**（channel state information，CSI）。MIMO信道容量可以表示为：

$$C = \max_{Tr(R_{ss})=M_T} W\log_2 \det\left(I_{M_R} + \frac{E_s}{M_T N_0} HR_{ss}H^H\right) \text{(bit/s)} \tag{2-41}$$

其中W是带宽，I_{M_R}是和矩阵M_R同样大小的单位矩阵。条件$Tr(R_{ss}) = M_T$限制了整个符号周期中全部平均能量的传输。

如果传输者不知道信道，向量s的选择可能使得$R_{ss} = I_{MT}$。这就意味着，发射天线中的信号是独立的并且有相同的功率。这种情况下，MIMO信道的容量可以表示为：

$$C = W\sum_{i=1}^{r} \log_2\left(1 + \frac{E_s}{M_T N_0}\lambda_i\right) \text{(bit/s)} \tag{2-42}$$

其中r是信道的秩，λ_i (i=1, 2, …, r)是HH^H的正的特征值。非常有意思的是，对于某个传输者未知的MIMO信道，其容量是r个SISO 信道之和，每一个SISO信道有功率增益λ_i (i=1, 2, …, r)和相同的传输功率E_s/M_T。可以这样解释，在发送者和接收者之间，使用多个发射和接收天线可以引起多个并行的数据通道。这些标量数据通道的数目依赖于H的秩。

接下来，我们考虑满秩的MIMO信道，此信道满足$M_T = M_R = M$。因此$r = M$。当H是一个正交矩阵（也就是HH^H=H^HH）时，就可以达到最大信道容量。此MIMO的信道容量可以表示为：

$$C = WM\log_2\left(1 + \frac{E_s}{M_T N_0}\right) \text{(bit/s)} \tag{2-43}$$

一个正交的MIMO信道的容量是M乘以标量信道容量。如果传输者知道信道，通过在传输者端和接收者端的处理，不同的标量数据通道可以被分别访问。基本思想是，为了最大化互信息，根据不同的数据通道来对变量的能量进行分配。迭代地运用Waterpouring**算法**就可以达到最优能量。当传输者知道信道的时候，MIMO信道的容量一定大于或者等于当传输者不知道信道时MIMO信道的容量。

2.8　码的随机选取

考虑由一组n维二元码字构造的一组M个编码后的信号波形。我们把这些码字表示如下：

$$C_i = [c_{i1}\ c_{i2} \cdots c_{in}],\ i = 1,2, \cdots, M \tag{2-44}$$

因为我们考虑的是二元码，c_{ij}是0或1。设码字的每一个比特都映射到一个BPSK波型p_j，因此码字可以表示为

$$s_i(t) = \sum_{i=1}^{n} s_{ij}\, p_j(t),\ i = 1,2, \cdots, M \tag{2-45}$$

其中

$$s_{ij}=\begin{cases}\sqrt{E} & \text{若 } c_{ij}=1\\ -\sqrt{E} & \text{若 } c_{ij}=0\end{cases} \tag{2-46}$$

这里$\sqrt{E}$是每个码字比特的能量。波形$s_i(t)$又可表示为n维向量

$$\boldsymbol{s}_i=[s_{i1}\ s_{i2}\ \cdots\ s_{in}]\quad i=1,2,\cdots,M \tag{2-47}$$

我们观察到这对应到n维空间中的超立方体。现在让我们将k比特的信息编码成n比特长的码字，再将这个码字映射到M个波形之一。注意对应$M=2^k$个不同码字的可能波形总共有2^k个。

设编码器的信息率为每秒R比特。编码器每次接受k比特，然后把这k比特块映射到M个波形之一。所以$k=RT$，而且需要$M=2^k$个信号。

我们定义如下的参数D

$$D=\frac{n}{T}\ (\text{维/秒}) \tag{2-48}$$

$n=DT$称为空间的**维数**。上述超立方体有$2^n=2^{DT}$个顶点，我们必须从中选$M=2^{RT}$个来传递信息。在$D>R$的约束条件下，可用作信号点的顶点所占比例为

$$F=\frac{2^k}{2^n}=\frac{2^{RT}}{2^{DT}}=2^{-(D-R)T} \tag{2-49}$$

对$D>R$，当$T\to\infty$时，有$F\to 0$。因为$n=DT$，这表明当$n\to\infty$时，有$F\to 0$。设计一个好的编码方案的问题转化为如何从超立方体的2^n个顶点中选取M个顶点，当增加n时错误概率趋于零的问题。我们已经看到了当不断增大n的时候，分数F趋于零，这说明当$n\to\infty$时，增加这M个信号点之间的最小距离是可能的。增加信号点之间的最小距离将使错误概率$P_e\to 0$。

从总共2^n个顶点中选取M个共有$(2^n)^M$种不同的选法。每种选法都对应一种编码方案。对每一组M个波形，有可能设计由一个调制器和一个解调器组成的通信系统。因此有2^{nM}个通信系统，每一个都对应M个编码波形的选择。每一个这样的通信系统都用它的错误概率描述。当然，很多这类通信系统在错误概率方面表现得很差。

让我们从2^{nM}组码中随机选取一个码。随机选取第m个码字时的概率为

$$P(\{\boldsymbol{s}_i\}_m)=\frac{1}{2^{nM}} \tag{2-50}$$

设这样选取的码的错误概率为$P_e(\{s_i\}_m)$。那么全部码的集合的平均错误概率为

$$\begin{aligned}\overline{P}_e&=\sum_{m=1}^{2^{nM}}P_e(\{\boldsymbol{s}_i\}_m)P(\{\boldsymbol{s}_i\}_m)\\&=\frac{1}{2^{nM}}\sum_{m=1}^{2^{nM}}P_e(\{\boldsymbol{s}_i\}_m)\end{aligned} \tag{2-51}$$

下面将给出这个平均错误概率的上界。如果我们有$\overline{P}_e$的一个上界，可以肯定至少存在一个码满足该上界条件。另外，如果$n\to\infty$时有$\overline{P}_e\to 0$，则我们可以猜测当$n\to\infty$时也有$P_e(\{s_i\}_m)\to 0$。

考虑对k比特消息$\boldsymbol{X}_k=[x_1\ x_2\ \cdots\ x_k]$的传输问题，其中$x_j$ ($j=1,2,\cdots,k$)是二元的。在所有可能码上平均条件错误概率为

$$\overline{P_e(\boldsymbol{X}_k)} = \sum_{\text{所有码}} P_e(\boldsymbol{X}_k, \{\boldsymbol{s}_i\}_m) P(\{\boldsymbol{s}_i\}_m) \tag{2-52}$$

其中$P_e(\boldsymbol{X}_k, \{\boldsymbol{s}_i\}_m)$是给定通过码$\{\boldsymbol{s}_i\}_m$传输的$k$比特消息$\boldsymbol{X}_k=[x_1\ x_2\ \cdots\ x_k]$的条件错误概率。对第$m$个码有

$$P_e(\boldsymbol{X}_k, \{\boldsymbol{s}_i\}_m) \leqslant \sum_{\substack{l=1 \\ l\neq k}}^{M} P_{2m}(\boldsymbol{s}_l, \boldsymbol{s}_k) \tag{2-53}$$

其中$P_{2m}(\boldsymbol{s}_l, \boldsymbol{s}_k)$是用信号向量$\boldsymbol{s}_l$和$\boldsymbol{s}_k$传输两个等可能出现的$k$比特消息之一的二元通信系统的错误概率。故

$$\overline{P_e(\boldsymbol{X}_k)} \leqslant \sum_{\text{所有码}} P_e(\{\boldsymbol{s}_i\}_m) \sum_{\substack{l=1 \\ l\neq k}}^{M} P_{2m}(\boldsymbol{s}_l, \boldsymbol{s}_k) \tag{2-54}$$

通过改变求和次序，可得

$$\overline{P_e(\boldsymbol{X}_k)} \leqslant \sum_{\substack{l=1 \\ l\neq k}}^{M} \left[\sum_{\text{所有码}} P_e(\{\boldsymbol{s}_i\}_m) P_{2m}(\boldsymbol{s}_l, \boldsymbol{s}_k) \right] \leqslant \sum_{\substack{l=1 \\ l\neq k}}^{M} \overline{P_2(\boldsymbol{s}_l, \boldsymbol{s}_k)} \tag{2-55}$$

其中$\overline{P_2(\boldsymbol{s}_l, \boldsymbol{s}_k)}$表示$P_{2m}(\boldsymbol{s}_l, \boldsymbol{s}_k)$在$2^{nM}$个码上的总平均值。对加性白高斯噪声信道，有

$$P_{2m}(\boldsymbol{s}_1, \boldsymbol{s}_k) = Q\left(\sqrt{\frac{d_{lk}^2}{2N_0}}\right) \tag{2-56}$$

其中$d_{lk}^2 = |\boldsymbol{s}_l - \boldsymbol{s}_k|^2$，$Q(x) = \dfrac{1}{2\pi}\displaystyle\int_x^{\infty} \mathrm{e}^{-x^2/2}\mathrm{d}z$。若$\boldsymbol{s}_l$与$\boldsymbol{s}_k$有$d$个坐标不同，则

$$d_{lk}^2 = |\boldsymbol{s}_l - \boldsymbol{s}_k|^2 = \sum_{j=1}^{n} (s_{lj} - s_{kj})^2 = d(2\sqrt{E})^2 = 4dE \tag{2-57}$$

因此，

$$P_{2m}(\boldsymbol{s}_l, \boldsymbol{s}_k) = Q\left(\sqrt{\frac{2dE}{N_0}}\right) \tag{2-58}$$

在所有码都均等的假设下，向量$\boldsymbol{s}_l$可以等可能地是超立方体的2^n个顶点中的任一个。进一步，$\boldsymbol{s}_l$和$\boldsymbol{s}_k$是统计独立的，因此$\boldsymbol{s}_l$和$\boldsymbol{s}_k$恰好在d处不相同的概率为

$$P(d) = \left(\frac{1}{2}\right)^n \binom{n}{d} \tag{2-59}$$

于是$P_{2m}(\boldsymbol{s}_l, \boldsymbol{s}_k)$在全部码上的期望值为

$$P_2(\boldsymbol{s}_l, \boldsymbol{s}_k) = \sum_{d=0}^{n} P(d)\, Q\left(\sqrt{\frac{2dE}{N_0}}\right) = \left(\frac{1}{2^n}\right) \sum_{d=0}^{n} \binom{n}{d} Q\left(\sqrt{\frac{2dE}{N_0}}\right) \tag{2-60}$$

利用下面的上界

$$Q\left(\sqrt{\frac{2dE}{N_0}}\right) < \mathrm{e}^{\frac{-dE}{N_0}} \tag{2-61}$$

可得

$$\overline{P_2(s_l,s_k)} \leqslant \left(\frac{1}{2^n}\right)\sum_{d=0}^{n}\binom{n}{d}e^{\frac{-dE}{N_0}} \leqslant \left[\frac{1}{2}\left(1+e^{-\frac{E}{N_0}}\right)\right] \tag{2-62}$$

由式（2-55）和式（2-62）我们得到

$$\overline{P_e(X_k)} \leqslant \sum_{\substack{l=1\\ l\neq k}}^{M}\overline{P_2(s_l,s_k)} = (M-1)\left[\frac{1}{2}\left(1+e^{\frac{-E}{N_0}}\right)\right]^n < M\left[\frac{1}{2}\left(1+e^{-\frac{E}{N_0}}\right)\right]^n \tag{2-63}$$

我们需要平均错误概率$\overline{P}_e$的一个上界。要得到$\overline{P}_e$，我们将$\overline{P_e(X_k)}$在所有可能的k比特信息序列上取平均值。于是

$$\overline{P}_e = \sum_k \overline{P_e(X_k)}\,P(X_k) < M\left[\frac{1}{2}\left(1+e^{-\frac{E}{N_0}}\right)\right]^n \sum_k P_e(X_k) < M\left[\frac{1}{2}\left(1+e^{-\frac{E}{N_0}}\right)\right]^n \tag{2-64}$$

现在定义如下的参数。

定义2.4 定义**切断率**（cutoff rate）R_0如下：

$$R_0 = \log_2\frac{2}{1+e^{-\frac{E}{N_0}}} = 1-\log_2\left(1+e^{-\frac{E}{N_0}}\right) \tag{2-65}$$

切断率的单位为比特/维。观察上式可知$0\leqslant R_0\leqslant 1$。$R_0$相对每维的SNR的平面图由图2-8给出。

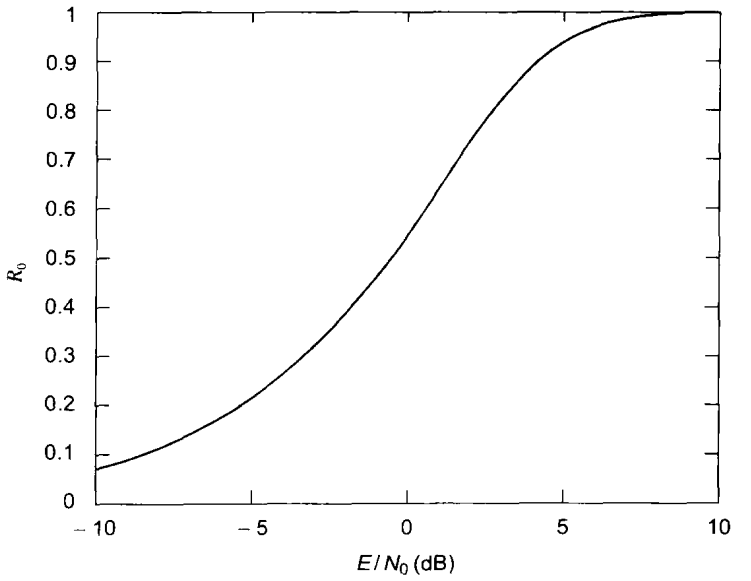


图2-8 切断率R_0相对每维的SNR（dB数）

现在式(2-64)可以简写为

$$\overline{P}_e < M2^{-nR_0} = 2^{-RT}2^{-nR_0} \tag{2-66}$$

将$n=DT$代入，得到

$$\overline{P}_e < 2^{-T(DR_0 - R)} \tag{2-67}$$

如果将$T=n/D$代入，则得到

$$\overline{P}_e < 2^{-n(R_0 - R/D)} \tag{2-68}$$

观察到

$$\frac{R}{D} = \frac{R}{n/T} = \frac{RT}{n} = \frac{k}{n} = R_c \tag{2-69}$$

这里R_c表示码率。于是平均错误概率可以写成下面的形式

$$\overline{P}_e < 2^{-n(R_0 - R_c)} \tag{2-70}$$

从上述等式我们可以得到下列结论：

(1) 对于$R_c < R_0$，当$n \to \infty$时，平均错误概率$\overline{P}_e \to 0$。因为通过选取一个大的n值，可以使$\overline{P}_e$任意小，因此理论上存在一个好码使错误概率小于$\overline{P}_e$。

(2) 我们观察到$\overline{P}_e$是总体平均值。因此若一个码是随机选取的，错误$\overline{P}_e > \alpha\overline{P}_e$的概率小于$1/\alpha$。这说明只有不超过10%的码其错误概率超过$10\overline{P}_e$。因此有很多好码存在。

(3) 那些错误概率超出$\overline{P}_e$的码并不都是差的码。这些码的错误概率可以通过增加维数n来降低。

对二元编码信号，当切断率R_0在E/N_0取很大的值时，比如说$E/N_0 > 10$，即达到饱和值1比特/维。因此要取得低的错误概率，必须减小码率R_c。另外，可以使用很长的组块，但这不是一种有效的方法。因此二元码在高SNR情况下是有效的。对那些高SNR的情况，应该用非二元编码信号集来增加每维的比特数。通过将不同振幅水平（例如脉冲振幅调制）映射到码元素，多级振幅编码信号集也可以很容易从非二元码构造。对M辐多级振幅信号的随机码，Shannon（在1959年）证明了

$$R_0^* = \frac{1}{2}\left[1 + \frac{E}{N_0} - \sqrt{1 + \left(\frac{E}{N_0}\right)^2}\right]\log_2 e + \frac{1}{2}\log_2\left[\frac{1}{2}\left(1 + \sqrt{1 + \left(\frac{E}{N_0}\right)^2}\right)\right] \tag{2-71}$$

现在让我们将切断率R_0^*同AWGN信道的容量联系起来，该信道容量为

$$C = W\log_2\left(1 + \frac{P}{N_0 W}\right) \text{ (bit/s)} \tag{2-72}$$

每码比特的能量等于

$$E = \frac{PT}{n} \tag{2-73}$$

回想一下采样定理，一个带宽为W的信号可以由速率为$2W$的采样表示。因此在长度为T的时间段内有$n=2WT$个采样。因此我们可以写为$D=n/T=2W$，可得

$$P = \frac{nE}{T} = DE \tag{2-74}$$

定义规范容量$C_n = C/2W = C/D$，替换式(2-72)中的W和P得到

$$C_n = \left(\frac{1}{2}\right)\log_2\left(1+2\frac{E}{N_0}\right) = \left(\frac{1}{2}\right)\log_2(1+2R_c\gamma_b) \tag{2-75}$$

其中γ_b是每比特的SNR。规范容量C_n和切断率R_0^*的平面图如图2-9所示。从图中我们可以得到下述结论：

(1) 对所有E/N_0的值，都有$R_0^* < C_n$。这是预料之内的，因为C_n是传输速率R/D的最终限。

(2) C_n和R_0^*之间的较小的差大概在3dB左右。这说明随机选取的、有限平均功率的多级振幅信号使R_0^*在信道容量的3dB之内。

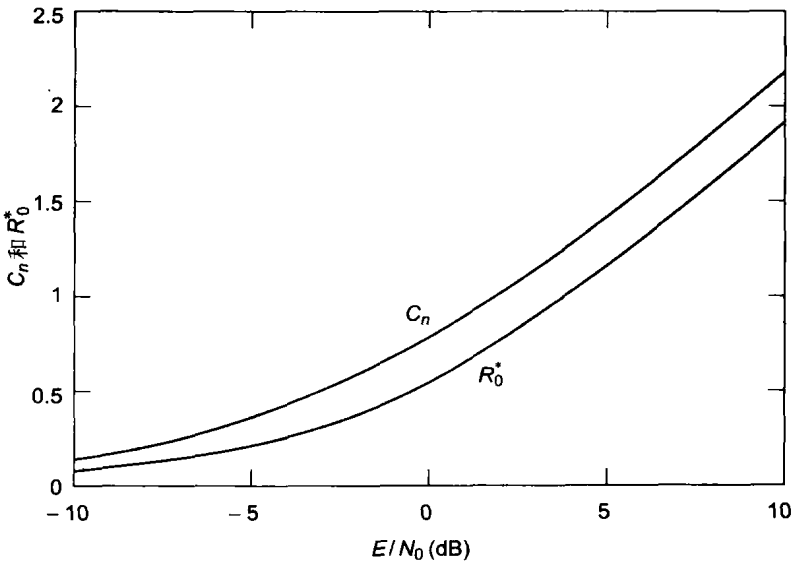


图2-9 AWGN信道的规范容量C_n和切断率R_0^*

2.9 评注

Shannon在1948年进行了开创性工作。Shannon的第二定理在发表时确实是一个惊人的结果。它声明只要码率小于信道容量，一个BSC的错误概率便可做到想多小就多小。这个定理为系统研究在不可靠（有噪）信道中实现可靠通信铺平了道路。Shannon第三定理——信息容量定理，是信息论中最辉煌的成果之一，它给出了信道带宽、信噪比和信道容量的关系。在20世纪50年代和60年代，Gilbert、Gallager、Wyner、Forney和Viterbi等人又作出了进一步的贡献。

切断率的概念也是Shannon提出来的，但后来被Wozencraft、Jacobs和Kennedy用作通信系统的设计参数。Jordan用切断率的概念设计了具有相关检测和非相关检测的M辐正交信号的编码波形。切断率被广泛用作不同信道的设计标准，包括无线通信中遇到的衰退信道。

2.10 小结

- 条件概率$P(y_i|x_j)$称为信道转移概率，记为p_{ji}。描述DMC的条件概率$\{P(y_i|x_j)\}$可以表示为

矩阵$\boldsymbol{P}=[p_{ji}]$的形式。矩阵$\boldsymbol{P}$称为信道的概率转移矩阵。

- 离散无记忆信道（DMC）的容量定义为任何一次信道使用的最大平均互信息，其中最大值是在所有输入概率上的取值，即

$$C=\max_{P(x_j)} I(X;Y)=\max_{P(x_j)}\sum_{j=0}^{q-1}\sum_{i=0}^{r-1}P(x_j)P(y_i|x_j)\log\frac{P(y_i|x_j)}{P(y_i)}$$

- 信道编码的基本目标是增加数字通信系统对信道噪声的抵抗力。这通过在要传输的数据流中有控制地添加冗余度来实现。信道编码又称为错误控制编码。
- 比率$r=k/n$称为码率。任何编码方案的码率都总是小于1。
- 设一个字母集为X，熵为$H(X)$ 的DMS每T_s秒产生一个符号。设该DMS的容量为C，并且每T_c秒被使用一次。那么若$H(X)/T_s\leqslant C/T_c$，则存在一种编码方案使通过该有噪信道的信源输出可以以任意低的错误概率恢复。这就是信道编码定理或有噪编码定理。
- 若$H(X)/T_s>C/T_c$，则通过该信道传输的信息不可能以任意小的错误概率恢复。参数C/T_c称为**临界率**。
- 一个SISO信道的信息容量可以表示为每秒$C=W\log_2(1+P/(N_0W))$比特。这是计算有限带宽的AWGN波形信道当输入为有限带宽和有限功率时的容量的基本公式。这是信息容量定理的重要之处。这一定理也称作信道容量定理。
- 频率平坦确定MIMO信道容量可以表示为

$$C=\max_{T_r(\boldsymbol{R}_{ss})=M_T} W\log_2\det\left(\boldsymbol{I}_{MR}+\frac{E_s}{M_TN_0}\boldsymbol{H}\boldsymbol{R}_{ss}\boldsymbol{H}^H\right)(\text{bit/s})$$

基中W是带宽。条件$\mathrm{Tr}(\boldsymbol{R}_{ss})=M_T$限制了整个符号周期中全部平均能量的传输。在此处假设接收者知道信道。

- 如果传输者不知道这个信道，那么MIMO信道容量可表示为

$$C=W\sum_{i=1}^{r}\log_2\left(1+\frac{E_s}{M_TN_0}\lambda_i\right)(\text{bit/s})$$

其中r是信道的秩，$\lambda_i(i=1,2,\cdots,r)$是$\boldsymbol{H}\boldsymbol{H}^H$的正的特征值。

- 如果传输者知道信道，通过在传输者端和接收者端的处理，不同的标量数据通道可以被分别访问。迭代地运用Waterpouring算法就可以为每个数据通道达到最优能量。

- 切断率R_0由公式 $R_0=\log_2\dfrac{2}{1+\mathrm{e}^{-\frac{E}{N_0}}}=1-\log_2\left(1+\mathrm{e}^{-\frac{E}{N_0}}\right)$给出。切断率的单位为bit/维。注意$0\leqslant R_0\leqslant 1$。平均错误概率用切断率可以表示为$\overline{P}_e<2^{-n(R_0-R_c)}$。对于$R_c<R_0$的情况，当$n\to\infty$时，有平均错误概率$\overline{P}_e\to 0$。

数学就像象棋，适合年轻人，不太难，很有趣，而且对国家没危险。

——Plato（公元前429—347）

习题

2.1　考虑图2-10所示的二元信道。设发送二元符号的先验概率为p_0和p_1，其中$p_0+p_1=1$。求

后验概率$P(X=0|Y=0)$和$P(X=1|Y=1)$。

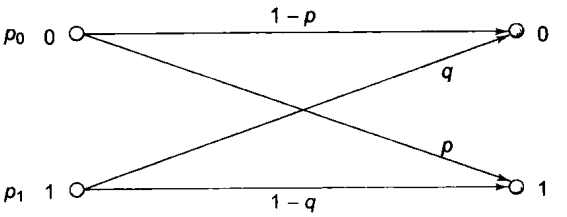


图 2-10

2.2 求图2-11中所示的二元涂抹信道的容量，其中p_0和p_1为先验概率。

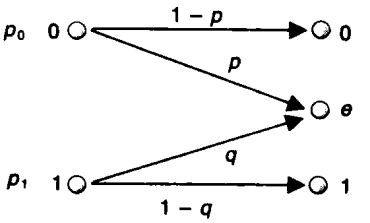


图 2-11

2.3 考虑图2-12所示的信道A、B和级联信道AB。

(1) 求信道A的容量C_A。

(2) 求信道B的容量C_B。

(3) 将两个信道级联并求联合信道容量C_{AB}。

(4) 解释C_A、C_B和C_{AB}之间的关系。

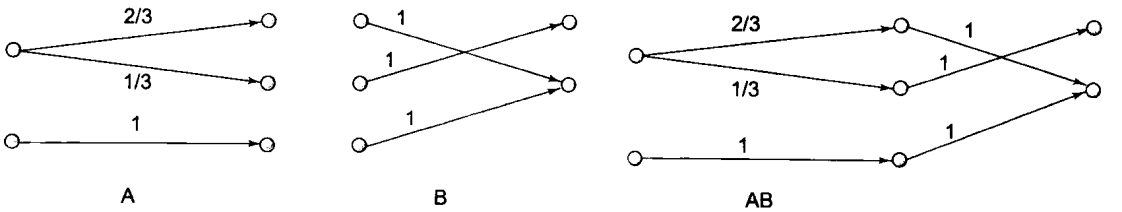


图 2-12

2.4 求图2-13所示信道的容量。

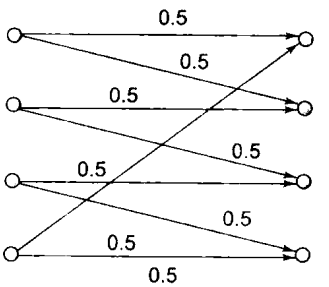


图 2-13

2.5 (1) 一个电话信道具有带宽3000Hz且SNR＝20dB。求信道容量。

(2) 若SNR增加到25dB，求信道容量。

2.6 确定图2-14所示信道的容量。

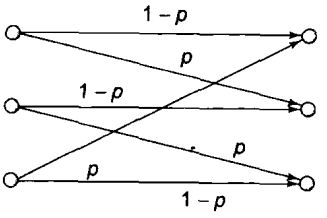


图 2-14

2.7 假定电视每秒钟显示30个画面，每个画面大约有2×10^5个像素，每个像素需要16bit的彩色显示。假定SNR为25dB，计算支持电视图像信号传输所需要的带宽（利用信息容量定理）。

2.8 考虑图2-15所示的Z型信道。

(1) 求获得信道容量所需的输入概率。

(2) 若将N个这样的信道相级联，证明联合信道可以用一个信道转移概率为$(1-p)^N$的等价Z信道表示。

(3) 当$N\to\infty$时联合信道的容量是什么？

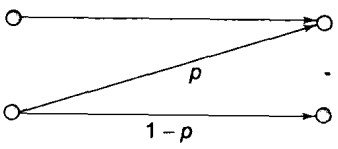


图 2-15

2.9 考虑一个使用对跖信号（antipodal signaling）的通信系统。它的SNR是20dB。

(1) 求切断率R_0。

(2) 我们想设计一种码使平均错误概率为$\overline{P}_e<10^{-6}$。我们能得到的最好码率是多少？

(3) 这个码的维数n将是多少？

(4) 对SNR＝5dB的情况重复上述步骤(1)、(2)和(3)。对所得结果进行比较。

2.10 (1) 证明对有限方差σ^2，高斯随机变量具有所有随机变量可能获得的最大微分熵。

(2) 证明该熵由公式$1/2\log_2(2\pi e\sigma^2)$给出。

上机习题

2.11 写一个程序，它在输入信道转移概率矩阵后计算出信道容量。

2.12 对M−PSK画图显示操作点在带宽效率图上的位置，其中M＝2、4、8、16和32，错误概率为

(1) $P_e=10^{-6}$；

(2) $P_e=10^{-8}$。

2.13 写一个程序实现码率为$1/n$的二元重复码，其中n是一个奇数。对该重复码建立一个解码器。用信道转移概率为p的BSC来测试该编码方案的性能。将该程序推广到$GF(q)$上码率为$1/n$的重复码。画出剩余比特错误率（BER）对p和q的图（画个三维网图）。

2.14 编写一个程序，实现对应M_T条发射天线和M_R条接收天线的容量相对于SNR的曲线图。

(1) 实现下列组合的曲线图：$(M_T, M_R)=(1,1),(1,2),(1,3),(2,1),(2,2),(3,1)$。

(2) 比较MISO和SIMO的容量，并进行相关评价。

(3) 如果我们有四条天线，则下面的组合哪种最优：(1, 3), (2, 2)或(3, 1)？并证明。

第二部分　错误控制编码（信道编码）

第3章　纠错线性分组码

数学是一种有趣的智力运动，但不允许它阻挡获取关于物理过程的有用信息。

——Richard W. Hamming

3.1　纠错码简介

在这个信息时代，存储、恢复和传输数据不但对速度而且对准确性有着越来越高的需求。用来传输消息的信道通常不是完美的。机器也会出错，它们所出的那些非人工错误可以使完好的程序变成无用的，甚至是有害的垃圾。就像建筑师们设计即使在地震中也不会倒的建筑一样，计算机专家们已经使用了精确技术能够抵消Murphy定律（"如果什么东西可能出错，那么它将会出错"）的数字表现。**纠错码**（error correcting code）是一种安全措施——针对不完美数字世界的奇异变化的数学保险。

纠错码，就像它的名字表明的那样，是当消息经过有噪信道传输或要恢复存储的数据时用来纠错的。用来传输消息的物理介质叫做信道（如电话线、卫星连接、用于移动通信的无线信道等）。不同种类的信道易产生不同种类的噪声，对传输的数据造成损害。噪声的产生可以是因为光、人类错误、设备故障、电压起伏等。因为这些纠错码试图克服信道中噪声造成的损害，其编码过程又称为**信道编码**。错误控制码也用于将信息从一个地方到另一个地方精确传输，例如存储数据，然后从一个CD上读出来。在这种情况下，错误可能是由于CD表面的一点划痕所致。纠错码的编码方案将试图从受损的数据中恢复原始数据。学习错误控制编码的动机来自这样一个现实，即现在日常交流用到大量数据和在通信、存储和获取数据方面需要具有更高可靠性。

纠错码的基本思想是在消息通过一个有噪信道传输前以多余符号的形式在消息中增添冗余度，这种冗余度是在控制下添加的。编码后的消息在传输时可能还会遭到信道中噪声的损害。在接收端，如果错误数在该编码策略所设计的限度内，原始消息可以从受损的消息中恢复。图3-1显示了数字通信系统的框图，注意图中最重要的部分就是噪声部分，没有它的话就用不着信道编码器了。

例3.1　让我们来看冗余度是如何同噪声的影响做"斗争"的。我们用来交流的正常语言（如英语）有很大的冗余度。考虑下面的句子：

CODING THEORY IS AN INTRSTNG SUBJCT

我们看到，在这个句子中有几处错误。但由于对这种语言的熟悉，我们可以猜到原来的句子应该是

CODING THEORY IS AN INTERESTING SUBJECT

我们刚才用到的就是一种纠错策略，它利用了英语中固有的冗余度来从受损的消息中恢复原始消息。

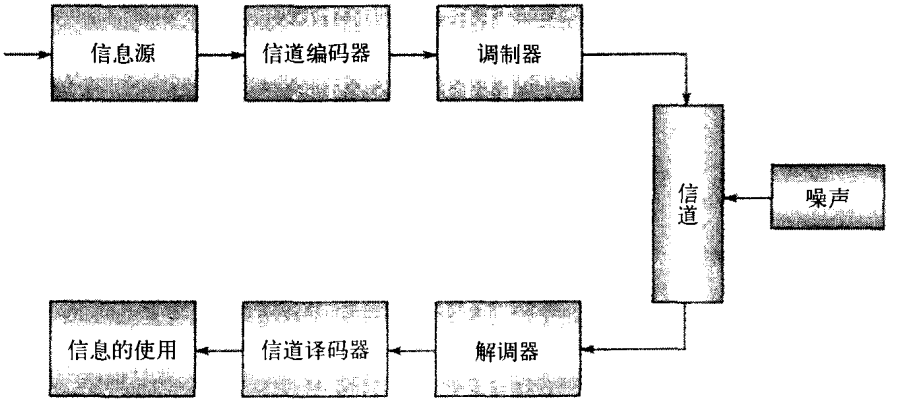


图3-1　数字通信系统框图（和原理），这里没显示信源编码器/译码器

一个好的错误控制编码方案的目标是：

(1) 用可以纠正的错误个数来衡量的纠错能力。

(2) 快速有效地对消息进行编码。

(3) 快速有效地对接收到的消息进行译码。

(4) 单位时间内所能传输的信息比特数尽量大（即有较少的冗余度）。

第一个目标是最基本的。为了增加一个编码方案的纠错能力，必须引入更多的冗余度。但增加的冗余度会造成实际信息传输速率的降低。因此第1条和第4条的目标不完全相容。另外，为了能纠正更多的错误，编码策略会变得更复杂，于是第2条和第3条的目标也很难达到。

本章中我们将首先学习错误控制编码的基本定义。我们将看到这些定义贯穿全书。然后我们引入**线性分组码**（linear block code）的定义。**线性分组码**构成一大类有用的码。我们将看到用矩阵来描述这类码很方便。本章后面部分我们将学习如何有效地对这些**线性分组码**译码。我们同时会学习**汉明码**（Hamming Code）、**低密度奇偶校验码**（Low Density Parity Check Code）和**空时分组码**（Space Time Block Code），最后引入完备码和最优线性码的概念。

3.2　基本定义

本节给出的基本定义在本章以及以后的章节中会频繁地用到。

定义3.1　**字**是一些符号的序列。

定义3.2　**码**是称为**码字**的向量的集合。

定义3.3　一个码字（或任何向量）的**汉明重量**（Hamming Weight）等于该码字中的非零元素个数。码字c的汉明重量记为$w(c)$。两个码字之间的**汉明距离**（Hamming Distance）是码字不相同的位置的数目。两个码字c_1和c_2之间的汉明距离记为$d(c_1, c_2)$。容易看出$d(c_1, c_2)=w(c_1-c_2)$。

例3.2 考虑有两个码字{0100，1111}的码C，码字的汉明重量为$w(0100)=1$和$w(1111)=4$。这两个码字间的汉明距离为3，因为它们在第1、第3和第4个位置上不同。观察得到$w(0100-1111)=w(1011)=3=d(0100,1111)$。

例3.3 对于码$C=\{01234, 43210\}$，每个码字的汉明重量都是4，码字之间的汉明距离也是4（因为两个码字只在第3个位置相同，在其余4个位置都不同）。

定义3.4 一个**分组码**由具有固定长度的码字集合构成。这些码字的固定长度称为**分组长度**（block length），通常记为n。因此一个分组长度为n的码由一组有n个**分量**（component）的码字的集合构成。

定义在q个符号的字母集上的大小为M的分组码是M个q元序列的集合，每个序列的长度为n。对$q=2$的特殊情况，那些符号称为比特，而码称为二元码。通常对某个整数k有$M=q^k$，我们称这样的码为(n,k)码。

例3.4 码$C=\{00000, 10100, 11110, 11001\}$是分组长度等于5的一个分组码。该码可用来表示两个比特的二元数字，如下所示：

未编码的比特	码字
00	00000
01	10100
10	11110
11	11001

这里$M=4$，$k=2$且$n=5$。假设我们要用上述编码方案传输由0和1构成的一个序列，比如，要编码的序列为1001010011…第一步是把这个序列分成两个比特一组（因为我们要每次编码两个比特），那么我们做如下分割

$$10\ 01\ 01\ 00\ 11\cdots$$

接下来把每个组用它们对应的码字代换：

$$11110\ 10100\ 10100\ 00000\ 11001\cdots$$

因此对每两个比特未编码的消息，我们发送5比特（编码后）。应该观察到对每2比特的信息，我们发送3个额外比特（冗余度）。

定义3.5 一个(n,k)码的**码率**定义为比率(k/n)，它表示码字所含信息符号的分数。

码率总是小于1。码率越小，冗余度就越大，即在一个码字中添加给每个信息符号的冗余符号越多。一个码有越多的冗余度，就有检测和纠正更多错误符号的能力，但也降低了传输信息的实际速率。

定义3.6 一个码的**最小距离**（Minimum Distance）就是任何两个码字之间的最小汉明距离。如果码C由码字集合$\{c_i, i=0, 1, \ldots, M-1\}$中的码字组成，那么该码的最小距离为$d^*=\min d(c_i, c_j)$，$i\neq j$。一个最小距离为$d^*$的$(n,k)$码有时候记为$(n,k,d^*)$码。

定义3.7 一个码的**最小重量**（Minimum Weight）是所有非零码字的最小重量，记为w^*。

定义3.8 一个**线性码**具有下述性质：

(1) 两个属于该码的码字的和仍是一个属于该码的码字。

(2) 全零字总是一个码字。

(3) 一个线性码的两个码字之间的最小距离等于任何非零码字的最小重量，即$d^*=w^*$。

注意，如果两个码字的和是另外一个码字，则该两个码字的差也将仍然是一个合法码字。例如，若c_1、c_2和c_3是码字，且$c_1+c_2=c_3$，那么有$c_3-c_1=c_2$。所以，对一个线性分组码，全零码字必为一个合法码字（一个码字自减的结果）。

定理3.1 对于一个线性码，它的最小距离就等于码的最小重量，即$d^*=w^*$。

直观证明：任何码字c_i和c_j之间的距离d_{ij}就是由c_i-c_j所形成的码字的重量。因为该码为线性的，任何两个码字的差是另一个合法码字。故非零码字的最小重量就反映了码的最小距离。

例3.5 码$\boldsymbol{C}=\{0000, 1010, 0101, 1111\}$是一个分组长度$n=4$的线性分组码。观察码字之间所有十种可能的和：

$0000+0000=0000$，$0000+1010=1010$，$0000+0101=0101$，

$0000+1111=1111$，$1010+1010=0000$，$1010+0101=1111$，

$1010+1111=0101$，$0101+0101=0000$，$0101+1111=1010$，

$1111+1111=0000$

它们都在$\boldsymbol{C}$中，全零码字也在$\boldsymbol{C}$中。该码的最小距离为$d^*=2$。为了验证这个线性码的最小距离，我们可确定所有码字对（共$\binom{4}{2}=6$对）之间的距离：

$d(0000, 1010)=2$，$d(0000, 0101)=2$，$d(0000, 1111)=4$

$d(1010, 0101)=4$，$d(1010, 1111)=2$，$d(0101, 1111)=2$

我们观察到这个码的最小距离为2。

注意例3.4中给出的码不是线性的，因为$1010+1111=0101$不是一个合法的码字。即使全零字是个合法码字，它也并不保证线性。**全零码字的出现是线性的必要条件但不是充分条件。**

为了使纠错码容易使用、理解和分析，对它们加一些代数结构会有所帮助。我们将很快证明，用一些能实现数学运算（如加、减、乘、除）的字母表将会很有用。

定义3.9 一个域F就是一些元素的集合，具有满足下列性质的两种运算+（加法）和·（乘法）：

(1) F在“+”和“·”下是封闭的，即若a和b在F中，则$a+b$和$a\cdot b$也在F中。对F中所有的a、b和c，满足下列性质：

(2) 交换律：$a+b=b+a$，$a\cdot b=b\cdot a$

(3) 结合律：$(a+b)+c=a+(b+c)$，$a\cdot(b\cdot c)=(a\cdot b)\cdot c$

(4) 分配律：$a\cdot(b+c)=a\cdot b+a\cdot c$

进一步，F中必须存在元素0和1，满足

(5) $a+0=a$

(6) $a\cdot 1=a$

(7) 对F中任意a，存在加法逆元$(-a)$，使$a+(-a)=0$。

(8) 对F中任意$a\neq 0$，存在乘法逆元(a^{-1})，使$a\cdot a^{-1}=1$。

以上性质对含有有限和无限个元素的域都成立。具有有限个元素（比如说q个）的域称为

伽罗瓦域（Galois Field）并记为$GF(q)$。若只满足前7条性质，则称为**环**（ring）。

例3.6 考虑其中有4个元素{0，1，2，3}的$GF(4)$，加法和乘法表为：

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

.	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

应该注意到这里的$GF(4)$中的加法不是模4加法。

我们定义一个向量空间$GF(q^n)$，它是$GF(q)$中元素的n元数组的集合。线性分组码可以看做$GF(q)$上的n元数组（长为n的向量）的一个集合，满足任意两个码字的和还是一个码字，任意一个码字与域上一个元素的乘积也是一个码字。因此线性分组码是$GF(q^n)$的一个子空间。

设S为一个长度为n且分量在$GF(q)$上的向量集合。S中所有向量的线性组合构成的集合称为S的线性扩张，记为$<S>$。因此线性扩张是由S生成的$GF(q^n)$的一个子空间。给定$GF(q^n)$的任意子集S，可以得到一个由S生成的线性码$C=<S>$，它恰好包含下列码字：

(1) 全零码字。

(2) S中所有的字。

(3) S中两个或两个以上的字的所有线性组合。

例3.7 设$S=\{1100, 0100, 0011\}$。S的所有可能线性组合为 $1100+0100=1000$，$1100+0011=1111$，$0100+0011=0111$，$1100+0100+0011=1011$。

因此，$C=<S>=\{0000, 1100, 0100, 0011, 1000, 1111, 0111, 1011\}$。该码的最小距离为$w(0100)=1$。

例3.8 设$S=\{12, 21\}$是定义在$GF(3)$上的。$GF(3)=\{0, 1, 2\}$上的加法和乘法表为：

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

.	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

12和21的所有可能线性组合为：

$12+21=00$，$12+2(21)=21$，$2(12)+21=12$。

因此$C=<S>=\{00, 12, 21, 00, 21, 12\}=\{00, 12, 21\}$。

3.3 线性分组码的矩阵描述

正如我们前面观察到的，任意线性码C都是$GF(q^n)$的一个子空间。任意基向量组都可以生成这个码空间。因此我们可以定义一个生成矩阵G，它的行构成子空间的基向量。G的行将是线性独立的，因此，行的线性组合可以用于生成C中的码字。生成矩阵将是秩为k的$k\times n$阶矩阵。因为基向量组的选取不是惟一的，所以对一个给定的线性码，它的生成矩阵不是惟一的。

生成矩阵把一个长为k的向量转化（编码）为一个长为n的向量。设输入向量（未编码的

符号）用i表示，则编码后的符号将由

$$c = iG \tag{3-1}$$

给出，其中c称作码字，而i称作信息字。

生成矩阵提供了一种简明而有效地表示一个线性分组码的方法。$k \times n$阶矩阵可以生成q^k个码字。因此，我们只需要一个生成矩阵而不需要含q^k个码字的查询表。这对大码的储存空间是极大的节省。例如二元（46, 24）码的总码字数为$2^{24} = 1\ 777\ 216$，它的码字的查询表将有$n \times 2^k = 771\ 751\ 936$（bit）。另一方面，若我们使用生成矩阵，需要的总储存量将是$n \times k = 46 \times 24 = 1104$（bit）。

例3.9 考虑一个生成矩阵

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$c_1 = [00]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [000], \qquad c_2 = [01]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [010]$$

$$c_3 = [10]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [101], \qquad c_4 = [11]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [111]$$

因此，这个生成矩阵生成的码为$C = \{000, 010, 101, 111\}$，可以观察得到这是一个(3, 2)码。

3.4 等价码

定义3.10 集合$S = \{x_1, x_2, \cdots, x_n\}$上的一个置换是$S$到自身的一个一对一映射。一个置换可以表示如下：

$$\begin{matrix} x_1 & x_2 & \cdots & x_n \\ \downarrow & \downarrow & \cdots & \downarrow \\ f(x_1) & f(x_2) & \cdots & f(x_n) \end{matrix} \tag{3-2}$$

定义3.11 两个q元码称为**等价**的，如果一个可以由另一个通过下面一种或两种运算得到：

(1) 对在固定位置上的符号做置换。

(2) 对码的位置做置换。

假定一个包含M个码字的码用$M \times n$阶矩阵列出，其中行表示码字。运算(1)所对应的是对给定列中的符号进行重贴标签，运算(2)表示矩阵的列的重新排列。

例3.10 考虑分组长度为3的三元码（分量属于{0, 1, 2}的码）

$$C = \begin{Bmatrix} 2 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 1 & 2 \end{Bmatrix}$$

如果我们在第2列使用置换0→2，2→1，1→0，在第3列使用置换1→2，0→1，2→0，可得到

$$C_1 = \begin{Bmatrix} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{Bmatrix}$$

码C_1与一个长度为3的重复码等价。注意原来的码不是线性的，但它与一个线性码等价。

定义3.12 两个线性q元码称为**等价的**，如果一个可以由另一个通过下面一种或两种运算得到：

(1) 用非零常量去乘它的分量。

(2) 对码的位置做置换。

注意定义3.11中我们定义的等价码不一定是线性的。

定理3.2 两个$k\times n$矩阵中若一个可以由另一个通过一系列下述变换得到，则它们生成的$GF(q)$上的(n,k)线性码等价：

(1) 对行置换。

(2) 对行乘以一个非零常量。

(3) 把一行乘以一个常量然后加到另一行上。

(4) 对列置换。

(5) 对任意列乘以一个非零常量。

证明：前三种运算（只是行变换）保留了生成矩阵的行的线性独立性，那些变换只是改变了基。最后两种运算（列变换）把矩阵变成能生成等价码的一个矩阵。

定理3.3 一个生成矩阵可以简化成**系统型**（systematic form，也称为生成矩阵的标准型）$\boldsymbol{G}=[\boldsymbol{I}|\boldsymbol{P}]$，其中$\boldsymbol{I}$是$k\times k$单位矩阵，$\boldsymbol{P}$是一个$k\times(n-k)$阶矩阵。

证明：任意生成矩阵的k个行（大小为$k\times n$）是线性独立的。所以通过基本行变换和列置换可以得到一个阶梯型的等价的生成矩阵，这个矩阵将是$[\boldsymbol{I}|\boldsymbol{P}]$型。

例3.11 考虑$GF(3)$上一个$(4,3)$码的生成矩阵

$$\boldsymbol{G}=\begin{bmatrix}0&1&2&1\\1&0&1&0\\1&2&2&1\end{bmatrix}$$

我们用r_i表示第i行，用c_j表示第j列。将r_3代换为$r_3-r_1-r_2$可得到（注意在$GF(3)$中，$-1=2$且$-2=1$，因为$1+2=0$，见例3.6中的表格）

$$\boldsymbol{G}=\begin{bmatrix}0&1&2&1\\1&0&1&0\\0&1&2&0\end{bmatrix}$$

接下来我们将r_1代换为r_1-r_3，得到

$$\boldsymbol{G}=\begin{bmatrix}0&0&0&1\\1&0&1&0\\0&1&2&0\end{bmatrix}$$

最后，做移位$c_4\to c_1$，$c_1\to c_2$，$c_2\to c_3$及$c_3\to c_4$，可以得到标准型的生成矩阵

$$\boldsymbol{G}=\begin{bmatrix}1&0&0&0\\0&1&0&1\\0&0&1&2\end{bmatrix}=[I\mid P]$$

3.5 奇偶校验矩阵

设计好码的目标之一就是有好的编码和译码方法。到目前为止，我们学习了用一个生成矩阵来有效生成一个线性分组码。用输入向量（未编码的字）乘以生成矩阵就得到码字。可以用类似的构想检测一个码字是否合法吗？答案是肯定的，对一个给定的码，这样的矩阵称为**奇偶校验矩阵**（Parity Check Matrix），记为$\boldsymbol{H}$。对一个奇偶校验矩阵有

$$\boldsymbol{c}\boldsymbol{H}^T = 0 \tag{3-3}$$

其中$\boldsymbol{c}$是个合法的码字。因为$\boldsymbol{c} = \boldsymbol{i}\boldsymbol{G}$，故有$\boldsymbol{i}\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$。要使这个等式对所有合法的信息字都成立，必有

$$\boldsymbol{G}\boldsymbol{H}^T = 0 \tag{3-4}$$

奇偶校验矩阵的大小为$(n-k)\times n$。奇偶校验矩阵提供了一个检测是否发生错误的简单方法。如果收到的字（在接收端）与$\boldsymbol{H}$的转置的乘积是一个非零向量，这说明发生了错误。但当传输的码字中的错误数目超过编码方案所设计的错误数目允许范围时，这种方法可能会失败。我们将很快看到非零乘积$\boldsymbol{c}\boldsymbol{H}^T$不仅帮我们检测错误，而且在某些条件下还可以纠正错误。

假定生成矩阵以系统型$\boldsymbol{G} = [\boldsymbol{I}|\boldsymbol{P}]$给出。矩阵$\boldsymbol{P}$称为**系数矩阵**（Coefficient Matrix）。那么奇偶校验矩阵将定义为

$$\boldsymbol{H} = [-\boldsymbol{P}^T \mid \boldsymbol{I}] \tag{3-5}$$

其中$\boldsymbol{P}^T$表示矩阵$\boldsymbol{P}$的转置。这是因为

$$\boldsymbol{G}\boldsymbol{H}^T = [\boldsymbol{I} \mid \boldsymbol{P}]\begin{bmatrix} -\boldsymbol{P} \\ \boldsymbol{I} \end{bmatrix} = 0 \tag{3-6}$$

因为一个码的生成矩阵的选取不是惟一的，**奇偶校验矩阵也不是惟一的**。给定一个生成矩阵$\boldsymbol{G}$，我们可以确定相应的奇偶校验矩阵，反过来也一样。因此奇偶校验矩阵$\boldsymbol{H}$可以完全确定一个码。

从式（3-3）可得，在对应$\boldsymbol{H}^T$的行向量相加为零向量“**0**”的位置上，$\boldsymbol{c}$的分量必为1。现在我们知道一个码字中“1”的个数与其汉明重量一致。因此，*一个线性分组码的最小距离d^*就是$\boldsymbol{H}^T$中满足和为零向量的行向量的最小个数。对于非二元的情况，由码字确定重量时需考虑$\boldsymbol{H}$的列向量。*

例3.12 一个（7,4）线性分组码的生成矩阵为

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

矩阵$\boldsymbol{P}$为$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$，它的转置$\boldsymbol{P}^T$为$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$。观察到在二元情况下，$-1=1$的事实，我

们可以将奇偶校验矩阵写为

$$\boldsymbol{H}=[-\boldsymbol{P}^T \mid \boldsymbol{I}]=\begin{bmatrix}1&1&0&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&0&0&0&1\end{bmatrix}$$

注意奇偶校验矩阵$\boldsymbol{H}$的第1、5和7列的和为零向量，因此，对这个码有$d^*=3$。

定理3.4 码$\boldsymbol{C}$有一个汉明重量等于或小于w的非零码字的充分必要条件为$\boldsymbol{H}$中必存在w个线性相关的列。

证明：考虑一个码字$c\in\boldsymbol{C}$，设c的重量为w，这说明c中有w个非零分量和$(n-w)$个零分量。如果我们把那w个零分量扔掉，则从关系式$c\boldsymbol{H}^T=0$，可以肯定$\boldsymbol{H}$中有w个列线性相关。

相反，如果$\boldsymbol{H}$有w个线性相关的列，则有一个至多有w个列的线性组合为零，这w个非零系数就定义了一个重量小于或等于w的码字满足$c\boldsymbol{H}^T=0$。

定义3.13 一个(n, k)**系统码**（systematic code）是一个分组长度为n的码字的前k个符号为信息符号本身（即未编码的向量），而后$(n-k)$个符号为**奇偶校验符号**（parity symbol）的码。

例3.13 下面是$GF(3)$上的一个$(5,2)$系统码

序　号	信息符号（$k=2$）	码字（$n=5$）
1.	00	00 000
2.	01	01 121
3.	02	02 220
4.	10	10 012
5.	11	11 221
6.	12	12 210
7.	20	20 020
8.	21	21 100
9.	22	22 212

注意总的码字数为$3^k=3^2=9$。每一个码字开始于信息符号，后面跟了三个奇偶校验符号。在上表中信息字01的奇偶校验符号为121。一个系统型（标准型）的生成矩阵将生成一个系统码。

定理3.5 一个(n, k)线性码的最小距离（最小重量）有下述界

$$d^* \leqslant n-k+1 \tag{3-7}$$

这称为Singleton界（Singleton Bound）。

证明：我们可以把所有线性分组码化简为它们等价的系统型。一个系统码可以有一个信息符号和$(n-k)$个奇偶校验符号。最多全部奇偶校验符号都为非零的，致使码字的总重量为$(n-k+1)$。因此没有任何码字的重量可以超过$(n-k+1)$，这也导出了下述最大距离码的定义。

定义3.14 一个**最大距离码**（Maximum Distance Code）满足$d^*=n-k+1$。

熟悉了线性码最小距离的概念后，我们将研究这个最小距离是怎么跟这个码所能检测的并可能纠正的错误的个数联系起来。所以我们到接收端看一下线性分组码的译码方法。

3.6 线性分组码的译码

信道编码的基本目标就是检测并纠正消息在通过有噪信道传输时产生的错误。信道中的噪声随机地将所传输的码字的符号转变为其他符号。假如噪声只改变传输码字的一个符号，则该有错码字与原来传输的码字的汉明距离为1。如果噪声改变了t个符号（即码字的t个符号都发生了错误），则收到的字与原来传输的码字的汉明距离为t。给定一个码，它能检测到多少个错误，又能纠正多少个错误呢？让我们先看一下检测方面的问题。

一个码字只要不是转变成另一个合法的码字，就能检测到错误。如果码字之间的最小距离为d^*，要使一个码字转变成另一个码字，错误模式的重量必须至少为d^*。因此一个(n, k, d^*)码将至少可以检测所有重量小于或等于(d^*-1)的非零错误模式。另外，至少有一个重量为d^*的错误模式不能被检测到，这与两个距离最近的码字相对应。可能有些重量大于或等于d^*的错误模式也能被检测到，但并不是所有重量为d^*的错误模式都能被检测到。

例3.14　码$C_1=\{000, 111\}$的最小距离为3，因此重量为2或1的错误模式可以被检测到。这说明任意属于集合{011, 101, 110, 001, 010, 100}的错误模式都能检测到。

下面考虑码$C_2=\{001, 110, 101\}$，它的$d^*=1$。我们没法说这个码能检测多少错误，因为$d^*-1=0$。但是重量为1的错误模式010可以被这个码检测到，但它不能检测所有重量为1的错误模式，例如错误向量100就不能被检测到。

接下来让我们看一下纠错问题。它的目标就是在接收的字的基础上对原来传输的码字作最佳猜测。怎样才算一个明智的译码策略呢？因为传输的只可能是一个合法码字，逻辑上应该判断那个与接收到的字最近的（以汉明距离衡量）合法码字是实际传输的。换句话说，就是假定与接收到的字最相像的那个码字是被传送的。这种策略称为**最近邻居译码**（Nearest Neighbour Decoding），因为我们选取了在汉明距离度量下最接近接收到的字的码字。

也可能有多个码字与接收到的字有相同的汉明距离。在这种情况下接收方可做下面的一件事：

(1) 任意选择相同距离的邻居中的一个。

(2) 要求发送端重新传送。

为了确保收到的字（最多有t个错误）离原始码字最近，而离所有其他码字都更远些，我们必须对码的最小距离提出下面的条件：

$$d^* \geqslant 2t+1 \tag{3-8}$$

能纠正不多于t个错误的条件可以由图3-2表示。考虑所有q元n维数组空间。每个长为n的q元向量可以表示为这个空间的一个点。于是每个码字可以用这个空间的一个点描述，并且所有汉明距离小于或等于t的字都位于以该码字为中心、半径为t的球中。如果码的最小距离为d^*且满足条件$d^* \geqslant 2t+1$，那么这些球不会重叠。任意一个接收到的在某个球内的

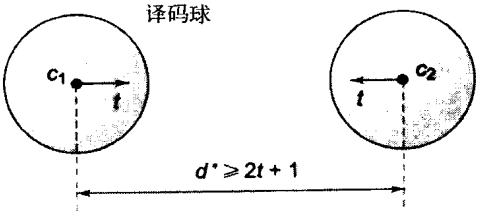


图3-2　译码球

向量（仅仅是一个点）将离它的中心（表示一个码字）比离其他码字的距离更近。我们称与码字相关的球为**译码球**（Decoding Sphere）。因此可以毫不含糊地用“最近邻居”方法对接收到的向量译码。

图3-2说明在以c_1为中心、半径为t的球内的字将被译码为c_1。对于明确译码，应满足$d^* \geqslant 2t+1$。

条件$d^* \geqslant 2t+1$考虑到了最坏的情况。但也可能不满足上述条件，而仍然可以纠正t个错误，如下例所示。

例3.15 考虑码$\boldsymbol{C}=\{00000, 01010, 10101, 11111\}$。它的最小距离为$d^*=2$。假设被传送的码字是11111，而接收到的字是11110，即$t=1$（在第5个分量上发生了1个错误）。现在有：

$d(11110, 00000)=4$，$d(11110, 01010)=2$

$d(11110, 10101)=3$，$d(11110, 11111)=1$

根据最近邻居译码法，我们可以确定11111是所传输的码字。尽管一个纠错（$t=1$）在这种情况下做到了，却有$d^*<2t+1=3$。因此即使在$d^*<2t+1$时也可能纠错。但是很多情况下这个码一个纠错都不可能实现。例如若传送的码字是00000而接收到的码字是01000，

$d(01000, 00000)=1$，$d(01000, 01010)=1$

$d(01000, 10101)=4$，$d(01000, 11111)=4$

在这种情况下，如果没有一个明显的判决，就需要掷硬币来选择了。

定义3.15 一个**不完全译码器**（Incomplete Decoder）接收到的码字只在能找到一个离它最近的码字的情况下才能译码。在不确切的情况下，译码器声明收到的字无法辨认。发送方就被要求重新传送。一个**完全译码器**（Complete Decoder）对收到的任何字都译码，即它试图对任何收到的字都找一个对应的码字，即使有时它需要猜。

例3.15就是一个完全译码器。当有一种好的猜测比无任何猜测要好的情况下，就用到完全译码器。现实生活中的多数译码器都是不完全译码器，它们通常发送一个消息给发送端要求重新传送。

例3.16 **国际标准书号**（ISBN）是一个在GF(11)上的分组长度n=10的线性分组码。所使用的符号是0，1，…，9，X。为避免使用‘10’（这可能造成‘10’和两个连续符号‘1’和‘0’的混淆），我们使用符号‘X’。ISBN满足下面的限制条件：

$$\sum_{i=0}^{9}(10-i)c_i = 0 \text{计算}(\text{mod } 11) \tag{3-9}$$

比如，考虑ISBN为0-07-048297-7，我们对此ISBN做下面的验证：

$$10\times 0+9\times 0+8\times 7+7\times 0+6\times 4+5\times 8+4\times 2+3\times 9+2\times 7+1\times 7=176=0 \text{ (mod 11)}$$

因此，0-07-048297-7是一个合法的ISBN。

由于ISBN是一个线性分组码，因此全零码字也是一个合法的码字。同样1000000001也是合法的ISBN。所以此码的最小重量是d^*=2。理论上，它不能纠错并且只能检测只有一位的错误。但是，假设ISBN的一位被涂脏，我们有0-07-048e97-7。我们可以通过解方程

$$10\times 0+9\times 0+8\times 7+7\times 0+6\times 4+5\times 8+4\times e+3\times 9+2\times 7+1\times 7=176=0(\text{mod } 11)$$

计算e的值，从而恢复被涂脏的数字位，即e=2。由于我们知道错误数字的位置，所以可以恢复正确的ISBN。

定义3.16　当一个收到的符号是含糊的，或在接收过程中检测到干扰时，收信人就声明发生了一个**删除**（erasure，即一个收到的符号被删除了）。

例3.17　考虑一个二元**脉冲振幅调制**（PAM）**方案**，其中5V表示1，0V表示0。噪声的范围为1V，这表明在接收端：

如果收到的电压在4V～5V⇒发送的比特为1，

如果收到的电压在0V～1V⇒发送的比特为0，

如果收到的电压在1V～4V⇒发生了删除。

因此如果收信人在比特区间收到2.9V，将声明发生了删除。

一个信道更易于发生错误或删除。如果在这样一个信道中发生了t个错误和r个删除，纠错方案应该能弥补删除和纠错。如果出现r个删除，该码在最坏条件下的最小距离为d^*-r，这是因为删除了的符号只能被抛弃，而如果它们在计算最小距离时用得到，距离将会减小。一个简单例子将说明这一点。考虑这样的**重复码**：

$$0\rightarrow 00000$$

$$1\rightarrow 11111$$

这里$d^*=5$。如果$r=2$，即有两个比特被删除了（假定是前两个比特），我们将得到

$$0\rightarrow ??000$$

$$1\rightarrow ??111$$

现在，有效的最小距离为$d_1^*=d^*-r=3$。

因此，对一个有t个错误和r个删除的信道，有$d^*-r\geqslant 2t+1$，或者

$$d^*\geqslant 2t+r+1 \tag{3-10}$$

对一个没有错误（$t=0$），只有r个删除的信道，有

$$d^*\geqslant r+1 \tag{3-11}$$

下面我们稍微正式地介绍一下译码过程。我们能否构造一些数学工具来简化最近邻居译码？假设通过有噪信道传输的码字是$c=c_1, c_2, \cdots, c_n$。信道中的噪声改变了码字的某些或全部符号。令收到的向量为$v=v_1, v_2, \cdots, v_n$。定义**错误向量**（error vector）为

$$e=v-c=v_1, v_2, \cdots, v_n-c_1, c_2, \cdots, c_n=e_1, e_2, \cdots, e_n \tag{3-12}$$

译码器需要从收到的向量v来确定传输的是哪个码字，也可以说，它必须确定错误向量e。

定义3.17　设$\boldsymbol{C}$是$GF(q)$上的一个(n,k)码，$\boldsymbol{a}$是长为n的任意向量。则集合

$$\boldsymbol{a}+\boldsymbol{C}=\{\boldsymbol{a}+\boldsymbol{x}|\boldsymbol{x}\in\boldsymbol{C}\} \tag{3-13}$$

称为$\boldsymbol{C}$的一个**陪集**（Coset）。当$(\boldsymbol{a}-\boldsymbol{b})\in\boldsymbol{C}$时，我们称$\boldsymbol{a}$和$\boldsymbol{b}$属于同一个陪集。

定理3.6　假设$\boldsymbol{C}$是$GF(q)$上的一个(n,k)码，则

(1) 任意长为n的向量$\boldsymbol{b}$都属于$\boldsymbol{C}$的某个陪集。

(2) 每个陪集恰好包含q^k个向量。

(3) 两个陪集或者不相交或者完全重合（不可能部分相交）。

(4) 若$\boldsymbol{a}+\boldsymbol{C}$是$\boldsymbol{C}$的一个陪集，而且$\boldsymbol{b}\in\boldsymbol{a}+\boldsymbol{C}$，则$\boldsymbol{b}+\boldsymbol{C}=\boldsymbol{a}+\boldsymbol{C}$。

证明：

(1) $b=b+0\in b+C$。

(2) 注意到$x\to a+x$是$C\to a+C$上的一一映射，因此$a+C$含有的元素个数与C相同，即等于q^k。

(3) 假设$a+C$与$b+C$相交，即它们至少有一个公共向量。设$v\in(a+C)\cap(b+C)$。则必存在$x,y\in C$，使

$$v=a+x=b+y$$

或

$$b=a+x-y=a+z$$

其中$z\in C$（因为两个码字的差也是一个码字）。故有

$$b+C=a+C+z\text{或}\ (b+C)\subseteq(a+C)$$

类似地可以证明$(a+C)\subset(b+C)$。从这两条我们得到$(b+C)=(a+C)$。

(4) 因为$b\in(a+C)$，这表明对某个$x\in C$有$b=a+x$。对任意$b+y\in b+C$，有

$$b+y=(a+x)+y=a+(x+y)\in a+C$$

因此$b+C\in a+C$。另一方面，对任意$a+z\in(a+C)$，有

$$a+z=(b-x)+z=b+(z-x)\in b+C$$

因此$a+C\in b+C$，从而$b+C=a+C$。

定义3.18 一个陪集中具有最小重量的向量称为**陪集首**（Coset Leader）。如果有多于一个向量具有最小重量，则从中随机选择一个定为陪集首。

例3.18 设C为一个二元（3,2）码，其生成矩阵如下：

$$C=\begin{bmatrix}1&0&1\\0&1&0\end{bmatrix}$$

例如$C=\{000,\ 010,\ 101,\ 111\}$。$C$的陪集为

$$000+C=000,\ 010,\ 101,\ 111$$
$$001+C=001,\ 011,\ 100,\ 110$$

注意所有的8个向量都被这两个陪集覆盖了。正如我们已经看到的（在上面定理中），如果$a+C$是C的一个陪集而且$b\in a+C$，则我们有$b+C=a+C$。

因此，上述已经列出了所有陪集。为了演示，我们在下面写出全部可能情况

$$010+C=010,\ 000,\ 111,\ 101$$
$$011+C=011,\ 001,\ 110,\ 100$$
$$100+C=100,\ 110,\ 001,\ 011$$
$$101+C=101,\ 111,\ 000,\ 010$$
$$110+C=110,\ 100,\ 011,\ 001$$
$$111+C=111,\ 101,\ 010,\ 000$$

可以看到所有陪集已经覆盖了。

因为两个陪集或不相交或重合（由定理3.6），$GF(q^n)$上的所有向量可以写为

$$GF(q^n)=C\cup(a_1+C)\cup(a_2+C)\cup\cdots\cup(a_t+C)$$

其中$t=q^{n-k}-1$。

定义3.19　一个 (n, k) 码$\boldsymbol{C}$的**标准阵列**（Standard Array）是一个$GF(q^n)$上全部向量的 $q^{n-k} \times q^k$阵列，它的第一行由码$\boldsymbol{C}$构成（0在最左边），其他行是陪集$\boldsymbol{a}_i + \boldsymbol{C}$，都以相应次序排列，陪集首放在最左边。

构造标准阵列的步骤为：

(1) 在第一行从全零码字开始写下所有的合法码字。

(2) 选择不在第一行的一个最小重量的向量$\boldsymbol{a}_1$，在第二行中列出陪集$\boldsymbol{a}_1 + \boldsymbol{C}$的元素使$\boldsymbol{a}_1 + \boldsymbol{x}$在$\boldsymbol{x} \in \boldsymbol{C}$的下边。

(3) 接下来再选一个最小重量的向量$\boldsymbol{a}_2$（不在前两行中），然后在第三行中列出陪集$\boldsymbol{a}_2 + \boldsymbol{C}$的元素使$\boldsymbol{a}_2 + \boldsymbol{x}$在$\boldsymbol{x} \in \boldsymbol{C}$的下边。

(4) 继续此过程直到列出所有陪集并且$GF(q^n)$中所有的元素都恰好出现过一次。

例3.19　考虑码$C = \{0000, 1011, 0101, 1110\}$。相应的标准阵列为

码字→ 0000 1011 0101 1110
1000 0011 1101 0110
0100 1111 0001 1010
0010 1001 0111 1100
↑
陪集首

注意每项都是它的陪集首和码字的和。

我们现在介绍一下利用标准阵列译码的概念（从接收到的码字中得到信息符号）。因为标准阵列包括了$GF(q)^n$中所有的字，接收到的字总是标准阵列中的某个元素。如果接收到的字是个合法码字，那么可以下结论说没有错误发生（这个结论可能是错的，就是当噪声把一个合法码字改变成另一个合法码字时，但它的错误概率很低）。在接收到的字$\boldsymbol{v}$不属于合法码字集合时，我们推测错误。译码器则声明陪集首就是错误向量$\boldsymbol{e}$，然后译码为$\boldsymbol{v} - \boldsymbol{e}$，这就是在$\boldsymbol{v}$同一列最上边的那个码字。因此，我们把接收到的字译为包含该字的列的最上边的那个码字。

例3.20　假设使用的码为前例中的码$\boldsymbol{C} = \{0000, 1011, 0101, 1110\}$，接收到的字为$\boldsymbol{v} = 1101$。因为它不是一个合法的码字，我们推测错误。接着我们估计四个可能的码字中的哪一个是实际被传送的。如果我们利用前例中的标准阵列，可以发现1101在第三列，该列的最上边为0101，于是估计的码字为0101。进一步观察到：

$$d(1101, 0000) = 3,\ d(1101, 1011) = 2$$
$$d(1101, 0101) = 1,\ d(1101, 1110) = 2$$

错误向量为$\boldsymbol{e} = 1000$，即陪集首。

具有较大分组长度的码是可取的（尽管不总是如此，见3.16节），因为大的码在码率方面更接近Shannon限。当我们取越来越大的码时（大的k值和n值），标准阵列方法越来越不实际，因为大小为 $(q^{n-k} \times q^k)$ 的标准阵列会变得大到不可操作。编码理论的基本目标之一就是要设计有效的译码方法。如果我们要建造能实时工作的译码器，译码方案应该在所需内存和计算量方面都可以实现。能不能缩减标准阵列呢？答案在我们下面将要讨论的伴随式译码的概念中。

3.7 伴随式译码

如果我们只存储标准阵列的第一列，并在需要时计算其他列，就可以简化标准阵列。为了做到这点，我们引进错误模式的**伴随式**（Syndrome）的概念。

定义3.20 假定$\boldsymbol{H}$是一个(n,k)码的奇偶校验矩阵。对任意向量$v\in GF(q^n)$，向量

$$s=vH^T \tag{3-14}$$

称为v的**伴随式**。v的伴随式有时候明确地写为$s(v)$。因为它给出错误症状以帮助我们诊断错误，所以称为伴随式（又译为综合症）。

定理3.7 两个向量x和y在C的同一个陪集中的充分必要条件是它们的伴随式相同。

证明：向量x和y属于同一个陪集

$$\begin{aligned}
&\Leftrightarrow x+C=y+C\\
&\Leftrightarrow x-y\in C\\
&\Leftrightarrow (x-y)H^T=0\\
&\Leftrightarrow xH^T=yH^T\\
&\Leftrightarrow s(x)=s(y)
\end{aligned}$$

因此在陪集和伴随式之间一一对应。

我们可以通过只列出伴随式和对应的陪集首来缩减标准阵列的大小。

例3.21 我们现在把例3.19中的标准阵列添加一列伴随式进行扩张。原码为$\boldsymbol{C}=\{0000, 1011, 0101, 1110\}$。相应的标准阵列为

					伴随式
码字→	0000	1011	0101	1111	00
	1000	0011	1101	0110	11
	0100	1111	0001	1010	01
	0010	1001	0111	1100	10
	↑				
	陪集首				

伴随式译码的步骤如下：

(1) 对接收到的字v确定伴随式（$s=vH^T$）。

(2) 在“伴随式列”中给伴随式定位。

(3) 判断相应的陪集首，这就是错误向量e。

(4) 从接收到的字中减掉该错误向量就得到码字$y=v-e$。

用伴随式译码方式建立了一种有效的译码方法后，让我们看一下编码实际上提供了多少优点。

3.8 译码后的错误概率（纠错概率）

定义3.21 任意译码方案的**错误概率**（或**字出错率**）P_{err}是译码器输出一个错误码字的概率。它也称为**剩余错误率**（Residual Error Rate）。

假设有M个码字（长度为n），它们被使用的概率相同。假设译码用标准阵列。记α_i为重量

为i的陪集首的个数。我们假定信道为BSC，符号错误概率为p。如果错误向量e不是一个陪集首，则译码出错。因此正确译码概率为

$$P_{cor}=\sum_{i=0}^{n}\alpha_i\, p^i\,(1-p)^{n-i} \tag{3-15}$$

故错误概率为

$$P_{err}=1-\sum_{i=0}^{n}\alpha_i\, p^i\,(1-p)^{n-i} \tag{3-16}$$

让我们考虑在分组长度为n比特的一个分组中存在m个错误的情况。对于一个具有最小距离d^*的线性分组码，最多有$\left\lfloor\frac{1}{2}(d^*-1)\right\rfloor$的错误都是可以被纠正的。在一个$n$比特的分组中，出现$m$个错误的概率是

$$P_{m,n}=\binom{n}{m}p^m(1-p)^{n-m} \tag{3-17}$$

其中，p是出现1比特错误的概率。如果多于t个错误发生时，码字就将会出错。因此，码字出错的概率的上界可以表示为

$$P_M\leqslant\sum_{m=t+1}^{n}P_{m,n} \tag{3-18}$$

上式是等号的时候，我们称为理想码，后面将简单介绍。我们注意到，P_M不可能小于把某个码字错误译成具有距离d^*的另外一个码字的概率。所以，

$$P_M\geqslant\sum_{m=\lfloor d^*/2\rfloor+1}^{d^*}\binom{d^*}{m}p^m(1-p)^{d^*-m} \tag{3-19}$$

如果我们考虑另外所有的$(M-1)$个码字，则可以得到上界。所有这些码字的距离至少是d^*。因此，联合界可以表示为

$$P_M\leqslant(M-1)\sum_{m=\lfloor d^*/2\rfloor+1}^{d^*}\binom{d^*}{m}p^m(1-p)^{d^*-m} \tag{3-20}$$

例3.22 考虑例3.19中的标准阵列，陪集首为0000、1000、0100、0010。因此$\alpha_0=1$（只有一个重量等于零的陪集首），$\alpha_1=3$（剩余的三个重量都为1）而且所有其余的$\alpha_i=0$。因此

$$P_{err}=1-[(1-p)^4+3p(1-p)^3]$$

回顾该码有4个码字，且可用来每次发送2比特。如果我们不采用编码，接收到的2比特有错的概率为

$$P_{err}=1-P_{cor}=1-(1-p)^2$$

注意到$p=0.01$，则**字出错概率**（在编码条件下）为$P_{err}=0.0130$，而对无编码的情况$P_{err}=0.0199$。因此编码几乎把字出错概率减小了一半。图3-3对有编码和无编码情况下的P_{err}进行了比较。可以看出只有在$p<0.5$的情况下编码才胜过无编码。注意编码带来的改进是以信息传输速率为代价的，由于我们对任意两个信息比特都要发送2个奇偶校验比特，因此信息传输速率减少了一半。

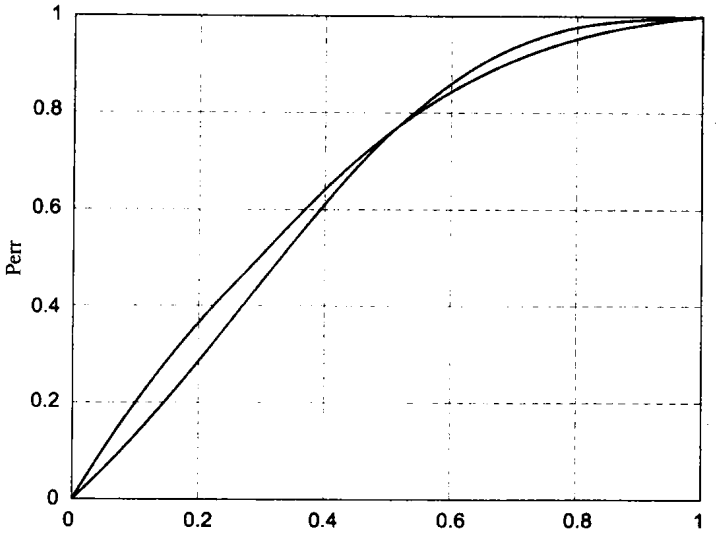


图3-3 对2比特消息有编码和无编码下的P_{err}的比较

例3.23 这个例子帮助我们更好地理解编码的功能。考虑一个符号出错概率为$p=10^{-7}$的BSC。假设10比特长的字*未经编码*就被传输了。设发送端的比特速率为10^7bit/s，这表明每秒可以发送10^6个字。一个字不能被正确接收到的概率为

$$\binom{10}{1}(1-p)^9 p+\binom{10}{2}(1-p)^8 p^2+\binom{10}{3}(1-p)^7 p^3+\cdots\approx\binom{10}{1}(1-p)^9 p=10^{-6}\quad(字/秒)$$

因此一秒将有$10^{-6}\times 10^6=1$个字出错！这意味着每秒都有一个错误而且它*未被检测到*。

下面我们在未编码的字中加进一个奇偶校验比特使它们变为11比特长。该奇偶校验使所有码字为偶校验，这样可以确保单个错误可以被检测到。仅当两个或更多比特有错时编码后的字也将有错，即至少有2比特有错。少于2比特有错的情况可以以概率1计算出来。因此字错误概率将为

$$1-(1-p)^{11}-\binom{11}{1}(1-p)^{10}p\approx 1-(1-11p)-11(1-10p)p=110p^2=11\times 10^{-13}$$

新的码率为$10^7/11$字/秒，因为每个字有11比特而比特速率与以前相同。因此一秒将有$(10^7/11)\times(11\times 10^{-13})=10^{-6}$个字有错。这表明编码后每$10^6$秒$=11.5$天将有一个字被错误地接收而*未被检测*！

仅仅把字长从10比特（未编码）增加到11比特（编码），我们能使字出错率惊人地减小。对第二种情况，每次检测到错误字时要求发送端重新传输该字。

这种要求重新传送的策略叫做自动重复请求（Automatic Repeat Request, ARQ）。

如果我们有线性码(n, k, d^*)，我们使用它要么为了纠错，要么为了错误检测（一般与ARQ结合使用）。假设我们有一个单个纠错码($d^*=3$)并在某个接收到的码字中有两个错误。此种情况下的伴随式非零，说明有1位或者2位错误。但是它不能指出是1位错误还是2位错误。在此时，需要进行两个互斥操作。一种策略是使用此码作为错误检测工具并请求重传。这种策略中我们不尝试去纠错。另外一个可能是假设只有1位错误并尝试纠正它。当有2位错误发生时，使用上述策略去尝试纠正1位错误时可能会引入第三个错误。如果我们计算此经过错误纠正的码字的伴随式，伴随式可能是零。这是因为编码最多只能检测出2位错误。

考虑由于信道编码而节省的能量值是非常重要的。在图3-4中描绘了错误概率(P_e)关于一个普通通信系统E_b/N_0的曲线。从图中可以看出，对于相同的P_e值，使用编码时的E_b/N_0比不使用编码时的E_b/N_0要少。因此编码方案可以提供**编码增益**（Coding Gain）。编码增益使用dB来度量。通过观察可以知道，如果P_e没有足够小，编码增益可能会是负数。这就表明，如果信道一开始不是太差，那么信道编码能够提供某种改善。同样，编码增益随着P_e的减小而增加。其极限值，即当$P_e \to 0$时，称为**渐近编码增益**（Asymptotic Coding Gain）。

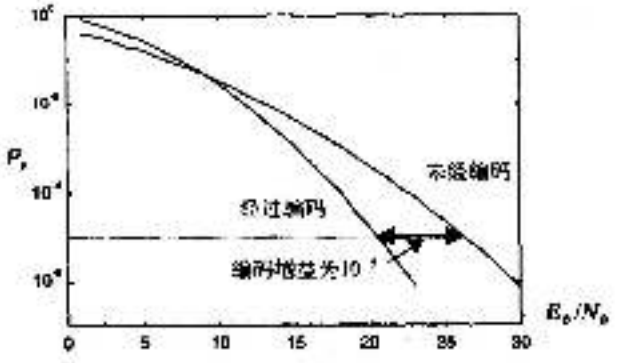


图3-4　编码增益图示

3.9　完备码

定义3.22　对$GF(q^n)$中的任意向量u和任意整数$r \geqslant 0$，以u为中心、以r为半径的球，记为$S(u, r)$，是集合$\{v \in GF(q^n) | d(u, v) \leqslant r\}$。

这个定义可以用图3-5来解释。考虑码C，其最小距离为$d^*(C) \geqslant 2t+1$。则C的那些以码字$\{c_1, c_2, \cdots, c_M\}$为中心、以$t$为半径的球将不相交。现在考虑译码问题。任何接收到的向量可以用这个空间的一个点来表示。若这个点在一个球内，则根据最近邻居译码，它将被译码为所在球的中心点。如果发生的错误不多于t个，则接收到的字将肯定在所传输的码字所在的球内，从而可以正确译码。但是，如果有多于t个错误发生，它将脱离这个球，从而导致译码错误。

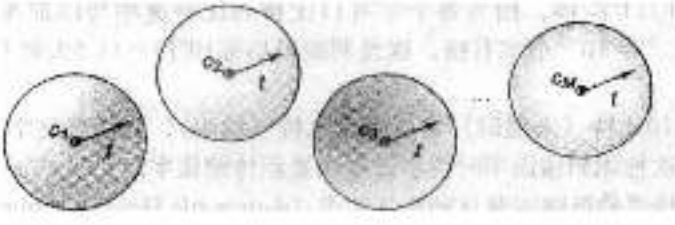


图3-5　$GF(q^n)$中球的概念

当$d^*(C) \geqslant 2t+1$时，C的码字为这些不相交的球的中心。

定理3.8　半径为r（$0 \leqslant r \leqslant n$）的球包含向量的个数恰好为

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r \tag{3-21}$$

个向量。

证明：考虑$GF(q)^n$中的向量$\boldsymbol{u}$和另一个与$\boldsymbol{u}$的距离为m的向量$\boldsymbol{v}$。这说明向量$\boldsymbol{u}$和$\boldsymbol{v}$在m个位置不同。从n个位置中选取m个位置的总选取方法共有$\binom{n}{m}$种。现在这m个位置中的每一个都可以被$(q-1)$个可能的符号取代，这是因为总的符号数为q，其中一个已经在$\boldsymbol{u}$的那个特定位置上了。因此与$\boldsymbol{u}$的距离恰为m的向量个数为

$$\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{r}(q-1)^r$$

个向量。

例3.24 考虑分组长度$n=4$的二元码（即$q=2$）。与任意码字的距离小于或等于2的向量个数为

$$\binom{4}{0}+\binom{4}{1}(1)+\binom{4}{2}(1)^2=1+4+6=11$$

不失一般性，我们选取固定向量$\boldsymbol{u}=0000$。距离小于或等于2的向量为：

距离为2的向量：0011，1001，1010，1100，0110，0101

距离为1的向量：0001，0010，0100，1000

距离为0的向量：0000

因此，总共有11个这样的向量。

定理3.9 一个有M个码字，最小距离为$(2t+1)$的q元(n,k)码满足

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}\leqslant q^n \tag{3-22}$$

证明：设C是一个q元(n,k)码。考虑以M个码字为中心、以t为半径的球。每个半径为t的球有

$$\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t$$

个向量（定理3.8）。因为这样的球都不相交，这M个不相交的球中包含的总向量个数为$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}$，它以$GF(q^n)$中长为$n$的总向量数$q^n$为上界。这个界称为**汉明界**（Hamming Bound）或**填球界**（Sphere Packing Bound），而且它对非线性码也适用。对于二元码，汉明界变为

$$M\left\{\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{t}\right\}\leqslant 2^n \tag{3-23}$$

这里需要注意的是，满足汉明界的整数n、M和t的存在性不一定保证有这样的二元码。例如集合$n=5$、$M=5$和$t=1$满足汉明界，但不存在这样的二元码。

观察到$M=q^k$的情况，汉明界也可以写为

$$\log_q\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}\leqslant n-k \tag{3-24}$$

定义3.23 一个能达到汉明界的码称为**完备码**，即满足

$$M\left\{\binom{n}{1}+\binom{n}{1}(q-1)+\binom{n}{2}(q-2)^2+\cdots+\binom{n}{t}(q-1)^t\right\}=q^n \tag{3-25}$$

对一个完备码，以码字为中心的等半径不相交的球完全填满空间。因此一个纠t个错误的完备码最有效地利用了整个空间。

例3.25 考虑分组长度为n的二元重复码

$$C=\begin{cases}00\cdots0\\11\cdots1\end{cases}$$

其中n为奇数。在这种情况下，$M=2$且$t=(n-1)/2$。将这些值代入汉明界不等式左边，可以得到

$$\text{左式}=\left\{\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{(n-1)/2}\right\}=2.2^{n-1}=2^n=\text{右式}$$

因此，重复码是个完备码。它实际称为**平凡**（trivial）**完备码**。在下一章中我们将看到一些有意义的完备码。

寻找完备码的方法之一就是得到一组汉明界方程中参数n、q、M和t的整数解。由计算机穷举搜索得到的一些结果如下：

序　号	n	q	M	t
1	23	2	2^{12}	3
2	90	2	2^{78}	2
3	11	3	3^6	2

3.10 汉明码

有二元和非二元汉明码。这里我们将把讨论限定在二元码上。二元汉明码具有性质

$$(n,\ k)=(2^m-1,\ \ 2^m-1-m) \tag{3-26}$$

其中m是任意正整数。例如当$m=3$时，我们有（7，4）汉明码。汉明码的奇偶校验矩阵$\boldsymbol{H}$是个很有趣的矩阵。回顾可以知道（n，k）码的奇偶校验矩阵有$n-k$个行和n个列。对二元（n，k）汉明码，$n=2^m-1$个列由长为$n-k=m$的除全零向量之外的所有可能的二元向量构成。

例3.26 二元（7，4）汉明码的生成矩阵为

$$\boldsymbol{G}=\begin{bmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\0&0&1&1&0&1&0\\0&0&0&1&1&0&1\end{bmatrix}$$

相应的奇偶校验矩阵为

$$\boldsymbol{H}=\begin{bmatrix}1&0&1&1&1&0&0\\0&1&0&1&1&1&0\\0&0&1&0&1&1&1\end{bmatrix}$$

观察到该奇偶校验矩阵的列由（100）、（010）、（101）、（110）、（111）、（011）和（001）构成，这7个是所有长为3的非零二元向量。很容易可以得到一个系统汉明码。该奇偶校验矩阵$\boldsymbol{H}$可以被安排成如下的系统型：

$$\boldsymbol{H}=\begin{bmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&1&0&0&1\end{bmatrix}=[-\boldsymbol{P}^T|\boldsymbol{I}]$$

于是该二元汉明码的生成矩阵的系统型为

$$\boldsymbol{G}=[\boldsymbol{I}|\boldsymbol{P}]=\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{bmatrix}$$

从上例中我们观察到$\boldsymbol{H}$的任何两列都线性独立（否则它们会完全相同）。但当$m>1$时，有可能找到$\boldsymbol{H}$的三个列使它们的和为零。因此一个(n, k)汉明码的最小距离d^*等于3，这表明它是一个单一错误纠错码。汉明码是完备码。

通过在添加一个整体奇偶校验比特，一个(n, k)汉明码可以修改为$d^*=4$的$(n+1, k)$码。另一方面，通过删掉生成矩阵$\boldsymbol{G}$中的l行，或等价地删掉它的奇偶校验矩阵$\boldsymbol{H}$中的l列，一个(n, k)汉明码可以被缩短为$(n-l, k)$码。我们现在可以给出汉明码的更正规的定义。

定义3.24 设$n=(q^k-1)/(q-1)$。$GF(q)$上的(n, k)汉明码是这样一个码，它的奇偶校验矩阵的列两两线性独立（在$GF(q)$上），即那些列是两两线性独立向量的最大集合。

3.11 低密度奇偶校验（LDPC）码

定义3.25 一个Gallager码(r, s)是一个线性码，其校验矩阵$\boldsymbol{H}$满足如下条件：每一列有r个1，每一行有s个1。

定义3.26 当一个Gallager码的r和s很小时，它是**低密度奇偶校验**（LDPC）码。因此，一个LDPC码有一个稀疏的奇偶校验矩阵，它的每行和每列只有很少的1。一般情况下，$r\leqslant\log_2 n$，其中n是分组长度。这个码可以写成(n, r, s)LDPC 码。

例3.27 考虑下面的奇偶校验矩阵：

$$\boldsymbol{H}=\begin{bmatrix}1&0&0&0&1&1&1&0&0&0\\0&1&0&0&1&0&0&1&1&0\\0&0&1&0&0&1&0&1&0&1\\0&0&0&1&0&0&1&0&1&1\\1&1&1&1&0&0&0&0&0&0\end{bmatrix}\tag{3-27}$$

此矩阵每一列有$(r=2)$个1，每一行有$(s=4)$个1。在这个5×10的矩阵中，不可能再有更

多的线性无关的列，因为在5个位置中任意选择两个位置设置1的所有可能（也就是$\binom{5}{2}$）均已经穷尽。由于奇偶校验矩阵的维数是$(n-k)\times n$，我们有$n=10$，$k=5$。我们还注意到矩阵的最后三列加起来是零向量。所以此码最小的距离是3。因此，这是一个(10, 2, 4) LDPC码。

定义3.27　一个有着固定的r和s的LDPC码称为**正则的**LDPC码。如果列中1的个数和行中1的个数分别近似r和s，则称为**非正则的**LDPC码。

LDPC码可以通过随机构造、代数构造，或者两种构造的组合进行构造。以下是LDPC码的一种简单随机构造的算法：

第1步：设置$i=1$。

第2步：生成一个长度为nr/s和汉明重量为r的随机二元向量。把它作为$\boldsymbol{H}$的第i列。

第3步：如果此时$\boldsymbol{H}$每行的重量均小于等于s，并且任意两列的标量积均小于等于1，则设置$i=i+1$。否则，转到第2步。

第4步：如果$i=n$，算法停止。否则，转到第2步。

随机算法并不能保证矩阵的每行都有s个1。因此，我们通过这种方法可能生成一个非正则的LDPC码。为了使得LDPC码是正则的，需要在算法中加入一些必要的限制。

下面介绍一种LDPC码的代数构造算法：

第1步：选择$p>\dfrac{r-1}{s-1}$。

第2步：通过对单位矩阵$\boldsymbol{I}_p$的每一行进行循环向右移1位，构造出一个$p\times p$的矩阵：

$$\boldsymbol{J}=\begin{bmatrix}0&1&0&0&\cdots&0\\0&0&1&0&\cdots&0\\0&0&0&1&\cdots&0\\0&0&0&0&\cdots&1\\1&0&0&0&\cdots&0\end{bmatrix} \tag{3-28}$$

第3步：奇偶校验矩阵可以得到：

$$\boldsymbol{H}=\begin{bmatrix}\boldsymbol{J}^0&\boldsymbol{J}^0&\boldsymbol{J}^0&\cdots&\boldsymbol{J}^0\\\boldsymbol{J}^0&\boldsymbol{J}^1&\boldsymbol{J}^2&\cdots&\boldsymbol{J}^{s-1}\\\boldsymbol{J}^0&\boldsymbol{J}^2&\boldsymbol{J}^3&\cdots&\boldsymbol{J}^{(2(s-1))}\\&&&\cdots&\\\boldsymbol{J}^0&\boldsymbol{J}^{(r-1)}&\boldsymbol{J}^{(2(r-1))}&\cdots&\boldsymbol{J}^{((r-1)(s-1))}\end{bmatrix} \tag{3-29}$$

其中$\boldsymbol{J}_0=\boldsymbol{I}_p$，并且$\boldsymbol{J}$的$l$次方是通过对矩阵$\boldsymbol{I}_p$的每一行循环向右移$(l \bmod p)$位。此矩阵每列有$r$个1，每行有$s$个1。因此这是一个正则的LDPC码。

对于一个正则码，每一个收到的码字都由r个方程进行校验，每一个校验方程合计s个码符号。所有的线性码，包含LDPC码，都可以用Tanner图表示。

定义3.28　Tanner图是基于校验方程集合的线性码的图表示。它是一个二分图，因此它有两种节点：符号节点和校验节点。每一个符号节点只和校验节点连接，同样校验节点只和符号节点连接。

Tanner图中，码字的每一个部分都可以通过一个符号节点表示。因此，图中有n个符号节点。图中每一个码的校验方程都可以通过一个校验节点表示。因此，途中有$n-k$个校验节点。每一个校验节点通过一条边和它所校验的符号节点相连。

一个LDPC码能够通过简单的**迭代flipping算法**进行解码。下面是这种次优算法的描述：

1) 对从信道上收到的符号进行硬判决译码后形成一个二元向量v。如果我们在最开始已经收到了字，则此步骤可以省略。v是n—比特长的向量。

2) 计算伴随式$s = vH^T$。注意v每一部分只影响伴随式的s个部分。如果只有1比特出错，则伴随式的s个部分将等于1。

3) 计算所有的校验和，以及对向量v每一个比特计算不满足奇偶校验的数目。

4) 对向量v中具有不满足奇偶校验的最大数目的所有比特位进行比特翻转。

5) 返回第3步，迭代执行直到以下情况出现：

(1) 所有校验均满足，或者

(2) 到达一个预定义的最大迭代数。

Flipping 算法不能保证改正接近最小距离一半的错误。但是对于一个最长的分组长度，此次优算法能够很好地解决问题。

例3.28 通过下面的奇偶校验矩阵，我们考虑线性率1/3 码。

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{3-30}$$

此处，$n = 6, r = 3, s = 2$。这不是一个LDPC码，因为n不是足够大以致矩阵$\boldsymbol{H}$是稀疏的。此码对应的Tanner 图如图3-6所示。

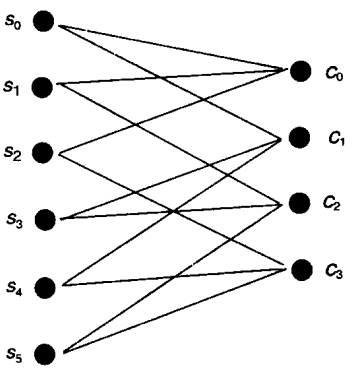


图3-6 例3-28中1/3线性码对应的Tanner 图

假设收到的字是v=[001000]。通过计算伴随式，我们得到$s = vH^T$=[1 0 0 1]。因为非零，所以这不是一个有效的码字。从伴随向量，我们注意到失败的奇偶校验是1和4。这就意味着在Tanner图中连接校验节点1和4的符号当中有一个是错误的。我们进行下面的观察：

(1) 接收到的向量中的比特4对应一个成功的校验，因为它连接校验节点2和3，这两个节点在伴随向量中都是0。

(2) 接收到的向量中的比特1对应一个成功的校验，因为它连接校验节点1。

(3) 接收到的向量中的比特5和比特6对应一个失败的校验，因为它连接校验节点4。

(4) 接收到的向量中的比特3对应两个失败的校验，因为它连接校验节点1和4，这两个节点在伴随向量中都是1。

通过flipping算法，我们把接收到的向量中的第3比特进行翻转，从而得到[0 0 0 0 0 0]。我们再次进行奇偶校验，发现此时它已经满足所有的校验。因此，这个例子中正确的向量是全零。

3.12 最优线性码

定义3.29 对一个 (n, k, d^*) **最优码**，不存在 $(n-1, k, d^*)$、$(n+1, k+1, d^*)$ 或 $(n+1, k, d^*+1)$ 码。

最优线性码给出了分组长度限制下的最好距离属性。多数最优码都是通过计算机长时间搜索找到的。对给定的一组参数n、k和d，可能存在不止一个最优码。例如存在两个不同的（25，5，10）二元最优码。

例3.29 考虑二元（24，12，8）码。可以验证：
(1) 不存在（23，12，8）码（只存在（23，12，7）码）。
(2) 不存在（25，13，8）码。
(3) 不存在（25，12，9）码。
所有二元（24，12，8）码是最优码。

3.13 最大距离可分（MDS）码

在这一节中我们考虑对给定的冗余度r，找到最大可能的最小距离d^*。

定理3.10 一个$(n, n-r, d^*)$码满足$d^* \leqslant r+1$。

证明：由Singleton界我们得到$d^* \leqslant n-k+1$。将$k=n-r$代入即得$d^* \leqslant r+1$。

定义3.30 一个$(n, n-r, r+1)$码称为**最大距离可分**（Maximum Distance Separable, MDS）码。一个MDS码是冗余度为r，最小距离等于$r+1$的线性码。

3.14 最小距离的界

一个(n, k)二元或非二元线性分组码最小距离的上界可以简单表示为式(3-7)中的Singleton边界，即$d^* \leqslant n-k+1$。此边界的正则形式可以写为：

$$\frac{d^*}{n} \leqslant (1-r)+\frac{1}{n} \tag{3-31}$$

其中r是码率。当分组长度很大时，因子$1/n$可以被忽略。另外一个上界可以直接通过式(3-25)中的汉明边界得到。通过不等式两边同时除以n，并且对二元情况设置$q=2$，我们可以得到：

$$\frac{1}{n}\log_2\left(\sum_{i=0}^{t}\binom{n}{i}\right) \leqslant 1-r \tag{3-32}$$

由于最小距离d^*以及纠错能力t是相关的，上面的关系是d^*的上界。对任意的n，记t_0表示使得式(3-32)成立的t的最大值。则容易得到当$n\to\infty$时，t/n不会超过t_0/n，其中t_0/n满足方程：

$$1-r=H\left(\frac{t_0}{n}\right) \tag{3-33}$$

泛化的汉明界可以写成

$$\frac{1}{n}\log_q\left(\sum_{i=0}^{i}\binom{n}{i}(q-1)^i\right) \leqslant 1-r \tag{3-34}$$

Plotkin给出了d^*另外一个界

$$\frac{1}{n}\left(\frac{qd^*-1}{q-1}-1-\log_q d^*\right) \leqslant 1-r \tag{3-35}$$

Elias给出了d^*的一个更紧的界，此界是一个渐近的形式：

$$\frac{d^*}{n} \leqslant 2A(1-A) \tag{3-36}$$

其中$r = 1+A\log_2 A + (1-A)\log_2(1-A)$，$0 \leqslant A \leqslant 1/2$。 (3-37)

d^*同样存在着一个下界。基于Gilbert和Varshamov的定理，d^*的下界是

$$\frac{d^*}{n} \geqslant \alpha \tag{3-38}$$

其中$r = 1+\alpha\log_2\alpha + (1-\alpha)\log_2(1-\alpha) = 1-H(\alpha)$，$0 \leqslant \alpha \leqslant 1/2$。 (3-39)

3.15 空时分组码

空时分组码是为多天线发射而设计的编码技巧。此编码是在空间域和瞬时域增加设计合适的冗余。空时分组码是由Tarokh、Seshadri和Calderbank首先提出来。在空时编码系统中，系统的符号错误率（SER）大约为

$$\overline{P}_e \approx \frac{c}{(G_c S)^{G_d}} \tag{3-40}$$

其中S是信噪比SNR，c是一个采用调制和信道特质的度量常数。G_c($G_c \geqslant 1$)表示**编码增益**，G_d表示系统的**分集数**。分集增益/分集数决定了SNR函数的误码率曲线的斜率，而编码增益决定了把非编码系统的误码率曲线水平平移到对应相同分集数的空时编码误码率曲线。有两种主要的空时编码机制：空时网格码（STTC）和空时分组码（STBC）。STTC的设计是为提供最大的编码增益和分集增益，而STBC的设计只为提供完全的分集增益和零或最小的编码增益。我们将学习STBC，并在第7章学习STTC。

1998年提出的**Alamouti方案**是史上第一个空时分组码，它给两发射天线的系统提供完全的发射分集。随后，Tarokh把Alamouti的发射分集方案泛化到任意数目的发射天线环境中，并且基于正交矩阵提出了更加复杂的STBC。STBC通过在接收端的**线性处理**方式能够达到**最大似然**（ML）解码。图3-7 给出了Alamouti空时编码器的框图。

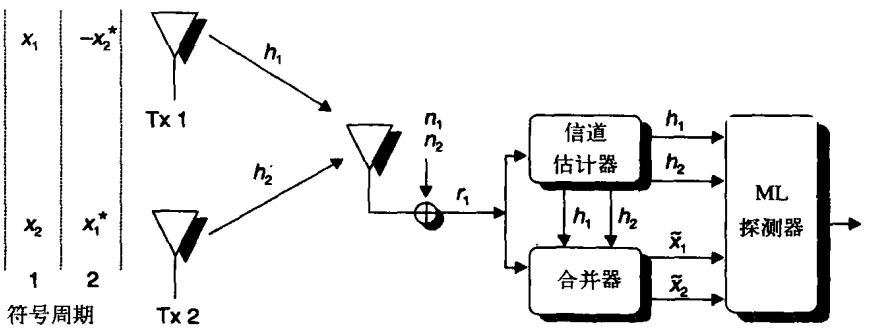


图3-7 Alamouti的两天线发射分集机制

在每次编码操作中，输入编码器两个符号x_1和x_2，然后根据下面的正交编码矩阵进行发射。

$$X = \begin{bmatrix} x_1 & -x_2^* \\ x_2 & x_1^* \end{bmatrix} \tag{3-41}$$

其中 x_1^*是x_1的复共轭。在符号周期1中，天线1和天线2分别发射x_1和x_2。

在符号周期2中，天线1和天线2分别发射 $-x_2^*$ 和 x_1^*。注意：

$$\mathbf{X}\mathbf{X}^H = (|x_1|^2 + |x_2|^2)I_2 \tag{3-42}$$

其中I_2是一个2×2的矩阵。假设衰弱信道系数h_1和h_2在连续的两个符号周期中均是常数。

在连续的两个符号周期中收到的信号，r_1和r_2可以表示为：

$$r_1 = h_1x_1 + h_2x_2 + n_1 \tag{3-43}$$

$$r_2 = -h_2x_2^* + h_2x_1^* + n_2 \tag{3-44}$$

其中n_1和n_2分别是两个符号周期的AWGN取样。

组合器把收到的信号进行如下组合：

$$\tilde{x}_1 = h_1^*r_1 + h_2r_2^* \tag{3-45}$$

$$\tilde{x}_2 = h_2^*r_1 - h_1r_2^* \tag{3-46}$$

对等能量M元（例如M-PSK）信号群，ML决策规则可以简化为

$$\hat{x}_1 = \arg\min_{\hat{x}_1\in S} d^2(\tilde{x}_1, \hat{x}_1) \tag{3-47}$$

$$\hat{x}_2 = \arg\min_{\hat{x}_2\in S} d^2(\tilde{x}_2, \hat{x}_2) \tag{3-48}$$

其中，$d^2(x, y) = |x-y|^2$。

基于信号群的种类，STBC可以分成**实信号的**STBC和**复信号的**STBC。一般说来，一个空时分组码是通过一个$N_t\times p$的发射矩阵来定义的。这里N_t是发射天线数量，p是发射一组编码符号的时间周期。如果变量为$x_1, x_2, \cdots, x_k$的一个$N_t\times p$实发射矩阵$\mathbf{X}_N$满足

$$\mathbf{X}_{N_t}.\mathbf{X}_{N_t}^T = c(|x_1|^2 + |x_2|^2 + \cdots + |x_k|^2)\mathbf{I}_{N_t} \tag{3-49}$$

则空时分组码可以提供码率为k/p的完全发射分集N_t。这种STBC被称为**实正交设计**。对于任意的实信号群有$N_t\times N_t$的发射方阵$\mathbf{X}_{N_t}$的STBC存在，当且仅当N_t=2, 4, 8。这些码是全码率(R = 1)并且提供全发射分集。例如：

当N_t=2个发射天线时

$$\mathbf{X}_2 = \begin{bmatrix} x_1 & -x_2 \\ x_2 & x_1 \end{bmatrix} \tag{3-50}$$

当N_t=4个发射天线时

$$\mathbf{X}_4 = \begin{bmatrix} x_1 & -x_2 & -x_3 & -x_4 \\ x_2 & x_1 & x_4 & -x_3 \\ x_3 & -x_4 & x_1 & x_2 \\ x_4 & x_3 & -x_2 & x_1 \end{bmatrix} \tag{3-51}$$

一般情况下，对N_t个发射天线，如果要达到全码率，发射周期的最小值是：

$$\begin{aligned} &N_t = 2, p = 2 \qquad && N_t = 6, p = 8 \\ &N_t = 3, p = 4 \qquad && N_t = 7, p = 8 \\ &N_t = 4, p = 4 \qquad && N_t = 8, p = 8 \\ &N_t = 5, p = 8 \end{aligned} \tag{3-52}$$

和Alamouti方案类似，由于发射矩阵的正交性，ML解码是通过接收端的线性处理在正交的STBC中完成的。

3.16 评注

1948年，Claude Elwood Shannon在*Bell System Technical Journal*上发表的经典论文导致了两个重要领域的产生：信息论和编码理论。当时Shannon才32岁。Shannon的信道编码定理说，“如果信息率小于信道容量，通过有限带宽有噪信道传输的数据的错误率可以减小到任意小的程度”。Shannon预测了存在好的信道码但没有构造它们。从那以后人们就开始了寻找好码的工作。Shannon的精辟论文可以从下面的网址得到：

http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html

在1950年，R. W. Hamming引入了第一个单一错误纠错码，至今还在使用。Golay（他的码将在下一章学习）对线性码方面的工作进行了扩展。Golay还提出了完备码的概念。Golay和Cocke在20世纪50年代后期开发了非二元汉明码。后来许多计算机用来搜索有趣的码。但是有些最著名的码是由真正天才发现的，而不是计算机穷搜索。

根据Shannon的定理，若$C(p)$表示比特错误概率等于p的BSC的容量，则对任意低的符号错误概率，我们一定要有码率$R<C(p)$。即使信道容量提供可以达到的码率的上界（$R=k/n$），抛开信道容量来评估一个码可能是误导。码的分组长度，直接转换为延迟，也是一个重要的参数。即使一个码的性能很不理想，但它可能对于给定的码率和长度而言是最好的码。我们已经看到通过增加码的分组长度，码率的界比小的分组长度的码更接近信道容量。但是分组长度越长意味着译码时的延迟越大，这是因为对一个码字的译码只能在收到整个码字后才能开始。最大允许的延迟受实际约束的限制。例如在移动无线电通信中，数据包限制在200比特之内。在这种情况下，具有很大分组长度的码字不能被采用。

低密度奇偶校验（LDPC）码是Gallager于1963年在他的博士论文中提出的。在30年后的20世纪90年代又被重新研究并成为无线通信标准的一部分。在无线通信中，使用多个天线变得非常普及。发射分集技术正在融入3G无线通信标准。空-时编码由Tarokh、Seshadri和Calderbank在1998年提出。1998年提出的Alamouti机制成为首个空-时分组码。它通过使用两个天线为系统提供了完全发射分集。

3.17 小结

- 字是符号的序列。码是一些称为码字的向量的集合。
- 码字（或任何向量）的汉明重量等于该码字中非零元素的个数。一个码字$\boldsymbol{c}$的汉明重量记为$w(\boldsymbol{c})$。
- 一个分组码由一组固定长度的码字组成。这些码字的固定长度称为分组长度，而且习惯上记为n。一个分组编码方案把k个信息符号转换成n个编码符号。这样的一个码记为(n,k)。
- 一个(n,k)码的码率定义为比率(k/n)，它反映的是码字所含信息符号的分数。
- 一个码的最小距离是任意两个码字间的最小汉明距离。一个最小距离为d^*的(n,k)码有时记为(n,k,d^*)码。一个码的最小重量是所有非零码字的最小重量，记为w^*。对一个线性码，它的最小距离等于其最小重量，即$d^*=w^*$。
- 一个线性码具有下述性质：

(1) 属于一个码的两个码字的和也是属于该码的码字。

(2) 全零码字总是个码字。

(3) 一个线性码两个码字之间的最小汉明距离等于任意非零码字的最小汉明重量，即$d^*=w^*$。

- 生成矩阵将一个长为k的向量转变（编码）成长为n的向量。设输入向量（未编码的符号）

表示为i，则编码符号为$\boldsymbol{c}=i\boldsymbol{G}$。

- 两个q元码称为等价的，如果其中一个可由另一个经下列变换中的一个或两个得到：

(1) 对固定位置上的符号进行置换。

(2) 对码的位置进行置换。

- 一个(n,k)系统码其中分组长度为n的码字的前k个符号为信息符号本身。型为$\boldsymbol{G}=[\boldsymbol{I}\mid\boldsymbol{P}]$的生成矩阵称为生成矩阵的系统型或标准型，其中$\boldsymbol{I}$为$k\times k$单位矩阵，而$\boldsymbol{P}$为$k\times(n-k)$矩阵。
- 一个给定码的奇偶校验矩阵$\boldsymbol{H}$满足$\boldsymbol{cH}^T=0$，其中$\boldsymbol{c}$是一个合法码字。因为$\boldsymbol{c}=i\boldsymbol{G}$，所以$i\boldsymbol{GH}^T=0$。对一个给定的码，它的奇偶校验矩阵不是惟一的。
- 一个最大距离可分码满足$d^*=n-k+1$。
- 要使一个码能纠t个错，必须满足$d^*\geqslant 2t+1$，其中d^*是码的最小距离。
- 设$\boldsymbol{C}$是$GF(q)$上的(n,k)码，$\boldsymbol{a}$是长为n的任意向量。则集合$\boldsymbol{a}+\boldsymbol{C}=\{\boldsymbol{a}+\boldsymbol{x}|\boldsymbol{x}\in\boldsymbol{C}\}$称为$\boldsymbol{C}$的一个陪集。$\boldsymbol{a}$和$\boldsymbol{b}$属于同一个陪集，当且仅当$(\boldsymbol{a}-\boldsymbol{b})\in\boldsymbol{C}$。
- 假设$\boldsymbol{H}$是一个(n,k)码的奇偶校验矩阵。则对任意向量$\boldsymbol{v}\in GF(q)^n$，向量$\boldsymbol{s}=\boldsymbol{vH}^T$称为$\boldsymbol{v}$的伴随式。它之所以称为伴随式（综合症）是因为它给出错误的症状，从而帮助我们诊断错误。
- 一个完备码达到汉明界，即

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}=q^n$$

- 二元汉明码具有性质$(n,k)=(2^m-1,2^m-1-m)$，其中m为任意正整数。汉明码是完备码。
- 一个Gallager码(r,s)是一个线性码，其校验矩阵$\boldsymbol{H}$满足条件：每一列有r个1，每一行有s个1。一个Gallager码的r和s很小时，它是低密度奇偶校验（LDPC）码。
- Tanner图是基于校验方程集合的线性码的图表示。它是一个二分图，因此它有两种节点：符号节点和校验节点。每一个符号节点只和校验节点连接，同样校验节点只和符号节点连接。
- 对一个(n,k,d^*)最优码，不存在$(n-1,k,d^*)$、$(n+1,k+1,d^*)$和$(n+1,k,d^*+1)$码。
- 一个$(n,n-r,r+1)$码称为最大距离可分（MDS）码。一个MDS码是一个冗余度为r，最小距离等于$r+1$的线性码。
- 空时分组码是为了多天线发射而设计的编码技巧。此编码是在空间域和瞬时域增加设计合适的冗余。
- Alamouti方案的正交编码矩阵表示为

$$\boldsymbol{X}=\begin{bmatrix}x_1 & -x_2^*\\ x_2 & x_1^*\end{bmatrix}$$

数学中最大的未解决的定理是为什么有些人比别人强。

——Mathesis, Adrian

习题

3.1　证明$\boldsymbol{C}=\{0000,1100,0011,1111\}$是一个线性码。它的最小距离是什么？

3.2　如果可能的话，构造下列参数的二元(n,k,d^*)码：

(1) (6, 1, 6)

(2) (3, 3, 1)

(3) (4, 3, 2)

3.3 考虑$GF(2)$上的下列生成矩阵

$$\boldsymbol{G}=\begin{bmatrix}1&0&1&0&0\\1&0&0&1&1\\0&1&0&1&0\end{bmatrix}$$

(1) 用此矩阵生成所有可能的码字。

(2) 求奇偶校验矩阵$\boldsymbol{H}$。

(3) 求与其等价的一个系统码的生成矩阵。

(4) 构造该码的标准阵列。

(5) 这个码的最小距离是多少?

(6) 这个码能检测多少错误?

(7) 写出该码能检测的所有错误模式。

(8) 这个码能纠多少个错?

(9) 如果我们用此编码方案,那么符号错误概率是多少?将它与未编码的错误概率进行比较。

(10) 这是一个线性码吗?

3.4 构造码$\boldsymbol{C}=\{00000, 10101, 01010, 11111\}$的生成矩阵。因为这个$\boldsymbol{G}$不是惟一的,给出另一个能生成这个码字集合的生成矩阵。

3.5 证明如果存在一个d^*为偶数的二元(n, k, d^*)码,则存在一个二元(n, k, d^*)码,其中所有码字都有偶数重量。

3.6 证明如果$\boldsymbol{C}$是一个二元线性码,则对$\boldsymbol{C}$加一个整体偶校验比特后得到的码仍是线性的。

3.7 标记向量$\boldsymbol{u}=u_1u_2u_3\cdots u_n$和$\boldsymbol{v}=v_1v_2v_3\cdots v_n$。同时标记$(\boldsymbol{u}|\boldsymbol{v})=u_1u_2u_3\cdots u_nv_1v_2v_3\cdots v_n$作为长度为$2n$的向量。接下来,假设$\boldsymbol{C}_1$是一个二元$(n, k_1, d_1)$码,$\boldsymbol{C}_2$是一个二元$(n, k_2, d_2)$码,并且码字$\boldsymbol{u}\in\boldsymbol{C}_1$,$\boldsymbol{v}\in\boldsymbol{C}_2$,$d_1$和$d_2$分别表示两个码的最小距离。产生一个$(\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v})$新的码$\boldsymbol{C}_3$。

(1) 证明$\boldsymbol{C}_3$是一个$(2n, k_1+k_2)$码。

(2) $\boldsymbol{C}_3$的最小距离是多少?

3.8 对下列每一个集合$\boldsymbol{S}$,列出扩张码$<\boldsymbol{S}>$:

(1) $\boldsymbol{S}=\{0101, 1010, 1100\}$

(2) $\boldsymbol{S}=\{1000, 0100, 0010, 0001\}$

(3) $\boldsymbol{S}=\{11000, 01111, 11110, 01010\}$

3.9 考虑(23, 12, 7)二元码。证明若将它用在一个比特错误概率为$p=0.01$的二元对称信道(BSC)中,字错误概率约为0.00008。

3.10 假设$\boldsymbol{C}$是一个二元码,它的奇偶校验矩阵为$\boldsymbol{H}$。证明由$\boldsymbol{C}$通过添加整体奇偶校验比特得到的扩展码$\boldsymbol{C}_1$的奇偶校验矩阵为

$$\boldsymbol{H}_1=\begin{bmatrix}&&&&|&0\\&&&&|&0\\&\boldsymbol{H}&&&|&\vdots\\&&&&|&0\\-&-&-&-&-&-\\1&1&\cdots&1&|&1\end{bmatrix}$$

3.11 $GF(4)$ 上的一个 $(5,3)$ 码的生成矩阵为

$$\boldsymbol{G}=\begin{bmatrix}1&0&0&1&1\\0&1&0&1&2\\0&0&1&1&3\end{bmatrix}$$

(1) 求奇偶校验矩阵。
(2) 这个码能检测多少个错？
(3) 这个码能纠多少个错？
(4) 这个码能纠多少个删除？
(5) 这是个完备码吗？

3.12 考虑一个$GF(11)$上的长度是$n=10$的线性分组码，并满足下面的两个限制条件：$\sum_{i=0}^{9} ic_i=0 \pmod{11}$和$\sum_{i=0}^{9}(10-i)c_i=0 \pmod{11}$。找出此码的最小距离。

3.13 设C是长度为n、最小距离为7的二元完备码。证明$n=7$或$n=23$。

3.14 用r_H表示二元汉明码的码率。求 $\lim_{k\to\infty} r_H$。

3.15 证明不存在 $(15,8,5)$ 码。

3.16 检验下面的STBC码是否是正交的。

$$\boldsymbol{X}=\begin{bmatrix}x_1&-x_2^*&-x_3^*&0\\x_2&x_1^*&0&-x_3^*\\x_3&0&x_1^*&x_2^*\end{bmatrix}$$

上机习题

3.17 写一个程序，当知道一个码的生成矩阵时，该程序可以求$GF(2)$上线性分组码的最小距离。

3.18 将上述程序推广到$GF(q)$上的任意线性分组码的情况。

3.19 写一个程序对汉明界等式中的参数n, q, M和t，穷举搜索所有的完备码。对$1\leqslant n\leqslant 200$，$2\leqslant q\leqslant 11$的情况进行搜索。

3.20 给码率为$(2^m-1)/(2^m-1-m)$的通用二元汉明码写一个程序，该程序在得到输入m和一个将被编码的比特流时输出一个编码后的比特流。也给译码器编个程序完成下面的任务：
(1) 写一个错误生成器模块，在给定一个比特流作为输入时，它的输出流的每个比特都以概率p发生了改变，即比特错误概率为p。
(2) 对$m=3$，将汉明码编码后的比特流输入到上述模块，然后对收到的字用译码器进行译码。
(3) 画出剩余错误概率（译码之后的错误概率）关于p的函数。注意如果你在$\mathrm{BER}=10^{-r}$范围内工作，你必须以10^{r+2}比特的数量级传输（考虑为什么）。
(4) 对$m=5$、8及15的情况重复上述模拟。当$m\to\infty$时会怎么样？

3.21 写一个程序，使用下列方式生成一个(20, 3, 4)的LDPC码：
(1) 随机收缩方式，
(2) 算术收缩方式。

3.22 写一个程序用于搜索$N_t>2$时的正交STBC。

第4章 循 环 码

我们所获取的真理不仅仅来自于理论，还要靠灵感。

——Blaise Pascal（1623—1662）

4.1 循环码简介

在第3章中，当处理线性分组码时，在分组码的结构上加进去了一些线性限制条件。这些结构上的性质可以帮我们寻找能够快速、简易地编码和译码的线性分组码。在本章中，我们将研究一个线性分组码的子类，该码在结构上有另外的限制条件。这种限制条件就是一个码字任意循环移位的结果都是另一个有效码字。这种条件使得循环码可用移位寄存器非常简单地实现。高效的电路实现是任何错误控制码的卖点。我们也将看到伽罗瓦（Galois）域理论可以有效地用于研究、分析和发现新的循环码。循环码的伽罗瓦域表示导致低复杂度编码和译码算法。

本章内容结构如下：在前三节中，我们研究多项式。我们将回顾一些概念并学习几个新概念。然后利用这些数学工具构造和分析循环码。接下来将介绍循环码的矩阵描述，并简要介绍准循环码和截短循环码。然后将讨论一些流行的循环码。本章最后讨论循环码的电路实现。

定义4.1 我们称码$\boldsymbol{C}$是循环码，如果

(1) $\boldsymbol{C}$是一个线性码；

(2) 一个码字的任意循环移位的结果还是一个码字，即若$a_0a_1\cdots a_{n-1}$为$\boldsymbol{C}$中的一个码字，则$a_{n-1}a_0\cdots a_{n-2}$也是$\boldsymbol{C}$中的一个码字。

例4.1 二元码$\boldsymbol{C}_1=\{0000, 0101, 1010, 1111\}$是个循环码。然而$\boldsymbol{C}_2=\{0000, 0110, 1001, 1111\}$却不是一个循环码，但与前面的码等价。将$\boldsymbol{C}_2$的第三位和第四位分量交换就得到$\boldsymbol{C}_1$。

4.2 多项式

定义4.2 一个**多项式**就是如下的数学表达式

$$f(x)=f_0+f_1x+\cdots+f_mx^m \tag{4-1}$$

其中符号x称为未知元，系数$f_0, f_1,\cdots, f_m$是$GF(q)$中的元素。系数f_m称为最高项系数。若$f_m\neq 0$，则m称为该多项式的**次数**，并记为$\deg f(x)$。

定义4.3 若一个多项式的最高项系数为单位元，则该多项式称为首一的。

例4.2 $f(x)=3+7x+x^3+5x^4+x^6$是$GF(8)$上的一个首一多项式。该多项式的次数为6。

多项式在研究循环码（本章的主题）方面起到重要作用。令$F[x]$为系数在$GF(q)$上的关于x的所有多项式的集合。$F[x]$中的不同多项式可以按通常方式进行加法、减法和乘法运算。$F[x]$就是一种叫做环的代数结构的一个例子。环满足定义域的（见3.2节）八个定理中的前七个。$F[x]$不是一个域，因为次数大于零的多项式不存在乘法逆元。容易证明，如果$f(x)$，

$g(x)\in F[x]$，则$\deg(f(x)g(x))=\deg f(x)+\deg g(x)$。但是，$\deg(f(x)+g(x))$ 不一定为$\max\{\deg f(x), \deg g(x)\}$。例如，考虑二元域$GF(2)$ 上的两个多项式$f(x)=1+x^2$和$g(x)=1+x+x^2$，则有$\deg(f(x)+g(x))=\deg(x)=1$。这是因为在$GF(2)$ 上，$1+1=0$，从而$x^2+x^2=(1+1)x^2=0$。

例4.3　考虑$GF(3)$ 上的多项式$f(x)=2+x+x^2+2x^4$和$g(x)=1+2x^2+2x^4+x^5$，则

$$\begin{aligned} f(x)+g(x) &= (2+1)+x+(1+2)x^2+(2+2)x^4+x^5 = x+x^4+x^5 \\ f(x)\cdot g(x) &= (2+x+x^2+2x^4)(1+2x^2+2x^4+x^5) \\ &= 2+x+(1+2.2)x^2+2x^3+(2+2+2.2)x^4+(2+2)x^5 \\ &\quad +(1+2+1)x^6+x^7+2.2x^8+2x^9 \\ &= 2+x+(1+1)x^2+2x^3+(2+2+1)x^4+(2+2)x^5+(1+2+1)x^6+x^7+x^8+2x^9 \\ &= 2+x+2x^2+2x^3+2x^4+x^5+x^6+x^7+x^8+2x^9 \end{aligned}$$

注意系数的加法和乘法都是在$GF(3)$ 上进行的。

例4.4　考虑$GF(2)$ 上的多项式$f(x)=1+x$，有

$$(f(x))^2=1+(1+1)x+x^2=1+x^2$$

同样考虑$GF(3)$ 上的多项式$f(x)=1+x$，有

$$(f(x))^2=1+(1+1)x+x^2=1+2x+x^2$$

4.3　多项式的除法算法

多项式的除法算法是指对$F[x]$中的任意一对多项式$a(x)$ 和$b(x)\neq 0$，存在惟一一对多项式$q(x)$ 和$r(x)$，分别称为商和余数，满足$a(x)=q(x)b(x)+r(x)$，其中$\deg r(x)<\deg b(x)$。该余数有时又称为**剩余**，并记为$R_{b(x)}[a(x)]=r(x)$。

剩余的两个重要性质为

(1) $R_{f(x)}[a(x)+b(x)]=R_{f(x)}[a(x)]+R_{f(x)}[b(x)]$　(4-2)

(2) $R_{f(x)}[a(x)\cdot b(x)]=R_{f(x)}\{R_{f(x)}[a(x)]\cdot R_{f(x)}[b(x)]\}$　(4-3)

其中$a(x)$、$b(x)$ 和$f(x)$ 都为$GF(q)$ 上的多项式。

例4.5　设$a(x)=x^3+x+1$和$b(x)=x^2+x+1$为定义在$GF(2)$ 上的多项式。我们可进行$a(x)$ 对$b(x)$ 的竖式除法如下

$b(x)\longrightarrow x^2+x+1$

$$\begin{array}{r} x+1 \leftarrow q(x) \\ x^2+x+1 \overline{\big)\, x^3+\quad x+1} \leftarrow a(x) \\ \underline{x^3+x^2+x\quad\quad} \\ x^2+\quad 1 \\ \underline{x^2+x+1} \\ x \leftarrow r(x) \end{array}$$

因此，$a(x)=(x+1)b(x)+x$。可以写为$a(x)=q(x)b(x)+r(x)$，其中$q(x)=x+1$，$r(x)=x$。注意这里有$\deg r(x)<\deg b(x)$。

定义4.4　令$f(x)$ 为$F[x]$中的一个固定多项式。$F[x]$中的两个多项式$g(x)$ 和$h(x)$ 关于模$f(x)$ 同余的概念可描述为$g(x)\equiv h(x)\pmod{f(x)}$，如果$g(x)-h(x)$ 能被$f(x)$ 整除。

例4.6 设$g(x)=x^9+x^2+1$, $h(x)=x^5+x^2+1$和$f(x)=x^4+1$为定义在$GF(2)$上的多项式。因为$g(x)-h(x)=x^5f(x)$，故我们可写为$g(x)\equiv h(x) \pmod{f(x)}$。

下面我们记$F[x]/f(x)$为$F[x]$中次数小于$\deg f(x)$的多项式集合，其中的加法和乘法是在模$f(x)$意义下进行的，如下所述：

(1) 若$a(x)$和$b(x)$都属于$F[x]/f(x)$，则$a(x)+b(x)$在$F[x]/f(x)$中的形式同在$F[x]$中的一样，这是因为$\deg a(x)<\deg f(x)$，$\deg b(x)<\deg f(x)$，因而$\deg (a(x)+b(x))<\deg f(x)$。

(2) 乘积$a(x)b(x)$是次数小于$\deg f(x)$，且与在$F[x]$中的$a(x)b(x)$是模$f(x)$同余的惟一一个多项式。

我们称$F[x]/f(x)$为（$F[x]$上）模$f(x)$的**多项式环**。在前面曾提到过，环满足定义域的8个定理中的前7个。若在一个环中每一个元素都有乘法逆元，则它构成一个域。

例4.7 考虑定义在$GF(2)$上的$F[x]/(x^2+x+1)$中的乘积$(x+1)^2$。$(x+1)^2=x^2+x+x+1=x^2+1=x \pmod{x^2+x+1}$。

定义在$GF(2)$上的$F[x]/(x^2+1)$中的乘积$(x+1)^2$可表示为$(x+1)^2=x^2+x+x+1=x^2+1=0 \pmod{x^2+x+1}$。

定义在$GF(3)$上的$F[x]/(x^2+x+1)$中的乘积$(x+1)^2$可表示为$(x+1)^2=x^2+x+x+1=x^2+2x+1=x \pmod{x^2+x+1}$。

若$f(x)$的次数为n，则$GF(q)$上的环$F[x]/f(x)$由次数小于等于$n-1$的多项式构成。这个环的大小将是q^n，因为多项式的n个系数中的每一个可以是$GF(q)$中q个元素的任意一个。

例4.8 考虑定义在$GF(2)$上的环$F[x]/(x^2+x+1)$。这个环中的多项式的最高次数为1。这个环有$q^n=2^2=4$个元素（每个元素都是一个多项式）。环中的元素为0、1、x和$x+1$。加法和乘法表如表4-1所示。

表4-1 针对域$F[x]/(x^2+x+1)$的加法和乘法表

+	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

.	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

下面考虑定义在$GF(2)$上的$F[x]/(x^2+1)$。该环的元素为0、1、x和$x+1$。加法和乘法表如表4-2所示。

表4-2 针对环$F[x]/(x^2+1)$的加法和乘法表

+	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

.	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	1	$x+1$
$x+1$	0	$x+1$	$x+1$	0

值得注意的是$F[x]/(x^2+x+1)$实际上是域，因为所有非零元素的乘法逆元也都存在。另一方面，$F[x]/(x^2+1)$就不是域，因为元素$x+1$的乘法逆元不存在。

值得探索具有什么性质的$f(x)$能使$F[x]/f(x)$构成域。我们很快会发现，多项式$f(x)$必须是既约的（不可分解的）。

定义4.5　$F[x]$中的一个多项式称为**可约的**，如果存在$F[x]$中的元素$a(x)$和$b(x)$使$f(x)=a(x)b(x)$，其中$\deg a(x)$和$\deg b(x)$都小于$\deg f(x)$。若$f(x)$不是可约的，则称为**既约的**。次数至少为1的一个首一既约多项式称为**素多项式**。

有必要把可约多项式同可分解为素数乘积的正整数比较一下。$F[x]$中的任意一个首一多项式可惟一分解为一些既约首一多项式（素多项式）的乘积。检验素多项式的一种方法就是尝试所有可能的分解，这需要计算机搜索。任意伽罗瓦域上的任意次数的素多项式都存在。

定理4.1

(1) 当且仅当$f(a)=0$，多项式$f(x)$有一个线性因子$(x-a)$，其中a是域上的一个元素。

(2) $GF(q)$上的$F[x]$中的次数为2或3的一个多项式$f(x)$是既约的，当且仅当对$GF(q)$中的所有元素a，$f(a)\neq 0$。

(3) 在任意域上$x^n-1=(x-1)(x^{n-1}+x^{n-2}+\cdots+x+1)$都成立。第二个因子可能还可以进一步约分。

证明：

(1) 若$f(x)=(x-a)g(x)$，则显然有$f(a)=0$。另一方面，若$f(a)=0$，由除法算法知，$f(x)=q(x)(x-a)+r(x)$，其中$\deg r(x)<\deg (x-a)=1$。这说明$r(x)$是个常数。但由于$f(a)=0$，$r(x)$必然为零，因此$f(x)=q(x)(x-a)$。

(2) $GF(q)$上的次数为2或3的一个多项式是可约的，当且仅当它有一个线性因子，因此(2)中的结果由(1)直接可得。该结果对次数大于3的多项式并不成立。这是因为一个次数为4或更高的多项式可能会分解为几个多项式的积，但它们中不含形式为$(x-a)$的线性因子。

(3) 由(1)得，$(x-1)$是(x^n-1)的一个因子。利用(x^n-1)对$(x-1)$的竖式除法可得到$(x^{n-1}+x^{n-2}+\cdots+x+1)$。

例4.9　考虑$GF(2)$上的$f(x)=x^3-1$。利用定理4.1中的(3)我们可以写为$x^3-1=(x-1)(x^2+x+1)$。这种分解在任何域上都成立。现在让我们尝试分解第二项，$p(x)=(x^2+x+1)$。

$$\text{在}GF(2)\text{上有}p(0)=0+0+1=1$$
$$\text{在}GF(2)\text{上有}p(1)=1+1+1=1$$

因此$p(x)$不能进一步分解了（根据定理4.1的(2)）。

故在$GF(2)$上有$x^3-1=(x-1)(x^2+x+1)$。

下面考虑$f(x)=x^3-1$在$GF(3)$上的情况，

$$x^3-1=(x-1)(x^2+x+1)$$

同样令$p(x)=(x^2+x+1)$，

$$\text{在}GF(3)\text{上有}p(0)=0+0+1=1$$
$$\text{在}GF(3)\text{上有}p(1)=1+1+1=0$$
$$\text{在}GF(3)\text{上有}p(2)=2.2+2+1=1+2+1=1$$

因为$p(1)=0$，由(1)可得$(x-1)$是$p(x)$的一个因子。

因此在$GF(3)$上有

$$x^3-1=(x-1)(x-1)(x-1)$$

定理4.2 当且仅当$f(x)$是$F[x]$上的素多项式时，环$F[x]/f(x)$是域。

证明：为了证明一个环是域，我们必须证明该环上的每一个非零元素都有一个乘法逆元。设$s(x)$是该环上的一个非零元素，因为$s(x)$包含在$F[x]/f(x)$中，我们有$\deg\ s(x)<\deg\ f(x)$。可以证明两个多项式$f(x)$和$s(x)$的最大公因子（GCD）可表示为

$$\mathrm{GCD}(f(x),s(x))=a(x)f(x)+b(x)s(x)$$

其中$a(x)$和$b(x)$都是$GF(q)$上的多项式。因为$f(x)$在$F[x]$上是既约的，我们有

$$\mathrm{GCD}(f(x),s(x))=1=a(x)f(x)+b(x)s(x)$$

于是，$1=R_{f(x)}[1]=R_{f(x)}[a(x)f(x)+b(x)s(x)]$

$$\begin{aligned}&=R_{f(x)}[a(x)f(x)]+R_{f(x)}[b(x)s(x)] && \text{(剩余的性质 (1))}\\&=0+R_{f(x)}[b(x)s(x)]\\&=R_{f(x)}\{R_{f(x)}[b(x)]\cdot R_{f(x)}[s(x)]\} && \text{(剩余的性质 (2))}\\&=R_{f(x)}\{R_{f(x)}[b(x)]\cdot s(x)\}\end{aligned}$$

故$R_{f(x)}[b(x)]$是$s(x)$的乘法逆元。

下面让我们来证明定理的必要性。我们假定$f(x)$的次数至少为2，而且不是素多项式（次数为1的多项式总是既约的）。因此存在次数至少为1的多项式$r(x)$和$s(x)$，使

$$f(x)=r(x)s(x)$$

若环$F[x]/f(x)$确实为域，那么$r(x)$和$r^{-1}(x)$的乘法有意义，因为域中所有多项式都必存在对应的乘法逆元。因此

$$\begin{aligned}s(x)&=R_{f(x)}\{s(x)\}\\&=R_{f(x)}\{r(x)r^{-1}(x)s(x)\}=R_{f(x)}\{r^{-1}(x)r(x)s(x)\}=R_{f(x)}\{r^{-1}(x)f(x)\}=0\end{aligned}$$

但我们已经假定$s(x)\neq 0$，因此，产生矛盾。此矛盾表明该环不是域。

注意一个素多项式既为首一的，又是既约的。在上述定理中，要想得到一个域，$f(x)$为既约的条件就足够了。上述定理其实可以表述为“环$F[x]/f(x)$是域的充分必要条件是$f(x)$在$F[x]$中是既约的”。

这样，我们现在得到了一种产生伽罗瓦域的漂亮的过程！若我们能辨别$GF(q)$上次数为n的一个素多项式，我们就可以构造一个有q^n个元素的伽罗瓦域。这样的域将含有多项式作为其元素。这些多项式将由定义在$GF(q)$上的所有次数小于n的多项式构成。容易看到共有q^n个这样的多项式，它们组成了**扩域**上的元素。

例4.10 考虑$GF(2)$上的多项式$p(x)=x^3+x+1$。因为$p(0)\neq 0$且$p(1)\neq 0$，该多项式在$GF(2)$上是既约的。因为它又是首一的，故$p(x)$是素多项式。这里有$n=3$，所以我们可以用$p(x)$来构造$2^3=8$个元素的域，即$GF(8)$。这个域上的元素将为0、1、x、$x+1$、x^2、x^2+1、x^2+x、x^2+x+1，它们正好是次数小于$n=3$的全部可能的多项式。很容易构造这个域上的加法和乘法表（留作习题）。

建立了必要的数学工具之后，现在重新回到我们对循环码的学习。在本章中从现在开始我们固定$f(x)=x^n-1$。我们也把$F(x)/f(x)$记为R_n。在继续进行前我们注意到下面的结论：

(1) $x^n=1(\text{mod } x^n-1)$。因此任何多项式模x^n-1时，只需把x^n代换为1，把x^{n+1}代换为x，依此类推。

(2) 一个码字可以惟一地用一个多项式表示。一个码字由一个元素序列组成。我们可以用一个多项式来表示码字中所有这些元素的位置和值。例如码字$c_0c_1c_2\cdots c_n$可以用多项式$c(x)=c_0+c_1x+c_2x^2+\cdots+c_nx^n$来表示。另外一个例子是$GF(8)$上的码字$c=207735$可以用多项式表示为$c(x)=2+7x^2+7x^3+3x^4+5x^5$。

(3) 任意多项式乘以x对应于码字元素的单一右循环移位。更确切地说，在R_n中用x乘以$c(x)$，得到$x\cdot c(x)=c_0x+c_1x^2+c_2x^3+\cdots+c_nx^{n+1}=c_n+c_0x+c_1x^2+c_2x^3+\cdots+c_{n-1}x^n$。

定理4.3　R_n中的一个码$\boldsymbol{C}$是循环码的充分必要条件是$\boldsymbol{C}$满足下列条件：

(1) $a(x),b(x)\in\boldsymbol{C}\Rightarrow a(x)+b(x)\in\boldsymbol{C}$　(4-4)

(2) $a(x)\in\boldsymbol{C}$且$r(x)\in R_n\Rightarrow a(x)r(x)\in\boldsymbol{C}$　(4-5)

证明：

(1) 假设$\boldsymbol{C}$是R_n中的一个循环码。因为循环码是线性分组码的一个子集，故第一个条件满足。

(2) 设$r(x)=r_0+r_1x+r_2x^2+\cdots+r_nx^n$。乘以$x$后对应于一个右循环移位。但根据定义，一个循环码字的循环移位还是一个合法码字，即

$$x\cdot a(x)\in\boldsymbol{C},x\cdot(xa(x))\in\boldsymbol{C}$$

依此类推。因此

$$r(x)a(x)=r_0a(x)+r_1xa(x)+r_2x^2a(x)+\cdots+r_nx^na(x)$$

也在$\boldsymbol{C}$中，因为每一个被加数也在$\boldsymbol{C}$中。

下面我们证明定理的必要性。假定（1）和（2）都满足。把$r(x)$看做一个常量。则（1）表明$\boldsymbol{C}$是线性的。在（2）中令$r(x)=x$，这表明每个循环移位都会产生一个码字。因此（1）和（2）表明$\boldsymbol{C}$是个循环码。

在下一节中，我们将用到目前为止建立起来的数学工具来构造循环码。

4.4　一种循环码的生成方法

下面的步骤可用来生成一个循环码：

(1) 在R_n中取一个多项式$f(x)$。

(2) 用R_n中的所有多项式乘以$f(x)$得到一个多项式集合。

(3) 上述所得多项式集合对应属于一个循环码的码字的集合。该码的分组长度为n。

例4.11　考虑定义在$GF(2)$上的R_3中的多项式$f(x)=1+x^2$。一般地，$R_3=(F[x]/(x^3-1))$中的一个多项式可以表示为$r(x)=r_0+r_1x+r_2x^2$，其中系数取值为0或1（因为是定义在$GF(2)$上的）。

因此，定义在$GF(2)$上的R_3中的多项式有$2\times2\times2=8$个，它们是0、1、x、x^2、$1+x$、$1+x^2$、$x+x^2$、$1+x+x^2$。要生成一个循环码，我们用R_3中的这8个元素去乘以$f(x)$，然后将结果模(x^3-1)：

$(1+x^2)\cdot0=0,(1+x^2)\cdot1=(1+x^2),(1+x^2)\cdot x=1+x,(1+x^2)\cdot x^2=x+x^2,$

$(1+x^2)\cdot(1+x)=x+x^2,(1+x^2)\cdot(1+x^2)=1+x,(1+x^2)\cdot(x+x^2)=(1+x^2),$

$(1+x^2)\cdot(1+x+x^2)=0$。

故只有4个不同的码字：{0，$1+x$，$1+x^2$，$x+x^2$}，它们对应于{000，110，101，011}。

从上面的例子看，我们似乎有一个**生成多项式**（Generator Polynomial），它可以用来构造循环码。

定理4.4 设C是R_n中 (n,k) 非零循环码。则

(1) C中存在惟一一个最小次数的首一多项式$g(x)$。

(2) 循环码C由所有生成多项式$g(x)$ 与次数不大于$k-1$的多项式的乘积构成。

(3) $g(x)$ 是x^n-1的一个因子。

证明：

(1) 设$g(x)$ 和$h(x)$ 都是C中最小次数的首一多项式，则$g(x)-h(x)$ 也在C中，但有着更小的次数。如果$g(x)\neq h(x)$，则可以找到一个适当的常数使之与$g(x)-h(x)$ 的乘积是首一的，而且也在C中，但次数小于$g(x)$ 的次数。此为矛盾。

(2) 设$a(x)\in C$。则由除法公式可得$a(x)=q(x)g(x)+r(x)$，其中$\deg\ r(x)<\deg\ g(x)$。但$r(x)=a(x)-q(x)g(x)\in C$，因为等式右边是两个码字。但$g(x)$ 的次数必须为所有码字中最低的，这只有在$r(x)=0$或$a(x)=q(x)g(x)$ 时才可能。所以一个码字是由生成多项式$g(x)$ 乘以多项式$q(x)$ 得到的。对一个定义在$GF(q)$ 上的码，有q^k个不同的可能码字。这些码字对应于$g(x)$ 与q^k个不同多项式$q(x)$ 的乘积，在此$\deg\ q(x)\leqslant(k-1)$。

(3) 根据除法算法有$x^n-1=q(x)g(x)+r(x)$，这里$\deg\ r(x)<\deg\ g(x)$，或者$r(x)=\{(x^n-1)-q(x)g(x)\}$ modulo $(x^n-1)=-q(x)g(x)$。但是$-q(x)g(x)\in C$，因为我们在用生成多项式乘以另外一个多项式$-q(x)$。这与$g(x)$ 为最小次数的假设相矛盾，除非$r(x)=0$。这表明$x^n-1=q(x)g(x)$，即$g(x)$ 是x^n-1的一个因子。

定理的最后部分给出了获得一个循环码生成多项式的方法。我们只需要把x^n-1分解成既约的首一多项式。仅仅通过分解x^n-1我们也能找到所有分组长度为n的可能的循环码字。

注释1：一个循环码C可能包含除生成多项式之外的多项式，它也能生成C。但只有最小次数的那个多项式才称为生成多项式。

注释2：$g(x)$ 的次数为$n-k$（这一点将在后面得到证明）。

例4.12 要找到分组长度为3的所有二元循环码，我们首先分解x^3-1。注意在$GF(2)$ 上有$1=-1$，因为$1+1=0$。所以

$$x^3-1=x^3+1=(x+1)(x^2+x+1)$$

因此，我们制作了表4-3。

表4-3 分组长度为3的所有可能二元循环码

生成多项式	码（多项式）	码（二元）
1	$\{R_3\}$	{000, 001, 010, 011, 100, 101, 110, 111}
$(x+1)$	$\{0,\ x+1,\ x^2+x,\ x^2+1\}$	{000, 011, 110, 101}
(x^2+x+1)	$\{0,\ x^2+x+1\}$	{000, 111}
$(x^3+1)=0$	{0}	{000}

由生成多项式产生码字的一个简单编码规则为

$$c(x)=i(x)g(x) \tag{4-6}$$

其中$i(x)$为信息多项式，$c(x)$为码字多项式，而$g(x)$为生成多项式。我们已经看到在字（向量）和多项式之间一一对应。错误向量也可以表示为错误多项式$e(x)$。因此通过一个有噪信道后在接收端收到的字可以表示为

$$v(x)=c(x)+e(x) \tag{4-7}$$

我们定义**伴随式多项式**（Syndrome Polynomial）$s(x)$为$v(x)$除以$g(x)$之后的余数，即

$$s(x)=R_{g(x)}[v(x)]=R_{g(x)}[c(x)+e(x)]=R_{g(x)}[c(x)]+R_{g(x)}[e(x)]=R_{g(x)}[e(x)] \tag{4-8}$$

因为由式(4-6)得$R_{g(x)}[c(x)]=0$。

例4.13　考虑分组长度为$n=4$的三元循环码（即在$GF(3)$上）的生成多项式$g(x)=x^2+1$。我们这里处理的是循环码，$g(x)$的最高幂为$n-k$。因为$n=4$，k必为2。所以我们将要构造一个（4, 2）三元循环码。总共有$q^k=3^2=9$个码字。信息多项式和对应的码字多项式在表4-4中列出。

表4-4　使用$g(x)=x^2+1$构造的三元循环码

i	$i(x)$	$c(x)=i(x)g(x)$	c
00	0	0	0000
01	1	x^2+1	0101
02	2	$2x^2+2$	0202
10	x	x^3+x	1010
11	$x+1$	x^3+x^2+x+1	1111
12	$x+2$	x^3+2x^2+x+2	1212
20	$2x$	$2x^3+2x$	2020
21	$2x+1$	$2x^3+x^2+2x+1$	2121
22	$2x+2$	$2x^3+2x^2+2x+2$	2222

可以看出任意码字的循环移位将得到另一个合法码字。通过观察我们发现这个码的最小距离为2（有4个非零码字，其最小汉明重量=2）。因此这个码能检测1个错和纠0个错。

观察到码字多项式能被生成多项式整除的事实，我们可以检测到比码的最小距离所显示的更多的错误。因为我们讨论的是循环码，它是线性分组码的一个子集，因此为了不失一般性，我们可以用全零码来说明这个问题。

假设$g(x)=x^2+1$，要传送的码字是全零码字。那么接收到的码字就是错误多项式，即

$$v(x)=c(x)+e(x)=e(x) \tag{4-9}$$

在接收端，如果$g(x)$不能整除接收到的字$v(x)=e(x)$，则检测到一个错误。现在$g(x)$只有两项，因此如果$e(x)$有奇数个项，即如果错误个数为奇数，就能被译码器捕捉到。例如我们试图用$g(x)$去除$e(x)=x^3+x+1$，总是得到一个余数。对（4, 2）循环码的情况，$g(x)=x^2+1$，$d^*=2$说明我们可以检测$d^*-1=1$个错，但由此我们发现它可以检测任何奇数个数小于等于n的错误。在这种情况下，它可以检测1个错或3个错，但不能检测2个错。

4.5　循环码的矩阵描述

定理4.5　*假设C是一个循环码，其生成多项式$g(x)=g_0+g_1x+\cdots+g_rx^r$的次数为$r$。$C$的生成矩阵为*

$$G=\begin{bmatrix} g_0 & g_1 & \cdots & g_r & 0 & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_r & 0 & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & g_r & 0 & \cdots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \cdots & g_r \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \quad k=(n-r)\text{行}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{n\text{列}} \tag{4-10}$$

证明：矩阵的 $(n-r)$ 个行显然是线性独立的，因为矩阵是阶梯型的。这 $(n-r)$ 个行表示码字$g(x)$，$xg(x)$，$x^2g(x)$，…，$x^{n-r-1}g(x)$。因此该矩阵可以生成这些码字。现在要证明该矩阵能生成所有可能的码字，我们必须证明任何一个码字都能表示为码字$g(x)$，$xg(x)$，$x^2g(x)$，…，$x^{n-r-1}g(x)$ 的线性组合。

我们知道如果$c(x)$是个码字，则对某个多项式$q(x)$，它可以表示为

$$c(x)=q(x)\cdot g(x)$$

由于$c(x)$的次数小于n（因为码字的长度是n），我们有$q(x)$的次数小于$n-r$。所以

$$q(x)\cdot g(x)=(q_0+q_1x+\cdots+q_{n-r-1}x^{n-r-1})g(x)=q_0g(x)+q_1xg(x)+\cdots+q_{n-r-1}x^{n-r-1}g(x)$$

因此任意码字可以表示为$g(x)$，$xg(x)$，$x^2g(x)$，…，$x^{n-r-1}g(x)$ 的一个线性组合。这就证明了矩阵G确实是生成矩阵。

我们还知道该生成矩阵的维数为$k\times n$，因此$r=n-k$，即$g(x)$的次数为$n-k$。

例4.14 要找出分组长度为$n=4$的所有三元码（即在$GF(3)$上的码）的生成矩阵，我们首先分解x^4-1。

$$x^4-1=(x-1)(x^3+x^2+x+1)=(x-1)(x+1)(x^2+1)$$

我们知道x^4-1的所有因子都能生成一个循环码，结果生成矩阵在表4-5中给出。注意在$GF(3)$上有$-1=2$。

表4-5 $GF(3)$上分组长度$n=4$的循环码

$g(x)$	(n,k)	d^*	G
1	(4,4)	1	$[I_4]$
$(x-1)$	(4,3)	2	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$(x+1)$	(4,3)	2	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
(x^2+1)	(4,2)	2	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
(x^2-1)	(4,2)	2	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
(x^3-x^2+x-1)	(4,1)	4	$[-1\ \ 1\ \ -1\ \ 1]$
(x^3+x^2+x+1)	(4,1)	4	$[1\ \ 1\ \ 1\ \ 1]$
(x^4-1)	(4,1)	0	$[0\ \ 0\ \ 0\ \ 0]$

从表中可以看出没有一个（4, 2）三元循环码是单一纠错码（因为它们的最小距离小于3）。有趣的是我们看到没有一个三元（4, 2）汉明码是循环码！记住汉明码是单一纠错码，其中$n=(q^r-1)/(q-1)$且$k=(q^r-1)/(q-1)-r$，这里r是大于等于2的整数。因此（4, 2）三元汉明码存在，但不是循环码。

下一步是研究我们能否发现对应于生成多项式$g(x)$的奇偶校验多项式。我们已经知道$g(x)$是x^n-1的一个因子，因此可以写为

$$x^n-1=h(x)g(x) \tag{4-11}$$

其中$h(x)$是某个多项式。通过观察上式可以得到下述结论：

(1) 因为$g(x)$是首一的，$h(x)$也必定为首一的，因为等号左边是首一的（最高项系数为1）。

(2) 因为$g(x)$的次数为$n-k$，$h(x)$的次数必定为k。

假设C是R_n中的一个循环码，其生成多项式为$g(x)$。我们把$F[x]/f(x)$记为R_n，其中$f(x)=x^n-1$。在R_n中，$h(x)g(x)=x^n-1=0$，则任何属于C的码字可以写为$c(x)=a(x)g(x)$，其中多项式$a(x)\in R_n$，因此在R_n中有

$$c(x)h(x)=a(x)g(x)h(x)=a(x)\cdot 0=0$$

因此$h(x)$的表现就像一个**奇偶校验多项式**。任何合法码字乘以奇偶校验多项式都变成一个零多项式。这个概念跟第3章中介绍的奇偶校验矩阵可以相提并论。因为我们仍然在线性分组码领域，继续定义与奇偶校验多项式相关的奇偶校验矩阵。

设$\boldsymbol{C}$是一个循环码，它的奇偶校验多项式为$h(x)=h_0+h_1x+\cdots+h_kx^k$，则$\boldsymbol{C}$的奇偶校验矩阵为

$$\boldsymbol{H}=\underbrace{\begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & \cdots & h_0 \end{bmatrix}}_{n\text{列}} \Big\}(n-k)\text{ 行} \tag{4-12}$$

前面提到$\boldsymbol{cH}^T=0$，因此对任意信息向量$\boldsymbol{i}$有$\boldsymbol{iGH}^T=0$，故$\boldsymbol{GH}^T=0$。进一步有$\boldsymbol{s}=\boldsymbol{vH}^T$，其中$\boldsymbol{s}$是伴随式向量，而$\boldsymbol{v}$是接收到的字。

例4.15 对分组长度$n=7$的二元码，我们有

$$x^7-1=(x-1)(x^3+x+1)(x^3+x^2+1)$$

考虑$g(x)=(x^3+x+1)$。因为$g(x)$是x^7-1的一个因子，有一个由它生成的循环码。对应于$g(x)$的生成矩阵为

$$\boldsymbol{G}=\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

它的奇偶校验多项式$h(x)$为$(x-1)(x^3+x^2+1)=(x^4+x^2+x+1)$，其对应的奇偶校验矩阵为

$$\boldsymbol{H}=\begin{bmatrix}1&0&1&1&1&0&0\\0&1&0&1&1&1&0\\0&0&1&0&1&1&1\end{bmatrix}$$

该码的最小距离为3，它恰好为（7, 4）汉明码。因此二元（7, 4）汉明码也是个循环码，并具有生成多项式$g(x)=(x^3+x+1)$。

4.6 准循环码和截短循环码

在本节中，我们将学习和循环码紧密相联的一些编码。

定义4.6 一个(n, k)**准循环码**是一个线性分组编码需要满足条件：对和n互素的某个m，如果$c(x)$是一个有效的码字多项式，则多项式$x^m c(x)$对(x^n-1)取模也是一个码字多项式。

例4.16 考虑由以下生成矩阵给出的一个(12, 4)准循环码

$$G=\begin{bmatrix}1&1&1&0&1&1&0&0&0&0&0&1\\0&0&1&1&1&1&0&1&1&0&0&0\\0&0&0&0&0&1&1&1&1&0&1&1\\0&1&1&0&0&0&0&0&1&1&1&1\end{bmatrix}$$

此时，m=3，因为对G中任何行进行循环移3位就能产生此矩阵的另外一列。

通过丢弃每次的第m个符号（m是n的一个因子），任何循环码均可以成为一个准循环码。如果所丢弃的符号都不是校验符号，则此准循环码是一个截短的码。这就意味着所有丢弃的符号都是数据符号，而这些符号都可以首先被设置为0。

定义4.7 一个(n, k)线性码可以成为一个**截短循环码**，如果它是从$(n+m, k+m)$循环码中通过删除m个连续的位置而得到的。

截短码中未使用的符号通常被设置为0，并且不被传输。接收端在解码之前重插入这些0。因此，虽然这些码不再是循环码，但如果这些0被正确对待后，同样的编码器和译码器仍然可以被使用。

例4.17 考虑一个(7, 4)汉明码（它也是一个循环码）。我们可以通过如下方法得到一个截短的(5, 3)码：码字中的前两位被设置为0，随后紧接着两个信息比特和同样的三个奇偶校验比特。此截短码的最小距离是3。因此，它是一个单个错误的校正码。

4.7 突发错误纠错

在许多现实生活的信道中，错误不是随机的，而是突发的。例如在一个移动通信信道中，信号衰退导致突发错误。当错误连续发生时，我们称它们为**突发错误**（burst error）。

例4.18 设要通过一个无线信道以10Kbit/s的速度传输的比特序列为

$$c=0100011101010000101101$$

假设在开始传输后的0.5ms后信道经历了持续1ms的衰退，在这段时间内信道把传输的比特搞

乱了。错误序列可以写为

$$\boldsymbol{b}=00000110110111100000000$$

这是一个突发错误的例子，这里传输序列出错部分是由信道造成的。这里突发的长度为10比特。但是，并不是所有10个位置都发生了错误。

定义4.8　一个长为t的**循环突发**是一个向量，它的非零分量在t个连续位置，第一个和最后一个分量不为零。

如果我们在为易发生长度为t的突发性错误（不同于任意t个随机错误的模式）的信道构造码，一定可以设计出更有效的码。我们可以把一个突发错误描述为

$$e(x)=x^i\cdot b(x) \tag{4-13}$$

其中$e(x)$是次数小于等于$t-1$的多项式，$b(x)$是突发模式。x^i标记了当码字开始发送以后突发错误模式的起始位置。

一个为纠突发错误设计的码必须对每一个错误模式有惟一的伴随式，即

$$s(x)=R_{g(x)}[e(x)] \tag{4-14}$$

对每一个表示长度为t的突发的多项式都不同。

例4.19　对一个分组长度$n=15$的二元码，考虑生成多项式

$$g(x)=x^6+x^3+x^2+x+1$$

这个码有能力纠长度不大于3的突发错。要证明这一点必须证明对应于不同突发错误的所有伴随式都不同。不同的突发错误如下：

(1) 长度为1的突发

$$e(x)=x^i,\ i=0,1,\cdots,14$$

(2) 长度为2的突发

$$e(x)=x^i\cdot(1+x),\ i=0,1,\cdots,13$$

(3) 长度为3的突发

$$e(x)=x^i\cdot(1+x+x^2),\ i=0,1,\cdots,12\text{和}e(x)=x^i\cdot(1+x^2),\ i=0,1,\cdots,12$$

可以证明，所有这55（15+14+13+13）个错误模式的伴随式都不相同。可以创建一个错误模式和对应伴随式的表，这个表可用来纠长度不大于3的突发错误。应该强调的是为纠突发错误专门设计的码具有更好的码率。这里讨论的是一个（15, 9）循环码，码率为$k/n=0.6$，最小距离$d^*=3$。这个码只能纠1个随机错误（但可以纠3个突发错误）。注意纠一个随机错相当于纠长度为1的突发错误。

类似于第3章中学习的Singleton界，对纠突发错误的线性纠错码所需的最小奇偶校验比特数也有一个界："一个能纠所有长度不大于t的突发性错误的线性分组码必须有至少$2t$个奇偶校验符号。"

在接下来的三节中，我们将学习三个不同的循环码的子类，每个子类都有一个特殊目的。

4.8 Fire码

定义4.9 Fire码是$GF(q)$上纠突发错误的循环码，它的生成多项式为

$$g(x)=(x^{2t-1}-1)p(x) \tag{4-15}$$

其中$p(x)$是$GF(q)$上的一个素多项式，它的次数m不小于t而且$p(x)$不整除$x^{2t-1}-1$。Fire码的分组长度是满足$g(x)$整除x^n-1的最小整数n。Fire码可以纠长度不大于t的所有突发错误。

例4.20 考虑满足$t=m=3$的Fire码。$GF(2)$上的一个次数为3的素多项式为$p(x)=x^3+x+1$，它不整除(x^5-1)。该Fire码的生成多项式为

$$\begin{aligned}g(x)&=(x^5-1)p(x)=(x^5-1)(x^3+x+1)\\&=x^8+x^6+x^5-x^3-x-1\\&=x^8+x^6+x^5+x^3+x+1\end{aligned}$$

$g(x)$的次数为$n-k=8$。分组长度是满足$g(x)$且能整除x^n-1的最小的整数n。经过尝试我们得到$n=35$。因此该Fire码的参数为$(35, 27)$，$g(x)=x^8+x^6+x^5+x^3+x+1$。这个码可以纠长度不大于3的突发错误。该码的码率为0.77，比由$g(x)=x^6+x^3+x^2+x+1$生成的码更有效，后者的码率为0.6。

当增加t时Fire码会变得更有效。二元Fire码的码率（当$m=t$时）对不同的t值如图4-1所示。

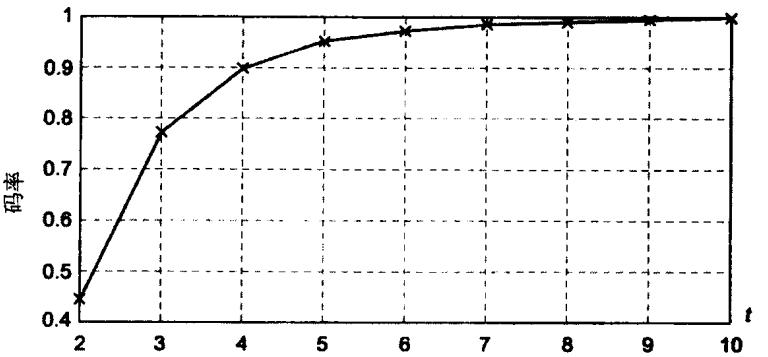


图4-1 不同Fire码的码率

4.9 Golay码

4.9.1 二元Golay码

在3.9节中我们看到（23, 12）完备码是存在的，它的$d^*=7$。对一个完备码，

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}=q^n \tag{4-16}$$

它满足下列一组值：$n=23$，$k=12$，$M=2^k=2^{12}$，$q=2$以及$t=(d^*-1)/2=3$。这个（23, 12）完备码就是二元Golay码。我们现在将把这个完备码当做循环码来研究。我们从分解$(x^{23}-1)$开始。

$$(x^{23}-1) = (x-1)(x^{11}+x^{10}+x^6+x^5+x^4+x^2+1)(x^{11}+x^9+x^7+x^6+x^5+x+1)$$
$$= (x-1)\,g_1(x)\,g_2(x) \tag{4-17}$$

$g_1(x)$ 的次数为$n-k=11$，因此$k=12$，表明存在（23, 12）循环码。为了证明它是完备码，我们必须证明这个（23, 12）循环码的最小距离为7。一种方法是写出奇偶校验矩阵H并证明没有任何6列是线性相关的。另一种方法是用分析法证明，这是一种冗长的证明。最简单的方法是写一个程序列出所有2^{12}个码字并找出最小重量（一台快速计算机需要30秒）。它的码率为0.52，而且它是一个纠三个错误的码。但是这个完备码相对较小的分组长度使得它在很多实际应用中都不实用。

4.9.2 三元Golay码

我们下面检查三元（11, 6）循环码，它也是三元Golay码。这个码的最小距离=5，而且可以验证为一个完备码。我们从在$GF(3)$上分解（$x^{11}-1$）开始。

$$(x^{11}-1) = (x-1)(x^5+x^4-x^3+x^2-1)(x^5-x^3-x^2-x-1)$$
$$= (x-1)\,g_1(x)\,g_2(x) \tag{4-18}$$

$g_1(x)$ 的次数为$n-k=5$，因此$k=6$，表明存在（11, 6）循环码。为了证明它是完备码，我们必须证明这个（11, 6）循环码的最小距离为5。我们再一次用计算机穷举搜索的方法发现它的最小距离确实是5。

可以证明（x^p-1）在$GF(2)$上的分解形式为$(x-1)g_1(x)g_2(x)$，其中p是型为$8m\pm1$（m是一个正整数）的素数。在这种情况下，$g_1(x)$和$g_2(x)$生成等价码。如果由$g_1(x)$所生成的码的最小距离是奇数，那么它满足**平方根界**（square root bound）

$$d^* \geqslant \sqrt{p} \tag{4-19}$$

注意p表示分组长度。

4.10 循环冗余校验（CRC）码

错误检测码中的一种常见的码是循环冗余校验码。对一个k比特块，(n, k) CRC编码器产生（$n-k$）比特长的帧检查序列（FCS）。我们给出如下定义：

$T=n$比特要传输的帧

$D=k$比特消息组（信息比特）

$F=(n-k)$比特FCS，T的最后（$n-k$）比特

$P=$预先确定的除数，（$n-k+1$）比特长的模块

预先确定的除数P应该能整除码字T。因此T/P没有余数。现在D是一个k比特消息组。因此$2^{n-k}D$相当于把k比特向左移位（$n-k$）比特，并在移出的空位添加零（记住一个二元序列向左移动一比特等价于对该二元序列所表示的数乘以2）。码字T可以表示为

$$T = 2^{n-k}D + F \tag{4-20}$$

在上述等式中添加F构成D和F的连接。如果我们用$2^{n-k}D$除以P，可得到

$$\frac{2^{n-k}D}{P} = Q + \frac{R}{P} \tag{4-21}$$

其中Q是商，R/P是余数。假设我们用R作为FCS，则有

$$T=2^{n-k}D+R \tag{4-22}$$

在此情况下，用T除以P我们得到

$$\begin{aligned}\frac{T}{P}&=\frac{2^{n-k}D+R}{P}=\frac{2^{n-k}D}{P}+\frac{R}{P}\\&=Q+\frac{R}{P}+\frac{R}{P}=Q+\frac{R+R}{P}=Q\end{aligned} \tag{4-23}$$

因此没有余数，即T正好被P整除。要生成这样一个FCS，我们只需用$2^{n-k}D$除以P，然后用$(n-k)$比特余数作为FCS即可。

设当T通过有噪信道传输时发生了错误E。接收到的码字为

$$V=T+E \tag{4-24}$$

仅当V完全被P整除时，CRC方案不能检测错误。这就是当E可以完全被P整除时的情况（因为T是可以被P整除的）。

例4.21 设消息$D=1010001101$，即$k=10$且模块为$P=110101$。FCS的比特数为5。因此$n=15$。我们希望确定FCS。

首先，将消息乘以2^5（左移5位并添加5个零），得到

$$2^5D=101000110100000$$

接着把得到的结果除以$P=110101$。通过除法运算我们得到$Q=1101010110$和$R=01110$。将余数加到2^5D中得到

$$T=101000110101110$$

T是要传送的码字。如果信道中不发生错误，接收到的码字除以P时得到的余数为0。

CRC码也可以用多项式表示来定义。设消息多项式为$D(x)$，预先确定的除数为$P(x)$。于是

$$\begin{aligned}\frac{x^{n-k}D(x)}{P(x)}&=Q(x)+\frac{R(x)}{P(x)}\\T(x)&=x^{n-k}D(x)+R(x)\end{aligned} \tag{4-25}$$

在接收端，将收到的码字除以$P(x)$。假设收到的码字是

$$V(x)=T(x)+E(x) \tag{4-26}$$

其中$E(x)$是错误多项式。则有$[T(x)+E(x)]/P(x)=E(x)/P(x)$，这是因为$T(x)/P(x)=0$。那些正巧包含$P(x)$为一个因子的错误将会溜掉，其余的将在CRC译码器网上捕捉到。多项式$P(x)$又称为CRC码的生成多项式。CRC码又称为**多项式码**（Polynomial Code）。

例4.22 假设要传输的码字发生单比特错误。错误多项式$E(x)$可以表示为$E(x)=x^i$，其中i决定错误比特的位置。如果$P(x)$包含两个或两个以上的项，则$E(x)/P(x)$永远都不可能为零。因此CRC码可以捕获所有单个错误。

例4.23 假设发生了两个单独的错误，即$E(x)=x^i+x^j$，$i>j$。我们也可以写成$E(x)=x^j(x^{i-j}+1)$。如果我们假设$P(x)$不能被x整除，那么可以检测所有两个错误的充分条件是对所有不大于$i-j$

（即帧的长度）的值k，$P(x)$ 都不能整除x^k+1。例如对所有小于32 768的k值，$x^{15}+x^{14}+1$都将不整除x^k+1。

例4.24　假设错误多项式有奇数个项（对应于奇数个错误）。一种有趣的现象是在二元算数（即模2运算）下，没有一个含奇数个项的多项式以$x+1$作为一个因子。把（$x+1$）作为$P(x)$ 的一个因子，则可以捕捉到所有由奇数个比特组成的错误（即我们可以捕捉到至少所有可能错误的一半）。

CRC码的另一个有趣的特点是它的检测突发错误的能力。一个长为k的突发错误可以表示为$x^i(x^{k-1}+x^{k-2}+\cdots+1)$，其中$i$决定突发错误在接收到的码字从右端数有多远。如果$P(x)$ 有x^0项，则它不含因子x^i，故若（$x^{k-1}+x^{k-2}+\cdots+1$）的次数小于$P(x)$ 的次数，则余数永远不可能为零。因此一个有r个校验比特的多项式码可以检测到所有长度小于等于r的突发错误。如果突发错误的长度为$r+1$，当且仅当突发错误与$P(x)$ 完全一样时，它除以$P(x)$ 的余数为零。根据定义，一个突发错误的第一个和最后一个比特必须为1，中间的比特可以为1或0。因此突发错误与$P(x)$ 的完全吻合决定于中间的$r-1$个比特。假如所有的组合都等可能，一个遗缺的概率为$1/2^{r-1}$。我们可以证明当长度大于$r+1$的突发错误发生时，或几个短的突发错误发生时，一个坏帧被漏掉的概率为$1/2^r$。

例4.25　有4个版本的$P(x)$ 已变成国际标准：

CRC-12：$P(x)=x^{12}+x^{11}+x^3+x^2+x^1+1$

CRC-16：$P(x)=x^{16}+x^{15}+x^2+1$

CRC-CCITT：$P(x)=x^{16}+x^{15}+x^5+1$

CRC-32：$$P(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1 \tag{4-27}$$

所有这四个多项式都含有（$x+1$）作为一个因子。CRC-12用来传送6比特字符的数据流并生成12比特FCS。CRC-16和CRC-CCITT都对8比特的字符很流行，它们产生16比特的FCS并能捕获所有单个和两个错误、所有奇数比特的错、所有长度小于或等于16的突发错误、99.997%的17比特的突发错误和99.998%的18比特及更长的突发错误。CRC-32用来作为某些点对点**同步传输标准**（Synchronous Transmission Standard）的可选项。

4.11　循环码的电路实现

移位寄存器可以很容易实现循环码的编码和译码。循环码的编码和译码需要多项式的乘法和除法，移位寄存器的移位性质对这些运算是很适合的。移位寄存器就是一些存储单元库，它们能够在每次时钟脉冲时将一个单元的内容移到下一个单元。这里我们将把注意力集中在$GF(2^m)$ 上的码的电路实现。除移位寄存器外，我们还将用到下面的电路元素：

(1) 常量，它的任务就是对输入乘以一个固定的域上的元素。

(2) 加法器，它把两个输入加在一起。加法器的一种简单电路实现就是“异或”门或简称为“xor”。

(3) 乘法器，实际上就是“与”门。

这些元素如图4-2所示。

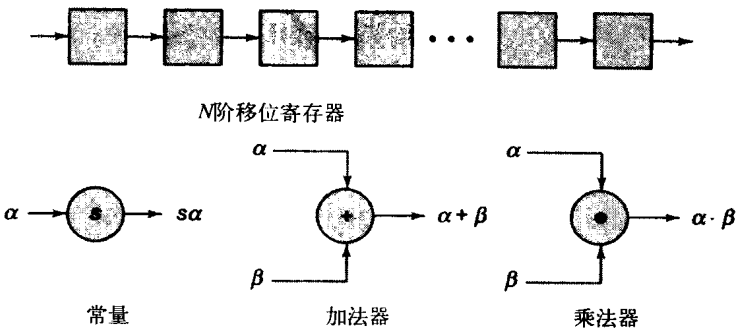


图4-2 用于构造循环码编码器和译码器的电路元素

$GF(2)$ 上的一个域元素可以用单一比特表示。对$GF(2^m)$ 我们需要m比特来表示一个元素。例如$GF(8)$ 上的元素可以表示为{000，001，010，011，100，101，110，111}。对这样的一种表示我们需要3个时钟脉冲才能将移位寄存器的一个阶段的元素移至下一个状态。图4-3给出了$GF(8)$ 的效果移位寄存器。这个域上的任意元素可以表示为ax^2+bx+c，其中a、b和c为二元数，未定元x的幂用来表示位置。例如$101=x^2+1$。

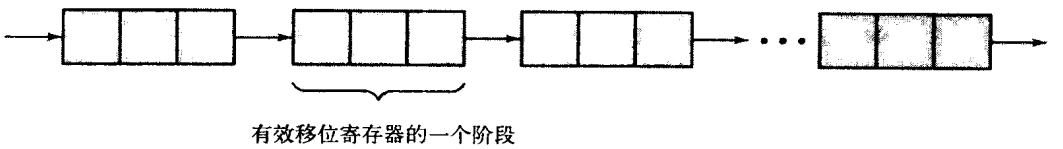


图4-3 $GF(8)$ 的效果移位寄存器

例4.26 我们现在考虑任意元素与$GF(8)$ 上另一域元素的乘法。回顾由$GF(2)$ 构造$GF(8)$ 的过程中我们用到素多项式$p(x)=x^3+x+1$，该域上的元素将为0、1、x、$x+1$、x^2、x^2+1、x^2+x、x^2+x+1。我们想得到任意域元素 (ax^2+bx+c) 和另一元素，比如x^2+x的乘法的电路表示。我们得到

$$(ax^2+bx+c)(x^2+x)=ax^4+(a+b)x^3+(b+c)x^2+cx \quad (\text{modulo } p(x))$$
$$=(a+b+c)x^2+(b+c)x+(a+b)$$

一种可能的电路实现如图4-4所示。

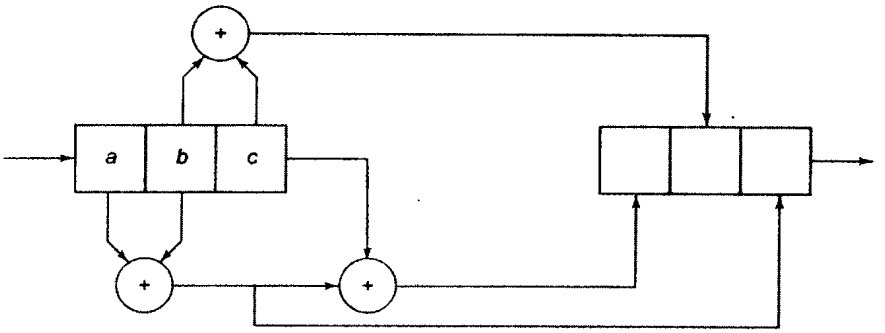


图4-4 $GF(8)$上任意域元素(ax^2+bx+c)与元素(x^2+x)的乘法

我们下面把注意力放在任意多项式$a(x)$ 和$g(x)$ 的乘法。设多项式$g(x)$ 可以表示为

$$g(x)=g_L x^L+\cdots+g_1 x+g_0 \tag{4-28}$$

多项式$a(x)$表示为

$$a(x)=a_k x^k+\cdots+a_1 x+a_0 \tag{4-29}$$

乘积多项式$b(x)=a(x)g(x)$可以表示为

$$b(x)=b_{k+L}x^{k+L}+\cdots+b_1 x+b_0 \tag{4-30}$$

$b(x)$的电路实现如图4-5所示。这是个线性前馈移位寄存器。它也称为**有限脉冲响应**（Finite Impulse Response, FIR）过滤器。

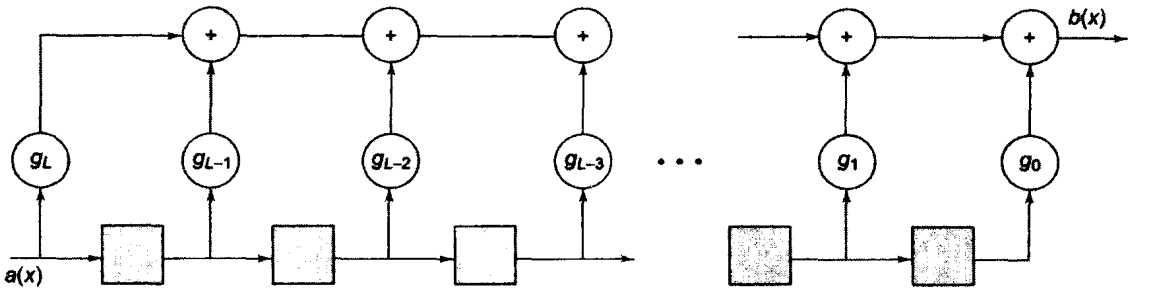


图4-5　两个多项式$b(x)=a(x)\,g(x)$产生的有限脉冲响应过滤器实现

按照电气工程术语，$a(x)$和$g(x)$的系数由移位寄存器做卷积运算。为此，我们有两个多项式乘法的电路实现。因此我们有实现循环码编码的有效方法，即用生成多项式乘以信息多项式。

例4.27　图4-6给出的是下述生成多项式的编码器电路图

$$g(x)=x^8+x^6+x^5+x^3+x+1$$

这是一个Fire码在$t=m=3$时的生成多项式。很容易理解该电路图：8个记忆单元对输入进行移位，每次一个单元。移位输出在适当的位置相加。有5个加法器对6个移位的输入本做加法。

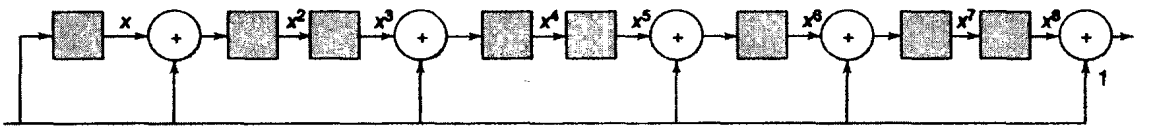


图4-6　Fire码生成多项式$g(x)=x^8+x^6+x^5+x^3+x+1$的编码器的电路实现

我们也可以用移位寄存器电路做任意一个多项式$a(x)$对一个固定多项式$g(x)$的除法。假定除数是首一多项式。我们已经知道怎样除以一个常数将任一多项式转化为首一多项式。除法过程可以用一对递归方程来描述。设$Q^{(r)}(x)$和$R^{(r)}(x)$为在第r步递归的商多项式和余数多项式，其初值条件为$Q^{(0)}(x)=0$，$R^{(0)}(x)=a(x)$。那么递归方程可以写为

$$\begin{aligned}Q^{(r)}(x)&=Q^{(r-1)}(x)+R_{(n-r)}x^{k-r}\\ R^{(r)}(x)&=R^{(r-1)}(x)-R_{(n-r)}x^{k-r}g(x)\end{aligned} \tag{4-31}$$

其中$R_{(n-r)}$表示余数多项式在第（$r-1$）步的最高次项的系数。

对于用一个固定多项式$g(x)$去除任意多项式$a(x)$的情况，电路实现如图4-7所示。经过n次

移位后，商由移位寄存器输出，而留在移位寄存器里的是余数。因此用移位寄存器实现译码器是非常简单的。移位寄存器在用生成多项式除接收到的多项式后所有单元的内容都与零做比较。如果移位寄存器有一个存储单元不为零，也可以检测到错误发生了。

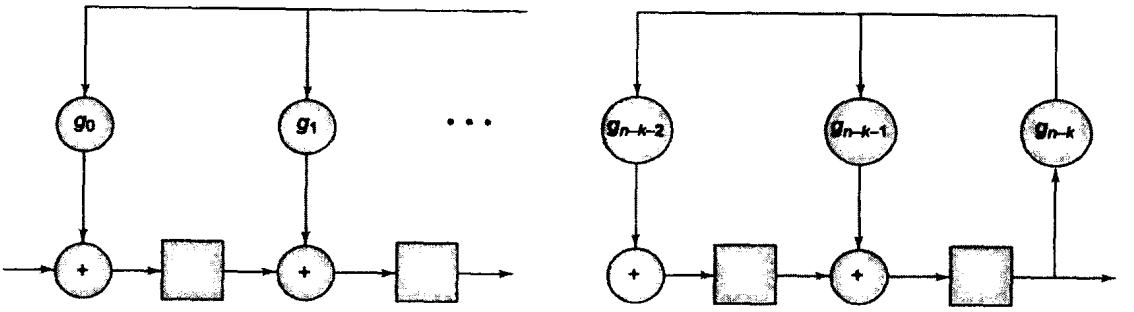


图4-7 用移位寄存器做对$g(x)$的除法

例4.28 对$g(x)=x^8+x^6+x^5+x^3+x+1$做除法的移位寄存器电路图如图4-8所示。

检测和纠错的过程如下：接收到的码字首先储存在一个缓冲器中，等待被$g(x)$去除。正如我们已经看到的，除法可用移位寄存器电路很有效地完成。然后移位寄存器中的余数与所有可能的（预先计算的）伴随式相比较。这个伴随式集合对应于可纠错误模式的集合。如果发现一个伴随式匹配，错误就从接收到的码字中减去。接收到的码字在纠错后移到接收单元的下一步做进一步处理。这种类型的译码器称为**Meggitt译码器**，它的流程图如图4-9所示。

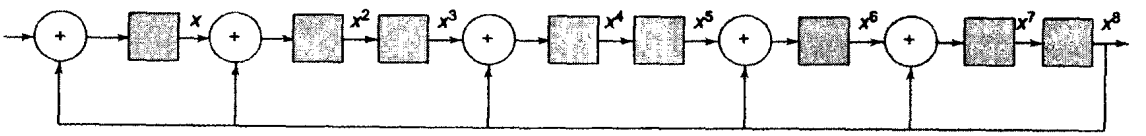


图4-8 对$g(x)=x^8+x^6+x^5+x^3+x+1$做除法的移位寄存器电路图

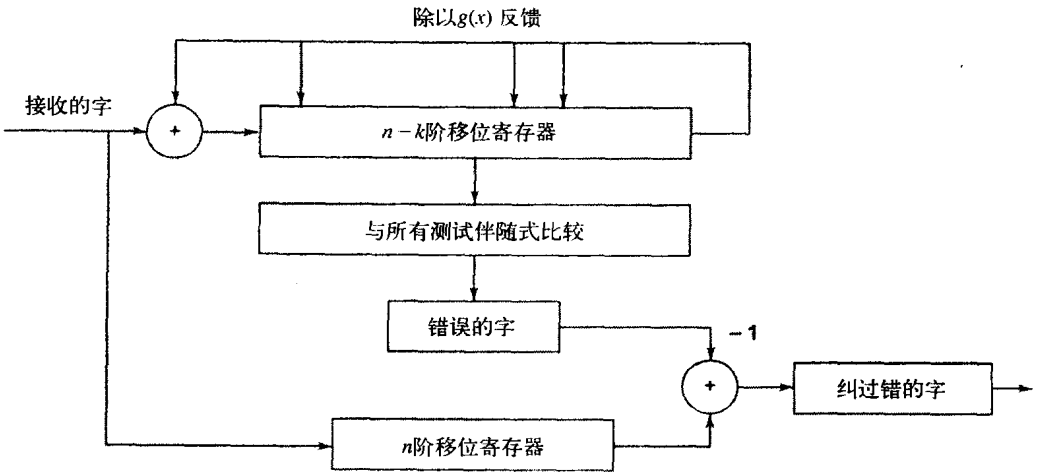


图4-9 Meggitt译码器的流程图

4.12 评注

由于存在下面两个原因，循环码引起了广泛的兴趣：

(1) 多项式环的结构导致的码结构；

(2) 存在大量基于多项式的数学工具，从而可以更好地分析循环码。

循环码的概念首先由Prange在1957年提出，Peterson和Kasami进一步发展了在循环码方面的研究工作。Bose和Raychaudhuri在20世纪60年代初进行了循环码最小距离方面的开创工作。另一类循环码的子类BCH码（是用Bose、Chaudhuri和Hocquenghem的名字命名）将在下一章中详细学习。读者很快就会发现几乎所有早期发现的线性分组码都可以变成循环码。在突发错误纠错方面的最初研究是Abramson在1959年进行的，Fire码也在同一年发表。Golay早在1949年就发表了二元和三元Golay码。

循环码的移位寄存器电路图是Peterson、Chien和Meggitt在20世纪60年代初期著的书中介绍的。Kasami、MacWilliams、Mitchell和Rudolph也在这方面作出了重大贡献。

4.13 小结

- 多项式就是数学表达式$f(x)=f_0+f_1x+\cdots+f_mx^m$，其中符号$x$称为不定元，系数$f_0, f_1, \cdots, f_m$是$GF(q)$中的元素。系数$f_m$称为最高次项的系数。若$f_m\neq 0$，则$m$称为多项式的次数并记为$\deg f(x)$。若一个多项式的最高次项的系数为1，则该多项式称为首一的。
- 除法算法是，对$F[x]$中的任意一对多项式$a(x)$和$b(x)\neq 0$，都存在惟一一对商多项式$q(x)$和余数多项式$r(x)$，使$a(x)=q(x)b(x)+r(x)$成立，其中$\deg r(x)<\deg b(x)$。余数有时候也称为剩余，记为$R_{b(x)}[a(x)]=r(x)$。
- 剩余的两个重要性质为

  (1) $R_{f(x)}[a(x)+b(x)]=R_{f(x)}[a(x)]+R_{f(x)}[b(x)]$

  (2) $R_{f(x)}[a(x)\cdot b(x)]=R_{f(x)}\{R_{f(x)}[a(x)]\cdot R_{f(x)}[b(x)]\}$

  其中$a(x)$、$b(x)$和$f(x)$为$GF(q)$上的多项式。
- $F[x]$中的多项式$f(x)$如果能写为$f(x)=a(x)b(x)$，其中$a(x)$和$b(x)$都是$F[x]$上的元素，且$\deg a(x)$和$\deg b(x)$都小于$\deg f(x)$，则$f(x)$称为可约的。如果$f(x)$不是可约的，则称$f(x)$为既约的。一个次数至少为1的首一既约多项式称为素多项式。
- 环$F[x]/f(x)$是一个域的充分必要条件为$f(x)$是$F[x]$中的一个素多项式。
- R_n上的码$\boldsymbol{C}$是循环码的充分必要条件为$\boldsymbol{C}$满足下列条件：

  (1) $a(x), b(x)\in\boldsymbol{C}\Rightarrow a(x)+b(x)\in\boldsymbol{C}$，

  (2) $a(x)\in\boldsymbol{C}$且$r(x)\in R_n\Rightarrow a(x)r(x)\in\boldsymbol{C}$。
- 下列步骤可用来生成一个循环码：

  (1) 在R_n中取一个多项式$f(x)$。

  (2) 用R_n中的所有可能的多项式与$f(x)$相乘得到一个多项式的集合。

  (3) 上述所得集合对应于一个分组长度为n的循环码码字的集合。
- 设$\boldsymbol{C}$是R_n中的一个(n, k)非零循环码。则

  (1) $\boldsymbol{C}$中存在惟一的次数最小的首一多项式$g(x)$。

  (2) 循环码$\boldsymbol{C}$由生成多项式$g(x)$与所有次数不大于$k-1$的多项式的乘积构成。

  (3) $g(x)$是x^n-1的一个因子。

(4) $g(x)$ 的次数为$n-k$。

- 一个生成多项式$g(x)=g_0+g_1x+\cdots+g_rx^r$次数为$r$的循环码$C$的生成矩阵为

$$\boldsymbol{G}=\begin{bmatrix} g_0 & g_1 & \cdots & g_r & 0 & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_r & 0 & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & g_r & 0 & \cdots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \cdots & g_r \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \downarrow \\ k=(n-r)\text{ 行} \end{matrix}$$

$$\underset{n\text{列}}{\longleftrightarrow}$$

- 一个循环码C的奇偶校验多项式为$h(x)=h_0+h_1x+\cdots+h_kx^k$，则$C$的奇偶校验矩阵为

$$\boldsymbol{H}=\begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & \cdots & h_0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \downarrow \\ (n-k)\text{ rows 行} \end{matrix}$$

$$\underset{n\text{列}}{\longleftrightarrow}$$

- $x^n-1=h(x)g(x)$，其中$g(x)$ 是一个生成多项式，而$h(x)$ 是奇偶校验多项式。
- 一个(n, k)准循环码是一个线性分组编码需要满足条件：对和n互素的某个m，如果$c(x)$是一个有效的码字多项式，则多项式$x^mc(x)$对(x^n-1)取模也是一个码字多项式。
- 一个(n, k)线性码可以成为一个截短循环码，如果它是从$(n+m, k+m)$循环码中通过删除m个连续的位置而得到的。
- Fire码是$GF(q)$ 上的纠突发错误的循环码，它的生成多项式为$g(x)=(x^{2t-1}-1)p(x)$，其中$p(x)$ 是$GF(q)$ 上的一个素多项式，它的次数m不小于t，而且$p(x)$ 不整除$(x^{2t-1}-1)$。Fire码的分组长度是满足$g(x)$ 整除(x^n-1)的最小整数n。一个Fire码可以纠所有长度不大于t的突发错误。
- 二元Golay码的生成多项式为：
  $g_1(x)=(x^{11}+x^{10}+x^6+x^5+x^4+x^2+1)$，或
  $g_2(x)=(x^{11}+x^9+x^7+x^6+x^5+x+1)$
- 三元Golay码的生成多项式为：
  $g_1(x)=(x^5+x^4-x^3+x^2-1)$，或
  $g_2(x)=(x^5-x^3-x^2-x-1)$
- 常见错误检测码之一是循环冗余校验（CRC）码。对一个k比特组，(n, k) CRC编码器生成（$n-k$）比特长的**帧检查序列**（FCS）。
- 用移位寄存器可以很容易实现循环码的编码和译码。循环码的编码和译码需要多项式的乘法和除法，移位寄存器的移位性质很适合处理这种运算。

所有的事情都应该弄得尽量简单，但不可过分简单。

——爱因斯坦（1879—1955）

习题

4.1 下面的哪个码是（a）循环码，（b）与一个循环码等价？

(1) $GF(2)$ 上的 {0000，0110，1100，0011，1001}。

(2) $GF(2)$ 上的 {00000，10110，01101，11011}。

(3) $GF(3)$ 上的 {00000，10110，01101，11011}。

(4) $GF(3)$ 上的 {0000，1122，2211}。

(5) 长度为n的q元重复码。

4.2 为下列定义的多项式环构造加法和乘法表：

(1) 定义在$GF(2)$ 上的$F[x]/(x^2+1)$。

(2) 定义在$GF(3)$ 上的$F[x]/(x^2+1)$。

4.3 列出如下定义的既约多项式：

(1) $GF(2)$ 上次数为$1\sim5$的。

(2) $GF(3)$ 上次数为$1\sim3$的。

4.4 找出所有分组长度为5的二元循环码，求每个码的最小距离。

4.5 设x^n-1是$GF(q)$ 上的r个不同既约多项式的乘积，$GF(q)$ 上存在多少个分组长度为n的循环码？对这些码的最小距离进行评价。

4.6 (1) 在$GF(3)$ 上分解x^8-1。

(2) 存在多少个长为8的三元循环码？

(3) 存在多少个长为8的四元循环码？

4.7 设多项式

$g(x)=x^{10}+x^8+x^5+x^4+x^2+x+1$

为$GF(2)$ 上分组长度为15的一个循环码的生成多项式。

(1) 求生成矩阵$\boldsymbol{G}$。

(2) 求奇偶校验矩阵$\boldsymbol{H}$。

(3) 这个码能检测多少个错误？

(4) 这个码能纠多少个错误？

(5) 将生成矩阵写成系统型。

4.8 考虑多项式

$g(x)=x^6+3x^5+x^4+x^3+2x^2+2x+1$

(1) 这是一个$GF(4)$ 上分组长度为15的一个循环码的有效生成多项式吗？

(2) 求奇偶校验矩阵$\boldsymbol{H}$。

(3) 这个码的最小距离是多少？

(4) 这个码的码率是多少？

(5) 接收到的字$v(x)=x^8+x^5+3x^4+x^3+3x+1$是个正确码字吗？

4.9 R_n中形如x^i+x^{i+1}的错误向量称为双连错误。证明由生成多项式$g_1(x)=(x-1)g_H(x)$ 生成的码能纠双连错，其中$g_H(x)$ 是二元汉明码的生成多项式。

4.10 确定二元（1023，1010）循环码的突发错误纠错长度，此循环码对应的生成多项式为

$$g(x)=x^{13}+x^{10}+x^7+x^6+x^5+x^4+x^2+1$$

4.11 证明Fire码的分组长度是$n=\mathrm{LCM}(2t-1,q^m-1)$。

4.12 证明$g(x)=(x^{11}+1)(x^6+x+1)$是一个Fire码的正确生成多项式。此编码的突发错误纠错

长度是多少?

4.13 证明一个$(n-k)$次生成多项式对应的截短二元循环码可以检测所有长度为$(n-k)+1$的突发模式的$1-2^{-(n-k)+1}$的部分。

4.14 对习题4.8中生成的码设计移位寄存器编码器和Meggitt译码器。

4.15 生成多项式为$g(x)=(x^{23}+1)(x^{17}+x^3+1)$的码在GSM中作检错和纠错标准。

(1) 这个码能纠多少个随机错误?

(2) 这个码能纠多少个突发错误?

上机习题

4.16 写一个程序来计算$GF(q)$上循环码的最小距离,输入为码的生成多项式(或生成矩阵)。

4.17 写一个程序对(35, 27) Fire码进行编码和译码。它应该自动纠正长度不大于3的突发错误。如果一个接收到的码字发生了长度为4的突发错误,你试图对它进行译码会怎样呢?

4.18 编写一个程序用于搜索GF(5)上的循环汉明码。在此Galois域上是否有长度为$n=156$的循环汉明码?

4.19 IEEE 802.16宽带无线接入组(broadband wireless access group)规定了CRC的多项式:

$$g(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

(1) 此编码可以检测的错误模式是什么?

(2) 此编码的突发错误纠错长度是多少?

第5章 BCH码

尽管这看上去是个悖论，但所有精确科学都受近似思想支配。

——Bertrand Russell（1872—1970）

5.1 BCH码简介

Bose-Chaudhuri Hocquenghem（BCH）码是人们所知道的线性循环分组码中功能最强大的一种。BCH码因其对多个错误的纠错能力和简易的编码和译码而著名。到目前为止，我们所做的只是构造一个码，然后计算它的最小距离，从而估计出它的纠错能力。在这类码中，我们将采用另一种方法——先说明我们希望码能纠随机错误的个数，然后构造这样的码的生成多项式。正如上面提到的，BCH码是循环码的一个子类，因此对任意循环码的译码方法也适用于BCH码。但是，已经知道有更有效的BCH码译码过程，本章也将对此进行讨论。

在接下来的几节中，我们先建立必要的数学工具，然后介绍BCH码生成多项式的构造方法。然后讨论对这类码的高效译码技术。BCH码的一个重要子类是Reed-Solomon码，它也将在本章进行讨论。

5.2 基本引理

定义5.1 *若$GF(q)$上的所有元素除零外都可表示为某元素α的幂，则称α为$GF(q)$上的*本原元（Primitive Element）。

例5.1 *考虑$GF(5)$。*因为$q=5$是个素数，模算数可以进行。考虑该域上的元素2，

$$2^0 = 1 \pmod 5 = 1$$
$$2^1 = 2 \pmod 5 = 2$$
$$2^2 = 4 \pmod 5 = 4$$
$$2^3 = 8 \pmod 5 = 3$$

因此，所有$GF(5)$上的非零元素，即{1，2，3，4}都可以表示为2的幂，故2是$GF(5)$上的本原元。

下面考虑元素3。

$$3^0 = 1 \pmod 5 = 1$$
$$3^1 = 3 \pmod 5 = 3$$
$$3^2 = 9 \pmod 5 = 4$$
$$3^3 = 27 \pmod 5 = 2$$

同样，所有$GF(5)$上的非零元素，即{1，2，3，4}都可以表示为3的幂，故3是$GF(5)$上的本原元。

但是可以检验其他非零元素{1，4}都不是本原元。

我们在上例中看到一个域中可以有多于一个的本原元。但是不是可以保证至少有一个本原元呢？答案是肯定的！每个伽罗瓦域的非零元素都构成一个循环群，因此一个伽罗瓦域将包括一个阶数为$q-1$的元素，这个元素就是本原元。本原元在构造域时很有用。一旦我们有了一个本原元，我们可以通过计算该本原元的幂来很容易地找到所有其他元素。

定义5.2 $GF(q)$ 上的一个**本原多项式** (Primitive Polynomial) $p(x)$ 具有下述性质：在模 $p(x)$ 运算下的扩域上，x所表示的元素是本原元。

在任何伽罗瓦域上任意次数的本原多项式都存在。一个本原多项式可用来构造扩域。

例5.2 我们可以用本原多项式$p(x)=x^3+x+1$来构造$GF(8)$。设$GF(8)$上的本原元为$\alpha=z$。则我们可以通过α的幂模$p(\alpha)$得到$GF(8)$上的所有元素，如表5-1所示。

表5-1 $GF(8)$ 中的元素

α的幂	$GF(8)$ 上的元素
α^1	z
α^2	z^2
α^3	$z+1$
α^4	z^2+z
α^5	z^2+z+1
α^6	z^2+1
α^7	1

定理5.1 用$\beta_1, \beta_2, \ldots, \beta_{q-1}$记$GF(q)$上的非零域元素。则

$$x^{q-1}-1=(x-\beta_1)(x-\beta_2)\ldots(x-\beta_{q-1}) \tag{5-1}$$

证明：$GF(q)$上非零元素的集合在乘法运算下构成一个有限群。设β为该域上的任意非零元素，则它可以表示为本原元α的幂。设对某个整数r有$\beta=(\alpha)^r$。则

$$\beta^{q-1}=((\alpha)^r)^{q-1}=((\alpha)^{q-1})^r=(1)^r=1$$

这是因为

$$(\alpha)^{q-1}=1$$

因此

$$\beta是x^{q-1}-1的根。$$

这对任意非零元素β都成立。因此

$$x^{q-1}-1=(x-\beta_1)(x-\beta_2)\ldots(x-\beta_{q-1})$$

例5.3 考虑域$GF(5)$。该域上的非零元素为{1，2，3，4}。故我们可写为

$$x^4-1=(x-1)(x-2)(x-3)(x-4)$$

5.3 极小多项式

在第4章中我们看到为了找到分组长度为n的循环码的生成多项式，我们必须首先分解x^n-1。

因此x^n-1可以写为它的素因子p的乘积

$$x^n-1=f_1(x)\,f_2(x)\,f_3(x)\cdots f_p(x) \tag{5-2}$$

这些因子的任意组合的乘积是一个生成多项式$g(x)$。如果x^n-1的素因子都不同，则有(2^p-2)个非平凡的分组长度为n的循环码。有两个需要忽视的平凡情况，它们是$g(x)=1$和$g(x)=x^n-1$。注意所有(2^p-2)个可能的循环码在最小距离方面都是好码。我们现在逐渐开发一些找到好码的策略，即可期望的最小距离。

在5.2节中我们学习了怎样从一个子域构造扩域。这一节我们将学习在扩域中有根的素多项式（在某些域中）。构造$g(x)$的方法如下：用在扩域中那些期望的根我们将找到在其子域上的本原多项式，将它们乘在一起就得到期望的$g(x)$。

定义5.3　*$GF(q)$上一个码的形如$n=q^m-1$的分组长度n称为*__本原分组长度__（Primitive Block Length）。*$GF(q)$上一个有本原分组长度的循环码称为*__本原循环码__。

域$GF(q^m)$是$GF(q)$的扩域。设本原分组长度为$n=q^m-1$。考虑域$GF(q)$上的分解

$$x^n-1=x^{q^m-1}-1=f_1(x)\,f_2(x)\cdots f_p(x) \tag{5-3}$$

这个分解在扩域$GF(q^m)$上也成立，因为子域上的加法表和乘法表是扩域上加法和乘法表的一部分。我们还知道$g(x)$整除x^n-1，即$x^{q^m-1}-1$，故$g(x)$必定为这些$f_i(x)$中某些多项式的乘积。另外$GF(q^m)$上的每个非零元素都是$x^{q^m-1}-1$的根，因此可以在扩域$GF(q^m)$上分解$x^{q^m-1}-1$得到

$$x^{q^m-1}-1=\prod_j (x-\beta_j) \tag{5-4}$$

其中β_j取遍$GF(q^m)$上的所有非零元素。这表明每个多项式$f_i(x)$在$GF(q^m)$上都可以表示为一些线性项的乘积，而且每一个β_j恰好是某一个$f_i(x)$的根。这个$f_i(x)$称为β_j的极小多项式（Minimal Polynomial）。

定义5.4　*系数定义在基域$GF(q)$上且在扩域$GF(q^m)$上有根β_j的最小次数多项式称为β_j的*__极小多项式__（Minimal Polynomial）。

例5.4　考虑子域$GF(2)$和它的扩域$GF(8)$。这里$q=2$，$m=3$。对$x^{q^m-1}-1$分解（在子域/扩域）得到

$$x^{q^m-1}-1=x^7-1=(x-1)\,(x^3+x+1)\,(x^3+x^2+1)$$

下面考虑扩域$GF(8)$上的元素，这些元素可以表示为0，1，z，$z+1$，z^2，z^2+1，z^2+z，z^2+z+1（根据第4章例4.10）。因此我们可以写为

$$\begin{aligned}x^{q^m-1}-1=\ & x^7-1=(x-1)(x-z)(x-z-1)(x-z^2)(x-z^2-1)(x-z^2-z)(x-z^2-z-1)\\ =\ & (x-1)\cdot[(x-z)(x-z^2)\,(x-z^2-z)]\cdot[(x-z-1)(x-z^2-1)(x-z^2-z-1)]\end{aligned}$$

可以看到在$GF(8)$上有

$$(x^3+x+1)=(x-z)(x-z^2)(x-z^2-z)\text{，且}$$
$$(x^3+x^2+1)=(x-z-1)(x-z^2-1)(x-z^2-z-1)$$

乘法和加法都是在$GF(8)$上进行的。有趣的是用一点点代数方法就可以发现极小多项式的系数都只属于$GF(2)$。我们现在可以依此形成表5-2。

表5-2 $GF(8)$上的元素用本原元α的幂表示

最小多项式$f_i(x)$	$GF(8)$中对应的元素β_j	元素用α的幂表示
$(x-1)$	1	α^0
(x^3+x+1)	z，z^2和z^2+z	α^1，α^2，α^4
(x^3+x^2+1)	$z+1$，z^2+1和z^2+z+1	α^3，α^6，α^5 $(=\alpha^{12})$

可注意到对应到同一个最小多项式的那些元素（用本原元α的幂表示）。我们观察到$\alpha^{12}=\alpha^7\cdot\alpha^5=1\cdot\alpha^5$，可以看到对应某一极小多项式的元素的一种模式。事实上，作为极小多项式根的那些元素在扩域中具有形式$\beta^{q^{r-1}}$，其中β是扩域上的一个元素。在上述例子中，极小多项式$f_2(x)=x^3+x+1$的根为α^1、α^2和α^4，而$f_3(x)=x^3+x^2+1$的根为α^3、α^6和α^{12}。

定义5.5 *若$GF(q^m)$有两个元素在$GF(q)$上具有同一个极小多项式，则称这两个元素在$GF(q)$上*__共轭__（Conjugate）。

例5.5 元素$\{\alpha^1,\ \alpha^2,\ \alpha^4\}$在$GF(2)$上共轭，它们具有同一极小多项式$f_2(x)=x^3+x+1$。

正如我们已经看到的，一个元素在扩域上可能有多于一个的共轭。两个元素的共轭关系取决于基础域。例如扩域$GF(16)$可通过$GF(2)$或$GF(4)$进行构造。两个在$GF(2)$上共轭的元素可能在$GF(4)$上不共轭。

如果$f(x)$是β的极小多项式，那么它也是所有集合$\{\beta,\ \beta^q,\ \beta^{q^2},\ \cdots,\ \beta^{q^{r-1}}\}$中元素的极小多项式，其中$r$是满足$\beta^{q^r}=\beta$的最小整数。集合$\{\beta,\ \beta^q,\ \beta^{q^2},\ \cdots,\ \beta^{q^{r-1}}\}$称为__共轭集合__。共轭集合中的元素为$f(x)$的全部根。因此$\beta$的极小多项式可以写为

$$f(x)=(x-\beta)(x-\beta^q)(x-\beta^{q^2})\cdots(x-\beta^{q^{r-1}}) \tag{5-5}$$

例5.6 考虑$GF(256)$为$GF(2)$的一个扩域。设α为$GF(256)$的本原元。则共轭集合为

$$\{\alpha^1,\ \alpha^2,\ \alpha^4,\ \alpha^8,\ \alpha^{16},\ \alpha^{32},\ \alpha^{64},\ \alpha^{128}\}$$

注意$\alpha^{256}=\alpha^{255}\cdot\alpha^1=\alpha^1$，故共轭集合到$\alpha^{128}$就结束了。$\alpha$的极小多项式为

$$f(x)=(x-\alpha^1)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)(x-\alpha^{16})(x-\alpha^{32})(x-\alpha^{64})(x-\alpha^{128})$$

等式右边在展开后将只含$GF(2)$上的系数。

类似地，α^3的极小多项式为

$$f(x)=(x-\alpha^3)(x-\alpha^6)(x-\alpha^{12})(x-\alpha^{24})(x-\alpha^{48})(x-\alpha^{96})(x-\alpha^{192})(x-\alpha^{129})$$

定义5.6 *定义在$GF(q)$上的分组长度为q^m-1的BCH码称为*__本元BCH码__。

建立了必要的数学工具后，我们现在将开始学习BCH码。我们将开发一种方法，用其构造能纠预先设定的t个随机错误的BCH码的生成多项式。

5.4 极小多项式作为生成多项式

我们知道$g(x)$是x^n-1的一个因子。因此循环码的生成多项式可以写成如下形式

$$g(x)=\mathrm{LCM}\left[f_1(x), f_2(x), \cdots, f_p(x)\right] \tag{5-6}$$

其中$f_1(x), f_2(x), \ldots, f_p(x)$为$g(x)$的根的极小多项式，每一个极小多项式对应于$g(x)$在扩域上的一个根。我们将用这种方法根据生成多项式期望的根来设计好码。

设$c(x)$为一个码字多项式，$e(x)$为错误多项式。则接收到的多项式为

$$v(x)=c(x)+e(x) \tag{5-7}$$

其中多项式的系数都在$GF(q)$上。现在考虑扩域$GF(q^m)$。设$\gamma_1, \gamma_2, \cdots, \gamma_p$为$g(x)$在$GF(q^m)$上的根，即$g(\gamma_i)=0$，$i=1, \cdots, p$。因为对某个多项式$a(x)$，有$c(x)=a(x)g(x)$，我们又得到$c(\gamma_i)=0$，$i=1, \cdots, p$。故

$$\begin{aligned} v(\gamma_i) &= c(\gamma_i)+e(\gamma_i) \\ &= e(\gamma_i),\ i=1, \cdots, p \end{aligned} \tag{5-8}$$

对分组长度n，我们有

$$v(\gamma_i)=\sum_{j=0}^{n-1} e_j \gamma_i^j \quad i=1, \cdots, p \tag{5-9}$$

因此，我们得到包含p个方程的方程组，它们只含有错误模式的分量。如果能求解该方程组得到e_j，便可以精确地确定错误模式。能否解此方程组决定于$g(x)$的根的个数p的值。为了求解错误模式，我们必须小心选取这p个方程。如果我们要设计能纠t个错误的循环码，我们的选择应该是方程组对至多t个非零e_j的情况可以求解。

我们定义伴随式$S_i=e(\gamma_i)$，$i=1, \cdots, p$。我们希望选取$\gamma_1, \gamma_2, \cdots, \gamma_p$使得由$S_1, S_2, \cdots, S_p$可以求解$t$个错误。若$\alpha$是个本原元，则能纠$t$个错误的$\gamma_i$的集合为$\{\alpha^1, \alpha^2, \alpha^3, \cdots, \alpha^{2t}\}$。因此，我们有一种确定能纠$t$个错误的BCH码的生成多项式的简单方法。

确定可纠t个错误的BCH码的生成多项式的步骤：

对一个本原分组长度$n=q^m-1$：

(1) 选取一个次数为m的素多项式并构造$GF(q^m)$。

(2) 求α^i，$i=1, \cdots, p$的极小多项式$f_i(x)$。

(3) 可纠t个错误的码的生成多项式为

$$g(x)=\mathrm{LCM}\left[f_1(x), f_2(x), \cdots, f_{2t}(x)\right] \tag{5-10}$$

用这种方法设计的码至少能纠t个错误。在很多情况下，这些码能纠多于t个错误。因此，

$$d=2t+1 \tag{5-11}$$

称为码的**设计距离**，其最小距离为$d^* \geqslant 2t+1$。它的生成多项式次数等于$n-k$（见定理4.4）。应该注意，一旦确定了n和t，我们便可以确定BCH码的生成多项式。信息长度k是由$g(x)$的次数确定的。从直觉上我们知道，当分组长度n确定后，t的值越大，就迫使信息长度k越小（因为要纠正更多的错误需要更高的冗余度）。在下一节中我们将介绍几个BCH码的具体实例。

5.5 一些BCH码实例

下面的例子展示了如何由$GF(2)$构造扩域$GF(16)$，所得的极小多项式将在接下来的例子

中用到。

例5.7 考虑$GF(2)$上的本原多项式$p(z)=z^4+z+1$，我们将以此来构造扩域$GF(16)$。设$\alpha=z$为本原元。$GF(16)$上以α的幂表示形式的元素及它们对应的极小多项式在表5-3中列出。

表5-3 $GF(16)$上的元素和对应的极小多项式

α的幂	$GF(16)$的元素	极小多项式
α^1	z	x^4+x+1
α^2	z^2	x^4+x+1
α^3	z^3	$x^4+x^3+x^2+x+1$
α^4	$z+1$	x^4+x+1
α^5	z^2+z	x^2+x+1
α^6	z^3+z^2	$x^4+x^3+x^2+x+1$
α^7	z^3+z+1	x^4+x^3+1
α^8	z^2+1	x^4+x+1
α^9	z^3+z	$x^4+x^3+x^2+x+1$
α^{10}	z^2+z+1	x^2+x+1
α^{11}	z^3+z^2+z	x^4+x^3+1
α^{12}	z^3+z^2+z+1	$x^4+x^3+x^2+x+1$
α^{13}	z^3+z^2+1	x^4+x^3+1
α^{14}	z^3+1	x^4+x^3+1
α^{15}	1	$x+1$

例5.8 我们希望确定纠单一错误的BCH码的生成多项式，即$t=1$且$n=15$。由式（5-10）可知，一个BCH码的生成多项式由LCM$[f_1(x), f_2(x), \cdots, f_{2t}(x)]$给出。我们将利用表5-3来获得极小多项式$f_1(x)$和$f_2(x)$。于是纠单一错误的BCH码的生成多项式为

$$\begin{aligned} g(x) &= \text{LCM}\,[f_1(x), f_2(x)] \\ &= \text{LCM}\,[(x^4+x+1), (x^4+x+1)] \\ &= x^4+x+1 \end{aligned}$$

因为$\deg\, g(x)=n-k$，可得$n-k=4$，从而得到$k=11$。于是我们得到纠单一错误的BCH（15, 11）码的生成多项式。该码的设计距离为$d=2t+1=3$。可以计算该码的实际最小距离d^*也是3。因此，在此情况下设计距离等于最小距离。

下面我们希望确定纠两个错误的BCH码的生成多项式，即$t=1$且$n=15$。该BCH码的生成多项式为

$$\begin{aligned} g(x) &= \text{LCM}\,[f_1(x), f_2(x), f_3(x), f_4(x)] \\ &= \text{LCM}\,[(x^4+x+1), (x^4+x+1), (x^4+x^3+x^2+x+1), (x^4+x+1)] \\ &= (x^4+x+1)(x^4+x^3+x^2+x+1) \\ &= x^8+x^7+x^6+x^4+1 \end{aligned}$$

因为$\deg\, g(x)=n-k$，可得$n-k=8$，从而得到$k=7$。于是我们得到纠两个错误的BCH（15, 7）码的生成多项式。该码的设计距离为$d=2t+1=5$。可以计算该码的最小距离d^*也是5。因此，在此情况下设计距离等于最小距离。

下面我们希望确定纠三个错误的二元BCH码的生成多项式。该BCH码的生成多项式为

$$\begin{aligned} g(x) &= \text{LCM}\,[f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)] \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) \\ &= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \end{aligned}$$

在这种情况下deg $g(x) = n - k = 10$，从而得到$k = 5$。于是我们得到纠三个错误的BCH（15, 5）码的生成多项式。该码的设计距离为$d = 2t + 1 = 7$。可以计算该码的最小距离d^*也是7。因此，在此情况下设计距离等于最小距离。

下面我们希望对$t = 4$的情况确定二元BCH码的生成多项式。该BCH码的生成多项式为

$$\begin{aligned} g(x) &= \text{LCM}\,[f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x), f_7(x), f_8(x)] \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x^4 + x^3 + 1) \\ &= x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \end{aligned}$$

在这种情况下deg $g(x) = n - k = 14$，从而得到$k = 1$。可以看出这是一个简单的重复码。该码的设计距离为$d = 2t + 1 = 9$，然而，这个码的最小距离d^*为15。因此，在此情况下设计距离不等于最小距离，码设计得太过度了。这个码实际可纠 $(d^* - 1)/2 = 7$个随机错误！

如果对$t = 5$、6和7的情况重复上述练习，我们会得到同样的生成多项式（重复码）。注意$GF(16)$ 中共有15个非零域元素，因此只有15个对应于这些域元素的极小多项式，故我们不能考虑$t = 7$以外的情况（因为对$t = 8$，我们需要$f_{16}(x)$，它还没有定义呢）。因此要想得到可以纠更多数量错误的BCH码，我们必须使用有更多元素的扩域！

例5.9　我们可以用$GF(4)$ 上的本原多项式$p(z) = z^2 + z + 1$构造$GF(4)$ 的扩域$GF(16)$。设$GF(4)$ 上的元素由集合{0，1，2，3}中的四元符号构成。为了方便查阅，$GF(4)$ 上的加法表和乘法表如表5-4所示。

表5-4　$GF(4)$上的加法表和乘法表

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

表5-5列出了$GF(16)$ 上以α的幂的形式表示的元素及对应的极小多项式。

表5-5　$GF(16)$ 上的元素和对应的极小多项式

α的幂	$GF(16)$ 的元素	极小多项式
α^1	z	x^2+x+2
α^2	$z+2$	x^2+x+3
α^3	$3z+2$	x^2+3x+1
α^4	$z+1$	x^2+x+2
α^5	2	$x+2$
α^6	$2z$	x^2+2x+1
α^7	$2z+3$	x^2+2x+2
α^8	$z+3$	x^2+x+3

（续）

α的幂	GF(16) 的元素	极小多项式
α^9	$2z+2$	x^2+2x+1
α^{10}	3	$x+3$
α^{11}	$3z$	x^2+3x+3
α^{12}	$3z+1$	x^2+3x+1
α^{13}	$2z+1$	x^2+2x+2
α^{14}	$3z+3$	x^2+3x+3
α^{15}	1	$x+1$

对$t=1$，

$$\begin{aligned} g(x) &= \text{LCM}\,[f_1(x), f_2(x)] \\ &= \text{LCM}\,[(x^2+x+2), (x^2+x+3)] \\ &= x^4+x+1 \end{aligned}$$

因为deg $g(x)=n-k$，可得$n-k=4$，从而得到$k=11$。于是我们得到$GF(4)$上纠一个错误的BCH（15, 11）码的生成多项式。它把11个四元信息符号编码成15个四元符号。注意一个四元符号相当于2比特，因此，事实上该（15, 11）BCH码把22个输入比特变成了编码后的30个比特（对于长度为30的二元序列，这个码可用于纠长度为2的突发性错误）。该码的设计距离为$d=2t+1=3$。可以计算该码的最小距离d^*也是3。因此，在此情况下设计距离等于最小距离。

对$t=2$，

$$\begin{aligned} g(x) &= \text{LCM}\,[f_1(x), f_2(x), f_3(x), f_4(x)] \\ &= \text{LCM}\,[(x^2+x+2), (x^2+x+3), (x^2+3x+1), (x^2+x+2)] \\ &= (x^2+x+2)(x^2+x+3)(x^2+3x+1) \\ &= x^6+3x^5+x^4+x^3+2x^2+2x+1 \end{aligned}$$

这是一个$GF(4)$上纠**两个**错误的BCH（15, 9）码的生成多项式。

对$t=3$，

$$\begin{aligned} g(x) &= \text{LCM}\,[f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)] \\ &= \text{LCM}\,[(x^2+x+2), (x^2+x+3), (x^2+3x+1), (x^2+x+2), (x+2), (x^2+2x+1)] \\ &= (x^2+x+2)(x^2+x+3)(x^2+3x+1)(x+2)(x^2+2x+1) \\ &= x^9+3x^8+3x^7+2x^6+x^5+2x^4+x+2 \end{aligned}$$

这是一个$GF(4)$上纠**三个**错误的BCH（15, 6）码的生成多项式。

类似地，对$t=4$，

$$g(x) = x^{11}+x^{10}+2x^8+3x^7+3x^6+x^5+3x^4+x^3+x+3$$

这是一个$GF(4)$上纠**四个**错误的BCH（15, 4）码的生成多项式。

类似地，对$t=5$，

$$g(x) = x^{12}+2x^{11}+3x^{10}+2x^9+2x^8+x^7+3x^6+3x^4+3x^3+x^2+2$$

这是一个$GF(4)$上纠**五个**错误的BCH（15, 3）码的生成多项式。

类似地，对$t=6$，

$$g(x) = x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

这是一个$GF(4)$上纠**六个**错误的BCH（15, 1）码的生成多项式。很显然，这是一个简单重复码，最多可以纠7个错误。

表5-6列出了码长直到2^5-1的二元BCH码的生成多项式。假设我们想构造BCH（15, 7）码的生成多项式，从表中查到（111010001）为该生成多项式的系数，于是

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1$$

表5-6　码长达2^5-1的二元BCH码的生成多项式

n	k	t	生成多项式的系数
7	4	1	1 011
15	11	1	10 011
15	7	2	111 010 001
15	5	3	10 100 110 111
31	26	1	100 101
31	21	2	11 101 101 001
31	16	3	1 000 111 110 101 111
31	11	5	101 100 010 011 011 010 101
31	6	7	11 001 011 011 110 101 000 100 111

5.6 BCH码的译码

到现在，我们学习了在给定要纠的随机错误数后如何构造一个BCH码的生成多项式。根据生成多项式，可以构造出快速的硬件编码器。我们现在把注意力转移到BCH码的译码上。由于BCH码是循环码的一个子类，任何对循环码的标准译码过程都适用于BCH码。但人们已经设计出专门针对BCH码的更高效的算法。我们来讨论Gorenstein-Zierler译码算法，它是首先由Peterson提出的二元译码算法的推广。

我们这里讨论可纠t个错误的BCH码的译码算法。假设BCH码是根据一个域元素α来构造的。考虑错误多项式

$$e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \cdots + e_1x + e_0 \tag{5-12}$$

其中最多有t个系数为非零。假设实际发生了v个错误，其中$0 \leqslant v \leqslant t$。设错误发生在位置$i_1$, $i_2, \cdots, i_v$。则错误多项式可以写为

$$e(x) = e_{i_1}x^{i_1} + e_{i_2}x^{i_2} + \cdots + e_{i_v}x^{i_v} \tag{5-13}$$

其中e_{i_k}为第k个错误的大小。注意我们在考虑一般情况。对二元码，$e_{i_k} = 1$。对于纠错问题，我们必须知道两件事情：

(1) 错误在**哪里发**生了，即错误的位置，

(2) 错误**程度**。

因此，未知量为$i_1, i_2, \cdots, i_v$和$e_{i_1}, e_{i_2}, \cdots, e_{i_v}$，它们分别表明错误发生的位置和程度。伴随式可以通过对接收到的关于α的多项式计算得到：

$$S_1 = v(\alpha) = c(\alpha) + e(\alpha) = e(\alpha)$$
$$= e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \cdots + e_{i_v} x^{i_v} \tag{5-14}$$

下面定义错误程度$Y_k = e_{i_k}$，$k = 1, 2, \cdots, v$和错误位置$X_k = \alpha^{i_k}$，$k = 1, 2, \cdots, v$，其中i_k为第k个错误的位置，X_k是与这个位置相关的域元素。现在伴随式可以写为

$$S_1 = Y_1 X_1 + Y_2 X_2 + \cdots + Y_v X_v \tag{5-15}$$

我们可以用在定义$g(x)$时用到的那些α的幂对接收到的多项式进行评估。对$j = 1, 2, \cdots, 2t$，我们定义伴随式

$$S_j = v(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) \tag{5-16}$$

于是我们得到下面的$2t$个联立方程组，它有v个错误位置未知量$X_1, X_2, \cdots, X_v$和v个错误程度未知量$Y_1, Y_2, \cdots, Y_v$：

$$\begin{aligned} S_1 &= Y_1 X_1 + Y_2 X_2 + \cdots + Y_v X_v \\ S_2 &= Y_1 X_1^2 + Y_2 X_2^2 + \cdots + Y_v X_v^2 \\ &\vdots \qquad\qquad\qquad\qquad \vdots \\ S_{2t} &= Y_1 X_1^{2t} + Y_2 X_2^{2t} + \cdots + Y_v X_v^{2t} \end{aligned} \tag{5-17}$$

下面定义错误定位多项式

$$\Lambda(x) = \Lambda_v x^v + \Lambda_{v-1} x^{v-1} + \cdots + \Lambda_1 x + 1 \tag{5-18}$$

这个多项式的根是错误位置的逆X_k^{-1}，$k = 1, 2, \cdots, v$，即

$$\Lambda(x) = (1 - xX_1)(1 - xX_2) \cdots (1 - xX_v) \tag{5-19}$$

所以，如果我们知道错误定位多项式$\Lambda(x)$的系数，便可以求得错误位置$X_1, X_2, \cdots, X_v$。经过一些代数变换我们得到

$$\Lambda_1 S_{j+v-1} + \Lambda_2 S_{j+v-2} + \cdots + \Lambda_v S_j = -S_{j+v}, j = 1, 2, \cdots, v \tag{5-20}$$

但这仅仅是一些将伴随式和$\Lambda(x)$的系数联系起来的线性方程而已。这些等式可以写为如下的矩阵形式：

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_{v-1} & S_v \\ S_2 & S_3 & \cdots & S_v & S_{v+1} \\ \vdots & & & & \vdots \\ S_v & S_{v+1} & \cdots & S_{2v-2} & S_{2v-1} \end{bmatrix} \begin{bmatrix} \Lambda_v \\ \Lambda_{v-1} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ \vdots \\ -S_{2v} \end{bmatrix} \tag{5-21}$$

错误定位多项式的系数值可以通过对伴随式矩阵求逆而得，这只有当该矩阵为非奇异的情况下才是可能的。可以证明当有v个错误时该矩阵是非奇异的。

BCH码的译码步骤

(1) 作为测试值，令$v = t$，计算伴随矩阵M的行列式。如果行列式的值为零，令$v = t - 1$，再一次计算M的行列式。重复这个过程直到找到这样一个v的值，使伴随矩阵的行列式不为零。这个值v就是实际发生错误的数目。

(2) 求M的逆并计算错误定位多项式$\Lambda(x)$的系数。

(3) 求解$\Lambda(x) = 0$的零点，从中可计算错误位置$X_1, X_2, \cdots, X_v$。如果是二元码，都到此为止(因为错误程度为1)。

(4) 如果不是二元码，回到方程组

$$\begin{aligned} S_1 &= Y_1X_1 + Y_2X_2 + \cdots + Y_vX_v \\ S_2 &= Y_1X_1^2 + Y_2X_2^2 + \cdots + Y_vX_v^2 \\ &\vdots \qquad\qquad\qquad \vdots \\ S_{2t} &= Y_1X_1^{2t} + Y_2X_2^{2t} + \cdots + Y_vX_v^{2t} \end{aligned}$$

因为错误位置未知，这形成了一组$2t$个线性方程。解这些方程构成的方程组就得到错误程度。

通过求$v \times v$阶矩阵的逆来计算Λ_i在计算上成本很高，需要的计算量为v^3的倍数。如果我们需要纠大量的错误（即大的v值），则需要更高效的求解矩阵方程的方法。通过提炼人们发现了很多不同的方法可以在很大程度在减少运算复杂度。可以看到这个$v \times v$阶矩阵在形式上不是任意的，它的与主对角线垂直方向的分量都是相同的，这种性质称为**斜对称**。Berlekamp (1968) 和Massey (1969) 利用这种结构找到了求解该方程组更简单的方法。

求$\Lambda(x)$ 的零点的最简单的方法就是对所有域元素一个接一个地测试。这种穷举搜索方法称为**Chien搜索**。整个BCH译码的过程见图5-1。

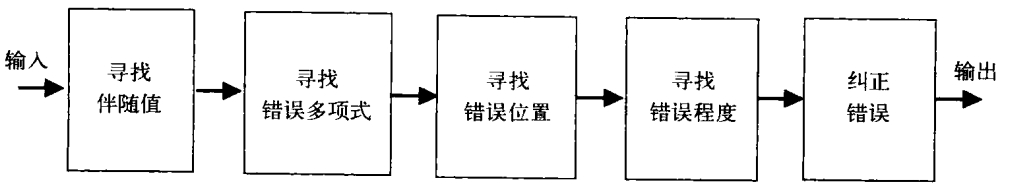


图5-1　BCH解码流程

例5.10　考虑纠三个错误的BCH (15, 5) 码，它的生成多项式为

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

设传输的是全零码字，接收到的多项式为$v(x) = x^5 + x^3$。故有两个错误分别在第4个位置和第6个位置，错误多项式为$e(x) = x^5 + x^3$。但是译码器并不知道这些，它连实际发生了几个错误都不知道。我们用Gorenstein-Zierler译码算法。首先我们用$GF(16)$ 上的算术计算出伴随式：

$$\begin{aligned} S_1 &= \alpha^5 + \alpha^3 = \alpha^{11} \\ S_2 &= \alpha^{10} + \alpha^6 = \alpha^7 \\ S_3 &= \alpha^{15} + \alpha^9 = \alpha^7 \\ S_4 &= \alpha^{20} + \alpha^{12} = \alpha^{14} \\ S_5 &= \alpha^{25} + \alpha^{15} = \alpha^5 \\ S_6 &= \alpha^{30} + \alpha^{18} = \alpha^{14} \end{aligned}$$

因为这是个纠三个错误的码，首先令$v = t = 3$。

$$\boldsymbol{M} = \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} = \begin{bmatrix} \alpha^{11} & \alpha^7 & \alpha^7 \\ \alpha^7 & \alpha^7 & \alpha^{14} \\ \alpha^7 & a^{14} & \alpha^5 \end{bmatrix}$$

$\mathrm{Det}(\boldsymbol{M}) = 0$，这表明发生的错误数少于三个。下面令$v = t = 2$，

$$\boldsymbol{M}=\begin{bmatrix}S_1 & S_2\\ S_2 & S_3\end{bmatrix}=\begin{bmatrix}\alpha^{11} & \alpha^{7}\\ \alpha^{7} & \alpha^{7}\end{bmatrix}$$

Det(M)=1，这表明实际发生了2个错误。我们下面计算M^{-1}。正巧这里有

$$\boldsymbol{M}^{-1}=\begin{bmatrix}\alpha^{11} & \alpha^{7}\\ \alpha^{7} & \alpha^{7}\end{bmatrix}$$

$$\begin{bmatrix}\Lambda_2\\ \Lambda_1\end{bmatrix}=\boldsymbol{M}^{-1}\begin{bmatrix}-S_3\\ -S_4\end{bmatrix}=\begin{bmatrix}\alpha^{11} & \alpha^{7}\\ \alpha^{7} & \alpha^{7}\end{bmatrix}\begin{bmatrix}\alpha^{7}\\ \alpha^{14}\end{bmatrix}$$

求解Λ_1和Λ_2可得$\Lambda_2=\alpha^8$和$\Lambda_1=\alpha^{11}$。从而

$$\Lambda(x)=\alpha^8x^2+\alpha^{11}x+1=(\alpha^5x+1)(\alpha^3x+1)$$

因此恢复出来的错误位置为α^5和α^3。因为该码是二元码，错误程度为1，故$e(x)=x^5+x^3$。

在下一节中我们将学习著名的Reed-Solomon码，一种BCH码的重要子类。

5.7 Reed-Solomon码

Reed-Solomon（RS）码是一种广泛应用于数字通信和数据储存的非二元BCH码的重要子类。RS码典型的应用领域为：

- 储存器件（包括磁带、CD、DVD、条形码等）。
- 无线或移动通信（包括移动电话、微波连接等）。
- 卫星通信。
- 数码电视或数码视频广播（DVB）。
- 高速调制解调器，比如那些用到ADSL、xDSL等的调制解调器。

这些都开始于1960年发表在*Journal of the Society for Industrial and Applied Mathematics*上的一篇只有5页的论文。由MIT Lincoln实验室的Irving S. Reed 和 Gustave Solomon在论文“Polynomial Codes over Certain Finite Fields”（某些有限域上的多项式码）引入的思想成为从计算机硬盘到CD播放器的所有设备的纠错技术的重要部分。Reed-Solomon码（当然连同许多工程技术）使得旅行者2号（Voyager Ⅱ）把外星球绝妙照片传回地球成为可能。它们使得用有划痕的CD听音乐也成为可能。而且在不远的将来，它们将使有线电视运营商们把500多个频道挤进他们的系统中去。

RS码的编码系统建立在比特组基础上，即字节，而不是单个的0和1，这使得它处理突发错误特别好：比如说6个连续的比特错误至多影响到两个字节，因此即使纠两个错误的Reed-Solomon码也能提供合适的安全因子。当前在CD技术上实现的Reed-Solomon码能处理长度可达到4000相继比特位的突发错误。

在这类BCH码的子类中，符号域$GF(q)$ 和错误定位域$GF(q^m)$ 相同，即$m=1$。因此在这种情况下有

$$n=q^m-1=q-1 \tag{5-22}$$

在相同的域$GF(q)$ 上任意元素β的极小多项式为

$$f_\beta(x)=x-\beta \tag{5-23}$$

因为符号域（子域）和错误定位域（扩域）是同一个，所有的极小多项式都是线性的。能纠t个错误的码的极小多项式为

$$\begin{aligned} g(x) &= \mathrm{LCM}\left[f_1(x), f_2(x), \cdots, f_{2t}(x)\right] \\ &= (x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{2t-1})(x-\alpha^{2t}) \end{aligned} \tag{5-24}$$

因此，生成多项式的次数将总是$2t$。故RS码满足

$$n-k=2t \tag{5-25}$$

一般地，一个RS码的生成多项式可以写为

$$g(x)=(x-\alpha^{i})(x-\alpha^{i+1})\cdots(x-\alpha^{2t+i-1})(x-\alpha^{2t+i}) \tag{5-26}$$

例5.11　考虑$GF(16)$上分组长度为15的纠两个错误的RS码，这里$t=2$。我们这里用到扩域$GF(16)$上的元素，该扩域由$GF(2)$以本原多项式$p(z)=z^4+z+1$构造。则码的生成多项式可以写为

$$\begin{aligned} g(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) \\ &= x^4+(z^3+z^2+1)x^3+(z^3+z^2)x^2+z^3x+(z^2+z+1) \\ &= x^4+(\alpha^{13})x^3+(\alpha^6)x^2+(\alpha^3)x+\alpha^{10} \end{aligned}$$

这里$n-k=4$，可得到$k=11$。因此我们得到了$GF(16)$上RS（15, 11）码的生成多项式。注意这个编码方案把11个符号（等价于$4\times11=44$bit）编码成15个符号（相当于60比特）。

定理5.2　Reed-Solomon码是最大距离可分码（MDS），它的最小距离为$n-k+1$。

证明：设RS码的设计距离为$d=2t+1$。最小距离d^*满足条件

$$d^*\geqslant d=2t+1$$

但对于一个RS码，有$2t=n-k$，从而

$$d^*\geqslant d=2t+1=n-k+1$$

但由任意线性码的Singleton界可知

$$d^*\leqslant n-k+1$$

故$d^*=n-k+1$，而且最小距离等于码的设计距离，即$d^*=d$。因为RS码为最大距离可分码，所有的码字都在码空间中在代数上离得尽可能地远。这表明码字在码空间中是均匀分布的。

表5-7列出了某些RS码的参数。注意对一个给定的最小距离，为了得到高码率，必须在高阶伽罗瓦域上进行。

表5-7　某些RS码参数

m	$q=2^m$	$n=q-1$	t	k	d^*	$r=k/n$
2	4	3	1	1	3	0.3333
3	8	7	1	5	3	0.7143
			2	3	5	0.4286
			3	1	7	0.1429
4	16	15	1	13	3	0.8667
			2	11	5	0.7333
			3	9	7	0.6000

（续）

m	$q=2^m$	$n=q-1$	t	k	d^*	$r=k/n$
			4	7	9	0.4667
			5	5	11	0.3333
			6	3	13	0.2000
			7	1	15	0.0667
5	32	31	1	29	3	0.9355
			5	21	11	0.6774
			8	15	17	0.4839
8	256	255	5	245	11	0.9608
			15	225	31	0.8824
			50	155	101	0.6078

例5.12 一种很流行的Reed-Solomon码是有8比特符号（字节），即$GF(256)$上的RS（255，223）码。每一个码字含有255个码字字节，其中233个字节是数据，32个字节为奇偶校验。对这个码，$n=255$，$k=223$，从而$n-k=32$。因此$2t=32$或$t=16$。译码器可以纠码字中的任意16个随机符号错误，即在码字的任何地方最多有16个字节的错误。

例5.13 Reed-Solomon纠错码对数字通信信道的效率有极其显著的作用。例如以每秒一百万字节传输的数据可以每秒处理大约4000个255字节的组。如果在信道中每秒注入1000个随机的短错误（长度小于17 bit），则每秒钟将有600～800个组被破坏，从而可能需要重新传输几乎所有的组。应用Reed-Solomon（255，235）码（它可以在每个235个信息字节及20个奇偶校验字节的组中纠最多10个错误）后，接收到不能纠错而需要重新传输的组的时间间隔约为800年。发生译码错误的平均时间间隔为2千万年以上！

5.8 Reed-Solomon码编码器和译码器的实现

5.8.1 硬件实现

把RS编码器的生成多项式表示为

$$g(x)=g_0+g_1x+g_2x^2+\cdots+g_{2t-1}x^{2t-1}+x^{2t} \tag{5-27}$$

注意方程右边x的最高次项在最右边。RS编码器的硬件实现如图5-2所示。具体编码步骤如下。

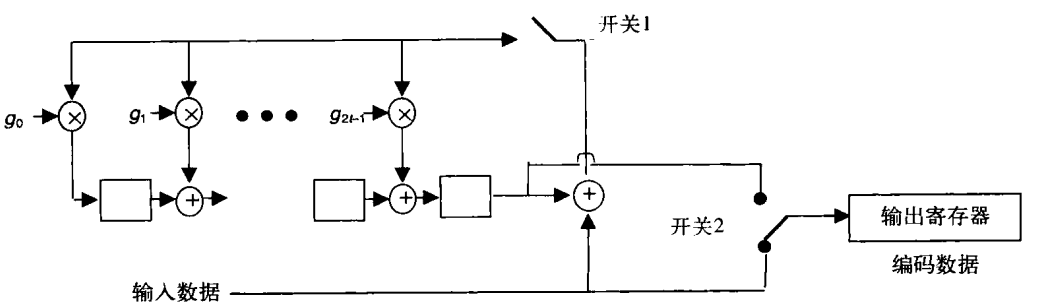


图5-2 生成多项式为$g(x)=g_0+g_1x+g_2x^2+\cdots+g_{2t-1}x^{2t-1}+x^{2t}$的RS编码器的硬件实现

(1) 在前k个时钟周期中，将开关1闭合，把消息符号移至$(n-k)$级移位寄存器。注意在这k个时钟周期中，根据移入的信息符号以及移位寄存器每个单元前面的加法器，所以移位寄存器的内容和反馈回路是不断改变的。

(2) 在前k个时钟周期中，开关2处于闭合状态，从而使得消息符号可以被同时直接送入输出寄存器中。注意，这是一个系统化的设计，码字的前k个符号是原始信息符号的。

(3) 当k个信息符号被送入输出寄存器后，将开关1开启，开关2置为开启状态。在移位寄存器中保留的内容都是校验位。这些符号将被移出，并跟在原先k个信息符号之后，从而成为一个完整的码字。

(4) 在剩下的$(n-k)$个时钟周期中，把移位寄存器中的校验符号转移到输出寄存器中，移位寄存器全部清零。

(5) 总时间周期是n，输出寄存器中的内容是最终对应于k个信息符号的码字多项式。

例5.14　IEEE 802.15.4a使用在GF(2^6)上的RS(63, 55)编码器。此时，$n-k = 63-55 = 8 = 2t$。编码器的生成多项式可以表示为

$$\begin{aligned} g(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)(x-\alpha^7)(x-\alpha^8) \\ &= x^8 + \alpha^{43}x^7 + \alpha^{59}x^6 + \alpha^{31}x^5 + \alpha^{10}x^4 + \alpha^{40}x^3 + \alpha^{14}x^2 + \alpha^7 x + \alpha^{36} \end{aligned} \tag{5-28}$$

编码器的移位寄存器部分如图5-3所示。

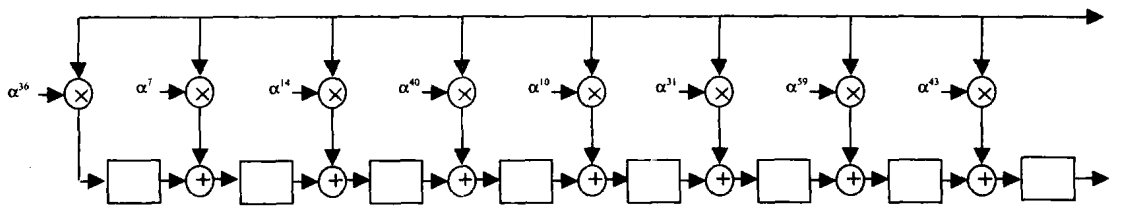


图5-3　RS(63, 55)编码器中移位寄存器部分

RS码的译码过程参见图5-1。每一个子分组，也就是伴随式计算模块分组，差错定位分组以及差错幅度分组均可以使用FPGA实现。由于所有的算术都在GF(q)上进行，因此加法、乘法、求逆可以查找表格。

存在许多RS码的商业硬件实现。许多已有的系统用现货供应的集成电路来对Reed-Solomon码进行编码和译码。这些集成电路逐渐支持一定程度的程式化功能，例如RS (255, k)码，其中t可以为1～16个符号。最近的趋势是朝着VHDL或Verilog设计（逻辑核或知识产权核）发展。它们有一些比标准集成电路优越的方面。一个逻辑核可以同其他VHDL或Verilog部件集成，并综合到一个FPGA（域可编程门阵列）或ASIC（特定应用集成电路）中——这使得所谓的“芯片上的系统”设计成为可能，其中多种模块可以结合到单一的集成电路块中。根据产品要求不同，逻辑核通常比标准的集成电路造价要低得多。采用逻辑核后，设计者避免了不断购买Reed-Solomon集成电路。

5.8.2　软件实现

直到最近，对Reed-Solomon码的“实时”软件实现除最简单的情况（即那些t值很小的码）外，都需要太大的运算量。软件实现Reed-Solomon码的最大困难是通用处理器不支持伽罗瓦

域上的算术运算。例如软件实现伽罗瓦域上的乘法需要测试0、两个日志表、模加和反日志表。但是仔细设计以及处理器性能的提高表明软件实现可以达到相对较高的数据速率。表5-8给出了在1.6GHz奔腾PC上的一些样本数字。这些数据速率只是译码的。编码要快一些，因为它需要较少的运算。

表5-8 RS码译码软件译码的一些样本数据

码	数据速率	t
RS(255, 251)	~120 Mbps	2
RS(255, 239)	~30 Mbps	8
RS(255, 223)	~10 Mbps	16

5.9 实信道上RS码性能

读者可能认为当我们降低RS码的码率时，BER性能可能得到提高。然而，在实信道上，调制方案也会对BER有所影响。因此，我们必须同时考虑调制和编码机制。其中一个机制会提升误差性能，而另外一个会降低误差性能。提升的那个机制是编码机制。冗余度越高，码的纠错能力就越强。降低的那个机制是由于冗余度增加（以及在实时通信系统中更加快速的信令）导致的每个信道符号（和数据符号相比）的能力下降。下降的符号能量导致解调产生很多错误。最终，第二种机制赢得胜利，因此在一个非常低的码率下，系统经历误差的性能降低。

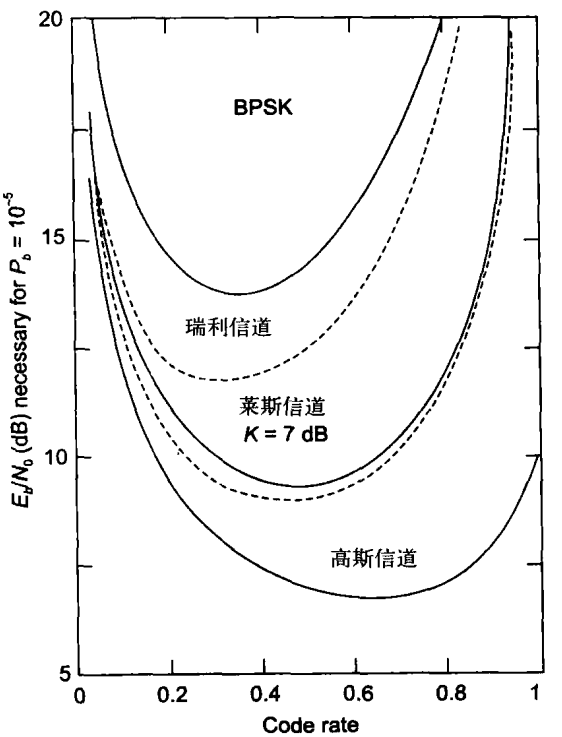


图5-4 使用BPSK的RS(31, k)解码器相对于码率的性能函数

图5-4展示了同时考虑差错控制编码和调制的实时通信系统的性能。考虑一个使用BPSK调制方案的RS(31, k)码。此时，为进行纠错编码而付出的代价是以等于码率的逆的因子而产生的带宽扩张。图中曲线清楚地显示了最小化所要求的E_b / N_0的最优码率。很容易看出，对于高斯信道，最优码率大概是0.6～0.7；对于莱斯衰减信道（直接的和反射的收到信号功率比K=7 dB），最优码率大概是0.5；对于瑞利衰落信道，最优码是0.3。

在实际环境中，通信系统的能量消耗也成为一个非常重要的设计限制，例如，无线传感器网络。如果我们考虑差错控制编码分组和调制分组的硬件实现，能量花费可以分成以下几种：

（1）由于信道编码和解码而需要的计算能量；

（2）由于调制和解调而需要的电路能量；

（3）由于传输冗余位而需要的信令能量（无线能量）。

短距离之间，收发器电路以及计算所花费的能量和信令能量差不多。因此，设计策略是最小化在编码器译码器分组和调制解调分机中所有的能量消耗，并同时考虑发射信令能量。差错控制码用来达到需要的比特错误率，并降低由于编码增益而产生的信令能量。然而，更多的能量花费是由于传输冗余比特以及编码和译码。因此，对于某一种编码调制配置，能量花费可以是最优的。此最优依赖于发射器和接收器之间的距离，因为这将影响信令（无线）能量。

例5.15　图5-5展示了不同的RS(31, k)编码器和BPSK调制（S Choubin 等人）为得到BER=10^{-5}而需要的能量。图中每一个数据比特的总能量对应一个纠错能力t。编码运行在StrongArm SA1100处理器上，发射器和接收器相隔110米，并且路径损耗指数等于4。所有的计算能量是编码和译码能量的总和。从图中可以看出，在此处理器上，RS(31, 20)达到了最小的能量。

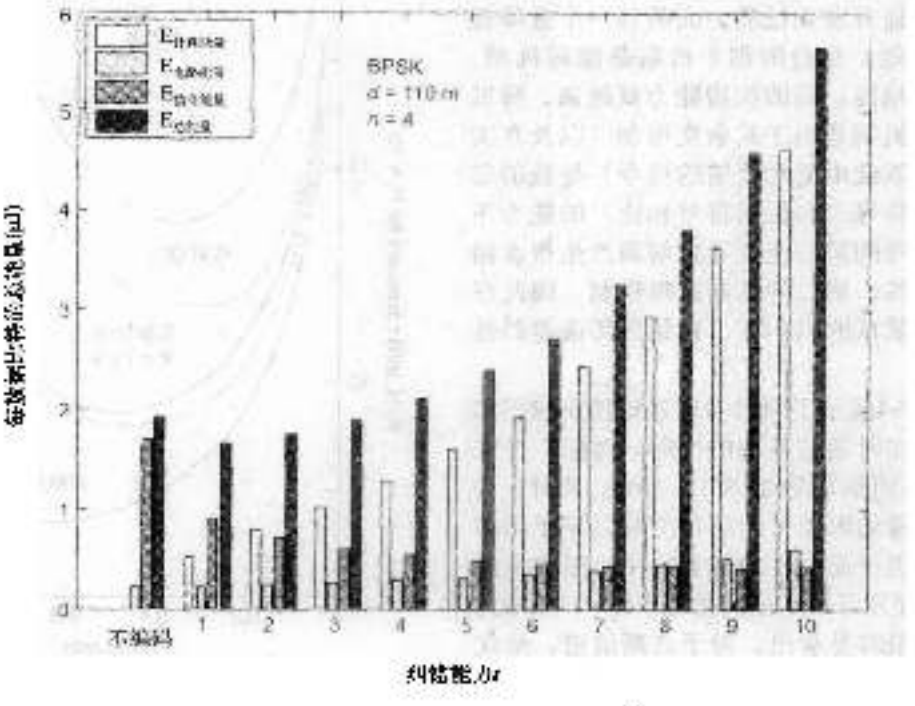


图5-5　RS(31,k)的总能量和纠错能力的对比

为达到最小的能量配置，一个非常有意思的问题是如何发掘码字长度（分组长度n），以及其与纠错能力t之间的关系。图5-6给出了在StrongArm SA1100处理器上，发射器和接收器相隔125米，使用BPSK调制的情况下的模拟结果。其中一个场景，即RS(127, 121, 7)达到了最小的能量。

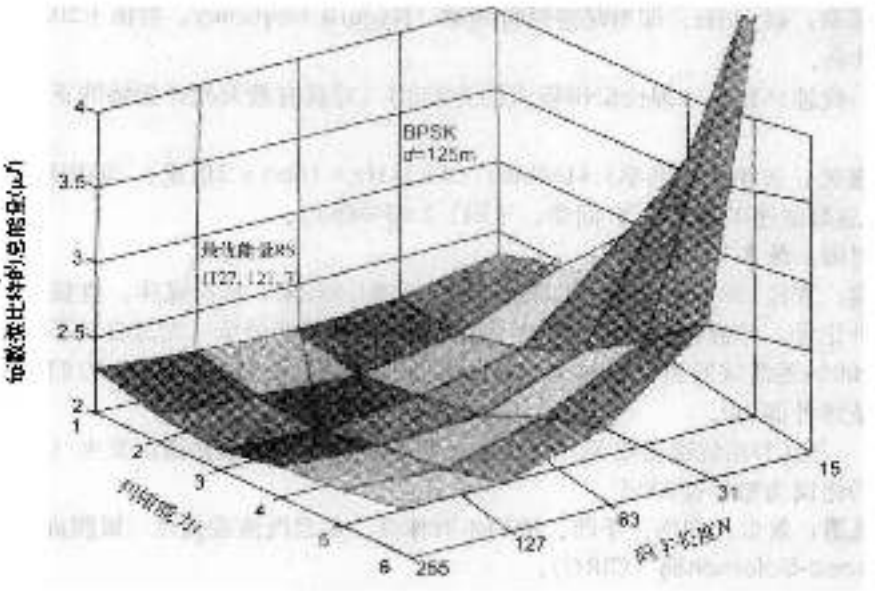


图5-6 总能量与码长及纠错能力的对比

5.10 嵌套码

要得到分组长度很大的码的方法之一就是将码嵌套起来。这种技术结合了一个具有小字母表的码和一个大字母表的码。设一个q元符号的长度为kK。这个组可以分段为K个k符号子组，每个子组可以看成是q^k元字母表中的一个元素。一个由K个这样的子组构成的序列可以用$GF(q^k)$上的(N, K)码进行编码。现在，N个q^k元符号中的每一个可以看做k个q元符号，进而可用q元(n, k)码进行编码。因此嵌套码有两个不同编码层。这种产生一个嵌套码的方法由图5-7给出。

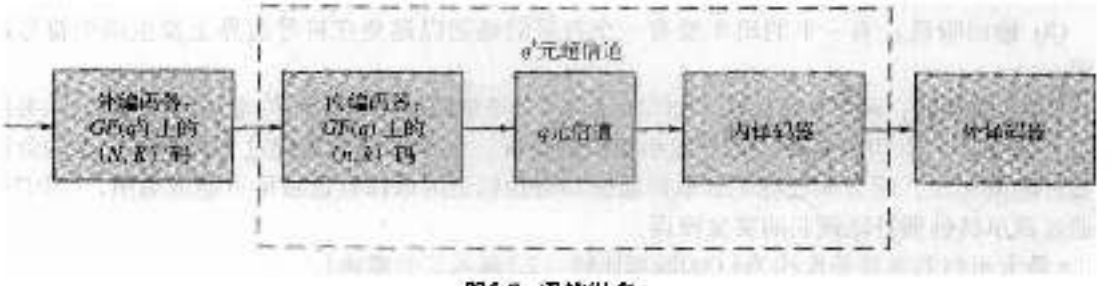


图5-7 码的嵌套

例5.16 下面的两个码可以通过嵌套形成一个有更大分组长度的码。

内码：$GF(8)$上纠两个错误的RS $(7, 3)$码。

外码：$GF(8^3)$上纠三个错误的RS $(511, 505)$码。

将这两个码嵌套后可得$GF(8)$上的$(3577, 1515)$码。这个码可以纠任意11个随机错误，码字长度为3577个符号，符号为$GF(8)$上的元素。

例5.17 RS码大量用于CD纠错。下面我们给出标准CD数据格式。

采样频率：44.1kHz，即对尼奎斯特频率（Nyquist frequency，指低于20kHz的听觉频率）有10%的富余。

量化：线性16比特⇒理论SNR值大约为98dB（对具有最大允许振幅的正弦曲线信号），2的补码。

信号格式：音频比特速率1.41Mbit/s（44.1kHz × 16bit × 2信道），编织Reed-Solomon码（CIRC），总数据速率（CIRC，同步，子码）2.034Mbit/s。

播放时间：最多74.7分钟。

盘规格：直径120毫米，厚度1.2毫米，磁轨槽1.6微米，单面媒体，盘按顺时针转动，信号由里向外记录，常数线性速度（CLV），以最大记录密度记录（光盘自转不是匀速的，它将从每分钟500转逐渐减低到每分钟200转），槽宽0.5微米，槽边界为1，而它们之间的区域，不管槽内还是槽外都为0。

纠错：一般CD系统错误率为10^{-5}，这表明每秒钟有20次数据错误发生（比特率 × BER）。大约可纠的错误为每秒钟200个。

错误来源：灰尘、划伤、手印、槽的不对称性、底层汽泡或疵点、镀膜疵点或次品。

编织Reed-Solomon码（CIRC）：

- C2可有效纠突发错误。
- C1可纠随机错误并检测突发错误。
- 数据在放入光盘之前要经历三个编织阶段。
- 交叉编织使奇偶校验能纠突发错误。

(1) 输入阶段：每个输入帧12个字（16比特，每信道6个字）的数据分成24个8比特的符号。

(2) C2 Reed-Solomon码：24个符号的数据通过一个（28, 24）RS码进行编码，4个奇偶校验符号用于纠错。

(3) 交叉编织：防范发生突发错误和分离纠错码，一个码可以检查另一个码的准确性，加强纠错功能。

(4) C1 Reed-Solomon码：交叉编织的C2码的28个符号再次用（32，28）RS码（4个奇偶校验符号用于纠错）编码。

(5) 输出阶段：有一半的码字要有一个符号的延迟以避免在符号边界上发生两个符号的错误。

CIRC的性能：两个RS码（C1和C2）都有4个奇偶校验位，它们的最小距离为5。如果错误位置未知，可以纠最多两个符号发生错误的情况。如果错误个数超过可纠的界，它们会通过插补隐藏起来。因为偶数标记的取样数据和奇数标记的取样数据被尽可能地编织，CIRC可以通过简单线性插补隐藏长的突发错误。

- 最大可纠的突发长度约为4 000数据比特（2.5毫米长的磁轨）。
- 在最坏情况下插补能纠的突发长度最大值约为12 320数据比特（7.7毫米长的磁轨）。

取样插补率在BER（比特错误率）$=10^{-4}$时为每10个小时一个取样，而在BER$=10^{-3}$时为每1000个小时一个插补。不能检测的错误取样在BER$=10^{-3}$时为每750小时少于一个，BER$=10^{-4}$时可以忽略。

5.11 评注

BCH码是由Hocquenghem在1959年、Bose和Ray Chaudhuri在1960年独立发现的。BCH码

构成最重要和功能强大的线性分组码的一类，即循环码。

Reed-Solomon码是由Irving S. Reed 和 Gustave Solomon发现的。他们1960年在*Journal of the Society for Industrial and Applied Mathematics*上发表了题为“Polynomial Codes over Certain Finite Fields”的只有5页的论文。尽管Reed-Solomon码有很多优点，它们在被发现之后并没有马上被应用，而不得不等待硬件技术的发展跟得上。在1960年没有快速数字电子学，至少按今天的标准当时没有。Reed-Solomon论文建议了9种处理数据的方法，但当时没人知道是不是可行，在1960年很可能是不可行的。

终于，技术赶上来了，许多研究人员开始了码的实现工作。其中关键人物之一是Elwyn Berle-kamp，他是加利福尼亚大学伯克利分校的电气工程教授，他发明了Reed-Solomon码的一种高效译码算法。Berlekamp算法在旅行者2号上得到了应用，也是CD播放器译码的基础。其他许多锦上添花（有些具有重要的理论意义）的工作也加进来。例如，CD所用的就是称为交叉编织的Reed-Solomon码，或简称为CIRC。

5.12 小结

- $GF(q)$ 上的本原元是这样一个元素α，即所有域元素除零外都能表示为α的幂。一个域可以有多于一个本原元。
- $GF(q)$ 上的一个本原多项式$p(x)$ 是$GF(q)$ 上的一个满足下列性质的素多项式：在由模$p(x)$构造的扩域上，x所表示的域元素是一个本原元。
- 形如$n=q^m-1$的分组长度n称为$GF(q)$ 上一个码的本原分组长度。$GF(q)$ 上具有本原分组长度的循环码称为本原循环码。
- 在$GF(q^m)$ 上有可能将$x^{q^m-1}-1$分解为 $x^{q^m-1}-1=\prod_j(x-\beta_j)$，其中$\beta_j$取遍$GF(q^m)$ 上所有非零元素。这表明每一个多项式$f_i(x)$ 在$GF(q^m)$ 上可以表示为一些线性项的乘积，而β_j恰好是$f_i(x)$ 中一个多项式的零点。这个$f_i(x)$ 成为β_j的极小多项式。
- $GF(q^m)$ 上两个在$GF(q)$ 具有相同极小多项式的元素称为在$GF(q)$ 上共轭。
- 定义在$GF(q)$ 上的分组长度为q^m-1的BCH码称为本原BCH码。
- 要确定纠t个错误的本原分组长度为$n=q^m-1$的BCH码的生成多项式，（1）选取一个次数为m的素多项式并构造$GF(q^m)$；（2）对$i=1,2,\cdots,p$求α^i的极小多项式$f_i(x)$；（3）求生成多项式$g(x)=\mathrm{LCM}[f_1(x),f_2(x),\cdots,f_{2t}(x)]$。用这种方法构造的码至少可以纠$t$个错。由于这个原因，$d=2t+1$称为码的设计距离，而码的最小距离为$d^*\geqslant 2t+1$。
- BCH码的译码步骤：

(1) 作为测试值，令$v=t$，计算伴随矩阵$\boldsymbol{M}$的行列式的值。如果行列式的值为零，则令$v=t-1$，再一次计算$\boldsymbol{M}$的行列式的值。重复这个过程直到得到一个v值使伴随矩阵的行列式的值不为零。这个v的值就是实际发生的错误的个数。

(2) 求矩阵$\boldsymbol{M}$的逆并计算错误定位多项式$\Lambda(x)$。

(3) 解方程$\Lambda(x)=0$，求得一些零点，从中计算错误位置$X_1, X_2, \cdots, X_v$。如果是二元码，到此为止（因为错误的大小为1）。

(4) 如果码不是二元码，回到方程组

$$\begin{aligned}
S_1 &= Y_1X_1 + Y_2X_2 + \cdots + Y_vX_v \\
S_2 &= Y_1X_1^2 + Y_2X_2^2 + \cdots + Y_vX_v^2 \\
&\vdots \qquad\qquad\qquad \vdots \\
S_{2t} &= Y_1X_1^{2t} + Y_2X_2^{2t} + \cdots + Y_vX_v^{2t}
\end{aligned}$$

因为错误位置未知，这形成了$2t$个线性方程。解这些方程构成的方程组就得到错误程度。

- 能纠t个错误的RS码的生成多项式为

$$\begin{aligned} g(x) &= \mathrm{LCM}[f_1(x), f_2(x), \cdots, f_{2t}(x)] \\ &= (x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{2t-1})(x-\alpha^{2t}) \end{aligned}$$

因此，生成多项式的次数总是$2t$。故RS码满足$n-k=2t$。

- Reed-Solomon码是最大距离可分（MDS）码，其最小距离为$n-k+1$。
- 获得大的分组长度码的方法之一就是将码嵌套。这种技术将一个小字母表的码和一个大字母表的码结合起来。设一个q元符号组长度为kK，这个组可以分为K个有k个符号的子组，每一个子组可以看做q^k元字母表中的一个元素。

“当你排除那些不可能的事件后，留下的不管有多么荒谬，都必然是真理。”

——福尔摩斯

习题

5.1　用一个合适的本原多项式由$GF(3)$构造$GF(9)$。

5.2　(1) 找出分组长度为8的纠两个错误的三元BCH码的生成多项式$g(x)$。该码的码率是多少？与（11, 6）三元Golay码比较码率和最小距离。

(2) 找出分组长度为26的纠三个错误的三元BCH码的生成多项式$g(x)$。

5.3　找出分组长度为31的一个二元BCH码的生成多项式$g(x)$。用本原多项式$p(x)=x^5+x^2+1$来构造$GF(32)$。该码的最小距离是多少？

5.4　$GF(2)$上的RS码的生成多项式是什么？

5.5　求下列码的生成多项式和最小距离：

(1) RS (15, 11) 码。

(2) RS (15, 7) 码。

(3) RS (31, 21) 码。

5.6　考虑一个$GF(16)$上的三错误纠错BCH(15, 5)码。给定一个接收多项式$v(x)=x^7+x^2$，则发送多项式是什么？

5.7　证明任意BCH码都是有相同设计距离的Reed-Solomon码在子域上的子码。在什么条件下该BCH码的码率等于RS码的码率？

5.8　证明Vandermonde多项式

$$A=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_1 & X_2 & \cdots & X_\mu \\ X_1^2 & X_2^2 & \cdots & X_\mu^2 \\ \vdots & & & \vdots \\ X_1^{\mu-1} & X_2^{\mu-1} & \cdots & X_\mu^{\mu-1} \end{bmatrix} \tag{5-29}$$

有一个非零的行列式，当且仅当所有的X_i ($i=1, 2, \cdots, \mu$)是不同的。

5.9　考虑$GF(11)$上具有下列奇偶校验矩阵的码

$$\boldsymbol{H}=\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & 2 & 3 & \cdots & 10\\ 1 & 2^2 & 3^2 & \cdots & 10^2\\ 1 & 2^3 & 3^3 & \cdots & 10^3\end{bmatrix} \tag{5-30}$$

(1) 证明该码为纠三个错误的码。

(2) 求该码的生成多项式。

5.10 考虑$GF(11)$上具有下列奇偶校验矩阵的码

$$\boldsymbol{H}=\begin{bmatrix}1 & 1 & 1 & 1 & \cdots & 1\\ 1 & 2 & 3 & 4 & \cdots & 10\\ 1 & 2^2 & 3^2 & 4^2 & \cdots & 10^2\\ 1 & 2^3 & 3^3 & 4^3 & \cdots & 10^3\\ 1 & 2^4 & 3^4 & 4^4 & \cdots & 10^4\\ 1 & 2^5 & 3^5 & 4^5 & \cdots & 10^5\end{bmatrix} \tag{5-31}$$

(1) 证明该码为纠三个错误的码。

(2) 求该码的生成多项式。

5.11 高速超宽带（UWB）通信中的ECMA标准使用$GF(2^8)$上的RS(255, 249)码，

(1) 找出此码的生成多项式。

(2) 给出系统化的RS(255, 249)码的基于移动寄存器的实现。

上机习题

5.12 写一个程序，使它当输入为一个本原多项式的系数以及q和m的值，它能构造扩域$GF(q^m)$。

5.13 写一个程序，使它能做$GF(2^m)$上的加法和乘法运算，其中m是个整数。

5.14 求分组长度为63的一个二元BCH码的生成多项式$g(x)$。用本原多项式$p(x)=x^6+x+1$构造$GF(64)$。该码的最小距离是多少？

5.15 写一个程序，当给出n、q、t和接收到的向量时它能进行BCH译码。

5.16 写一个程序，它能输出码字长度为n及信息长度为k的Reed-Solomon码的生成多项式。一个有效的n应该为2^M-1，其中M为不小于3的整数。该程序应该还列出码的最小距离。

5.17 写一个程序，它能进行标准CD中的两级RS编译码。

5.18 使用基于移动寄存器实现方法编写一段程序，实现$GF(2^6)$上的RS(63, 55)的系统化的编码器。

第6章 卷 积 码

现实中两个真理之间的最短路径要经过复杂的中间过程。

——Jacques Hadamard（1865—1963）

6.1 卷积码简介

到目前为止，我们学习了分组码，它把k个信息符号的组编码成为n个编码符号的组。在未编码的符号组（信息字）和编码后的符号组（码字）之间总有一一对应关系。这种方法特别适用于高数据率的应用，它将进入的未编码的数据流首先分成组，然后编码、传输（见图6-1）。由于下面的原因，大的分组长度是很重要的：

(1) 许多具有大距离性质的好码都有大的分组长度（如RS码）。

(2) 分组长度越大说明编码的附加耗费越低。

但是，大分组长度的缺点是，收信人只有在收到整个编码后的数据组后才能进行译码，这样可能会造成延迟。对比之下，还有另外一种编码方案，它用到小得多的未编码数据长度k_0。这些称为**信息帧**（information frame）。一个信息帧一般只包含几个符号，而且少的时候可以只有一个符号！这些信息帧被编码成长度为n_0的**码字帧**（codeword frame）。但是，只有一个信息帧不能用来获得码字帧，而是当前信息帧连同原来的m个信息帧用于获得单一的码字帧。这表明这样的编码器有记忆，它保留以前进来的m个信息帧。用这种方式得到的码称为**树码**（tree code）。直到现在我们所讨论的译码技术都是代数的，且无记忆的，即译码判决只决定于当前码字。卷积码在判决时用到过去的信息，即需要有记忆。

图6-1 用分组编码器编码

在本章中，我们从介绍**树码**和**网格码**开始，然后将用到必要的数学工具来构造卷积码。我们将看到卷积码可以很容易地用多项式表示。接下来我们将给出卷积码的矩阵描述。本章将继续讨论著名的Viterbi**译码技术**。我们将以介绍Turbo**编码和译码**来结束本章。我们同时会为Turbo编码简要讨论Interleaver设计。

6.2 树码和网格码

假设我们有一个无限长的输入符号流。（感谢今天发送的信息量，这种假设不算太差！）该符号流首先分成k_0个符号的片段，每个片段称为一个**信息帧**，如前所述。编码器由两部分组成（见图6-2）：

(1) 内存，其实就是一个移位寄存器。

(2) 一个逻辑电路。

编码器的内存可以储存m个信息帧。每次当一个新的信息帧到来时，它被移位到移位寄存

器，最旧的那个信息帧就被抛弃。在每一个帧时间段末，编码器在其内存中有m个最新的信息帧，它们对应于总共mk_0个信息符号。

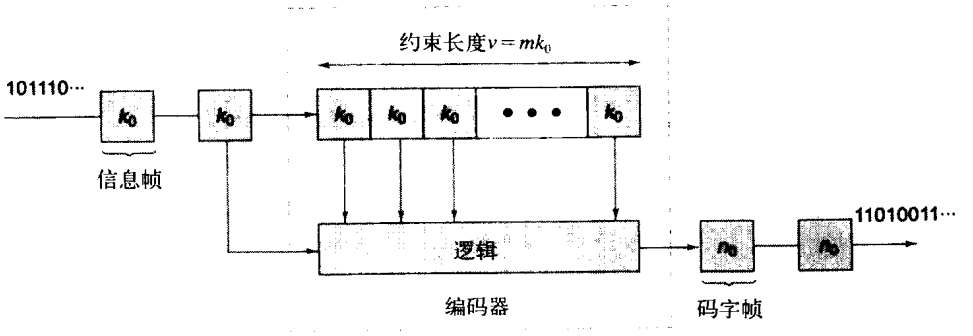


图6-2 生成树码的移位寄存器编码器

当一个新的帧到来时，编码器用刚到的帧和原来储存的m个帧计算码字帧。码字帧的计算是用逻辑电路完成的。该码字帧被移位出去。最旧的信息帧被抛弃，最新的信息帧被移位进来。然后编码器又可以对接下来的信息帧编码了。因此，对每一个进来的信息帧（k_0个符号），编码器都产生一个码字帧（n_0个符号）。可以观察到同样的信息帧可能会产生不同的码字帧，因为码字帧与原来的m个信息帧也有关。

定义6.1 一个移位寄存器编码器的**约束长度**（constraint length）定义为它的内存中所能储存的符号的个数。我们将在本章后面部分给出约束长度更正式的定义。

如果移位寄存器编码器储存了m个长度为k_0的原来的信息帧，则该编码器的约束长度为$v=mk_0$。

定义6.2 由所有可能输入到一个移位寄存器编码器的序列所产生的所有无限长码字的无限集合称为一个(n_0,k_0)**树码**。该树码的码率定义为

$$R=\frac{k_0}{n_0} \tag{6-1}$$

一种更正式的定义是，(n_0,k_0)**树码**是$GF(q)$上元素的伪无限序列的集合到自身的一个映射，满足对任意m，若两个伪无限序列的前mk_0个分量相同，则它们的映像也在前mk_0个分量相同。

定义6.3 移位寄存器编码器的**码字长度**（Wordlength）定义为$k=(m+1)k_0$。移位寄存器编码器的分组长度定义为$n=(m+1)n_0=k\frac{n_0}{k_0}$。注意码率为$R=\frac{k_0}{n_0}=\frac{k}{n}$。

一般情况下，对实用移位寄存器编码器来说，信息帧长度k_0很小（通常小于5）。因此，很难像分组码的情况（如RS码）得到树码接近1的码率R。

定义6.4 一个(n_0,k_0)树码如果是线性的、不随时间变化的，而且有有限码字长度$k=(m+1)k_0$，则称为(n,k)**卷积码**（Convolutional Code）。

定义6.5 一个不随时间变化且有有限码字长度k的(n_0,k_0)树码称为(n,k)**滑动分组码**（Sliding Block Code）。

例6.1 考虑图6-3所示的卷积码。

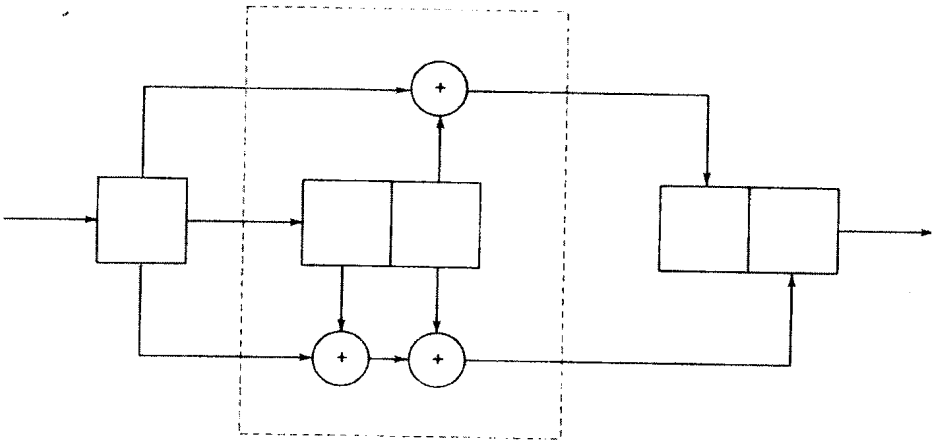
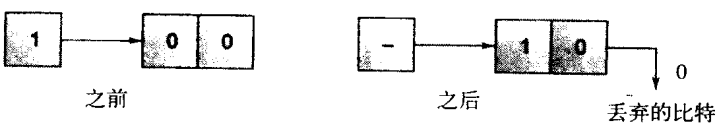


图6-3　例6.1的卷积编码器，此处$k_0=1, n_0=2$和$m=2$

这个编码器每次把1比特输入编码成2比特的输出。信息帧的长度为$k_0=1$，码字帧的长度为$n_0=2$，分组长度为$(m+1)n_0=6$。该编码器的约束长度为$v=2$，码率为1/2。输出数据的速率是输入数据的两倍。加法器为二元加法器，而且从电路实现的角度看，它们只是异或（XOR）门。

我们假定移位寄存器的初始状态为[0 0]。输入比特可能为“0”，也可能为“1”。假定进来的是“0”。在执行逻辑运算后，可知计算出的码字帧的值为[0 0]。0将被反馈到移位寄存器的内存，而那个最右边的“0”被丢弃了。移位寄存器的状态保持在[0 0]。下面假设“1”进入编码器。我们再一次执行逻辑运算来计算码字帧。这次我们得到[1 1]，它将作为编码后的帧输出。进来的“1”将被移位到内存中，而最右边的位被丢弃。故移位寄存器新的状态将为[0 0]。



此外，这里有两种关于输入比特的可能情况：“0”或“1”。表6-1列出了所有的可能情况。

表6-1　卷积编码器输入和输出的比特

输入比特	编码器当前状态	输出比特
0	0 0	0 0
1	0 0	1 1
0	0 1	1 1
1	0 1	0 0
0	1 0	0 1
1	1 0	1 0
0	1 1	1 0
1	1 1	0 1

我们观察到移位寄存器的状态有$2^2=4$种。因此我们可以构造编码器如图6-4所示的状态图。每个箭头所标的比特表示输入比特。可以看到同样的输入比特根据编码器当前的状态其编码

结果也不同。这与我们在前面章节中学习的线性分组码不同，那里输入的未编码符号组（信息字）与编码后的符号组（码字）总是存在一一对应的关系。

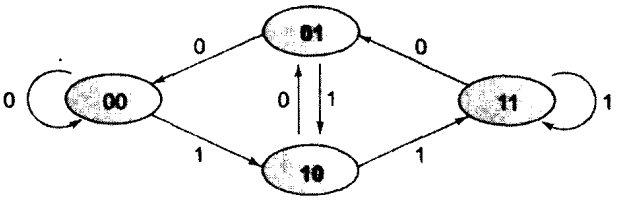


图6-4 例6.1中的编码器的状态图

状态图中所包含的信息可以有效地搬运到称作网格图的图中。网格图中的节点是长方形格子，它的右边是伪无限的。每一栏的节点的数字是无限的。下面的例子描述了网格图。

例6.2 例6.1中讨论的卷积编码器的网格图如图6-5所示。

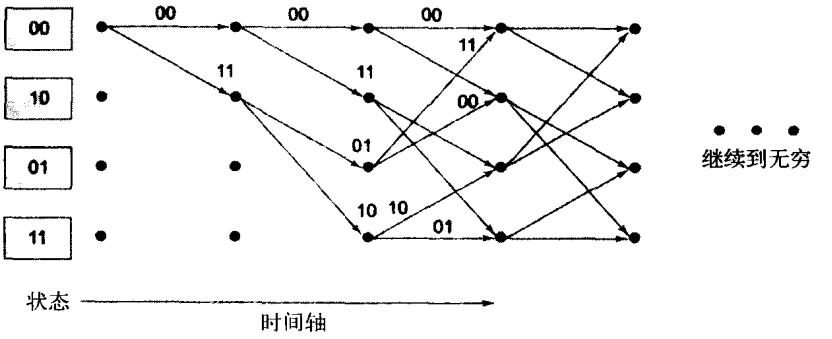


图6-5 图6-3所表示的编码器的网格图

网格图中的每个节点表示移位寄存器的一个状态。因为该编码器的码率为1/2，编码器每次处理一个比特。输入比特或为“0”或“1”。因此从每个节点发出两个分支。上面的分支表示输入为“0”的情况，下面的分支表示输入为“1”的情况。一般地，我们会在每个分支上标明它对应的输入符号。通常那些从左上方的节点开始并向右移动所不能到达的节点都不在网格图中显示。对应于某个状态和某特殊输入比特，编码器将产生一个输出。该输出被写在那个分支的上方。于是网格图给出了一种对输入数据流编码的非常简单的方法。用网格图进行编码的步骤如下：

- 我们从左上方的节点开始（因为编码器的初始状态为[0 0]）。
- 根据进入的是“0”还是“1”，我们沿上面的或下面的分支到达下一个节点。
- 编码器的输出从所经分支的上方读出。
- 再一次我们需要根据进入的是“0”还是“1”决定应该从当前节点（状态）沿上面还是下面的分支移动。
- 故编码过程只是沿图上分支读出编码器写在每个分支上方的输出而已。

将比特流1 0 0 1 1 0 1…编码后得到图6-6所示的网格图。从图中可以读出编码后的序列为 11 01 11 11 10 10 00 …。

可以证明在编码后的序列和网格图的路径之间一一对应。那么译码步骤是不是应该在网格图中寻找最可能的路径呢？答案是肯定的，我们在本章中将进一步证明这一点。

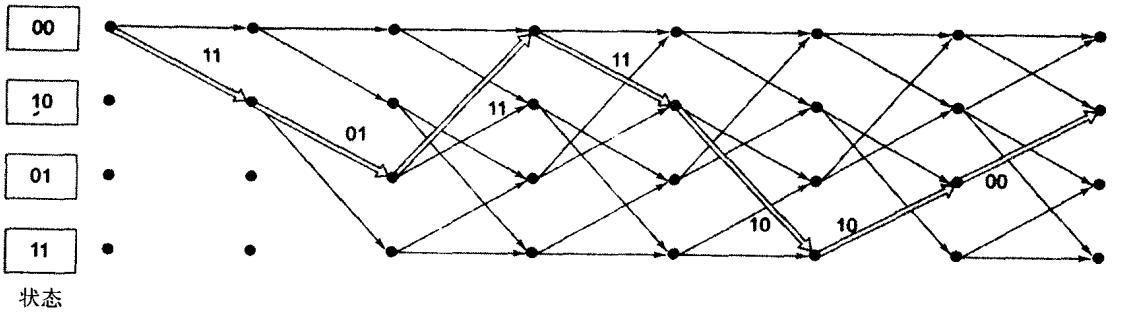


图6-6　用网格图对一个输入序列编码

6.3　卷积码的多项式描述（解析表示）

不同于前两个卷积码的图形表示（状态图和网格图），卷积码还有一种很有用的解析表示。该表示用到延迟算子D。我们前面看到了字（向量）和多项式之间的一一对应关系。该延迟算子使用方法类似。例如，考虑字10100011，最旧的数字在最左边。该信息字的解析表示$I(D)$将为

$$I(D)=1+D^2+D^6+D^7 \tag{6-2}$$

未知量D是相对于原始时间的延迟的时间单位数，它通常取自第一个比特。一般地，序列$i_0, i_1, i_2, i_3, \ldots$表示为$i_0+i_1D+i_2D^2+i_3D^3+\cdots$。

$GF(q)$上一个码字长度$k=(m+1)k_0$，分组长度$n=(m+1)n_0$，约束长度$v=mk_0$的卷积码可以用有限脉冲响应（FIR）过滤器的集合进行编码，每个过滤器集合包含$GF(q)$上的k_0个FIR过滤器。编码器的输入为k_0个符号的序列，而输出为n_0个符号的序列。图6-7显示的是一个$k_0=1$和$n_0=4$的二元卷积码的编码器。图6-8表示的是$k_0=2$和$n_0=4$的卷积过滤器。

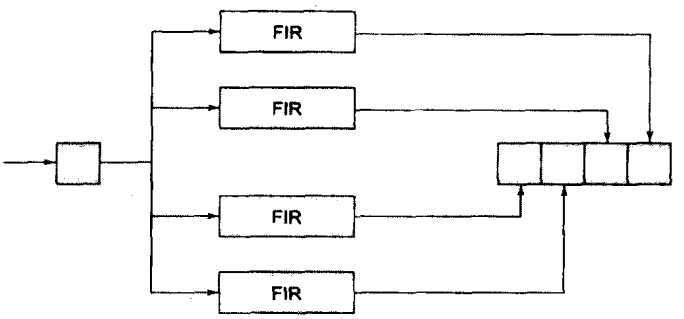


图6-7　$k_0=1$和$n_0=4$的FIR过滤器中的卷积编码器

每一个FIR过滤器可以表示为次数小于等于m的一个多项式，输入符号流也可以表示为一个多项式。于是过滤器的运算就变成了两个多项式的乘法。因此编码器（以及码）可以表示为一个多项式的集合，称为码的**生成多项式**（generator polynomial）。这个集合包含k_0n_0个多项式。在这个生成多项式集合中的多项式的最大次数为m。我们提醒读者：一个分组码用**单一**的生成多项式表示。因此对一个卷积码，我们可以定义$k_0\times n_0$阶的**生成多项式矩阵**（generator polynomial matrix）如下：

$$G(D)=[g_{ij}(D)] \tag{6-3}$$

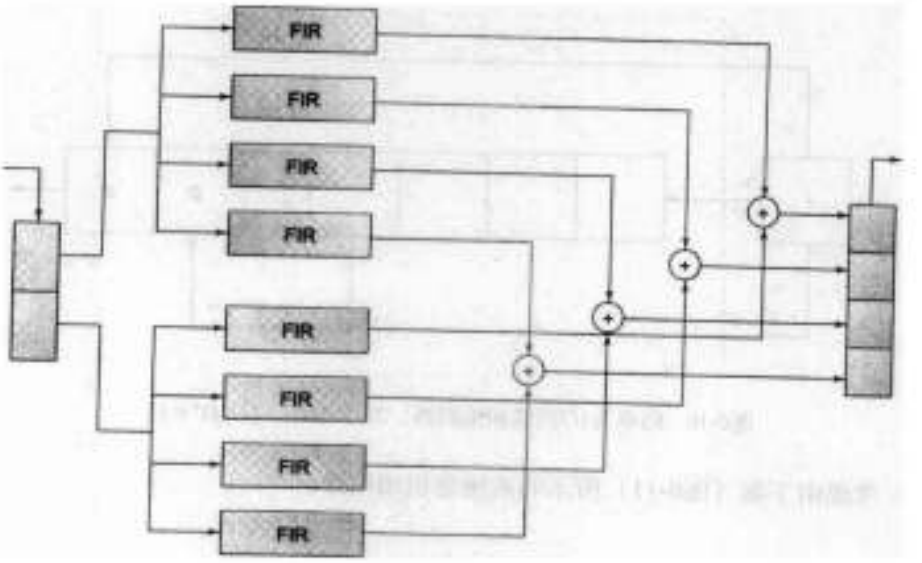


图6-8 $k_0=2$和$n_0=4$的FIR过滤器中的卷积编码器。该编码器的码率为$R=1/2$

例6.3 考虑图6-9给出的卷积码的编码器。

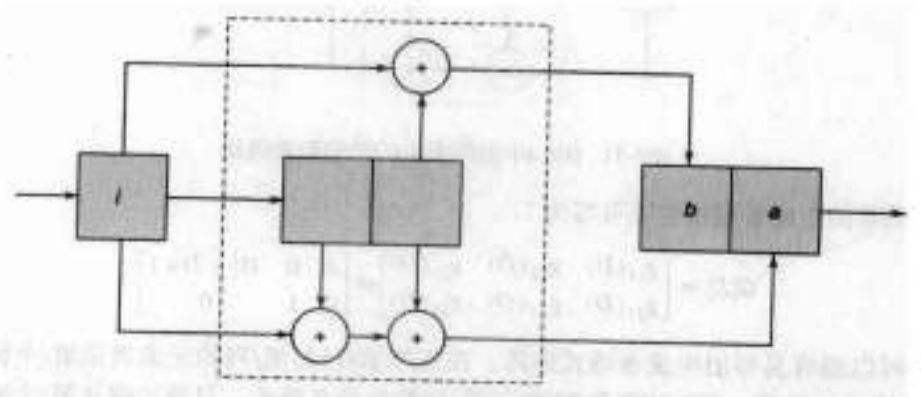


图6-9 码率为1/2的卷积编码器，其中$G(D)=[D^2+D+1 \quad D^2+1]$

输出的第一个比特为$a=i_{n-2}+i_{n-1}+i_n$，第二个比特为$b=i_{n-2}+i_n$，其中i_l表示提前l个时间单位到达的输入。设输入符号流由多项式表示。我们知道D乘任意多项式对应于对其元素的单一循环右移位，因此

$$g_{11}(D)=D^2+D+1 \text{且} g_{12}(D)=D^2+1$$

该编辑器的生成多项式矩阵可写为：

$$G(D)=[D^2+D+1 \quad D^2+1]$$

下面考虑如图6-10所示的编码器电路。

在这种情况下，$a=i_n$，$b=i_{n-4}+i_n$。因此该编码器的生成多项式矩阵可以写为

$$G(D)=[1 \quad D^4+1]$$

注意码字帧的前k_0比特（$k_0=1$）与信息帧相同。因此这是一个**系统卷积编码器**。

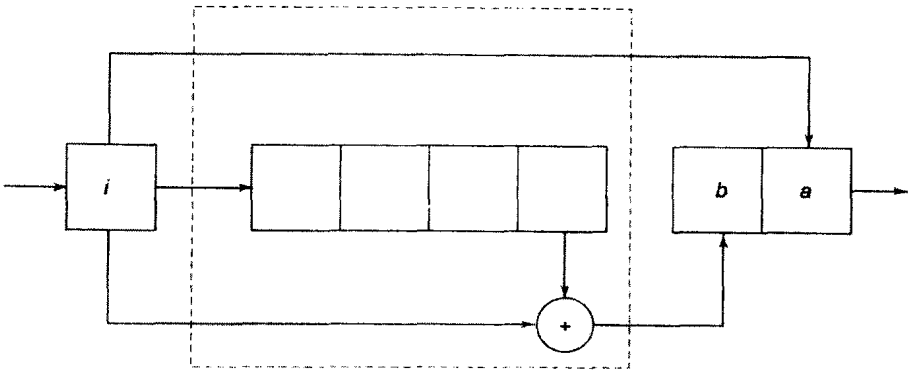


图6-10　码率为1/2的卷积编码器，其中$G(D)=[1 \quad D^4+1]$

例6.4　考虑由下图（图6-11）所示的系统卷积编码器。

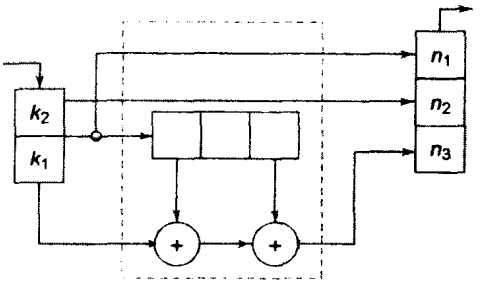


图6-11　例6.4中的码率为1/3的卷积编码器

该编码器的生成多项式矩阵可写为

$$\boldsymbol{G}(D)=\begin{bmatrix} g_{11}(D) & g_{12}(D) & g_{13}(D) \\ g_{21}(D) & g_{22}(D) & g_{23}(D) \end{bmatrix}=\begin{bmatrix} 1 & 0 & D^3+D+1 \\ 0 & 1 & 0 \end{bmatrix}$$

通过观察可以很容易写出生成多项式矩阵。在矩阵第i行与第j列的元素表示第i个输入比特与第j个输出比特的关系。要写出矩阵的第i行第j列的生成多项式，只要追踪从第i个输入比特到第j个输出比特的路径即可。如果不存在这样的路径，则生成多项式为零多项式，正如$g_{12}(D)$、$g_{21}(D)$和$g_{23}(D)$的情况。如果只有直接路径而没任何延迟元素，则多项式的值为1，正如$g_{11}(D)$和$g_{22}(D)$的情况。如果从第i个输入比特到第j个输出比特的路径涉及一系列的内存元素（延迟元素），将每一个延迟表示为D的多一次的幂，如$g_{13}(D)$的情况。注意在生成多项式集合中有三个生成多项式为零。当$k_0>1$时，生成多项式中有些为零的情况不常见。

我们现在可以给出一个卷积编码器的码字长度、分组长度和约束长度的正式定义。

定义6.6　给定一个卷积码的生成多项式矩阵$[g_{ij}(D)]$：

(1) 码的**码字长度**为

$$k=k_0 \max_{i,j}\left[\deg g_{ij}(D)+1\right] \tag{6-4}$$

(2) 码的**分组长度**为

$$n = n_0 \max_{i,j}\left[\deg g_{ij}(D)+1\right] \tag{6-5}$$

(3) 码的**约束长度**为

$$v = \sum_{i=j}^{k_0} \max_{i,j}\left[\deg g_{ij}(D)\right] \tag{6-6}$$

输入消息流$i_0, i_1, i_2, i_3, \cdots, i_{k_0}$的多项式表示为$I(D)=i_0+i_1D+i_2D^2+i_3D^3+\cdots+i_{k_0}D^{k_0}$，码字多项式可以写为$\boldsymbol{C}(D)=c_0+c_1D+c_2D^2+c_3D^3+\cdots+c_{n_0}D^{n_0}$。编码操作可以简单描述为向量矩阵的乘积：

$$\boldsymbol{C}(D) = \boldsymbol{I}(D)\boldsymbol{G}(D) \tag{6-7}$$

或等价地表示为

$$c_j(D) = \sum_{l=1}^{k_0} i_l(D) g_{l,j}(D) \tag{6-8}$$

观察到编码操作可以简单描述为向量矩阵的乘积，容易证明卷积码属于线性码（留作习题）。

定义6.7 **奇偶校验矩阵**$\boldsymbol{H}(D)$是有(n_0-k_0)个行和n_0个列的多项式矩阵，满足

$$\boldsymbol{G}(D)\boldsymbol{H}(D)^T = \boldsymbol{0} \tag{6-9}$$

伴随多项式向量，一个含(n_0-k_0)个分量的行向量，可由下式给出：

$$\boldsymbol{s}(D) = \boldsymbol{v}(D)\boldsymbol{H}(D)^T \tag{6-10}$$

定义6.8 一个卷积码的**系统编码器**具有如下形式的生成多项式矩阵：

$$\boldsymbol{G}(D) = [\boldsymbol{I} \mid \boldsymbol{P}(D)] \tag{6-11}$$

其中I是$k_0\times k_0$阶单位矩阵，$P(D)$为$k_0\times(n_0-k_0)$阶多项式矩阵。一个系统卷积编码器的奇偶校验多项式矩阵为

$$\boldsymbol{H}(D) = [-\boldsymbol{P}(D)^T \mid \boldsymbol{I}] \tag{6-12}$$

其中I是$(n_0-k_0)\times(n_0-k_0)$阶单位矩阵。于是得到

$$\boldsymbol{G}(D)\boldsymbol{H}(D)^T = \boldsymbol{0} \tag{6-13}$$

定义6.9 一个卷积码若对某个a，其生成多项式$g_1(D), g_2(D), \cdots, g_{n_0}(D)$满足

$$GCD[g_1(D), g_2(D), \cdots, g_{n_0}(D)] = D^a \tag{6-14}$$

则该卷积码称为**非灾难性卷积码**。否则称为**灾难性卷积码**。

为了不失一般性，我们令$a=0$，即$D^a=1$。因此寻找非灾难性卷积码的工作就等价于寻找一组好的互素生成多项式的问题。互素多项式可以很容易通过计算机搜索找到。但是，困难的是如何找到一组具有良好纠错能力的互素多项式。

例6.5 所有系统码都是非灾难性的，因为对于它们有$g_1(D)=1$，从而

$$GCD[1, g_2(D)], \cdots, g_{n_0}(D)] = 1$$

因此由生成多项式矩阵

$$G(D) = [1 \quad D^4 + 1]$$

所表示的系统卷积编码器为**非灾难性的**。

考虑如下的二元卷积编码器的生成多项式矩阵

$$G(D) = [D^2 + 1 \quad D^4 + 1]$$

我们观察到对二元编码器（mod 2）有 $(D^2+1)^2 = D^4 + (D^2 + D^2) + 1 = D^4 + 1$，故 $GCD[g_1(D), g_2(D)] = D^2 + 1 \neq 1$。因此这是一个**灾难性的**编码器。

下面考虑非系统型二元卷积编码器的生成多项式矩阵

$$G(D) = [D^2 + 1 \quad D^4 + D^2 + 1]$$

这两个生成多项式是互素的，即 $GCD[g_1(D), g_2(D)] = 1$，因此，这是个**非灾难性**卷积编码器。

在下一节中，我们将介绍卷积码的距离概念是决定一个卷积码能纠的错误个数的重要参数。

6.4　卷积码的距离概念

对分组码，两个码字间（汉明）距离的概念提供了量化描述两个向量之间的差异及作为一个好码必须如何拥有最大可能的最小距离的一种方法。卷积码也有一个决定一个码好的程度的距离的概念。

当一个码字从卷积编码器送到信道中去时，错误会不断出现。译码器的工作就是通过处理接收到的向量来纠这些错误。原则上，卷积码字的长度是无限的，但译码判决是根据码字的有限长片段做出的。译码器可以储存的符号的数量称为**译码窗宽度**（decoding window width）。不管这些有限片段的大小（译码窗宽度）如何，因为编码器有记忆，故原来的帧对当前的帧是有影响的。一般地，当增加译码窗宽度时会得到更好的效果，但当最终到达一个峰值后效果又会回落。

多数卷积码的译码过程是将注意力放在第一个帧的错误上。如果这个帧可以纠错并译码，则在接收端便得到第一个信息帧。这些信息符号对后面信息帧的影响可以从后续码字帧中计算得到。因此译码第二个码字帧的问题与译码第一帧的情况相同。

我们把这种逻辑进一步推广。如果前f个帧被成功译码了，则译码第（$f+1$）个帧的问题与译码第一帧是一样的。但如果中间的帧没有得到正确译码会怎么样呢？单一的译码错误事件可能会引发无限多个另外的错误，那么译码器称为遭受**错误传播**。这种译码算法造成的错误传播情况称为**普通错误传播**。当错误传播是由于选取了灾难性的生成多项式所致时，称为**灾难性错误传播**。

定义6.10　*卷积码的l级**最小距离**d_l^*等于任意两个有l个帧且最初的帧不相同的初始码字片段之间的最小汉明距离。如果$l = m + 1$，则这第（$m+1$）个最小距离称为码的**最小距离**，记为d^*，其中m为编码器内存所能储存的信息帧的个数。在文献中，最小距离也记为d_{min}。*

我们这里注意到卷积码是个线性码。因此决定码的最小距离的码字之一可以选取为全零码字。于是l级最小距离就等于第一个帧不为零（即不同于全零帧）且由l个帧组成的码片段的

最小重量。

假定卷积码的l级最小距离为d_l^*，当下列条件满足时该码可以纠在前l个帧中发生的t个错误：

$$d_l^* \geqslant 2t+1 \tag{6-15}$$

下面设$l=m+1$，在此情况下有$d_l^* = d_{m+1}^* = d^*$。下列条件满足时该码可纠发生在第一个分组长$n=(m+1)n_0$中的t个错误

$$d^* \geqslant 2t+1 \tag{6-16}$$

例6.6 考虑例6.1（图6-3）中的卷积编码器。该编码器的网格图如图6-12所示。

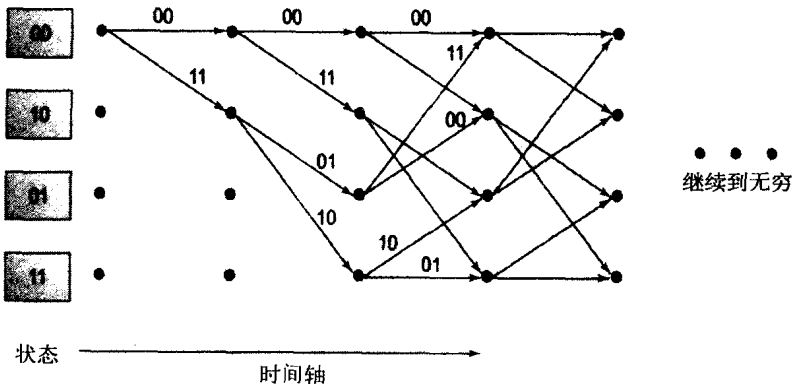


图6-12 例6.1中卷积编码器的网格图

在这种情况下$d_1^*=2$，$d_2^*=3$，$d_3^*=5$，$d_4^*=5$，…。我们观察到当$i\geqslant 3$时有$d_i^*=5$。对这个编码器，有$m=2$。因此该码的最小距离为$d_3^*=d^*=5$。这个码可以纠$(d^*-1)/2=2$个在一个分组长度$n=(m+1)n_0=6$中发生的随机错误。

定义6.11 一个卷积码的**自由距离**（Free Distance）为

$$d_{free} = \max_l [d_l^*] \tag{6-17}$$

由此得$d_{m+1} \leqslant d_{m+2} \leqslant \cdots \leqslant d_{free}$

d_{free}这一术语首先是由Massey在1969年提出的，用来表示一种距离，它是在卷积码译码技术中发现的一个重要参数。因为d_{free}表示的是任意长（可能为无限）编码序列之间的最小距离，d_{free}在文献中也被记为d_∞。参数d_{free}可以从网格图中直接计算。自由距离d_{free}是从全零路径偏离，后来又在网格后面的某一点回到全零路径的这样一条路径的最小距离，如图6-13所示。寻找具有大的最小距离和自由距离的码是一种单调乏味的过程，通常由计算机完成。已经有些很好的技术可以避免穷举搜索从而减少工作量。目前发现的大多数好的卷积码都是由计算机搜索找到的。

定义6.12 一个卷积码的**自由长度**n_{free}为最小重量的非零卷积码字的非零片段的长度。因此若$l=n_{free}$，则$d_l=d_{free}$，且若$l<n_{free}$，则$d_l<d_{free}$。在文献中，n_{free}也记为n_∞。

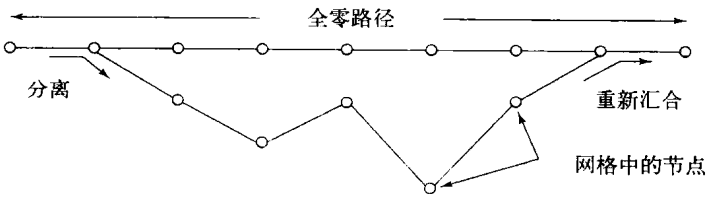


图6-13　自由距离d_{free}是指从全零路径偏离和后来又回到全零路径的一条路径的最小距离

例6.7　考虑例6.1（图6-3）中的卷积编码器。对这个编码器有$d_{free}=5$。通常有多于一对的路径可用于计算d_{free}。在图6-14中用于计算d_{free}的两条路径用双线标出。在这个例子中，$d_{min}=d_{free}$。

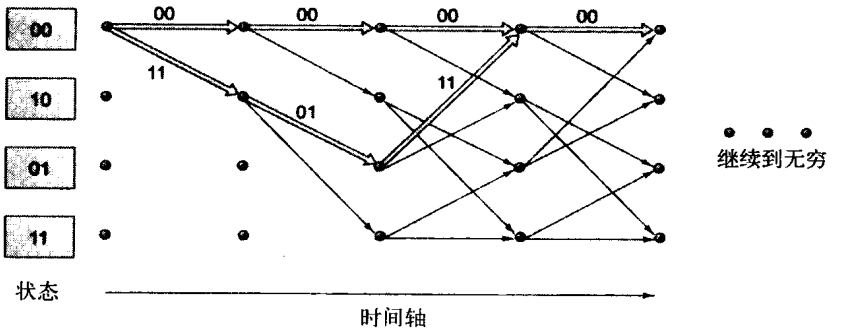


图6-14　在网格图中有多于一对的路径可用于计算d_{free}

该卷积码的自由长度为$n_{free}=6$。在本例中，n_{free}等于码的分组长度n。一般情况下它可以比分组长度要长。

6.5　生成函数

卷积码的性能取决于它的自由距离d_{free}。因为卷积码是线性码的一个子类，编码序列之间的汉明距离（Hamming distance）的集合与编码序列同全零序列之间的距离的集合是相同的。因此为了不失一般性，我们可以考虑卷积码相对全零序列的距离结构。在本节中我们将学习确定卷积码自由距离d_{free}的一种有效的方法。

要找到d_{free}，我们需要从全零路径偏离出去然后又回来的那些路径的集合。穷举搜索方法（耗时，不要提否则使人恼怒）是确定网格图中所有可能路径的距离。求找卷积码的d_{free}的另一种方法用到**生成函数**的概念，它的扩展直接给出所有的距离信息。生成函数可以通过下面的例子理解。

例6.8　再次考虑例6.1（图6-3）中的卷积编码器。它的编码器的状态图如图6-4所示。我们现在构造修改的状态图，如图6-15所示。

这个修改过的状态图的分支用分支增益D^i，$i=0,1,2$标记，其中D的幂为该分支的汉明重量。注意S_0的子回路被忽略了，因为它对码的距离性质没有贡献。沿着这个回路转只能产生全零序列。注意S_0被分为两个状态，初始状态和终点状态。任何从S_0分离出去后来又回到S_0的路径可以被认为等价于在这个修改过的状态图中从初始状态S_0开始，穿越该路径到达终点状态S_0。因此这个修改过的状态图包括了网格图中从全零路径分离后又回到该路径的所有可能的路径。

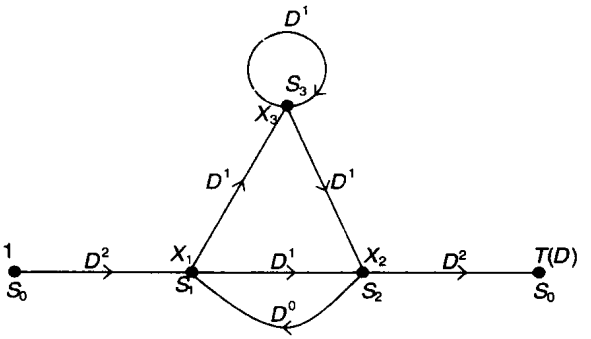


图6-15 图6-3中的卷积码的修改过的状态图

我们可以通过修改过的状态图的状态方程描述卷积码的距离。这些状态方程为

$$\begin{aligned} X_1 &= D^2 + X_2 \\ X_2 &= DX_1 + DX_3 \\ X_3 &= DX_1 \\ T(D) &= D^2 X_2 \end{aligned} \tag{6-18}$$

其中X_i为虚构变量。求解由这些方程构成的方程组，可以得到生成函数

$$\begin{aligned} T(D) &= \frac{D^5}{1-2D} \\ &= D^5 + 2D^6 + 4D^7 + \cdots + 2^k D^{k+5} + \cdots \\ &= \sum_{d=5}^{\infty} a_d D^d \end{aligned} \tag{6-19}$$

注意$T(D)$的表达式也可以由**Mason增益公式**（很容易地）得到，它是对学习**数字信号处理**的学生很著名的公式。

从生成函数的级数展开式可以得到下述结论：

(1) 从全零路径分离出去后又回到该路径的可能路径有无穷多个（这是很直观的）。

(2) 汉明距离为5的路径只有一个，汉明距离为6的路径有两个，一般地，与全零路径的汉明距离为$k+5$的路径有2^k个。

(3) 该码的自由汉明距离d_{free}为5。只有一个对应d_{free}的路径。例6.7清楚阐明了产生d_{free}的两条路径。

现在介绍修改过的状态图中的两个新标记。为了计算一个给定路径的长度，每个分支附带一个标签L。这样，当我们每次沿一个分支走的时候，会增加路径长度计数器的值。我们还对每个分支另加标签I^i，用来计算与修改过的状态图中每个分支相关的输入比特的汉明重量（见图6-16）。这种情况下的状态方程为

$$\begin{aligned} X_1 &= D^2 LI + LIX_2 \\ X_2 &= DLX_1 + DLX_3 \\ X_3 &= DLIX_1 + DLIX_3 \end{aligned}$$

进而**扩展生成函数**为

$$T(D) = D^2 LX_2 \tag{6-20}$$

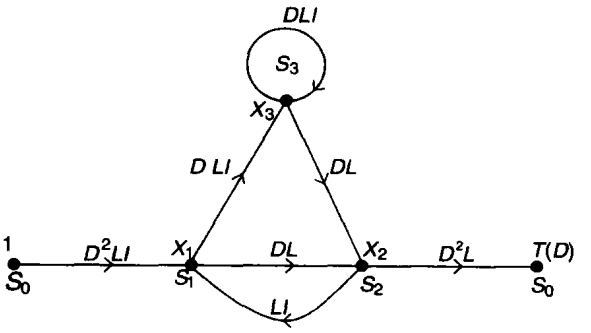


图6-16　用于确定扩张生成函数的修改状态图

同时解这些方程我们得到

$$T(D,L,I)=\frac{D^5L^3I}{1-DL(L+1)I}$$

$$=D^5L^3I+D^6L^4(L+1)I^2+\cdots+D^{k+5}L^{3+k}(L+1)^kI^{k+1}+\cdots \tag{6-21}$$

从扩展生成函数的级数展开式中我们可进一步得到下述结论：

(1) 汉明距离为5的路径的长度等于3。

(2) 对应于这个路径的输入序列的重量等于1。

(3) 与全零路径的汉明距离等于6的路径有两个，它们当中一个长度为4，另一个长度等于5（观察和式的第二项中L的幂）。这两个路径的输入序列的重量都为2。

在下一节中，我们学习卷积码的矩阵描述，它比线性分组码的矩阵描述要复杂些。

6.6　卷积码的矩阵描述

一个卷积码可以描述为由无数个无限长码字构成的，它们（从网格图中看到）属于线性码。它们可以用一个无限大的生成矩阵来描述。正如所预料的，卷积码的矩阵描述比线性分组码的矩阵描述要乱得多。

设一个卷积码的生成多项式表示为

$$g_{ij}(D)=\sum_l g_{ijl}\,D^l \tag{6-22}$$

为了得到生成矩阵，将系数g_{ijl}排成矩阵形式。对每一个l，记$\boldsymbol{G}_l$为$k_0\times n_0$阶矩阵

$$\boldsymbol{G}_l=[g_{ijl}] \tag{6-23}$$

则卷积码缩短成一个分组长度为n的分组码时的生成矩阵为

$$\boldsymbol{G}^{(n)}=\begin{bmatrix}\boldsymbol{G}_0 & \boldsymbol{G}_1 & \boldsymbol{G}_2 & \cdots & \boldsymbol{G}_m\\ \boldsymbol{0} & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_{m-1}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{G}_0 & \cdots & \boldsymbol{G}_{m-2}\\ \vdots & \vdots & \vdots & & \vdots\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{G}_0\end{bmatrix} \tag{6-24}$$

其中$\boldsymbol{0}$为$k_0\times n_0$阶零矩阵，m为用来生成该码的移位寄存器的长度。该卷积码的生成矩阵为

$$G=\begin{bmatrix} G_0 & G_1 & G_2 & \cdots & G_m & 0 & 0 & 0 & 0 & \cdots \\ 0 & G_0 & G_1 & \cdots & G_{m-1} & G_m & 0 & 0 & 0 & \cdots \\ 0 & 0 & G_0 & \cdots & G_{m-2} & G_{m-1} & G_m & 0 & 0 & \cdots \\ \vdots & \vdots & & & & & & & & \vdots \end{bmatrix} \tag{6-25}$$

该矩阵向右无限伸延。对于系统卷积码，其生成矩阵可以写为

$$G=\left[\begin{array}{cc|cc|cc|c|cc|cc|cc|c} I & P_0 & 0 & P_1 & 0 & P_2 & \cdots & 0 & P_m & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & I & P_0 & 0 & P_1 & \cdots & 0 & P_{m-1} & 0 & P_m & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & I & P_0 & \cdots & 0 & P_{m-2} & 0 & P_{m-1} & 0 & P_m & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & 0 & P_{m-2} & 0 & P_{m-1} & \cdots \\ & & & & & & & & & \vdots & \vdots & 0 & P_{m-2} & \cdots \\ & & & & & & & & & \vdots & \vdots & \vdots & \vdots & \end{array}\right] \tag{6-26}$$

其中I为$k_0\times k_0$阶单位矩阵，0为$k_0\times k_0$阶零矩阵，$P_0, P_1, P_2, \ldots, P_m$为$k_0\times(n_0-k_0)$阶矩阵。于是奇偶校验矩阵可写为

$$H=\begin{bmatrix} P_0^T & -I & & & & & & & & \cdots \\ P_1^T & 0 & P_0^T & -I & & & & & & \\ P_2^T & 0 & P_1^T & 0 & P_0^T & -I & & & & \\ \vdots & & \vdots & & \vdots & & & & & \vdots \\ P_m^T & 0 & P_{m-1}^T & 0 & P_{m-2}^T & 0 & \cdots & P_0^T & -I & \cdots \\ & & P_m^T & 0 & P_{m-1}^T & 0 & \cdots & & & \\ & & & & P_m^T & 0 & \cdots & & \cdots & \end{bmatrix} \tag{6-27}$$

例6.9 考虑图6-17所示的卷积编码器。我们首先写出该编码器的生成多项式矩阵。要做到这一点，我们只要一个接一个地计数从输入到输出的延迟就行了。得到的生成多项式矩阵为

$$G(D)=\begin{bmatrix} D+D^2 & D^2 & D+D^2 \\ D^2 & D & D \end{bmatrix}$$

生成多项式为$g_{11}(D)=D+D^2$，$g_{12}(D)=D^2$，$g_{13}(D)=D+D^2$，$g_{21}(D)=D^2$，$g_{22}(D)=D$，$g_{23}(D)=D$。

为了写出矩阵G_0，看一下生成多项式中的常数项（D^0的系数）。因为在所有生成多项式中都没有常数项，

$$G_0=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

接下来，我们要写出矩阵G_1，看一下生成多项式中D^1的系数。矩阵G_1第1行第1列的项对应于$g_{11}(D)$中D^1的系数，矩阵第1行第2列的项对应于$g_{12}(D)$中D^1的系数，依此类推。于是

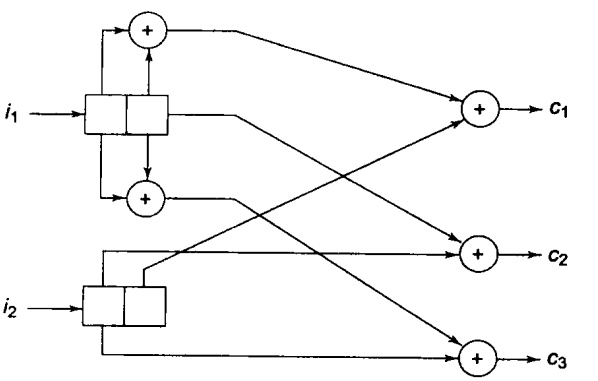


图6-17 码率为2/3的卷积编码器

$$\boldsymbol{G}_1=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}$$

类似地，可以写出

$$\boldsymbol{G}_2=\begin{bmatrix}1 & 1 & 1\\ 1 & 0 & 0\end{bmatrix}$$

现在可以写出生成矩阵：

$$\boldsymbol{G}=\left[\begin{array}{ccc:ccc:ccc:ccc:c}
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \cdots\\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & \cdots\\
\hdashline
 & & & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & \\
 & & & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & \\
\hdashline
 & \vdots & & & & & & & & 1 & 0 & 1 & \\
 & \vdots & & & \vdots & & & \vdots & & 0 & 1 & 1 & \cdots
\end{array}\right]$$

下面的任务是了解一种卷积码的有效译码方法。我们将详细讨论一种很流行的译码方法，**维特比**（Viterbi）**译码技术**。

6.7　卷积码的维特比译码

卷积码有三种重要的译码技术：序列译码、门限译码和维特比译码。**序列译码**技术是Wozencraft在1957年提出来的。序列译码的优点是它能很好地处理约束长度很长的卷积码，但它的译码时间是可变的。**门限译码**，也称为**择多逻辑译码**，是Massey于1963年在他在MIT的博士论文中提出的。门限译码器是卷积码的第一个商业化生产的译码器。**维特比译码**是Andrew J. Viterbi在1967年研发的，在20世纪70年代末已变成卷积码的主导技术。维特比译码具有下列优点：

(1) 比特错误方面的表现令人满意。

(2) 运行速度快。

(3) 容易实现。

(4) 成本低。

门限译码因其比特错误方面表现较差而变得不流行。它在概念上和实践上最接近分组码的译码，而且它需要计算一个伴随式集合，就像分组码的情况一样。在这种情况下，伴随式是个序列，因为信息和校验数字以序列形式出现。

维特比译码的优点是有着固定译码时间。它很适合于译码器的硬件实现。但它的计算量以约束长度的一个函数为指数增长，因此在实际中它通常限定约束长度为$v=9$或更小。到2000年初，有些主要的公司声明他们生产出了$v=9$的维特比译码器，其速度可以达到2Mbps。

自从维特比提出他的算法以来，其他研究人员通过寻找好的卷积码，探求该技术的性能局限性，用不同的译码器设计参数来优化该技术在硬件和软件上的实现，进而把他的工作做了进一步推广。维特比译码算法还用于对网格编码调制（TCM）的译码，这种技术用于电话线连接的调制解调器中将高速率的数据挤压后通过带宽为3kHz的模拟电话线传送。在下一章中我们将介绍更多TCM的内容。多年来，具有维特比译码技术的卷积编码成为空间通信的主导FEC（前向纠错）技术，特别是在定点卫星通信网中，如VSAT（微小孔端）网。VSAT网

中最常用的变体是码率为1/2的卷积编码，所用码的约束长度为$v=7$。用这个码，我们可以传输二元的或四元的相移键控（BPSK或QPSK）信号，比无编码时至少要少5dB的功率。也就是说会比这个增加三倍以上的瓦特数。这对降低发生器和天线成本或在同样发射器功率和天线大小一样的情况下提高数据传输率很有用。

我们现在将考虑怎样用维特比译码算法来对卷积码译码。这里用的术语是，我们有一个消息向量i，编码器由此生成一个码向量c，然后通过离散无记忆信道发送。接收到的向量r可能与传输的向量c不同（除非信道是理想的或我们很幸运！）。译码器需要对消息向量做出估算。因为码向量和消息向量之间是一一对应的，译码器要对码向量做出估算。

优化译码将导致最小的译码错误概率。设$p(r|c)$为在发送的码字为c时r被收到的条件概率。我们可以说最优译码器就是最大似然译码器，其判决准则是选取使log似然函数$\ln\ p(r|c)$的值最大的那个码向量估值$\hat{c}$。

如果我们考虑BSC，其中c和r中的向量元素记为c_i和r_i，则我们有

$$p(\boldsymbol{r}\mid\boldsymbol{c})=\prod_{i=1}^{N}p(r_i|c_i) \tag{6-28}$$

故log似然函数等于

$$\ln\ p(\boldsymbol{r}\mid\boldsymbol{c})=\sum_{i=1}^{N}\ln\ p(r_i|c_i) \tag{6-29}$$

我们假设

$$p(r_i|c_i)=\begin{cases}p, & r_i\neq c_i\\ 1-p & r_i=c_i\end{cases} \tag{6-30}$$

如果我们假定接收到的向量与传输的向量恰好有d个位置不同（向量c与r之间的汉明距离），我们可以将log似然函数重新写为

$$\begin{aligned}\ln\ p(\boldsymbol{r}\mid\boldsymbol{c}) &= d\ln p+(N-d)\ln(1-p)\\ &= d\ln\left(\frac{p}{1-p}\right)+N\ln(1-p)\end{aligned} \tag{6-31}$$

我们可以假设概率$p<1/2$，而且我们注意到$N\ln(1-p)$对所有码向量都是个常数。现在我们可以宣布一个二元对称信道的最大似然译码规则就是选取码向量的估值$\hat{c}$使得接收的向量r与传输的向量c之间的汉明距离最小。

对在有单面噪声功率N_0的加性白高斯噪声（AWGN）信道中的软判决译码，其似然函数为

$$\begin{aligned}p(\boldsymbol{r}\mid\boldsymbol{c}) &= \prod_{i=1}^{N}\frac{1}{\sqrt{\pi N_0}}e^{-\frac{|r_i-c_i|^2}{N_0}}\\ &= \left(\frac{1}{\pi N_0}\right)^{\frac{1}{2}}\exp\left(-\frac{1}{N_0}\sum_{i=1}^{N}|r_i-c_i|^2\right)\end{aligned} \tag{6-32}$$

故AWGN信道中软判决译码的最大似然译码规则就是使r和c之间的欧几里德距离平方最小。这个欧几里德距离平方由下式给出：

$$d_E^2(\boldsymbol{r}|\boldsymbol{c}) = \sum_{i=1}^{N} |r_i - c_i|^2 \tag{6-33}$$

维特比译码的工作方式是选取离接收的序列最近的测试信息序列。这里的汉明距离用来度量两个序列的近似程度。维特比译码过程可很容易通过下面的例子理解。

例6.10　考虑图6-18中给出的码率为1/3的卷积码及其对应的网格图。假设传输的序列为全零序列。设接收的序列为

$$r = 010000100001 \cdots$$

因为这是个码率为1/3的编码器，我们首先将接收的序列以三个比特为一组进行分段（因为$n_0 = 3$），即

$$r = 010\ 000\ 100\ 001 \cdots$$

现在的任务是在网格中找到最可能的路径。因为网格中的路径必须要通过一些节点，我们将试图找出网格中哪些节点属于一个最可能的路径。在任何时候，每个节点都有两个进入分支（→）。我们只需确定这两个分支中的哪个属于一个较可能的路径（从而抛弃另一个）。我们根据一些尺度（汉明距离）作此决定。这样的话我们在每个节点只保留一条路径和该路径的尺度。在这个例子中，随着译码的进程我们只保留了四个路径（因为在我们的网格中只有四种状态）。

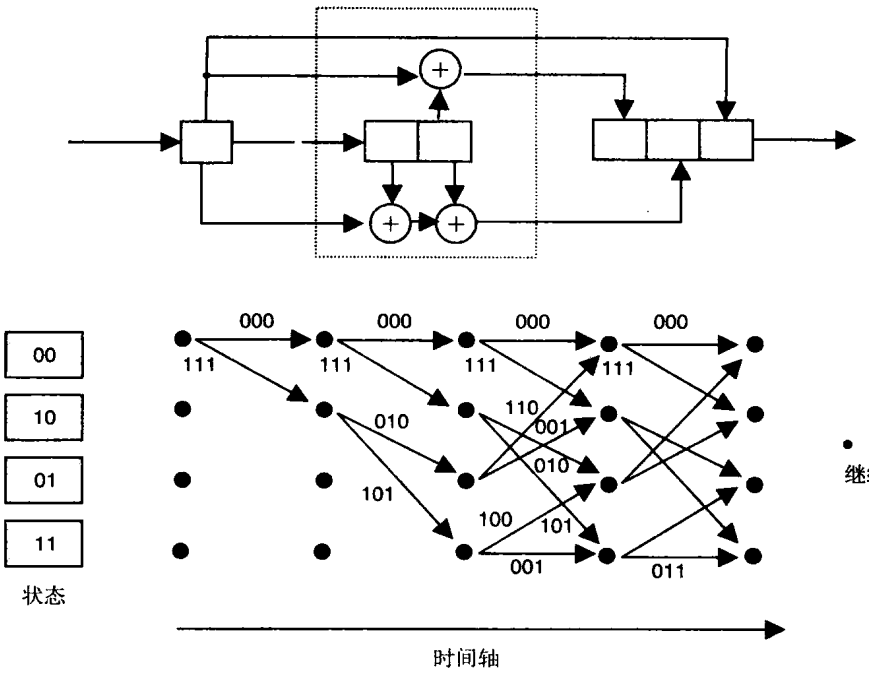


图6-18　码率为1/3的卷积编码器及其网格图

让我们考虑网格中第一个分支，它被标记为000。我们算出这个分支和接收到的第一个帧片段010之间的汉明距离$d(000, 010) = 1$。故这第一个分支的尺度为1，称为**分支尺度**（Branch Metric）。从起始节点到达最上面的节点后，该分支累加尺度＝1。接下来我们将接收到的帧片段同下面的路径比较，它在最上面的第二个节点结束。这种情况下汉明距离为$d(111, 010) =$

2。因此这第一个分支的尺度为2。在每一个节点我们写下路径累加的尺度，称为**路径尺度**(Path Metric)。路径尺度在图6-19的网格图中用圈起来的数字标记。在接下来的译码进程中，当两个路径在每一个节点结束时，我们将保留那个尺度值较小的路径。

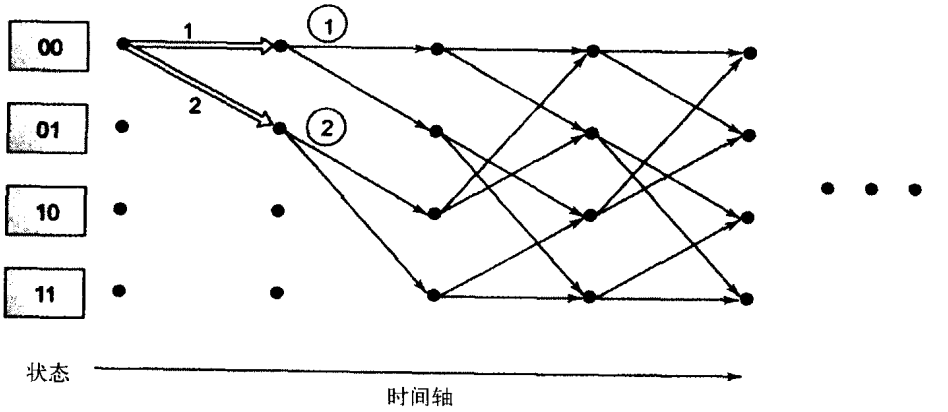


图6-19 维特比译码第一步后的路径尺度

我们现在转到网格的下一步。分支之间的汉明距离是根据接收到的第二个帧000计算的。从最上面的节点发出的两个分支的分支尺度为0和3。从第二个节点发出的两个分支的分支尺度为2和1。总的路径尺度在网格图中用圈起来的数字表示，如图6-20所示。

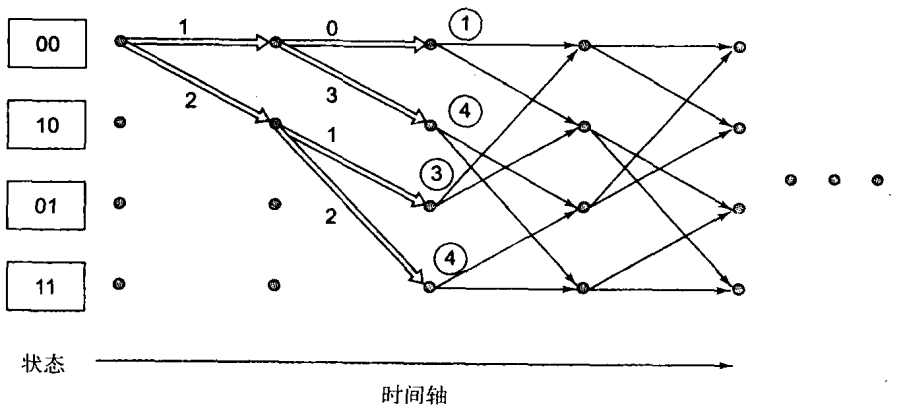


图6-20 维特比译码第二步后的路径尺度

现在进行下一步。我们再一次计算分支尺度并把它们加到相应的路径尺度以得到一个新的路径尺度。在这一步，考虑最上面的节点。有两个分支在这一节点结束。从前一步中的节点1过来的路径具有路径尺度2，而从前一步中的节点2过来的路径具有路径尺度6。保留具有低路径尺度的路径，而丢弃另一个。图6-21所示的网格图中给出了*留存下来的路径*（双线）和路径尺度（圈起来的数字）。维特比称这些留存下来的路径为**幸存者**（Survivor）。值得注意的是节点4收到两个具有相同路径尺度的路径。我们随意地选取它们中的一个作为留存下来的路径（用掷硬币决定）。

我们在下面的步骤中继续维特比译码过程。最终的分支尺度和路径尺度在图6-22中显示出来。最后我们选取尺度最小的路径。该路径对应于全零路径，因此译码过程能够正确对接

收到的向量译码。

该码的最小距离为$d^*=6$。它能纠的每个帧长度上的错误数为

$$t=\lfloor (d^*-1)/2 \rfloor = \lfloor (6-1)/2 \rfloor = 2$$

在这个例子中，每个帧长度所含的最大错误数为1。

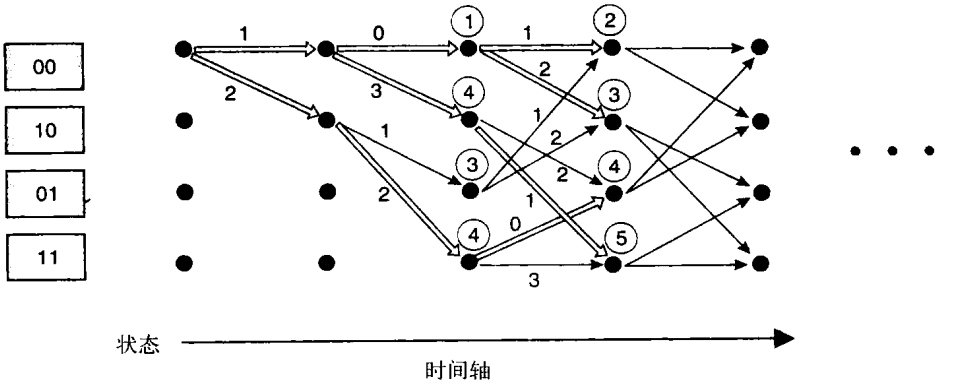


图6-21　维特比译码第三步后的路径尺度

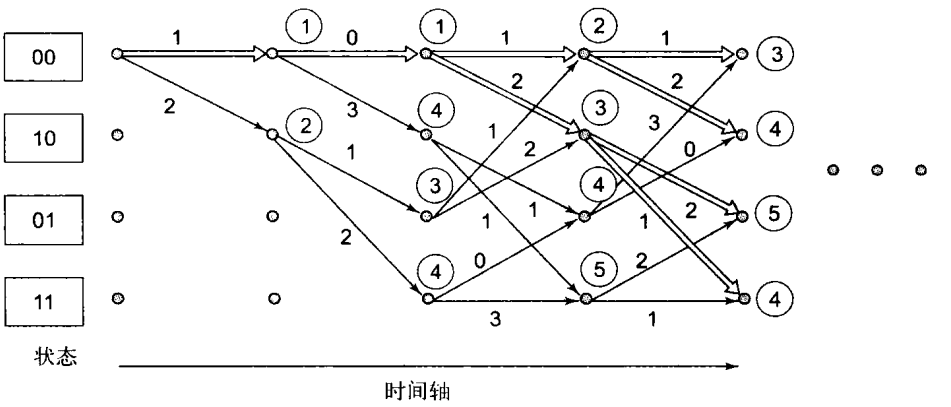


图6-22　维特比译码第四步后的路径尺度

考虑在第r个帧时间所残存的路径集合。如果所有的残存路径都经过相同的节点，则对最可能传输的路径的判决可以在那些共同节点处做出。要建立一个实际的维特比译码器，我们必须选取一个译码窗宽度w，它通常是分组长度的好几倍。在一个给定的帧时间f，译码器检查所有存活路径看它们是否在第一个分支相一致。这个分支确定了一个译过码的信息帧并从译码器中输出。在前面关于维特比译码的例子中我们看到，当译码器达到第四帧时，所有的存活路径都在它们第一个译码分支相一致（称为良定义判决）。译码器丢弃第一个分支（当送出译过码的帧之后）并从接收到的字中获取一个新的帧用于下一轮迭代。如果所有存活路径又一次通过最老的存活帧的同一个节点，则这个信息帧又被译码了。该过程就这样无限地继续下去。

如果选取充分宽的译码窗w，则良定义判决几乎总可以用得上。一个设计较好的码将导致

很高的正确译码概率。注意一个设计较好的码是指对某个特殊信道而言的。由信道产生的随机错误应该在码的纠错能力范围之内。维特比译码器可以看成是一个滑动窗，从这个窗中可以看到网格（见图6-23）。当要处理新的帧时，该窗就往右滑动。透过该窗可以看到在网格中那些标记出的存活的路径。随着窗的滑动，在其右边出现新的节点，有些存活的路径延伸到这些新的节点，而其他的路径就消失了。

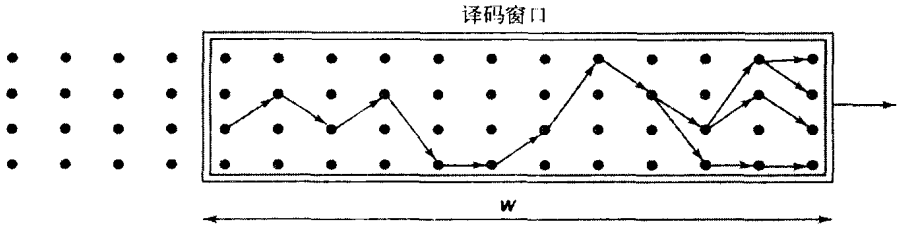


图6-23 维特比译码器就像一个滑动窗从中可以看到网格

如果存活的路径并不从同一个节点通过，我们把它标记为**译码失败**（decoding failure）。译码器可以打破常规用任意规则。在一定程度上译码器变成不完备译码器。让我们回到原来的例子。在第四步，那些存活的路径完全可以被选为图6-24所示的情况，那将致使译码器变成不完备译码器。

也可能在某些情况下译码器做出良定义判决，但结果是错的！如果发生了这种情况，该译码器无法知道它做出了错误判决。在这种错误判决的基础上，该译码器将做出更多的错误判决。但是，如果码不是灾难性的，该译码器将从错误中恢复正常。

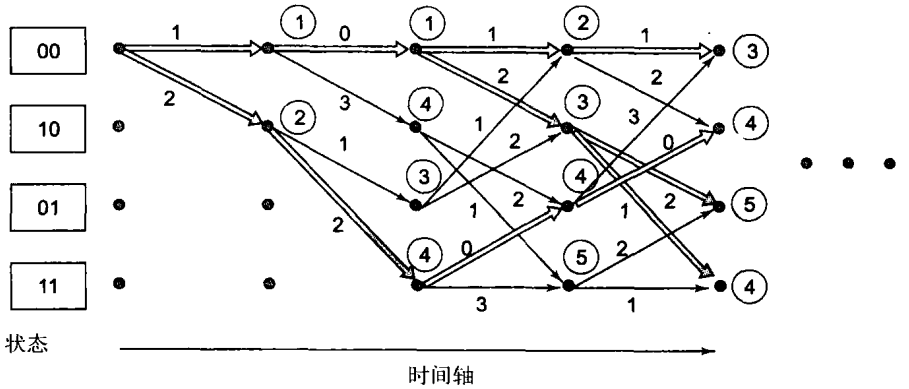


图6-24 在维特比译码过程中不完备译码器的例子

下一节讨论的是一些卷积码的**距离界**（distance bound）。这些界将有助于对不同卷积码方案进行比较。

6.8 卷积码的距离界

当卷积码的码率为$R = k_0/n_0$，码约束长度为$v = mk_0$时，其距离上界可根据最小距离算出。这些界来源于分组长度对应于约束长度的分组码的界，而且它们在本质上也是相同的。但正如我们将要看到的，这些界不是很紧。这些界只给我们关于码有多好的大致概念。这里我们对二元码的情况给出一些界（没有证明）。

对码率R和约束长度v，设d为满足下式的最大整数：

$$H\left(\frac{d}{n_0 v}\right) \leqslant 1-R \tag{6-34}$$

则至少有一个最小距离d满足上式的二元卷积码存在。这里$H(x)$是熟悉的对二元符号的熵函数

$$H(x) = -x\log_2 x-(1-x)\log_2(1-x),\ 0 \leqslant x \leqslant 1$$

对一个$R=1/n_0$的二元码，其最小距离d_{min}满足

$$d_{min} \leqslant \lfloor (n_0 v + n_0)/2 \rfloor \tag{6-35}$$

其中$\lfloor I \rfloor$表示小于或等于I的最大整数。

d_{free}的一个上界由下式给出（Heller，1968）

$$d_{free} = \min_{j \geqslant 1}\left\lfloor \frac{n_0}{2}\frac{2^j}{2^j-1}(v+j-1)\right\rfloor \tag{6-36}$$

要计算该上界，等式右边应该对j不同的整数值绘图，该上界就是这个图的最低点的值。

例6.11　让我们将距离界用到例6.1所给的卷积译码器。首先用式（6-34）所给的界。对于这个译码器，$k_0=1$，$n_0=2$，$R=1/2$且$v=2$。

$$H\left(\frac{d}{n_0 v}\right) = H(d/4) \leqslant 1-R = 1/2 \Rightarrow H(d/4) \leqslant 0.5$$

而我们有

$$H(0.11) = -0.11\log_2 0.11-(1-0.11)\log_2(1-0.11) = 0.4999, \text{且}$$
$$H(0.12) = -0.12\log_2 0.12-(1-0.12)\log_2(1-0.12) = 0.5294$$

故得$d/4 \leqslant 0.11$或$d \leqslant 0.44$。

满足这个界的最大整数为$d=0$。这表明至少有一个最小距离$d=0$的二元卷积码存在。这一声明没有说明什么问题（即该界对这个编码器不够精确）。

下面考虑图6-25所示的编码器。

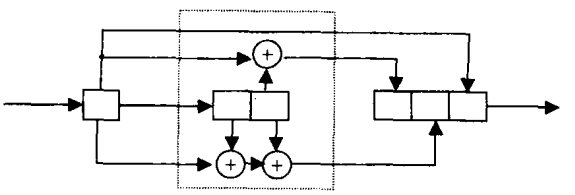


图6-25　例6.11的卷积编码器

对这个编码器有

$$\boldsymbol{G}(D) = [1 \quad D+D^2 \quad 1+D^2],$$
$$k_0 = 1,\ n_0 = 3,\ R = 1/3 \text{ 且 } v = 2.$$

$$H\left(\frac{d}{n_0 v}\right) = H(d/6) \leqslant 1-R = 2/3 \Rightarrow H(d/6) \leqslant 0.6666$$

而我们有

$$H(0.17) = -0.17\log_2 0.17 - (1-0.17)\log_2(1-0.17) = 0.6577, \text{ 且}$$
$$H(0.18) = -0.18\log_2 0.18 - (1-0.18)\log_2(1-0.18) = 0.6801$$

故得$d/6 \leqslant 0.17$或$d \leqslant 1.02$。

满足这个界的最大整数为$d=1$。这表明至少有一个最小距离$d=1$的二元卷积码存在。这是一个很松的界。

下面我们求式（6-36）给出的d_{free}的Heller界，函数

$$d(j) = \left\lfloor (n_0/2)\left(2^j/(2^j-1)\right)(\nu+j-1) \right\rfloor$$

对j取不同整数值时的平面图如图6-26所示。

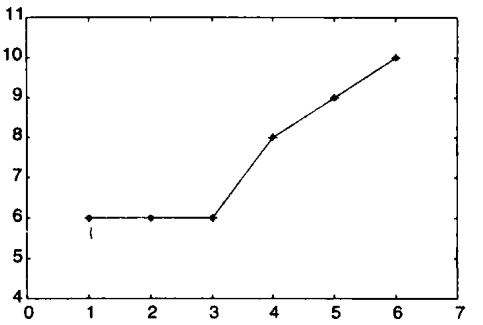


图6-26 Heler界平面图

我们得到$d_{min} \leqslant 6$，这是一个很好的上界，正如我们从该译码器的网格图（图6-26）所看到的。d_{free}=6的实际值可以从编码器的网格图中看出。用来计算d_{min}的两个路径在图6-27中用双线表示出来。在这个例子中，$d_{min} = d_{free} = 6$。

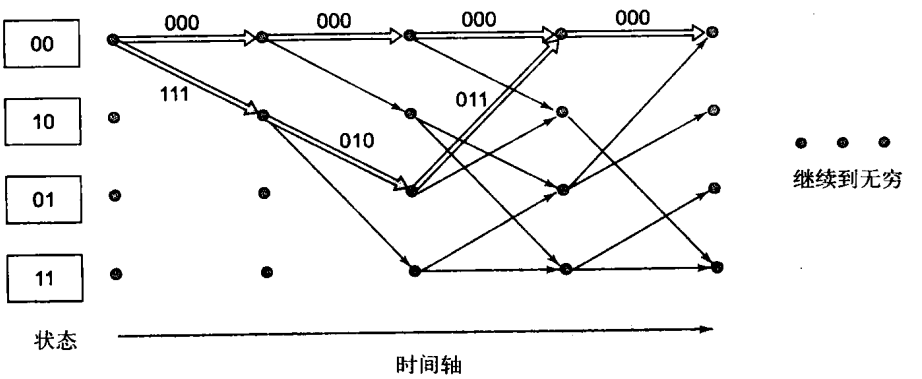


图6-27 图6-25中的卷积编码器的网格图

6.9 性能界

卷积码的有用的性能衡量指标之一是比特错误概率P_b。比特错误概率或比特错误率定义

为每个信息比特中译码信息比特错误的期望值。通常计算错误概率的一个上界而不是得到P_b的具体表达式。我们将首先确定**第一事件错误概率**（First Event Error Probability），它是在网格图中那些给定节点第一次与全零（正确）路径汇合的序列错误概率。

因为卷积码是线性的，我们假设被传送的是全零码字。如果译码器没有选全零路径而选了一个错误路径c'，它就制造了一个错误。假设c'与全零路径有d个比特不同，则如果发生了多于$\left\lfloor\frac{d}{2}\right\rfloor$个错误，最大似然译码器就做出了错误判决，其中$\lfloor x\rfloor$表示小于或等于$x$的最大整数。若信道转移概率为$p$，则出错概率的上界如下

$$P_d \leqslant \left[2\sqrt{p(1-p)}\right]^d \tag{6-37}$$

也可能有许多距离不同的路径在给定节点第一次与正确路径汇合。则第一事件错误概率的上界可以通过所有这些可能路径错误概率的和来获得：

$$P_e \leqslant \sum_{d=d_{free}}^{\infty} a_d P_d \tag{6-38}$$

其中a_d为与全零码字的汉明距离为d的码字的个数。比较式（6-19）和式（6-38）可得到

$$P_e \leqslant T(D)\big|_{D=2\sqrt{p(1-p)}} \tag{6-39}$$

现在比特错误概率P_b可以由下面的方法得到。通过对式（6-37）中的错误概率P_d和由相应错误路径导致的错误译码信息比特数n_d构成的每一个对进行加权，可以得到P_b的一个上界。对码率为k/n的译码器，P_b的平均值为

$$P_b \leqslant \frac{1}{k}\sum_{d=d_{free}}^{\infty} a_d n_d P_d \tag{6-40}$$

可以证明

$$\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1} = \sum_{d=d_{free}}^{\infty} a_d n_d D^d \tag{6-41}$$

因此

$$P_b \leq \frac{1}{k}\left.\frac{\partial T(D,I)}{\partial I}\right|_{I=1,\,D=2\sqrt{p(1-p)}} \tag{6-42}$$

6.10　著名的好卷积码

本节我们将学习一些著名的好卷积码。到目前为止，人们只知道少量的构造类卷积码。不存在一类与纠t个错误的BCH码有可比代数结构的卷积码。也没找到长约束长度的卷积码的构造方法。这里给出的大多数码都是由计算机搜索发现的。

具有最大自由距离的短卷积码的开创性工作是由Odenwalder（1970）和Larsen（1973）进行的。表6-2、6-3和6-4分别列出了码率为1/2、1/3和1/4这样的一些码。它们的生成器以序列1, $\gamma_1^{(i)}$, $\gamma_2^{(i)}$, …的形式给出，其中

$$g_i(D)=1+\gamma_1^{(i)}D+\gamma_2^{(i)}D^2+\cdots+\gamma_{K-1}^{(i)}D^{K-1} \tag{6-43}$$

例如$R=1/2$，$v=4$的编码器生成器的八进制表示为15和17（见表6-2）。八进制15可以解释为$15=1-5=1-101$。因此

$$g_1(D)=1+(1)D+(0)D^2+(1)D^3=1+D+D^3$$

类似地，

$$17=1-7=1-111$$

因此

$$g_2(D)=1+(1)D+(1)D^2+(1)D^3=1+D+D^2+D^3,\quad \text{从而}$$
$$\boldsymbol{G}(D)=[1+D+D^3\ 1+D+D^2+D^3]$$

表6-2 码率为1/2的最大自由距离码

v	n	生成器（八进制）		d_{free}	Heller界
非灾难性的					
3	6	5	7	5	5
4	8	15	17	6	6
5	10	23	35	7	8
6	12	53	75	8	8
7	14	133	171	10	10
灾难性的					
5	10	27	35	8	8
12	24	5237	6731	16	16
14	28	21645	37133	17	17

表6-3 码率为1/3的最大自由距离码

	v	n	生成器（八进制）			d_{free}	Heller界
码率为1/3	3	9	5	7	7	8	8
	4	12	13	17	17	10	10
	5	15	25	37	37	12	12
	6	18	47	75	75	13	13
	7	21	133	175	175	15	15

表6-4 码率为1/4的最大自由距离码

	v	n	生成器（八进制）				d_{free}	Heller界
码率为1/4	3	12	5	7	7	7	10	10
	4	16	13	15	15	17	15	15
	5	20	25	27	33	37	16	16
	6	24	53	67	71	75	18	18
	7	28	135	135	147	163	20	20

下面我们学习一类有趣的码，称为Turbo码，它介于线性分组码和卷积码之间。

6.11 Turbo码

Turbo码是1993年在国际通信学术会议（International Conference on Communications, ICC）上由Berrou、Glavieux和Thitimajshima在他们的论文“接近Shannon限的纠错编码和译码——Turbo码”中介绍的。在这篇论文中，他们引用了当E_b/N_0为0.7dB时只用1/2的码率就使BER达到10^{-5}的性能，从而产生了在该领域的巨大兴趣。在SNR很低的情况下Turbo码性能良好。在SNR很高的情况下，某些传统的码如Reed-Solomon码有着与Trubo码可比的或更好的性能。

尽管Turbo码被当做分组码考虑，他们并不完全像分组码那样运作。Turbo码实际是分组码和卷积码的准混合物。它们像分组码一样要求在编码前要给出整个组，但它们不是从一个方程组中计算奇偶校验比特，而是像卷积码那样用移位寄存器。

Turbo码用到至少两个卷积部件编码器和两个最大经验（MAP）算法部件译码器。这种做法称为**级联**。Turbo码有三种不同的排列，分别称为**并行级联卷积码**（Parallel Concatenated Convolutional Code, PCCC）、**串行级联卷积码**（Serial Concatenated Convolutional Code, SCCC），和**混杂级联卷积码**（Hybrid Concatenated Convolutional Code, HCCC）。通常Turbo码排列为PCCC。图6-28给出的一个Turbo码的PCCC编码器显示两个编码器并行工作。

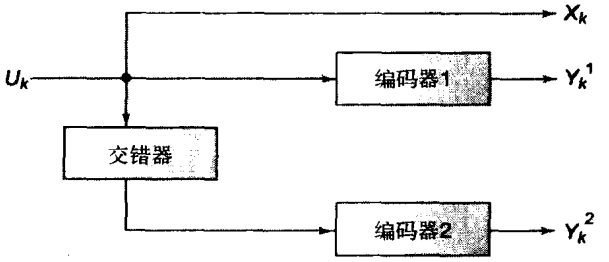


图6-28　码率为1/3的PCCC Turbo码编码器框图

定义6.13　交织器π是对具有N个输入符号$d_1, d_2, \cdots, d_N$的数据序列进行顺序改变的置换$i \to \pi(i)$。

如果输入数据序列是$\boldsymbol{d} = [d_1, d_2, \cdots, d_N]$，则置换后的数据序列是$\boldsymbol{dP}$，其中$P$是一个每行和每列均只有一个1，剩余位置为0的交织矩阵。每一个交织器都有一个对应的逆交织器π^{-1}，其作用于一个已交织后的数据序列，从而恢复成此序列之前的顺序。逆交织矩阵是交织矩阵的一种变换$\boldsymbol{P}^T$。

Turbo码性能良好的一个原因是它们产生重量大的码字。例如如果输入序列（U_k）原来重量很低，系统（X_k）和奇偶校验1的输出（Y_k^1）可能产生重量低的码字。但奇偶校验2的输出（Y_k^2）则不太可能是重量低的码字，原因是它前面的交错过程。该交错过程把输入序列U_k搅乱成在进入第二个编码器时很可能产生一个高重量的码字。这对于码来说是很理想的，因为大重量的码字能得到好的译码器性能。直观上，当一个编码器产生一个“弱”码字时，由于交错的原因，另一个编码器产生另一个“弱”码字的概率很低。因此两个码字级联之后就变成一个“强”的码字。这里“弱”被用作其他所有码字的平均汉明距离的一个度量。

尽管编码器决定了纠错的能力，而译码器才决定实际的性能。但是，这种性能依赖于所用的算法是哪一种。由于Turbo译码是一个迭代过程，它需要软输出算法，如译码最大经验算法（MAP）或软输出维特比算法（SOVA）。软输出算法强于硬判决算法，因为它们能更好地估算实际发送的数据是什么。这是因为软输出对计算出的信息比特有一种判决倾向，而不像硬输出直接选取1或0。图6-29给出了一个典型的Turbo译码器。

MAP算法常用于估算一个编码序列中最可能传送的信息比特。人们偏爱MAP算法因为在低SNR条件下它比其他算法性能好，比如SOVA。但是它的主要缺点是它比多数算法都复杂，因为它关注信息的每一个比特。在此领域的研究（在20世纪90年代后期）大大简化了MAP算法。

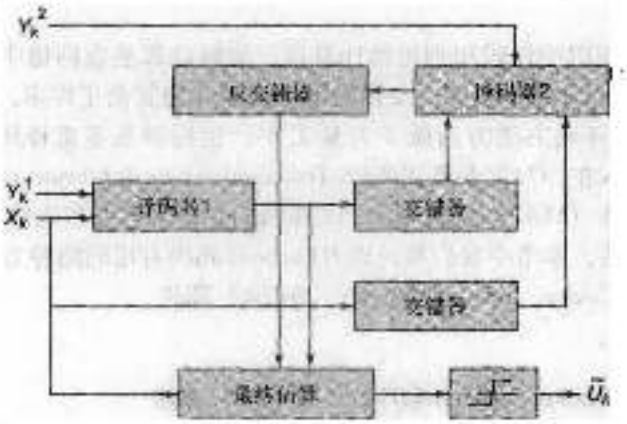


图6-29 Turbo译码器框图

Turbo译码器一般在至少一个部件译码器中使用MAP算法。当从信道中接收到部分信息（X_k和Y_k^1）并传递给第一个译码器时就开始了译码过程。余下的奇偶校验2信息（Y_k^2），则进入第二个译码器并等待其余信息的到来。第二个译码器在等待的同时，第一个译码器对传递的信息做出评估，进行交错使之与奇偶校验2的格式相同，然后发送到第二个译码器。第二个译码器根据从第一个译码器和信道来的信息重新评估传送的信息。这第二次的评估又被循环回到译码过程开始时的第一个译码器。Turbo译码器的迭代过程如图6-30所示。

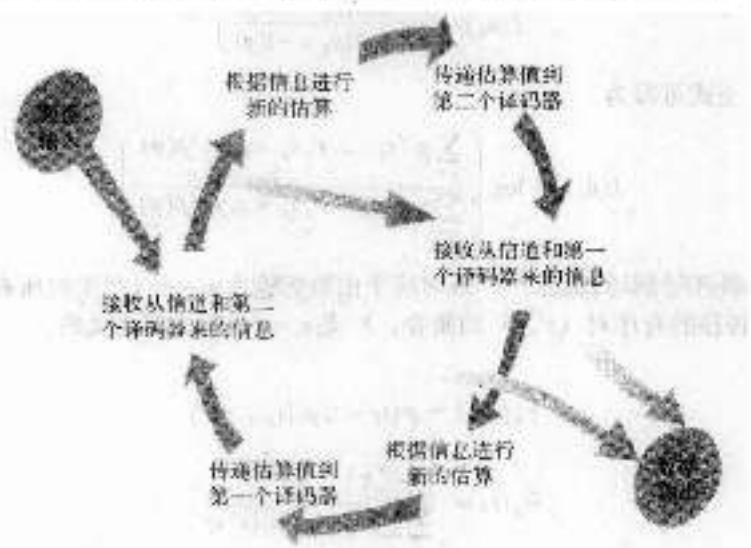


图6-30 Turbo码的迭代译码

这种循环将进行下去，直到满足某些条件，如已经迭代了多少次。Turobo码就是因为这种迭代而得名。译码器对所发送数据进行循环评估，就像涡轮引擎循环空气一样。当译码器做好评估后，所评估出的信息终于被从循环中踢出来，然后在门限组件部分做硬判决译码。

最后的结果是译码后的信息序列。

在下一节中我们学习两种Turbo码的详细译码方法。

6.12 Turbo译码

我们已经看到卷积码的译码用到维特比算法。维特比算法在网格中系统地排除路径。但是，对Turbo译码器就不这么幸运了。交错器的出现使问题复杂了许多。在发现Turbo码之前，人们在级联码的次优译码方法方面做了大量工作，包括涉及多重译码器的情况。由Bahl、Cocke、Jelinek和Raviv在1974年发表在*IEEE Transactions on Information Theory*上的依次按符号译码的最大经验算法（MAP）也受到人们的重视。Berrou等人在他们的Turbo码的迭代译码中使用的就是这种算法。本节中我们将讨论对Turbo译码很有用的两种方法：

(1) 改进的Bahl、Cocke、Jelinek和Raviv（BCJR）算法。

(2) 迭代MAP译码。

6.12.1 改进的Bahl、Cocke、Jelinek和Raviv（BCJR）算法

改进的BCJR算法是个符号接符号译码器。当满足

$$P(u_k = +1 \mid \mathbf{y}) > P(u_k = -1 \mid \mathbf{y}) \tag{6-44}$$

时，该译码器判决$u_k = +1$，否则它判决$u_k = -1$，其中$\mathbf{y} = (y_1, y_2, \ldots, y_n)$是接收到的有噪字。

更简洁地说，判决$\hat{u}_k$是根据

$$\hat{u}_k = \text{sign}\,[L(u_k)] \tag{6-45}$$

给出的，其中$L(u_k)$是**log后验概率比率**（LAPP），其定义为

$$L(u_k) = \log\left(\frac{P(u_k = +1|\mathbf{y})}{P(u_k = -1|\mathbf{y})}\right) \tag{6-46}$$

结合码的网格，上式可写为

$$L(u_k) = \log\left(\frac{\sum\limits_{S^+} p\,(s_{k-1} = s', s_k = s, \mathbf{y})/p(\mathbf{y})}{\sum\limits_{S^-} p\,(s_{k-1} = s', s_k = s, \mathbf{y})/p(\mathbf{y})}\right) \tag{6-47}$$

其中$s_k \in S$是编码器在时刻k的状态，S^+是对应于由数据输入$u_k = +1$产生的所有从状态（$s_{k-1} = s'$）到（$s_k = s$）转移的有序对（s', s）的集合，S^-是$u_k = -1$时类似定义的。

我们定义

$$\gamma_k(s', s) = p(s_k = s, y_k | s_{k-1} = s') \tag{6-48}$$

$$\tilde{\alpha}_k(s) = \frac{\sum\limits_{s'} \alpha_{k-1}(s')\gamma_k(s', s)}{\sum\limits_{s'}\sum\limits_{s'} \alpha_{k-1}(s')\gamma_k(s's)} \tag{6-49}$$

$$\tilde{\beta}_{k-1}(s') = \frac{\sum\limits_{s'} \tilde{\beta}(s)\gamma_k(s', s)}{\sum\limits_{s'}\sum\limits_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s', s)} \tag{6-50}$$

上述的边界条件为

$$\alpha_0(0)=1 \text{ 且 } \alpha_0(s\neq 0)=0$$
$$\beta_0(0)=1 \text{ 且 } \beta_0(s\neq 0)=0 \tag{6-51}$$

则改进的BCJR算法给出如下形式的LAPP比率

$$L(u_k)=\log\left(\frac{\sum_{s^+}\tilde{\alpha}_{k-1}(s')\gamma_k(s',s)\tilde{\beta}(s)}{\sum_{s^-}\tilde{\alpha}_{k-1}(s')\gamma_k(s',s)\tilde{\beta}_k(s)}\right) \tag{6-52}$$

6.12.2 迭代MAP译码

图6-31给出了译码器示意图。在图中D1和D2是两个译码器，S是2^m个要素译码器状态的集合，y是接收到的有噪字。根据贝叶斯规则（Baye's rule）我们可将$L(u_k)$写为

$$L(u_k)=\log\left(\frac{P(y|u_k=+1)}{P(y|u_k=-1)}\right)+\log\left(\frac{P(u_k=+1)}{P(u_k=-1)}\right) \tag{6-53}$$

其中第二项表示的是先验信息。典型的情况是$P(u_k=+1)=P(u_k=-1)$，从而对传统的译码器来说先验项通常为零。但是，对于迭代译码器，对每一个u_k，D1收到来自D2的外在或软信息作为先验信息。类似地，D2也收到来自D1的外在信息，于是译码迭代就这样在两个译码器之间进行下去：除最初的半轮迭代外，每半轮迭代从一个译码器到另一个译码器传递软信息。外在信息是指对于每一个u_k，D2利用D1得不到的信息为D1提供软信息，D1为D2做着类似的事。

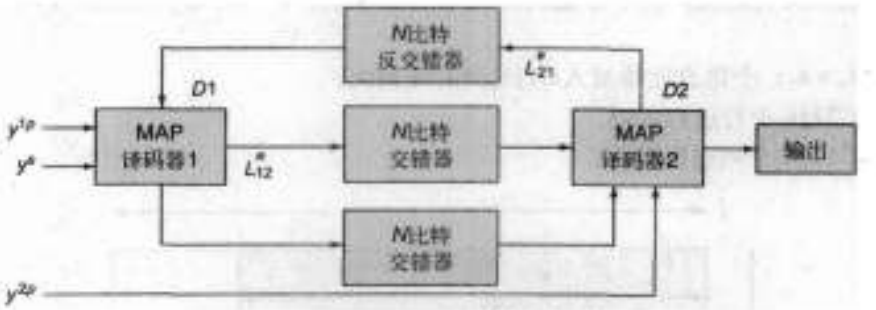


图6-31 Turbo译码器的迭代实现

每一次迭代，D1计算

$$L_1(u_k)=L_c y_k^1+L_{21}^e(u_k)+L_{12}^e(u_k) \tag{6-54}$$

其中第一项是信道值，$L_c=4E_c/N_0$（E_c=每信道比特的能量），$L_{21}^e(u_k)$是从D2传递给D1的外在信息，而$L_{12}^e(u_k)$是从D1传递给D2的外在信息。$L_{12}^e(u_k)$和$L_{21}^e(u_k)$的计算如下：

$$L_{12}^e(u_k)=\log\left(\frac{\sum_{S^+}\tilde{\alpha}_{k-1}^{(1)}(s')\gamma_k^e(s',s)\tilde{\beta}_k^{(1)}(s)}{\sum_{S^-}\tilde{\alpha}_{k-1}^{(1)}(s')\gamma_k^e(s',s)\tilde{\beta}_k^{(1)}(s)}\right) \tag{6-55}$$

$$L_{21}^e(u_k)=\log\left(\frac{\sum_{S^+}\tilde{\alpha}_{k-1}^{(2)}(s')\gamma_k^e(s',s)\tilde{\beta}_k^{(2)}(s)}{\sum_{S^-}\tilde{\alpha}_{k-1}^{(2)}(s')\gamma_k^e(s',s)\tilde{\beta}_k^{(2)}(s)}\right) \tag{6-56}$$

其中

$$\gamma_k^i(s',s) = \exp\left[\frac{1}{2}L_c\gamma_k^p x_k^p\right] \tag{6-57}$$

$$\gamma_k(s',s) = \exp\left[\frac{1}{2}u_k\left(L^e(u_k)+L_c y_k^s\right)\right]\cdot\gamma_k^i(s',s) \tag{6-58}$$

$$\bar{\alpha}_k(s) = \frac{\sum_{s'}\alpha_{k-1}(s')\gamma_k(s',s)}{\sum_s\sum_{s'}\alpha_{k-1}(s')\gamma_k(s',s)} \tag{6-59}$$

$$\bar{\beta}_{k-1}(s') = \frac{\sum_{s}\bar{\beta}_k(s)\gamma_k(s',s)}{\sum_s\sum_{s'}\bar{\alpha}_{k-1}(s')\gamma_k(s',s)} \tag{6-60}$$

对上面的算法，每一个译码器必须拥有组成编码器的网格的全部知识，即每个译码器必须有一个表包含所有从s'到s的状态转移的输入比特和奇偶校验比特。而且，应该注意要编码的N-比特信息字中的最后m比特必须使译码器1在第N个比特时状态为零。

卷积码的复杂性使低成本的Turbo卷积码（TCC）译码器发展很慢。另一方面，另一类Turbo码，称为Turbo乘积码（TPC），用到分组码，同时处理多个步骤，因此得到高硬件数据处理量。我们这里对乘积码给出简单的介绍。我们考虑两个系统线性分组码$C1$，其参数为(n_1, k_1, d_1)，和$C2$，它的参数为(n_2, k_2, d_2)，其中n_i，k_i和d_i（$i=1,2$）分别表示码字长度、信息比特数和最小汉明距离。两个分组码的串联（或乘积）$P=C_1*C_2$（见图6-32）可由下列步骤得到：

(1) 把$(k_1\times k_2)$个信息比特放入k_1行k_2列的阵列内，

(2) 用码$C2$对k_1个行进行编码，

(3) 用码$C1$对n_2个列进行编码。

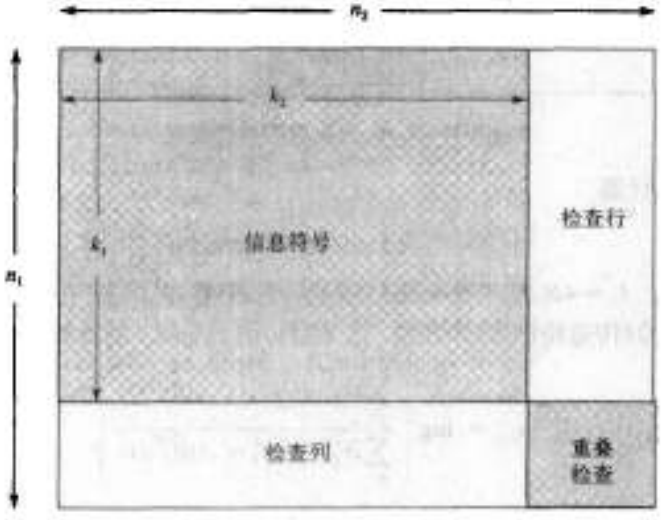


图6-32　一个乘积码$P=C_1*C_2$的例子

乘积码$\boldsymbol{P}$的参数为：$n=n_1*n_2$，$k=k_1*k_2$，$d=d_1*d_2$而且码率$R=R_1*R_2$，其中R_i是码$\boldsymbol{C}_i$的码率。因此我们可以通过把短的具有小的汉明距离的码结合起来得到具有大最小汉明距离的很长的分组码。根据构造乘积码的过程，很明显矩阵的最后（n_2-k_2）个列是$\boldsymbol{C}_1$的码字。利用矩阵生成器，我们可以证明矩阵$\boldsymbol{P}$的后面的行是$\boldsymbol{C}_2$的码字。因此矩阵$\boldsymbol{P}$的所有行都是$\boldsymbol{C}_1$的码字，矩阵$\boldsymbol{P}$的所有列都是$\boldsymbol{C}_2$的码字。

现在我们考虑用QPSK信号通过高斯信道后的乘积码$\boldsymbol{P}$的行和列的译码问题。当接收到对应于传输码字为$\boldsymbol{E}$的矩阵$\boldsymbol{R}$后，第一个译码器以矩阵$\boldsymbol{R}$为输入对$\boldsymbol{P}$的行（或列）进行软译码。软输入/输出译码使用由R. Pyndiah提出的新算法。从软输出中减除软输入可得到外在信息$W(2)$，其中索引号2表示我们考虑用于P的第二步译码的外部信息，它是在$\boldsymbol{P}$的第一步译码过程中计算而来的。在对$\boldsymbol{P}$的第二步译码中对列（或行）译码的软输入为

$$\boldsymbol{R}(2)=\boldsymbol{R}+a(2)\boldsymbol{W}(2) \tag{6-61}$$

其中$a(2)$为缩放因子，它应考虑到矩阵$\boldsymbol{R}$与矩阵$\boldsymbol{W}$中的采样标准方差是不同的。外在信息的标准方差在第一步译码中是很高的，但当我们迭代译码过程时它在不断降低。当BER相对较高时，缩放因子a也用于减少外部信息在第一步译码的软译码器中的影响。它在第一步译码中取一个较小的值，然后随BER趋于0而不断增加。上述的译码过程可以推广为基本级联译码器，如图6-33所示。

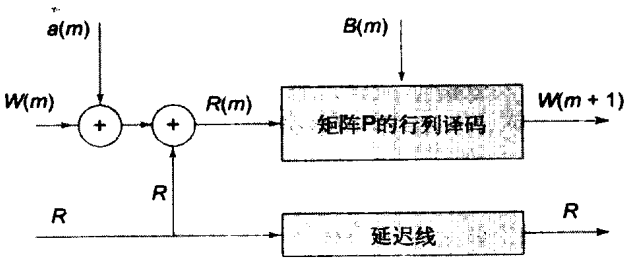


图6-33 基本分组Turbo译码器框图

现在让我们简单看一下Turbo码的性能，并将它同其他已有的方案作比较。正如图6-34所示，在低SNR条件下，Turbo码是最实用的码（在高SNR的情况下，Reed-Solomon码的性能可以超过Turbo码）。从图中可以明显看到，递归系统卷积（RSC）Turbo码是目前已知的最实用的码，因为它可以在低SNR的情况下得到低BER，而且最接近理论上的最大信道性能，即Shannon限。它的性能取决于编码增益。可以说编码增益就是对相同BER性能时编码信道和未编码信道的SNR之差。编码增益可以通过度量在某一给定BER下编码信道的SNR和未编码信道的SNR之间的距离得到。例如，码率为1/2，BER为10^{-5}的RSC Turbo码的编码增益约为8.5dB。其实际产生的结果如下所示。考虑空间通信，其中接收功率服从逆平方定律（$P_R\propto 1/d^2$）。这说明Turbo编码后的信号比未编码信号可以在2.56（$=\sqrt{7}$）倍远的地方收到（相同传输功率），或者只需要1/7的传输功率（对相同传输距离）。从另一个角度看，这个问题就是谈论移动装置的电池寿命问题。比如，因为RSC Turbo编码信道只需要未编码信道1/7的功率，我们可以说一个使用Turbo码的装置，比如一部手机，其电池寿命要比未使用任何信道编码的装置长7倍。

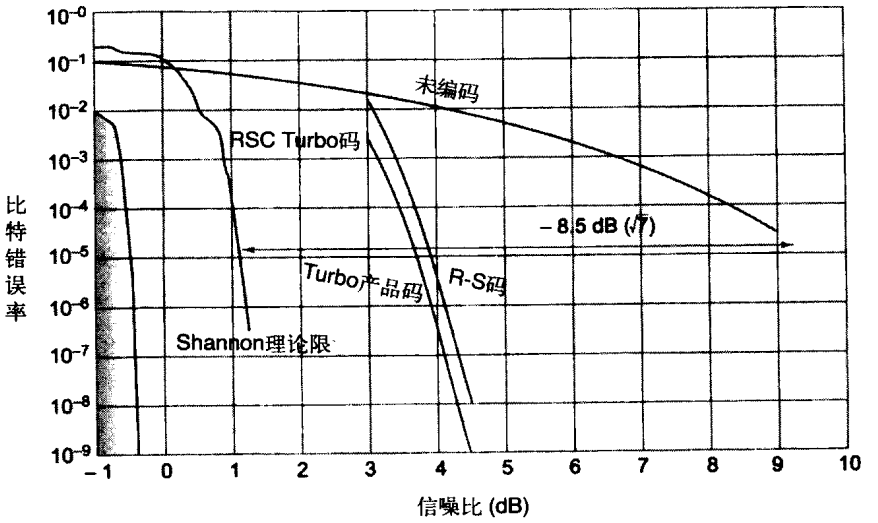


图6-34　不同编码系统的比较

6.13　Turbo码的交织器设计

当交织器的长度比较大，大约几千比特时，Turbo码相较于卷积码（convolutional code）有很大的性能优势。对大分组长度的交织器，大多数随机交织器效果不错。另一方面，为了降低交织/解交织操作对硬件的需求，某些应用更偏好于使用确定性交织器。对小分组长度的交织器，使用随机交织器的Turbo码的性能将明显降低到某一点，Turbo码在此点的误码率（BER）性能比相似计算复杂度的卷积码的BER要差。对小分组长度的交织器，交织器种类的选择对Turbo码的性能有显著的影响。在很多应用中（比如音频），在选择分组长度时延迟是一个需要考虑的问题。因此对这些应用，就需要设计一个小分组长度的交织器使得BER性能在可接受范围内。

在交织器设计中，有以下两个主要的标准：

(1) 码的距离频谱（重量分布）属性，

(2) 每一个对应于奇偶校验位的解码器的软输出和信息输入数据序列之间的相关性。

上面的第二个关键点有时称为**迭代译码有效性**（IDS）标准。

定义6.14　一个**随机交织器**（Random Interleaver）是一个简单的随机置换π。

定义6.15　一个**S随机交织器**(S=1, 2, 3, …)是一个按照下面程序构造出来的“半随机”交织器：每一个所选择的随机整数需要和前面S个已选择的随机数进行比较。如果此选择的随机数和前面S个随机数的差值小于S，则此选择的随机数被拒绝（需要重新选择）。这种选择过程一直持续到选择完所有N个不同的整数。

例6.12　3GPP计划（Third Generation Partnership Project）建议Turbo码的交织器算法由一系列把长度为K ($40 \leqslant K \leqslant 5114$) 的输入序列映射为交织序列的数学复杂过程构成。算法概述如下：

第1步：以列为主型，（逐行写入）把数据输入转换为一个$R \times C$的矩阵，如果$K<R \times C$，则用0填充。

第2步：基于一个递归构造而成的基序列s，对矩阵进行行内置换。

第3步：基于一个明确的列之间的置换模式T和一个置换的最小素数序列r（由T和一个有序的最小素数序列q得到），对矩阵进行行间置换。

第4步：以行为主型，（逐列读出）从矩阵中输出数据（输出过程进行过剪切）。

6.14 评注

卷积码的概念首先是由Elias（1954）提出的，之后由Wozencraft（1957）和Ash（1963）进行了发展。Massey（1963）提出了一类纠多个错误的卷积码。Massey（1968）和Forney（1970）对卷积码进行了代数结构的研究。

维特比译码是由Qualcomm有限公司创始人Andrew J. Viterbi发展的。他在该项技术上的精辟论文“卷积码的错误界和一种渐近最优译码算法”发表在1997年4月的*IEEE Transactions on Information Theory*第13期第260～269页。在1968年，Heller证明了若约束长度不是太大，维特比译码算法是实用的。

Turbo码代表了在纠错方面的又一次飞跃。Turbo码在1993年的国际通信学术会议（ICC）上由Berrou、Glavieux和Thitimajshima在他们的论文“近Shannon限纠错编码及译码——Turbo码”中提出来的。这类码名字的由来是因为译码数据在译码器中要来回循环好几次。发明者可能是发现了涡轮增压是怎么工作的。Turbo码被证明在BER为10^{-5}情况下，其性能离Shannon限只有1dB。它们把一个复杂的译码问题分解成简单的步骤，每一个步骤都在不断重复直到得到满意答案为止。术语“Turbo码”通常用于指Turbo卷积码（TCC）——Turbo码的一种形式。Bahl、Cocke、Jelinek和Raviv在1974年（比Turbo码的引入早了19年）发表的符号接符号最大后验（MAP）算法被Berrou等人用于他们的Turbo码的迭代译码。另一方面，另一种Turbo码，称为Turbo乘积码（TPC），用到分组码，同时求解多个步骤，因此可以得到高的硬件数据处理量。

6.15 小结

- 树码的一个重要子类是卷积码。卷积码根据过去的信息做出判决，即需要有记忆。一个(k_0, n_0)树码如果是线性的、不随时间变化的，而且有着有限码字长度$k=(m+1)k_0$，则称为一个(n,k)卷积码。
- 卷积码用到小得多的长度为k_0的未编码数据组。这些称为信息帧。这些信息帧被编码为长度为n_0的码字帧。该树码的码率定义为$R = k_0 / n_0$
- 一个移位寄存器编码器的约束长度定义为它的内存中所能储存的符号的个数。
- 卷积码大小为$k_0 \times n_0$的生成多项式矩阵可表示为$\boldsymbol{G}(D)=[g_{ij}(D)]$，其中$g_{ij}(D)$为该码的生成多项式。$g_{ij}(D)$是沿从输入$i$到输出$j$的路径得到的。
- 卷积码的码字长度为$k = k_0 \max\limits_{i,j}[\deg\ g_{ij}(D)+1]$，分组长度为$n = n_0 \max\limits_{i,j}[\deg\ g_{ij}(D)+1]$，约束长度为$v = \sum\limits_{i=1}^{k_0} \max\limits_{j}[\deg\ g_{ij}(D)]$。
- 编码操作可简单描述为向量矩阵乘积，$\boldsymbol{C}(D)=\boldsymbol{I}(D)\boldsymbol{G}(D)$，或等价地表示为

$$c_j(D) = \sum_{l=1}^{k_0} i_l\ (D) g_{ij}(D)$$

- 奇偶校验矩阵$\boldsymbol{H}(D)$是一个（n_0-k_0）行n_0列的多项式矩阵，且满足$\boldsymbol{G}(D)\boldsymbol{H}(D)^T=\boldsymbol{0}$，而有（$n_0-k_0$）个分量行向量的伴随式多项式向量表示为$\boldsymbol{s}(D)=\boldsymbol{v}(D)\boldsymbol{H}(D)^T$。
- 一个卷积码的系统编码器有形如$\boldsymbol{G}(D)=[\boldsymbol{I}|\boldsymbol{P}(D)]$的生成多项式矩阵，其中$\boldsymbol{I}$是$k_0\times k_0$阶单位矩阵，$\boldsymbol{P}(D)$为$k_0\times(n_0-k_0)$阶多项式矩阵。一个系统卷积编码器的奇偶校验多项式矩阵为$\boldsymbol{H}(D)=[-\boldsymbol{P}(D)^T|\boldsymbol{I}]$。
- 若一个卷积码的生成多项式$g_1(D), g_2(D), \cdots, g_{n_0}(D)$对某个整数$a$满足$GCD[g_1(D), g_2(D), \cdots, g_{n_0}(D)]=D^a$，则该码称为非灾难性卷积码。否则称为灾难性卷积码。
- 一个卷积码的第l级最小距离d_l^*等于任意两个最初l个帧不相同的码字之间的最小汉明距离。若$l=m+1$，则第（$m+1$）级最小距离称为该码的最小距离，记为d^*，其中m是编码器内存中所能储存的信息帧的个数。在文献中，最小距离又记为d_{min}。
- 如果一个卷积码的第l级最小距离为d_l^*，则当$d_l^*\geqslant 2t+1$时，该码可纠前l个帧上发生的t个错误。一个卷积码的自由距离为$d_{free}=\max\limits_{l}[d_l]$。
- 卷积码的自由长度n_{free}为最小非零重量的卷积码字的非零部分的长度。因此若$l=n_{free}$，则$d_l=d_{free}$，且当$l<n_{free}$时有$d_l<d_{free}$。在文献中，n_{free}又记为n_∞。
- 另外一种计算卷积码d_{free}的方法用到生成函数的概念，它的扩张直接提供了所有的距离信息。
- 卷积码的生成矩阵可表示为

$$\boldsymbol{G}=\begin{bmatrix}\boldsymbol{G}_0 & \boldsymbol{G}_1 & \boldsymbol{G}_2 & \cdots & \boldsymbol{G}_m & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots\\ \boldsymbol{0} & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{G}_0 & \cdots & \boldsymbol{G}_{m-2} & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m & \boldsymbol{0} & \boldsymbol{0} & \cdots\\ \vdots & \vdots & & & & & & & & \vdots\end{bmatrix}$$

- 维特比译码技术是卷积码的一种有效译码方法。它需要的计算量的增长速度是约束长度的某个指数函数。
- 对码率R和约束长度，设d为满足$H\left(\dfrac{d}{n_0 v}\right)\leqslant 1-R$的最大整数。则至少有一个最小距离$d$满足上述不等式的二元卷积码存在，这里$H(x)$是我们熟悉的关于二元字母集的熵函数。
- 对一个满足$R=1/n_0$的二元码，其最小距离满足$d_{min}\leqslant\lfloor(n_0 v+n_0)/2\rfloor$，其中$\lfloor I\rfloor$表示不大于$I$的最大整数。
- Heller给出的d_{free}的一个上界为$d_{free}=\min\limits_{j>1}\left\lfloor\dfrac{n_0}{2}\dfrac{2^j}{2^j-1}(v+j-1)\right\rfloor$。为了计算上界，等式右边需要对不同的整数值$j$做平面图，该图的最小值就是所求上界。
- 卷积码的第一个错误概率的上界可以由$P_e\leqslant T(D)\Big|_{D=2\sqrt{p(1-p)}}$得到，而比特错误概率可由$P_b\leqslant\dfrac{1}{k}\dfrac{\partial T(D,I)}{\partial I}\Big|_{I=1,\ D=2\sqrt{p(1-p)}}$得到。
- Turbo码实际为一种介于分组码和卷积码之间的类混合物。Turbo码典型地用到至少两个卷积部件编码器和两个最大后验（MAP）算法部件译码器。尽管编码器决定码的纠错能力，但还是译码器决定实际译码性能。
- 对小分组长度的交织器，交织器种类的选择对Turbo码的性能有显著的影响。

去做一些不可能的事情也是很有趣的。

——Walt Disney（1901—1966）

习题

6.1 设计一个码率为1/2的卷积码使其约束长度$v=4$且$d^*=6$。

(1) 构造该编码器的状态图。

(2) 构造该编码器的网格图。

(3) 该码的d_{free}是什么？

(4) 给出生成矩阵G。

(5) 该码是不是非灾难性的？为什么？

6.2 设计一个（12, 3）系统卷积编码器使其约束长度$v=3$且$d^*\geqslant 8$。

(1) 构造该编码器的网格图。

(2) 该码的d_{free}是什么？

6.3 考虑图6-35所示的二元编码器：

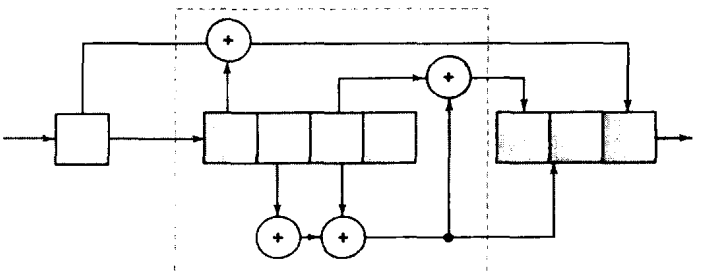


图 6-35

(1) 构造该编码器的网格图。

(2) 记下该编码器的k_0、n_0、v、m及R的值。

(3) 该码的d^*和d_{free}的值各是多少？

(4) 给出该编码器的生成多项式矩阵。

6.4 考虑图6-36所示的二元编码器。

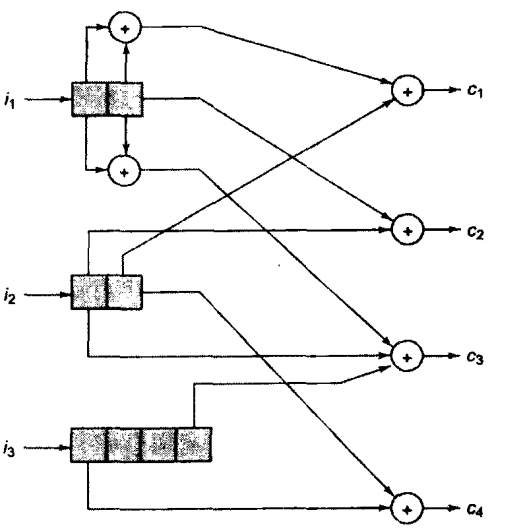


图 6-36

(1) 写出该编码器的k、n、v、m及R的值。

(2) 给出该编码器的生成多项式矩阵$\boldsymbol{G}(D)$。

(3) 给出该编码器的生成矩阵$\boldsymbol{G}$。

(4) 给出该编码器的奇偶校验矩阵H。

(5) 该码的d^*、d_{free}和n_{free}的值各是什么？

(6) 该编码器在d_{free}的Heller界上是最优的吗？

(7) 用该编码器将下列比特序列进行编码：101 001 001 010 000。

6.5 考虑一个由下列定义在$GF(2)$上的生成多项式矩阵所描述的卷积编码器：

$$\boldsymbol{G}(D)=\begin{bmatrix} D & 0 & 1 & D^2 & D+D^2 \\ D^2 & 0 & 0 & 1+D & 0 \\ 1 & 0 & D^2 & 0 & D^2 \end{bmatrix}$$

(1) 画出用移位寄存器实现该编码器的电路图。v的值是什么？

(2) 这个码是灾难性的吗？为什么？

(3) 该码在d_{free}的Heller界上是最优的吗？

6.6 当$m=2$时，(12, 9) Wyner-Ash码的奇偶校验矩阵由下列给出：

$$\boldsymbol{H}=\left[\begin{array}{cccc|cccc|cccc|cccc|c} 1&1&1&1 & & & & & & & & & & & & & \\ 1&1&0&0 & 1&1&1&1 & & & & & & & & & \\ 1&0&1&0 & 1&1&0&0 & 1&1&1&1 & & & & & \\ 0&0&0&0 & 1&0&1&0 & 1&1&0&0 & 1&1&1&1 & \cdots \\ 0&0&0&0 & 0&0&0&0 & 1&0&1&0 & 1&1&0&0 & \\ & &\vdots & & & &\vdots & & & &\vdots & & & &\vdots & & \end{array}\right]$$

(1) 确定生成矩阵$\boldsymbol{G}$。

(2) 确定生成多项式矩阵$\boldsymbol{G}(D)$。

(3) 给出 (12, 9) Wyner-Ash卷积码的电路实现。

(4) 该码的d^*和d_{free}的值各是什么？

6.7 考虑定义在$GF(4)$上的一个卷积编码器，它的生成多项式为

$$g_1(D)=2D^3+3D^2+1 \text{和}$$
$$g_2(D)=D^3+D+1$$

(1) 该码的最小距离是多少？

(2) 该码是非灾难性的吗？为什么？

6.8 设一个码率为1/3的二元卷积编码器的生成多项式如下：

$$g_1(D)=D^3+D^2+1$$
$$g_2(D)=D^3+D$$
$$g_3(D)=D^3+1$$

(1) 编码下列的比特流：01100011110101。

(2) 编码下列的比特流：1010101010…。

(3) 对下列接收到的比特流进行译码：001001101111000110011。

6.9 考虑$GF(3)$上的一个码率为1/2的卷积编码器，其生成多项式为

$$g_1(D) = 2D^3 + 2D^2 + 1 \text{ 和}$$
$$g_2(D) = D^3 + D + 2$$

(1) 给出该编码器的电路实现。

(2) 该码的最小距离是多少?

(3) 用该编码器将下列符号串编码：2012111002102。

(4) 假设错误向量为0010102000201。构造接收到的向量并用维特比算法对其译码。

上机习题

6.10 写一个程序，使其在给定n_0和v的值时能计算出d_{free}的Heller界。

6.11 写一个程序来穷举搜索好的系统卷积码。该程序应该对参数k_0、n_0、v、m等进行循环，以确定在这一范围内最好卷积码的生成多项式矩阵（以八进制形式给出）。

6.12 写一个程序，使其当给定任意卷积编码器的生成多项式矩阵时，能计算出d^*和d_{free}。

6.13 写一个程序，使其当给定一个约束长度v时，能构造所有码率为1/2的卷积编码器，且对给定的v，它能选取最好的码。利用该程序勾画出下列平面图：

(1) 最小距离d^*相对v的平面图。

(2) 自由距离d_{free}相对v的平面图。

对卷积码纠错能力在内存要求方面做出评论。

6.14 写一个维特比译码器软件，它接受下列输入：

(1) 以八进制形式给出的码的参数，以及

(2) 接收到的比特流。

该译码器产生幸存者和译码后的比特流。

6.15 对表6-4中列出的$v = 3, 4, \cdots, 7$情况验证Heller界。

6.16 写一个Turbo译码器的通用程序。该程序应该接受两个编码器的参数和交错器的类型，然后当输入（未编码的）比特流时产生编码比特流。

6.17 修改前一个问题中的Turbo编码器程序，用来确定Turbo编码器的d_{free}。

6.18 考虑图6-37中所示的码率为1/3的Turbo编码器。设随机交错器大小为256比特。

(1) 求该Turbo编码器的d_{free}。

(2) 若输入速率为28.8kb/s，该编码器造成的时间延迟为多少?

6.19 写一个用迭代MAP译码算法做一般Turbo译码的程序。该程序应该接受两个编码器的参数，用于编码的交错器类型和SNR，当输入有噪的、编码后的比特流时，它应该产生译码后的比特序列。

6.20 考虑码率为1/3的包含下面要素编码器的Turbo编码器：

$$G_1(D) = G_2(D) = \left(1 \quad \frac{1 + D^2 + D^3 + D^4}{1 + D + D^4}\right)$$

编码后的输出包含信息比特，后接两个来自两个编码器的奇偶校验比特，从而该编码器的码率为1/3。采用大小为256的随机交错器。

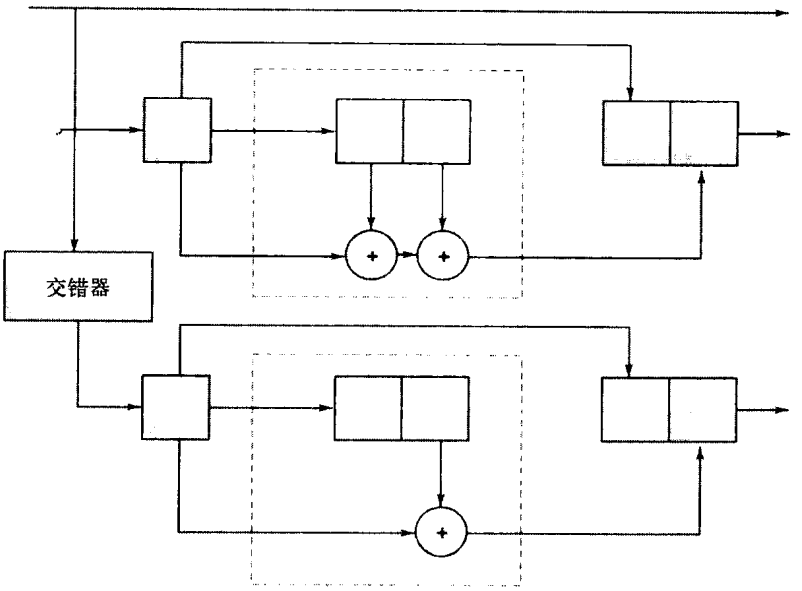


图6-37　习题6.18的Turbo编码器

(1) 对这个Turbo编码器，给出比特错误率（BER）相对信噪比（SNR）的平面图。将SNR在－2dB～10dB之间变化。

(2) 对大小为1024的交错器，重复上述过程。对你的结果给出评论。

6.21　我们希望找到图6-37 中Turbo编码器在AWGN信道上的性能。产生S随机交织器的一条BER相对于SNR的曲线。运行交织器规模分别为64、256、1024的模拟器。画出当S=5、10和15时BER相对于SNR的曲线族。在本题中使用迭代的MAP解码算法。

第7章 网格编码调制

一个数学理论在被搞得如此清楚以至于你能给马路上第一个遇到的人解释之前，不能算是完备的。

——David Hilbert.（1862—1943）

7.1 网格编码调制（TCM）简介

在前面的章节中我们学习了一些错误控制编码技术。在所有这些技术中，信息比特以一种已知的方式加入额外比特。但是，比特错误率的改善是以这些额外比特所占用的带宽为代价的。这种带宽扩张等于码率的倒数。

例如，一个RS（255，223）码的码率为$R=223/255=0.8745$，而$1/R=1.1435$。因此要传送100信息比特，我们需要传送14.35额外比特。这变成了14.35%的带宽扩张。即使对这个高效的RS（255，223）码，额外的带宽需求都不是个小数目。

在功率有限信道（如空间通信）中，我们可能会以带宽扩展为代价来换取期望的性能。但是对带宽有限信道（如电话信道），这可能不是理想的选择。在这样的信道中，一种带宽有效的信号方案如脉冲振幅调制（PAM）、正交振幅调制（QAM）或者多相移键控（MPSK），通常用来支持高带宽效率（以bit/s/Hz为单位）。

一般地，为了提高性能（错误率），或者需要额外带宽，或者需要更高的信号功率。是否可能在不牺牲带宽（表现为数据速率）或额外的功率来取得系统性能的改善呢？在本章中我们将学习一种称为**网格编码调制技术**的编码技术，它可以在不需要带宽扩张或不用额外功率的情况下取得更好的性能。

我们从介绍编码调制概念开始，然后学习一些能构造好的编码调制方案的设计技术。最后，讨论加性白高斯噪声（AWGN）信道和衰退信道等不同的编码调制方案的性能。

7.2 编码调制的概念

传统意义上，编码和调制被认为是数字通信系统中两个分开的部分。输入消息流首先通过信道编码（额外比特被加入进来），然后这些编码后的比特由调制器转化为模拟波形。信道编码器和调制器的目的都是纠正用一个不理想的信道时所产生的错误。这两部分（编码器和调制器）被独立地优化，尽管它们的目的是相同的，也就是说为了纠正由信道造成的错误！正如我们已经看到的，通过降低码率，即以带宽扩张和增加编码复杂性为代价，要得到更好的性能是可能的。但是，如果将信道编码器同调制器有机地结合起来，可以得到编码增益而不需要带宽扩张。我们通过一个例子来说明。

例7.1 考虑吞吐量为2bit/s/Hz的信道上的信息传输问题。一种可能的解决方法是用未编码的QPSK，另一种可能性是首先用码率为2/3的卷积编码器（它把2个未编码的比特转化为3个编

码后的比特），然后用吞吐量为3bit/s/Hz的8PSK信号集合。这种编码后的8PSK方案造成与未编码的QPSK同样的信息数据吞吐量（2bit/s/Hz）。注意不管是QPSK还是8PSK方案需要相同的带宽。但我们知道当每个符号的能量相同时，8PSK的符号错误率比QPSK的要差。但是，码率为2/3的卷积编码器可以提供一些编码增益。有可能由编码器提供的编码增益与由8PSK信号集合所造成的损失会持平。如果在相同SNR条件下编码调制方案远好于未编码的情况，我们可以说在没有牺牲数据率和带宽的情况下得到了一些改善。在这个例子中我们结合了网格编码器和调制器。这样的一种方案称为**网格编码调制**（TCM）方案。

我们观察到如果平均信号能量保持不变（图7-1），通过对信号集合的扩张来提供冗余度将导致信号点之间的欧几里得距离缩短。这种欧几里得距离的缩短将增加错误率，它应该由编码来补偿（增加汉明距离）。这里假定使用AWGN信道。我们也知道在编码方案中译码之前使用*硬判决*解调会造成不可逆转的信息丢失，这可以理解为SNR的损失。对编码调制方案，信号集合的扩张意味着在功率上的损失，因此必须使用*软判决*译码。结果是解调和译码应该结合为一个步骤，而且译码器应该在信道的软输出采样上操作。对使用软判决的最大似然译码算法，优化译码器选取离接收到的序列之欧几里得距离最近的码序列。因此有效编码方案应该以将编码序列之间的欧几里得距离，而不是汉明距离最大化为基础进行设计。

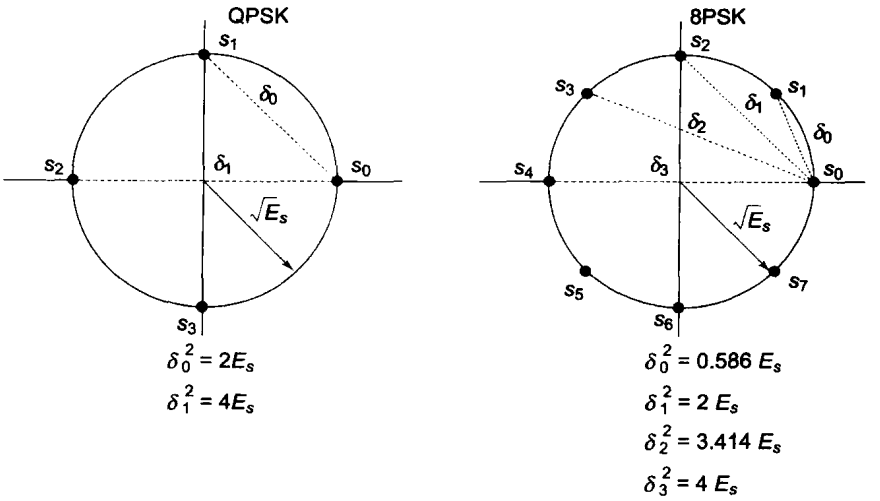


图7-1　QPSK和8PSK的信号点之间的欧几里得距离

对于TCM方案，维特比算法可以用来对接收到的符号译码。在前面的章节中我们已经看到维特比译码的基本思想是在网格中追踪最相像的路径。最相像的路径就是汉明距离最接近接收到的序列的那个路径。译码算法的性能依赖于一对构成错误事件的路径之间的最小欧几里得距离。

定义7.1 *网格中任意两个路径之间的最小欧几里得距离称为*__自由欧几里得距离__*，在TCM方案中记为*d_{free}。

在第6章中我们利用网格中任意两个路径之间的汉明距离定义了d_{free}。利用汉明重量确定的最小自由距离可以计算为从全零路径分离，后来在网格的某一点又回到全零路径的路径的最小重量。这是卷积码线性特性的结果。但是，这种方法对TCM则不适用，因为它不是线性

的。也可能d_{free}是网格中两个非全零路径之间的欧几里得距离。因此，为了计算TCM方案的自由欧几里得距离，所有路径对都需进行评估。

例7.2 考虑后面紧接着进行自然映射（$000\rightarrow s_0, 001\rightarrow s_1, \cdots, 111\rightarrow s_7$）的调制结构的卷积编码器，如图7-2所示。编码器的码率为2/3。它每次把两个比特的输入（a_1, a_2）变成三个比特的输出（c_1, c_2, c_3）。然后这三个输出比特被映射到8PSK信号集合的8个信号之一。

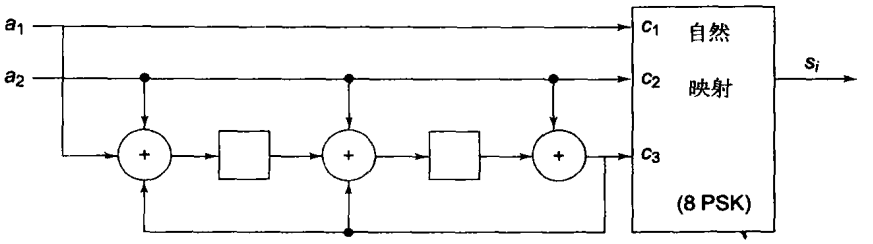


图7-2 例7.2中的TCM方案

这种编码和调制的结合也可以用将分支标记为输出符号s_i的网格表示。TCM方案可简单描述如下：有一个全连通的网格，每一个分支用8PSK星座图的一个符号标记。为了清楚地表示符号的安置，指派给分支的符号写在网格的前端。协议如下：考虑状态1。从状态1到状态1的分支标记为s_0，从状态1到状态2的分支标记为s_7，从状态1到状态3的分支标记为s_5，从状态1到状态4的分支标记为s_2。因此状态1前面的四元数组（s_0, s_7, s_5, s_2）表示发源于状态1的路径的有序分支标记。要对一个输入比特流编码时，我们沿用与卷积编码器相同的程序。然而对于TCM的情况，输出是一个符号序列而不是比特序列。假定我们要为比特流101110001001…进行编码，首先将输入序列按对分组，因为每次处理的输入是2比特。分组后的输入序列为

$$10\ 11\ 10\ 00\ \cdots$$

TCM编码器的输出可以简单地沿由输入序列指定的网格中的路径而获得。第一个输入对是10，从状态0的第一个节点开始，根据输入01的指示，我们穿越到第三个节点，这把我们带到了状态2。这一分支输出的符号为s_5。从状态2我们沿由下一个输入对11决定的第4个分支移动。该分支的输出符号为s_1。按照这种方式，对应于输入序列的输出符号为

$$s_5, s_1, s_3, s_3, \cdots$$

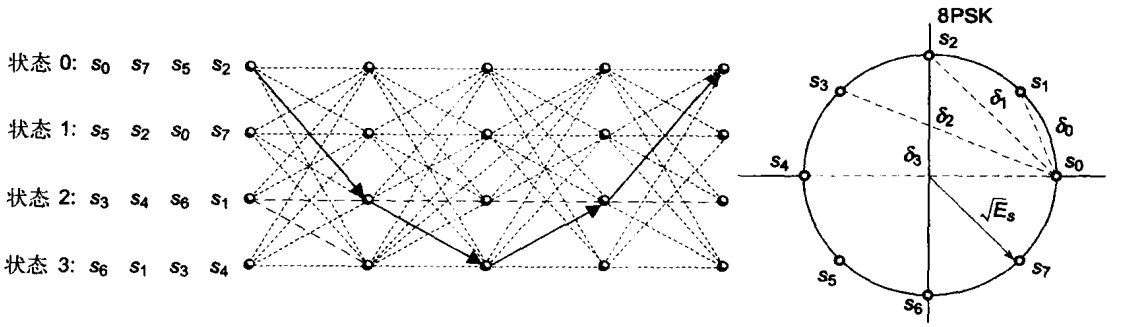


图7-3 对应于输入序列10 11 10 00…的网格中的路径

网格图中的路径在图7-3中用粗线表示。如在卷积编码器的情况下，在TCM中也是每一个

编码后的序列对应于网格中惟一的路径。译码器的目的就是从网格图中找出这个路径。

例7.3　考虑例7.2中的TCM方案。该TCM方案的自由欧几里得距离d_{free}可以通过检查网格中所有可能的路径对得到。网格图中两个由最小欧几里得距离平方分开的路径（由此导出d_{free}）在图7-4中用粗线表示。

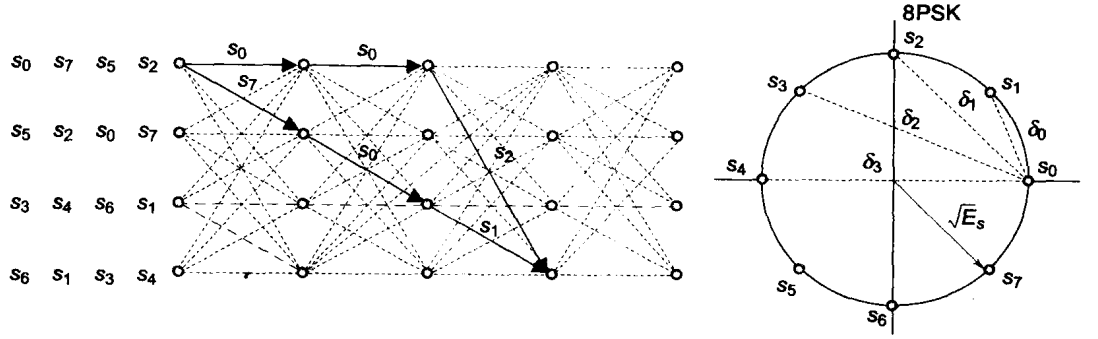


图7-4　网格中两个具有自由欧几里得距离d^2_{free}的路径

$$d^2_{free} = d^2_E(s_0, s_7) + d^2_E(s_0, s_0) + d^2_E(s_2, s_1)$$
$$= \delta_0^2 + 0 + \delta_0^2 = 2\delta_0^2 = 1.172\,E_s$$

可以看到在这种情况下，造成d_{free}的错误事件并没有涉及全零序列。正如前面提到的，为了求得d_{free}，我们必须对网格中所有可能的路径对进行评估。因为TCM的本质是非线性的，因此只评估那些从全零路径分离后又回到全零路径的路径是不够的。

我们现在必须找出一种将编码方案与未编码情况相比较的方法。下面介绍编码增益概念。

定义7.2　*要达到同样的错误概率，有编码的SNR和无编码的SNR的值之间的差定义为*编码增益（Coding Gain）g，

$$g = \mathrm{SNR}|_{\text{无编码}} - \mathrm{SNR}|_{\text{有编码}} \tag{7-1}$$

在高SNR的情况下，编码增益可以表示为

$$g_\infty = g|_{\mathrm{SNR}\to\infty} = 10\log\frac{\left(d^2_{free}/E_s\right)_{\text{有编码}}}{\left(d^2_{free}/E_s\right)_{\text{无编码}}} \tag{7-2}$$

*其中g_∞表示*渐近编码增益（Asymptotic Coding Gain），*E_s为平均信号能量。对无编码方案，d_{free}就是信号点之间的最小欧几里得距离。*

例7.4　考虑例7.2中讨论的TCM方案，它的编码器每次输入2个比特。如果我们要传送无编码的比特，那么将用到QPSK。从图7-1中可以得到该无编码方案（QPSK）的d_{free}为$2E_s$。从例7.3中我们得到，对TCM方案有$d_{free}=1.172E_s$。从而渐近编码增益为

$$g_\infty = 10\log\frac{1.172}{2} = -2.3\ \mathrm{dB}$$

这表明我们的TCM方案的性能实际比无编码方案*更差*。粗略看一下在这个例子中用到的

卷积编码器会了解到它在汉明距离方面有良好的性质。事实上，可以验证该卷积编码器在使自由汉明距离最大化方面是最优的。但是该编码器对TCM的情况则表现不佳。这说明TCM方案的设计应该是使欧几里得距离最大化，而不是汉明距离。

对一个例7.2中讨论的全连通网格，通过适当选取映射方案，可以提高其性能。为了设计好的TCM方案，可以直接从网格开始。我们的目的是给8PSK信号集中的8个符号赋值，使d_{free}达到最大值。一种方法是用计算机穷举搜索。从时刻$t_k \sim t_{k+1}$共有16个需要分配标记（符号）的分支，我们有8个符号可以选取，因此穷举搜索需要涉及8^{16}种不同情况！

另一种方法是探索性地在网格的分支上分配符号，以达到增加d_{free}的目的。我们知道一个错误事件由一个从某一状态分离经过一些变形后又回来的路径构成，如图7-5所示。这样一个错误的欧几里得距离可表示为

$$d^2_{total} = d^2_E(\text{分离的路径对}) + \cdots + d^2_E(\text{重新汇合的路径对}) \tag{7-3}$$

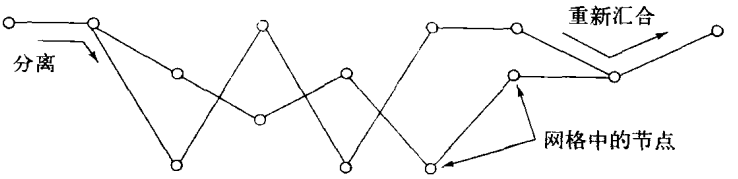


图7-5 一个错误事件

因此，为了设计TCM方案使d_{free}较大，我们至少可以保证d^2_E（分离的路径对）和d^2_E（重新汇合的路径对）尽可能大。在TCM方案中，在每次发信号的间隙经常用到冗余2^{m+1}重信号集来传送m比特。这m比特输入首先用码率为$m/(m+1)$的卷积编码器编码，得到的$(m+1)$比特输出对应到2^{m+1}重信号集的信号点上。现在回想一下用软判决译码时，AWGN信道的最大似然译码规则就是使接收到的向量与从网格图中估算出的码向量的欧几里得距离平方最小（见6.7节）。因此映射是根据能否使网格中不同路径间的欧几里得距离最大来确定的。这可以通过一种称为**集合分割**的映射完成。

7.3 通过集合分割的映射

通过集合分割的映射的基础是连续将一个扩展的2^{m+1}重信号集分割成最小欧几里得距离越来越大的子集。每一次我们分割集合时，减少了子集中信号点的数量，但增加了子集中信号点间的最小距离。集合分割可以借助下面的例子理解。

例7.5 考虑8PSK的集合分割问题。在分割之前，该信号集合的最小欧几里得距离为$\Delta_0 = \delta_0$。在第一步，该星座图的8个点被分成两个子集A_0和A_1，每个子集中有4个信号点，如图7-6所示。这第一步的结果是每个子集的最小欧几里得距离现在为$\Delta_1 = \delta_1$，它比原来8PSK的最小欧几里得距离大。我们继续这一过程，将集合A_0和A_1分别分割为两个更小的集合：$A_0 \rightarrow \{A_{00}, A_{01}\}$和$A_1 \rightarrow \{A_{10}, A_{11}\}$。这第二步的结果是每个子集的最小欧几里得距离现在为$\Delta_2 = \delta_2$。进一步分割将导致每个子集只有一个信号点。

考虑用于TCM的扩展2^{m+1}重信号集，没有必要将集合分割进行到最后一步。只要子集中的最小距离大于或等于要设计的TCM方案所期望的最小欧几里得距离就可以停止集合分割。

假定在第$\tilde{m}+1$步集合分割后得到所期望的欧几里得距离（$\tilde{m} \leqslant m$），可以看到在第$\tilde{m}+1$步后我们有$2^{\tilde{m}+1}$个子集合，每个子集合中含有$2^{m-\tilde{m}}$个信号点。

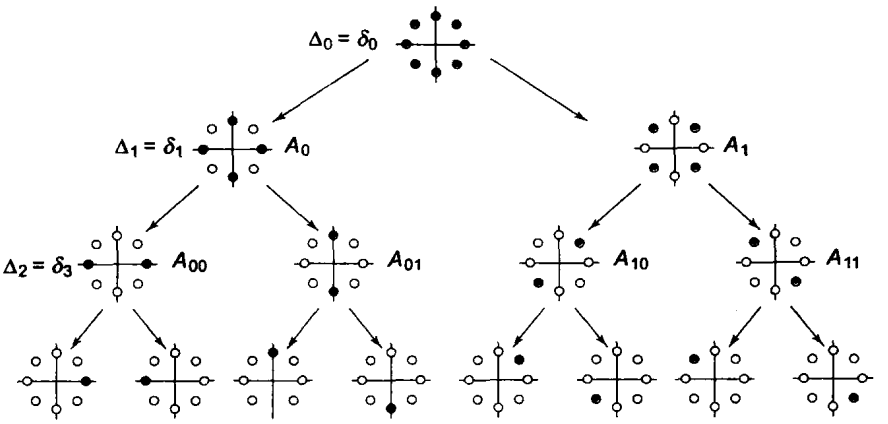


图7-6　8PSK信号集的集合分割

图7-7给出了TCM编码器的一般结构。它包括m个输入比特，其中$\tilde{m}$比特进入$\tilde{m}/(\tilde{m}+1)$卷积编码器，而剩余的$m-\tilde{m}$比特以无编码状态保留。编码器的$\tilde{m}+1$个输出比特连同$m-\tilde{m}$个无编码的比特然后进入信号映射。信号映射用卷积编码器输出的$\tilde{m}+1$比特来选取$2^{\tilde{m}+1}$个子集合中的一个，剩余的$m-\tilde{m}$无编码比特用于选取该子集合中的$2^{m-\tilde{m}}$个信号之一。因此该TCM编码器的输入为m比特，输出为从原来星座图中选取的一个信号点。

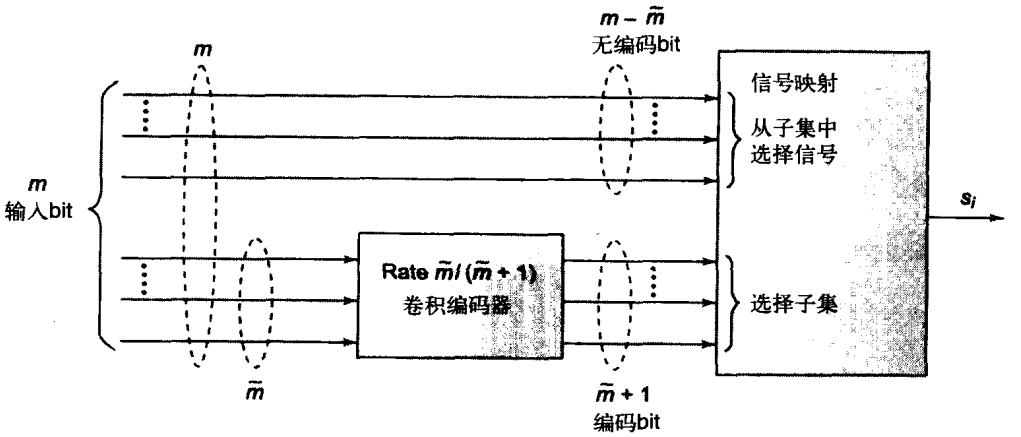


图7-7　TCM编码器的一般结构

对图7-7所示的TCM编码器，我们观察到这$m-\tilde{m}$个无编码的比特对卷积编码器的状态没影响，因为输入不会改变。因此我们可以在总的m个输入比特中改变前面的$m-\tilde{m}$个比特而不改变编码器的状态。这表明在状态间存在$2^{m-\tilde{m}}$个**平行转换**。这些平行转换与集合分割树最底层的子集中的信号有关。对$m=\tilde{m}$的情况，这些状态由单个的转换联合而成。

我们记平行转换之间的最小欧几里得距离为$\Delta_{\tilde{m}+1}$，网格图的非平行路径之间的最小欧几里得距离为$d_{free}(\tilde{m})$。则图7-7所示的TCM编码器的自由欧几里得距离可写为

$$d_{free} = \min\left[\Delta_{\tilde{m}+1},\ d_{free}(\tilde{m})\right] \tag{7-4}$$

例7.6 考虑Ungerboeck提出的TCM方案。它的设计目的在于使编码后序列之间的自由欧几里得距离最大。它包括码率为2/3的卷积编码器，且与8PSK信号集映射相连接。编码器由图7-8给出，而对应的网格图由图7-9给出。

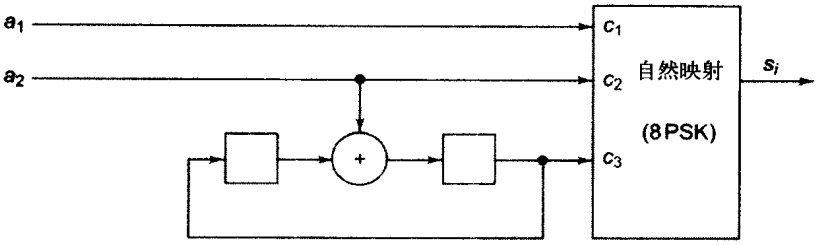


图7-8 例7.6中的TCM编码器

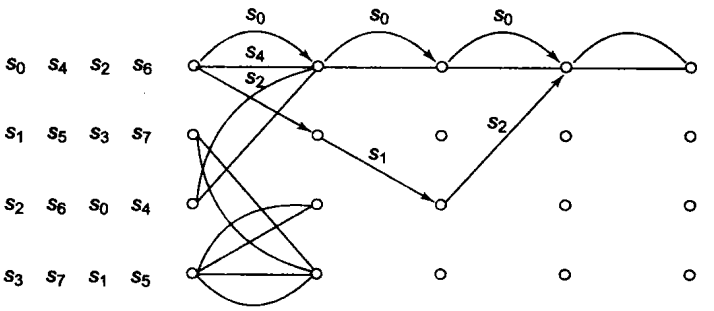


图7-9 例7.6中的编码器的网格图

对这个编码器有$m=2$且$\tilde{m}=1$，表明在每个状态之间有$2^{m-\tilde{m}}=2^1=2$个平行转换。这些平行转换间的最小平方欧几里得距离为

$$\Delta^2_{\tilde{m}+1}=\Delta^2_2=\delta^2_2=4E_s$$

网格中非平行路径间的最小平方欧几里得距离$d_{free}(\tilde{m})$由图7-9中用粗线表示的错误事件给出。从图中我们得到

$$\begin{aligned} d^2_{free}(\tilde{m}) &= d^2_E(s_0, s_2)+d^2_E(s_0, s_1)+d^2_E(s_0, s_2) \\ &= \delta^2_1+\delta^2_0+\delta^2_1=4.586\,E_s \end{aligned}$$

与平行路径有关的错误事件在所有可能的错误事件中具有最小平方欧几里得距离。因此该TCM方案的最小平方欧几里得距离为$d^2_{free}=\min[\Delta^2_{\tilde{m}+1}, d^2_{free}(\tilde{m})]=4E_s$。该方案的渐近编码增益为

$$g_\infty=10\log\frac{4}{2}=3\ \text{dB}$$

这表明由Ungerboeck提出的TCM方案比无编码的QPSK有3dB的改善。这个例子阐明：组合的编码调制方案可以弥补由卷积编码器的编码增益造成的信号集扩张的损失。更进一步，对于非平行路径有，

$$\begin{aligned} d^2_{total} &= d^2_E\ (\text{分离的路径对})+\cdots+d^2_E\ (\text{重新汇合的路径对}) \\ &= \delta^2_1+\cdots+\delta^2_1=(\delta^2_1+\delta^2_1)+\cdots=\delta^2_3+\cdots=4E_s+\cdots \end{aligned}$$

但是，平行转换的最小平方欧几里得距离为$\delta_3^2 = 4E_s$。所以，该TCM方案的最小平方欧几里得距离由那些平行转换决定。

7.4 Ungerboeck的TCM设计准则

在1982年，Ungerboeck提出了一组对TCM方案的设计准则，以便使得自由欧几里得距离最大。这些设计准则是建立在启发式基础上的。

准则1：平行转换，如果存在的话，必须与**集合分割树**最底层的子集中的信号相关联。这些信号具有最小欧几里得距离Δ_{m+1}。

准则2：发源于或汇合于一个状态的转换必须与集合分割第一步的信号相关联。这些信号之间的欧几里得距离至少为Δ_1。

准则3：所有信号都以相同频率在网格图中使用。

例7.7　下面我们希望在例7.6中提出的TCM方案的基础上进行改进。我们观察到在例7.6中平行转换限制了d_{free}^2。因此我们必须找到一种没有平行转换的网格。平行路径的缺少表明d_{free}^2并不限制在8PSK星座图中两个信号点之间最大可能距离d_3^2之内。考虑图7-10所示的网格图，该网格有8个状态，网格图中没有平行转换。我们希望根据Ungerboeck准则用8PSK信号集中的符号给该网格中的分支赋值。

因为这里没有平行转换，我们直接从Ungerboeck的第二条准则开始。我们必须用集合分割第一步中的信号为发源于或汇聚于某一状态的转换赋值。对8PSK的集合分割图我们将参照图7-6。集合分割的第一步产生两个子集A_0和A_1，每个子集中含有4个信号点。首先我们关注分离的路径。考虑最上面的节点（状态S_0）。我们将信号s_0，s_4，s_2和s_6赋值给这4个分开的路径。注意它们全部属于子集A_0。对下一个节点（状态S_1），我们用属于A_1的信号s_1，s_5，s_3和s_7赋值。对下一个节点（状态S_2），我们用属于A_0的信号s_4，s_0，s_6和s_2赋值。我们把次序打乱了，这样在它们重新汇合时我们仍然有集合分割第一步时的信号。如果我们观察汇合到状态S_0的节点的这四个路径，它们的分支被标记为s_0，s_4，s_2和s_6，它们都属于A_0。这种聪明的赋值方式保证了发源于或汇合于同一状态的转换可以用第一步集合分割时的信号标记，因此满足准则2。可以验证所有用过的信号都有相同的频率。我们并不需要为保证这一点而做特别的努力。

对应于平方自由欧几里得距离的错误事件在网格图中以粗线表示。该TCM方案的平方自由欧几里得距离为

$$\begin{aligned} d_{free}^2 &= d_E^2(s_0, s_6) + d_E^2(s_0, s_7) + d_E^2(s_0, s_6) \\ &= \delta_1^2 + \delta_0^2 + \delta_1^2 = 4.586\, E_s \end{aligned}$$

与无编码的QPSK相比，这解释为渐近编码增益

$$g_\infty = 10 \log \frac{4.586}{2} = 3.60\,(\text{dB})$$

故以增加的编码和译码复杂度为代价，对例7.6中讨论的TCM方案，我们得到0.6dB的增益。

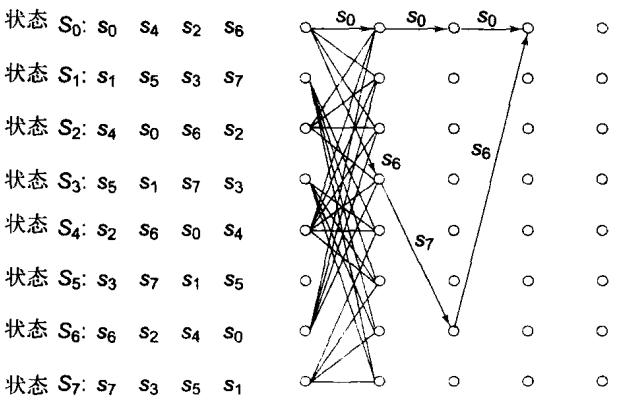


图7-10 例7.7中的编码器的网格图

例7.8 考虑例7.7中讨论的8个状态的8PSK TCM方案。等价的有反馈的系统编码器实现如图7-11所示。

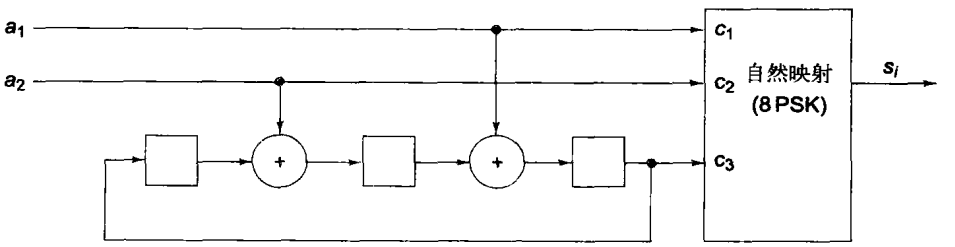


图7-11 例7.7中TCM的编码器

我们用输入和延迟的输入（关于卷积码的解析表示见6.3节）来表示图7-11所示的卷积编码器。从图中我们得到

$$c_1(D) = a_1(D)$$
$$c_2(D) = a_2(D)$$
$$c_3(D) = \left(\frac{D^2}{1+D^3}\right) a_1(D) + \left(\frac{D}{1+D^3}\right) a_2(D)$$

因此该编码器的生成多项式矩阵为

$$\boldsymbol{G}(D) = \begin{bmatrix} 1 & 0 & \dfrac{D^2}{1+D^3} \\ 0 & 1 & \dfrac{D}{1+D^3} \end{bmatrix}$$

因而满足$\boldsymbol{G}(D) \cdot \boldsymbol{H}^T(D) = 0$的奇偶校验多项式矩阵$\boldsymbol{H}(D)$为

$$\boldsymbol{H}(D) = [D^2 \quad D \quad 1+D^3]$$

我们可以将奇偶检验多项式矩阵重新写为$\boldsymbol{H}(D) = [H_1(D)\ H_2(D)\ H_3(D)]$，其中

$$H_1(D) = D^2 = (000\ 100)_{二进制} = (04)_{八进制}$$
$$H_2(D) = D = (000\ 010)_{二进制} = (02)_{八进制}$$

$$H_3(D)=1+D^3=(001\ 001)_{二进制}=(11)_{八进制}$$

表7-1给出了为8PSK信号星座构建的好的TCM码的编码器实现和渐近编码增益。几乎所有这些TCM方案都是由计算机穷举搜索找到的。编码增益是相对无编码的QPSK给出的。奇偶校验多项式用八进制形式表示。

表7-1　用8PSK的TCM方案

状态个数	H_1	H_2	H_3	d^2_{free}/E_s	g_∞(dB)
4	–	2	5	4.00	3.01
8	04	02	11	4.58	3.6
16	16	04	23	5.17	4.13
32	34	16	45	5.75	4.59
64	066	030	103	6.34	5.01
128	122	054	277	6.58	5.17
256	130	072	435	7.51	5.75

例7.9　我们现在来看一个涉及16QAM的TCM方案。该TCM编码器接收3比特输入后输出是16QAM星座图中的一个符号。该TCM方案的吞吐量为3bit/s/Hz，我们将把它与无编码的吞吐量也为3bit/s/Hz的8PSK进行比较。

设该16QAM信号星座图两点之间的最小距离为δ_0，如图7-12所示。假定所有信号是等概率的，则该16QAM信号的平均信号能量可由下式得到：

$$E_s=\frac{1}{16}(2\delta_0^2+10\delta_0^2+10\delta_0^2+18\delta_0^2)=\frac{10}{4}\delta_0^2$$

故我们得到

$$\delta_0=2\sqrt{\frac{E_s}{10}}$$

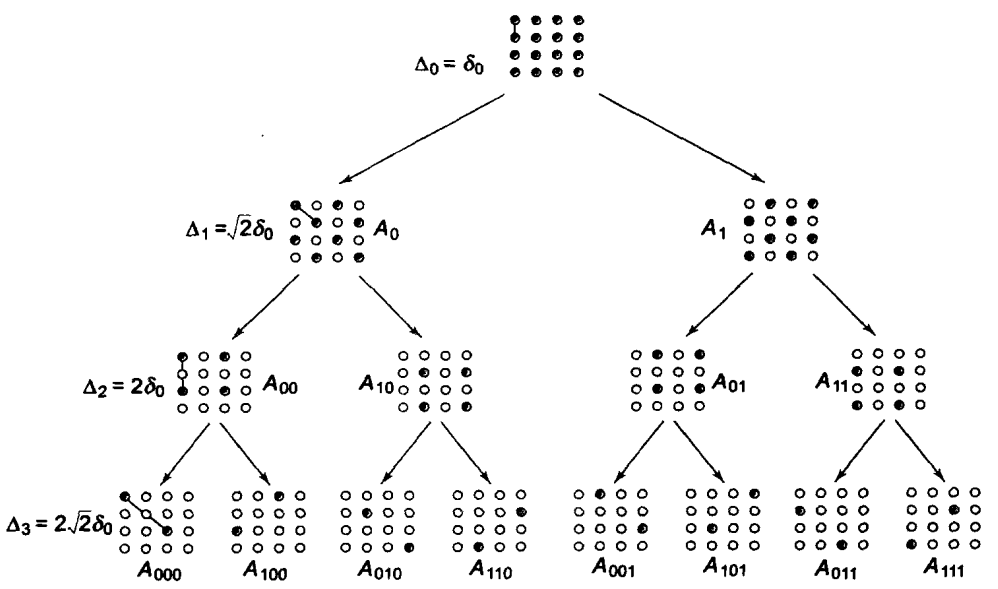


图7-12　16QAM的集合分割

该16QAM TCM方案的网格图由图7-13给出。该网格有8个状态，每一个节点有8个由此发出的分支，因为编码器每次接受3比特的输入（$2^3=8$）。图7-14给出了该编码器的实现。我们根据Ungerboeck的设计准则来对不同分支赋值不同的符号。从一个节点分离和汇聚到一个节点的分支将用集合A_0和A_1中的符号赋值。平行路径将用集合分割树的最底层（A_{000}，A_{001}等）的符号赋值。

任意两个平行路径间的平方欧几里得距离为 $\Delta_3^2=8\delta_0^2$，这是根据设计得到的，因为我们用集合分割树最底层的符号赋值。非平行路径间的最小平方欧几里得距离为

$$d_E^2=\Delta_1^2+\Delta_0^2+\Delta_1^2=5\delta_0^2$$

因此，该TCM方案的自由欧几里得距离为

$$d_{free}^2=\min\left[8\delta_0^2,\ 5\delta_0^2\right]=5\delta_0^2=2E_s$$

注意自由欧几里得距离由非平行路径而不是平行路径决定。我们现在将该TCM方案与具有同样吞吐量的无编码8PSK进行比较。对无编码8PSK，最小平方欧几里得距离为$(2-\sqrt{2})E_s$。故该TCM编码器的渐近编码增益为

$$g_a=10\log\frac{2}{2-\sqrt{2}}=5.3\ (\text{dB})$$

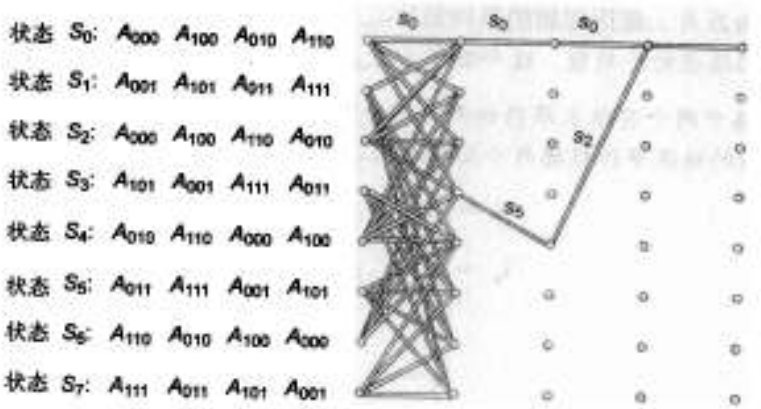


图7-13 16QAM TCM方案的网格图

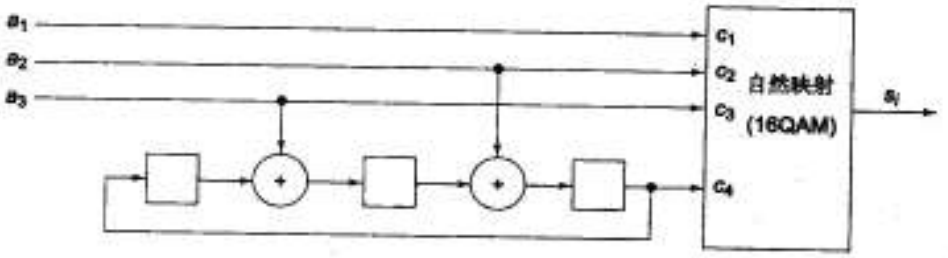


图7-14 与图7-13中网格图等价的系统编码器

7.5 TCM译码器

我们已经看到，就像卷积码一样，TCM方案也用网格图来描述。任何TCM编码器的输入

序列都根据该网格图编码成一个符号序列。编码后的序列对应于网格图中的一个特殊路径。在编码序列和网格中的路径之间存在一一对应关系。TCM译码器的任务只是识别网格中最可能的路径，它的依据是最大似然准则。正如在第6章中看到的，一种有效的搜索方法是用维特比算法（见6.7节）。

对用维特比算法接收序列的软判决译码，每一个网格分支都根据观察到的所接收的序列标记分支尺度。对加性白高斯噪声（AWGN）信道使用最大似然译码器时，分支尺度定义为编码序列和接收序列之间的欧几里得距离。维特比译码器在网格中找到一个在欧几里得距离度量下最靠近接收序列的路径。

定义7.3　为AWGN信道设计的TCM方案的**分支尺度**为接收到的信号与对应网格中分支的信号间的欧几里得距离。

在下一节中我们学习AWGN信道上TCM方案的性能，同时给出一些设计规则。

7.6 AWGN信道性能评估

对AWGN信道设计的TCM方案有不同的性能度量。我们已经讨论了渐近编码增益，它建立在自由欧几里得距离d_{free}的基础上。我们现在将学习用来刻画TCM码的其他一些参数。

定义7.4　自由距离上**最近邻居的平均数**$N(d_{free})$给出的是网格中与被传送序列相距为自由欧几里得距离d_{free}的路径的平均数。这个数字与d_{free}一起用作估算错误事件的概率。

定义7.5　网格中两个有限长路径如果从同一状态开始，分开后又汇聚，则它们形成一个**错误事件**。长度为l的错误事件根据两个编码序列s_n和$\tilde{s}_n$来定义，

$$s_n = (s_n, s_{n+1}, \cdots, s_{n+l+1})$$

$$\hat{s}_n = (\hat{s}_n, \hat{s}_{n+1}, \cdots, \hat{s}_{n+l+1})$$

满足

$$\begin{aligned} s_n &= \hat{s}_n \\ s_{n+l+1} &= \hat{s}_{n+l+1} \\ s_i &\neq \hat{s}_i,\ i = n+1, \cdots, n+l \end{aligned} \tag{7-5}$$

定义7.6　在译码器估算了时刻n正确的发射器状态的条件下，始于时刻n的错误事件的概率称为**错误事件概率**P_e。

TCM方案的性能一般用错误事件概率的上界来评估，这是根据生成函数处理方法得到的。我们再一次考虑码率为$m/(m+1)$的TCM方案的Ungerboek模型，如图7-7所示。编码器每次接受m比特并编码后得到$m+1$比特，然后由无记忆映射函数$f(.)$映射到一个符号s_i。我们称这个二元$(m+1)$数组c_i为信号s_i的标记。我们观察到在符号和其标记之间有一一对应关系。因此长度为l的一个错误事件可以等价地用下面的两个标记序列描述：

$$\boldsymbol{C}_l = (c_k, c_{k+1}, \cdots, c_{k+l-1}) \text{ 和 } \boldsymbol{C}'_l = (c'_k, c'_{k+1}, \cdots, c'_{k+l-1}) \tag{7-6}$$

其中 $c'_k = c_k \oplus e_k$, $c'_{k+1} = c_{k+1} \oplus e_{k+1}$, …, 且 $\boldsymbol{E}_l = (e_k, e_{k+1}, \cdots, e_{k+l-1})$ 为一个二元错误向量序列。数学符号⊕表示二元（模2）加法。当译码器没选择传送的序列$\boldsymbol{C}_l$，而是选择了$\boldsymbol{C}'_l$，它对应网格图中从原来传送的路径分开之后经过l个时间间隔后又汇聚的路径，这就产生了一个长度为l的

错误事件。要求出错概率，我们需要对l的所有可能值求长度为l的错误事件（即当传送$\boldsymbol{C}_l$而检测到$\boldsymbol{C}'_l$的概率）的和。错误概率上界通过下面的联合界获得：

$$P_e \leqslant \sum_{l=1}^{\infty} \sum_{s_l} \sum_{s'_l \neq s_l} P(s_l) P(s_l, s'_l) \tag{7-7}$$

其中$P(s_l, s'_l)$表示双事件错误概率（即传送序列s_l而检测到的序列为s'_l的概率）。假定符号和它的标记之间的一一对应关系，我们可以写为

$$\begin{aligned} P_e &\leqslant \sum_{l=1}^{\infty} \sum_{\boldsymbol{C}_l} \sum_{\boldsymbol{C}'_l \neq \boldsymbol{C}_l} P(\boldsymbol{C}_l) P(\boldsymbol{C}_l, \boldsymbol{C}'_l) \\ &= \sum_{l=1}^{\infty} \sum_{\boldsymbol{C}_l} \sum_{\boldsymbol{E}_l \neq 0} P(\boldsymbol{C}_l) P(\boldsymbol{C}_l, \boldsymbol{C}_l \oplus \boldsymbol{E}_l) \end{aligned} \tag{7-8}$$

双事件错误概率$P_2(\boldsymbol{C}_l, \boldsymbol{C}_l \oplus \boldsymbol{E}_l)$的上界可以用Bhattacharyya界（参见习题7.12），如下所示

$$\begin{aligned} P_2(\boldsymbol{C}_l, \boldsymbol{C}_l \oplus \boldsymbol{E}_l) &\leqslant e^{-\left\{\frac{1}{4N_0}\| f(\boldsymbol{C}_l) - f(\boldsymbol{C}'_l)\|^2\right\}} \\ &= e^{-\left\{\frac{1}{4N_0}\sum_{n=1}^{l}\| f(\boldsymbol{C}_l) - f(\boldsymbol{C}'_l)\|^2\right\}} \end{aligned} \tag{7-9}$$

其中$\|\cdot\|^2$代表平方欧几里得距离而$f(\cdot)$为无记忆映射函数。设$D = e^{-\left\{\frac{1}{4N_0}\right\}}$（边功率谱密度为$N_0$的加性白高斯噪声信道），则

$$P_2(\boldsymbol{C}_l, \boldsymbol{C}_l \oplus \boldsymbol{E}_l) \leqslant D^{\|f(\boldsymbol{C}_l) - f(\boldsymbol{C}'_l)\|^2} = D^{d_E^2(f(\boldsymbol{C}_l), f(\boldsymbol{C}'_l))} \tag{7-10}$$

其中$d_E^2(f(\boldsymbol{C}_l), f(\boldsymbol{C}'_l))$表示符号序列$s_l$和$s'_l$之间的平方欧几里得距离。下面定义函数

$$W(\boldsymbol{E}_l) = \sum_{\boldsymbol{C}_l} P(\boldsymbol{C}_l) D^{\|f(\boldsymbol{C}_l) - f(\boldsymbol{C}_l \oplus \boldsymbol{E}_l)\|^2} \tag{7-11}$$

我们现在可以把错误概率写为

$$P_e \leqslant \sum_{l=1}^{\infty} \sum_{\boldsymbol{E}_l \neq 0} W(\boldsymbol{E}_l) \tag{7-12}$$

从上式我们观察到错误概率的上界是所有可能错误事件$\boldsymbol{E}_l$的和。注意

$$d_E^2(f(\boldsymbol{C}_l), f(\boldsymbol{C}_l \oplus \boldsymbol{E}_l)) = \sum_{i=1}^{l} d_E^2(f(c_i), f(c_i \oplus e_i)) \tag{7-13}$$

我们现在介绍错误状态图的概念，它实质上是一个分支有矩阵标记的图。我们假设信源符号等可能地以概率$2^{-m} = 1/M$出现。

定义7.7 错误重量矩阵$G(e_i)$是一个$N \times N$阶矩阵，它的第p行第q列的元素定义为

$$[G(e_i)]_{p,q} = \begin{cases} \dfrac{1}{M} \displaystyle\sum_{c_{p \to q}} D^{\|f(c_{p \to q}) - f(c_{p \to q} \oplus e_i)\|^2} & \text{若存在从状态}p\text{到}q\text{的转换} \\ 0 & \text{若网格中不存在从状态}p\text{到}q\text{的转换} \end{cases} \tag{7-14}$$

其中$c_{p \to q}$为从状态p到状态q转换时产生的标记向量。

上述求和构成了网格图中状态间可能的平行转换。矩阵G中位置(p, q)的元素提供了开始于节点p且终止于节点q的错误事件概率的上界。类似地，$(1/N)G1$是一个向量，它的第p个元素为开始于节点p的所有错误事件概率的上界。现在，对任意序列$E_l=e_1, e_2, \ldots, e_l$，有一个相应的$l$个错误重量矩阵$G(e_1), G(e_2), \ldots, G(e_l)$的序列。因此，我们有

$$W(E_l)=\frac{1}{N}\mathbf{1}^T\prod_{n=1}^{l}G(\mathbf{e}_n)\mathbf{1} \tag{7-15}$$

其中$\mathbf{1}$是元素全为1的N维列向量。我们得到下面的观察结果：

(1) 对任意矩阵A，$\mathbf{1}^TA\mathbf{1}$表示A中所有元素的和。

(2) 矩阵$\prod_{n=1}^{l}G(e_n)$中位置(p, q)的元素列举了从状态p到状态q恰好l步转换中涉及的欧几里得距离。

我们的下一步工作就是将上面的分析同错误概率P_e联系起来。应注意到错误向量$e_1, e_2, \ldots, e_l$不是独立的。错误状态图的结构仅决定于线性卷积码，且仅在它的状态及分支标记$(G(e_i))$上不同于码状态图。因为错误向量e_i只是向量c_i上的差，向量e_i之间的联系同向量c_i之间的联系是一样的。因此，由式（7-12）和式（7-15）我们有

$$P_e\leqslant T(D)|_{D=e^{-1/4N_0}} \tag{7-16}$$

其中

$$T(D)=\frac{1}{N}\mathbf{1}^TG\mathbf{1} \tag{7-17}$$

且矩阵

$$G=\sum_{l=1}^{\infty}\sum_{E_l\neq 0}\prod_{n=1}^{l}G(e_n) \tag{7-18}$$

是错位状态图的矩阵转移函数。$T(D)$称为错误状态图的标量转移函数或简称为**转移函数**。

例7.10　考虑码率为1/2的TCM方案，其中$m=1$且$M=4$。它每次接受一比特并把它编码成为二比特，然后映射到四个QPSK符号中的一个。两个状态的网格图及4PSKC星座中符号的分配由图7-15给出。

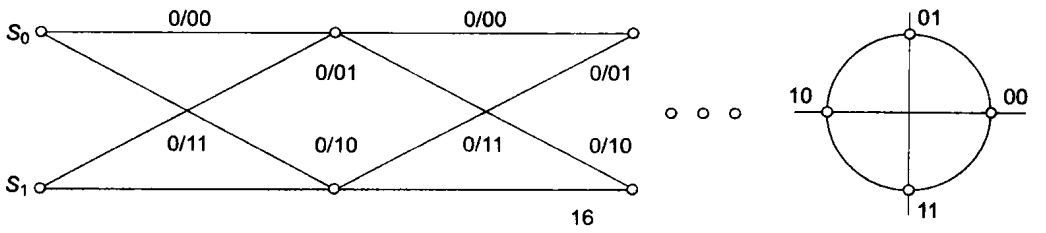


图7-15　两个状态的网格图

将错误向量表示为$e=(e_2\ e_1)$。于是由式（7-14）可得

$$\boldsymbol{G}(e_2e_1)=\frac{1}{2}\begin{bmatrix} D^{\|f(00)-f(00\oplus e_2e_1)\|^2} & D^{\|f(10)-f(10\oplus e_2e_1)\|^2} \\ D^{\|f(01)-f(01\oplus e_2e_1)\|^2} & D^{\|f(11)-f(11\oplus e_2e_1)\|^2} \end{bmatrix}$$
$$=\frac{1}{2}\begin{bmatrix} D^{\|f(00)-f(e_2e_1)\|^2} & D^{\|f(10)-f(\bar{e}_2e_1)\|^2} \\ D^{\|f(01)-f(e_2\bar{e}_1)\|^2} & D^{\|f(11)-f(\bar{e}_2\bar{e}_1)\|^2} \end{bmatrix} \tag{7-19}$$

其中$\bar{e}=1\oplus e$。该TCM方案的错误状态图由图7-16给出。

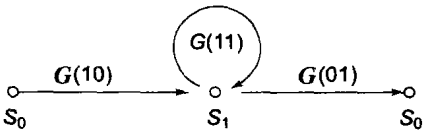


图7-16 错误状态图

该错误状态图的矩阵转移函数为

$$\boldsymbol{G}=\boldsymbol{G}(10)[\boldsymbol{I}_2-\boldsymbol{G}(11)]^{-1}\boldsymbol{G}(01) \tag{7-20}$$

其中I_2是2×2阶单位矩阵。在此情况下只有三种可能的错误向量，{01，10，11}。由式（7-19）我们可以计算出

$$\boldsymbol{G}(01)=\frac{1}{2}\begin{bmatrix} D^2 & D^2 \\ D^2 & D^2 \end{bmatrix},\ \boldsymbol{G}(10)=\frac{1}{2}\begin{bmatrix} D^4 & D^4 \\ D^4 & D^4 \end{bmatrix}\text{和}\ \boldsymbol{G}(11)=\frac{1}{2}\begin{bmatrix} D^2 & D^2 \\ D^2 & D^2 \end{bmatrix}$$

利用式（7-20）我们得到错误状态图的矩阵转换函数为

$$\boldsymbol{G}=\frac{1}{2}\frac{D^6}{1-D^6}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{7-21}$$

从而标量转换函数$T(D)$可由下式给出

$$T(D)=\frac{1}{2}\mathbf{1}^T\boldsymbol{G}\mathbf{1}=\frac{D^6}{1-D^2} \tag{7-22}$$

通过将$D=e^{-\left\{\frac{1}{4N_0}\right\}}$代入式（7-22）中即得错误概率的上界

$$P_e\leqslant\left.\frac{D^6}{1-D^2}\right|_{D=e^{\frac{-1}{4N_0}}} \tag{7-23}$$

例7.11 考虑另一个码率为1/2的TCM方案，其中$m=1$且$M=4$。它的两个网格图同例7.10中的一样。但是从4PSK星座的符号分配是不同的，由图7-17给出。

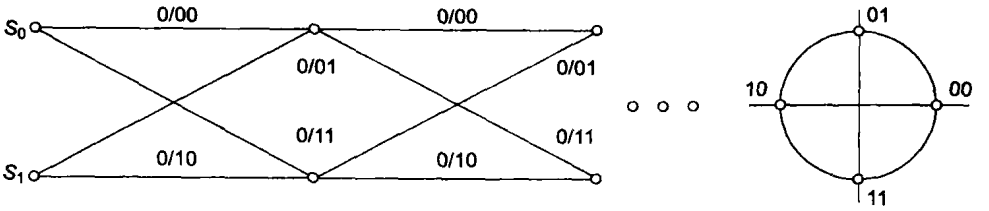


图7-17 两个状态的网格图

注意该符号分配违背了Ungerboek设计原理。我们仍然记错误向量为$\boldsymbol{e}=(e_2\ e_1)$。该TCM方案的错误状态图由图7-18给出。

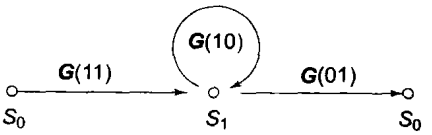


图7-18　错误状态图

该错误状态图的矩阵转换函数为

$$\boldsymbol{G}=\boldsymbol{G}(11)[\boldsymbol{I}_2-\boldsymbol{G}(10)]^{-1}\boldsymbol{G}(01) \tag{7-24}$$

其中$\boldsymbol{I}_2$是2×2阶单位矩阵。在这种情况下只有三种可能的错误向量{01，10，11}。由式（7-19）我们计算出

$$\boldsymbol{G}(01)=\frac{1}{2}\begin{bmatrix}D^2 & D^2\\ D^2 & D^2\end{bmatrix},\boldsymbol{G}(10)=\frac{1}{2}\begin{bmatrix}D^4 & D^4\\ D^4 & D^4\end{bmatrix}\text{ 和 }\boldsymbol{G}(11)=\frac{1}{2}\begin{bmatrix}D^2 & D^2\\ D^2 & D^2\end{bmatrix}$$

利用式(7-23)我们得到错误状态图的矩阵转换函数为

$$\boldsymbol{G}=\frac{1}{2}\frac{D^4}{1-D^4}\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} \tag{7-25}$$

从而标量转换函数$T(D)$ 可由下式给出

$$T(D)=\frac{1}{2}\mathbf{1}^T\boldsymbol{G}\mathbf{1}=\frac{D^4}{1-D^4} \tag{7-26}$$

于是错误概率的上界为

$$P_e\leqslant\frac{D^4}{1-D^4}\bigg|_{D=e^{\frac{-1}{4N_0}}} \tag{7-27}$$

比较式(7-23)和式(7-27)可知，仅仅通过改变网格图中对分支的符号分配，就可以很大程度地降低其性能。在第二个例子中，错位概率的上界放松了两个量级（假设$D<<1$，即SNR很高的情况）。

错误事件概率的一个更紧的上界为（留作习题）

$$P_e\leqslant\frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{d_{free}^2}{4N_0}}\right)e^{\frac{d_{free}^2}{4N_0}}T(D)|_{D=e^{\frac{-1}{4N_0}}} \tag{7-28}$$

根据式(7-28)，一个对错误事件概率的渐近估计可以通过只考虑自由欧几里得距离满足下式的错误事件来得到

$$P_e\approx\frac{1}{2}N(d_{free})\operatorname{erfc}\left(\sqrt{\frac{d_{free}^2}{4N_0}}\right) \tag{7-29}$$

通过一些与各错误向量相关联的不正确的输入比特数目可以求得成对事件的错误概率，然后将结果除以m就得到比特错误概率的上界。因此

$$P_b \leqslant \frac{1}{m}\frac{\partial T(D,I)}{\partial I}\bigg|_{I=1,\,D=e^{\frac{-1}{4N_0}}} \tag{7-30}$$

其中$T(D, I)$为修改的状态图的**扩展生成函数**（augmented generating function）。修改的状态图的概念在6.5节介绍。我们也可以得到如下比特错误概率的更紧上界：

$$P_e \leqslant \frac{1}{2m}\operatorname{erfc}\left(\sqrt{\frac{d_{free}^2}{4N_0}}\right)e^{\frac{d_{free}^2}{4N_0}}\frac{\partial T(D,I)}{\partial I}\bigg|_{I=1,D=e^{\frac{-1}{4N_0}}} \tag{7-31}$$

从式（7-31）中我们观察到比特错误概率的上界取决于d_{free}。在下一节中，我们将学习一些计算d_{free}的方法。

7.7 d_{free}的计算

我们已经看到欧几里得自由距离d_{free}是决定AWGN信道的TCM方案好坏的最重要的参数。它定义了该方案的渐近编码增益。在6.5节中我们看到生成函数可以用来计算汉明自由距离d_{free}。错误状态图的转换函数$T(D)$包括网格中所有路径与全零路径的距离信息。如果$T(D)$以封闭形式获得，则d_{free}的值可以直接从其幂级数展开式中得到。转换函数$T(D)$可以写为

$$T(D) = N(d_{free})D^{d_{free}^2} + N(d_{next})D^{d_{next}^2} + \cdots \tag{7-32}$$

其中d_{next}^2是次小平方欧几里得距离。故在级数展开式中D的最小指数为d_{free}。但是在大多数情况下，可能得不到封闭形式的$T(D)$，我们不得不求助于数值方法。

接下来考虑函数

$$\phi_1(D) = \ln\left[\frac{T(\varepsilon D)}{T(D)}\right] \tag{7-33}$$

则随着$D\to 0$，$\phi_1(D)$将单调趋于极限值d_{free}^2。因此在$D>0$的条件下我们得到d_{free}^2的一个上界。为了得到d_{free}^2的下界，考虑下面的函数

$$\phi_2(D) = \frac{\ln T(D)}{\ln D} \tag{7-34}$$

在式（7-32）两边取对数得到

$$d_{free}^2 \ln D = \ln T(D) - \ln N(d_{free}) - \ln\left[1 + \frac{N(d_{free})}{N(d_{next})}D^{d_{next}^2 - d_{free}^2} \cdots\right] \tag{7-35}$$

在$D>0$的条件下，令$D\to 0$，则由式（7-34）和式（7-35）得到

$$\frac{\ln T(D)}{\ln D} = d_{free}^2 - \varepsilon(D) \tag{7-36}$$

其中$\varepsilon(D)$大于零且随$D\to 0$而单调趋于零。所以当我们给$\phi_1(D)$和$\phi_2(D)$取越来越小的值时，可以得到非常接近d_{free}^2的值。

应该记住，尽管d_{free}^2是决定TCM方案质量的最重要的参数，但另两个参数也有影响：

(1) **错误系数**$N(d_{free})$：在错误率为10^{-6}时，该错误系数每增一倍就会使编码增益下降大约0.2dB。

(2) **次距离**d_{next}：它是形成一个错误事件的两个路径间次小欧几里得距离。若d_{next}与d_{free}很接近，则对P_e的上界好的近似需要有很大的SNR。

到目前为止，我们的注意力主要集中在AWGN信道上。我们发现最好的设计策略是使码的自由距离d_{free}最大。在下一节中，我们将考虑衰退信道的TCM方案设计规则。这里只想提醒读者，衰退信道在无线电和移动通信中经常遇到。造成衰退的一个常见原因是传播媒体的多路性。在这种情况下，信号从不同的路径（时间也有差别）到达接收端，然后叠加在一起。根据来自不同路径的信号是相位累加还是相位分离，下一个收到的信号呈随机变化的振幅和相位。接收信号振幅的下降（低于门限值）程度称为**衰退**。

7.8 衰退信道的TCM

在本节中，我们将考虑衰退信道上网格编码的M-元相移键控（MPSK）的性能。我们知道一个TCM编码器接受一个比特流输入并输出一个符号序列。在这种情况下我们将假设所有这些符号s_i都属于MPSK信号集。通过复数记号，每个符号可以表示为复平面上的一个点。为了使变化很慢的衰退过程造成的突发错误得以扩散，我们对编码信号进行**交错**。这些交错后的符号再变成脉冲信号以达到无**符号间干扰**，最后转化为RF频率在信道中传送。信道通过加入衰退增益（这是负增益或正丢失，取决于我们怎么看待）和AWGN来破坏这些传送的符号。在接收端，收到的序列经过解调和量化用于软判决译码。在许多实现中，信道评估者估算信道增益，它也称作**信道状态信息**。所以我们可以把在时刻i接收到的信号表示为

$$r_i = g_i\, s_i + n_i \tag{7-37}$$

其中n_i为均值为零、方差为$N_0/2$的高斯噪声过程的一个采样，g_i为复信道增益，也是方差为σ_g^2的复高斯过程的一个采样。复信道增益可以用相位量符号具体写为

$$g_i = a_i e^{j\phi_i} \tag{7-38}$$

其中a_i和ϕ_i分别为振幅和相位过程。我们现在做下面的假设：

(1) 收信人做连贯的检测。

(2) 交错过程是理想的，这表明衰退幅度是统计无关的，而且信道可以作为无记忆的。

因此我们可以写为

$$r_i = a_i\, s_i + n_i \tag{7-39}$$

我们知道对一个没有直接路径而只有扩散多路径的信道，衰退幅度服从Rayleigh分布，而**概率密度函数**（pdf）为

$$p_A(a) = 2ae^{-a^2},\ a \geqslant 0 \tag{7-40}$$

当除扩散多路径外还有一个直接路径时，观察到的结果是Rician**衰退**。Rician衰退幅度的概率密度函数为

$$p_A(a) = 2a(1+K)e^{-(K+a^2(K+1))} I_0\left(2a\sqrt{K(1+K)}\right) \tag{7-41}$$

其中$I_0(.)$为零级改进的第一类贝塞耳函数（Bessel Function），K为如下定义的Rician参数。

定义7.8 Rician参数K定义为直接路径能量与扩散多路径能量的比值。对$K=0$的极端情况，Rician分布的概率密度函数与Rayleigh分布的概率密度函数相同。

我们现在来看衰退信道上TCM方案的性能。设$r_l=(r_1, r_2,\cdots, r_{el})$为接收到的信号。则通常用维特比译码器来实现的最大似然译码器选取与收到的信号最相似的编码序列。这可以通过计算接收信号序列r_l和可能传送的信号s_l之间的尺度而得。我们前面已经看到，该尺度与下面的条件信道概率有关

$$m(\boldsymbol{r}_l, \boldsymbol{s}_l) = \ln p(\boldsymbol{r}_l|\boldsymbol{s}_l) \tag{7-42}$$

如果将信道状态信息也用上，则该尺度变为

$$m(\boldsymbol{r}_l, \boldsymbol{s}_l; \hat{\boldsymbol{a}}_l) = \ln p(\boldsymbol{r}_l|\boldsymbol{s}_l, \hat{\boldsymbol{a}}_l) \tag{7-43}$$

在理想交错的假设下，信道是无记忆的，从而这些尺度可以用下面的和式表示

$$m(\boldsymbol{r}_l, \boldsymbol{s}_l) = \sum_{i=1}^{l} \ln p(r_i|s_i) \tag{7-44}$$

且

$$m(r_l, s_l; \hat{a}_l) = \sum_{i=1}^{l} \ln p(r_i|s_i, \hat{a}_i) \tag{7-45}$$

我们首先考虑信道状态信息已知的情况，即$\hat{a}_i=a_i$。此时上述尺度可以写为

$$m(r_i, s_i; \hat{a}_i) = -|r_i - a_i s_i|^2 \tag{7-46}$$

因此，成对事件错误概率可表示为

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) = E_{al}[P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l|\boldsymbol{a}_l)] \tag{7-47}$$

其中

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l|\boldsymbol{a}_l) = P[m(\boldsymbol{r}_l, \hat{\boldsymbol{s}}_l; \boldsymbol{a}_l) \geqslant m(\boldsymbol{r}_l, \boldsymbol{s}_l; \boldsymbol{a}_l)|\boldsymbol{a}_l] \tag{7-48}$$

E是统计期望算子。用Chernoff界，可以求得成对事件错误概率的上界为

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) \leqslant \prod_{i=1}^{l} \frac{1+K}{1+K+\frac{1}{4N_0}|s_i - \hat{s}_i|} \exp\left[-\frac{K\frac{1}{4N_0}|s_i - \hat{s}_i|^2}{1+K\frac{1}{4N_0}|s_i - \hat{s}_i|^2}\right] \tag{7-49}$$

对高SNR的情况，上述方程可简化为

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) \leqslant \prod_{i \in \eta} \frac{(1+K)e^{-K}}{\frac{1}{4N_0}|s_i - \hat{s}_i|^2} \tag{7-50}$$

其中η为所有满足$s_i \neq \hat{s}_i$的i的集合。我们记η中元素个数为l_η，则可以写为

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) \leqslant \frac{\left((1+K)e^{-K}\right)^{l_\eta}}{\left(\frac{1}{4N_0}\right)^{l_\eta} d_p^2(l_\eta)} \tag{7-51}$$

其中

$$d_p^2(l_\eta) \leq \prod_{i \in \eta} |s_i - \hat{s}_i|^2 \tag{7-52}$$

为信号$s_i \neq \hat{s}_i$的**平方乘积距离**。其中的项l_η称为错误事件（s_i, $\hat{s}_i$）的**有效长度**。错误事件概率P_e的一般情况下的界在前面已经讨论过。对高SNR的情况，P_e的上界可表示为

$$P_e \leq \sum_{l_\eta} \sum_{d_p^2(l_\eta)} \alpha[l_\eta, d_p^2(l_\eta)] \frac{\left((1+K)e^{-K}\right)^{l_\eta}}{\left(\frac{1}{4N_0}\right)^{l_\eta} d_p^2(l_\eta)} \tag{7-53}$$

其中$\alpha[l_\eta, d_p^2(l_\eta)]$为有效长度为$l_\eta$且平方乘积距离为$d_p^2(l_\eta)$的码序列的平均数。错误事件概率实际主要由最小有效长度l_η和最小乘积距离$d_p^2(l_\eta)$决定。我们记最小有效长度l_η为L，其相应乘积距离为$d_p^2(L)$，则错误事件概率可以近似表示为

$$P_e \approx \alpha(L, d_p^2(L)) \frac{\left((1+K)e^{-K}\right)^{L}}{\left(\frac{1}{4N_0}\right)^{L} d_p^2(L)} \tag{7-54}$$

从式（7-54）我们观察到下列结论：

(1) 错误事件概率随SNR的L次幂而渐近变化。这与由时间差异技术得到的结果是相似的。因此，L又称为该TCM方案的时间差异。

(2) 衰退信道TCM设计重要的参数为时间差异L和乘积距离$d_p^2(L)$。这与AWGN信道的自由欧几里得距离参数不同。

(3) 为AWGN信道设计的TCM码在用于衰退信道时性能很差，反过来也一样。

(4) 对大的Rician参数值K。自由欧几里得距离对TCM方案性能的影响占主导地位。

(5) 在低SNR的情况下，自由欧几里得距离又对TCM方案的性能起到重要作用。

因此，衰退信道的TCM方案的基本设计规则在高SNR和小的K值的情况下为

(1) 使码的有效长度L最大化，且

(2) 使最小乘积距离$d_p^2(L)$达到最小值。

考虑一个有效长度为L且最小乘积距离为$d_{p1}^2(L)$的TCM方案。假设码被重新设计后产生具有相同L的最小乘积距离，$d_{p2}^2(L)$。因最小乘积距离的增加而产生的编码增益表示为

$$\Delta_g = SNR_1 - SNR_2 \big|_{P_{e_1} = P_{e_2}} = \frac{10}{L} \log \frac{d_{p_2}^2(L)\alpha_1}{d_{p_1}^2(L)\alpha_2} \tag{7-55}$$

其中a_i ($i=1, 2$)为有效长度为L的TCM方案i的码序列平均数。我们观察到对一个固定的L值，增加最小乘积距离使之对应于一个更小的L值对改善码的性能更有效。

我们假设已经提供信道状态信息。当信道状态信息未提供时，可以采用类似于信道状态信息已提供的情况予以分析。在缺少信道状态信息的情况下，尺度可以表示为

$$m(r_i, s_i; \hat{a}_i) = -|r_i - s_i|^2 \tag{7-56}$$

经过一些数学变换后，可以证明

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) \leqslant \frac{(2e/l_\eta)^{l_\eta}}{(1/N_0)^{l_\eta}} \frac{\left(\sum_{i \in l_\eta} |s_i - \hat{s}_i|^2\right)^{l_\eta}}{d_p^4(l_\eta)} (1+K)^{l_\eta} e^{-l_\eta K} \tag{7-57}$$

用前面讨论的论据可知，在信道状态信息未提供的情况下也可以确定错误事件概率P_e。

7.9 空时网格码

空时网格玛（Space Tim Trellis Code，STTC）是一个为多天线发射的编码技术。STTC是TCM 方案，其中网格的每一个分支均用对应于由N_t发射天线传输的N_t个信号进行标注。

例7.12 图7-19描绘了一个使用4PSK的2STTC。其中边附近的标注是c_1c_2，其中c_1是由天线①传输的，c_2是由天线②传输的。

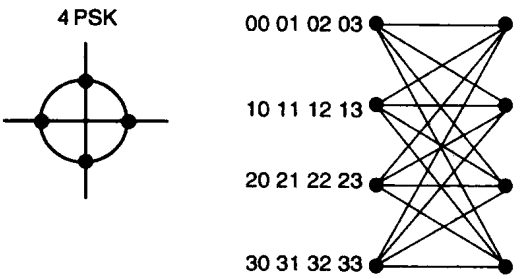


图7-19 2STTC, 4PSK, 4状态，2 bit/s/HZ

假定接收端是理想的信道状态信息，要对STTC码进行解码，我们需要使用**向量维特比算法**（vector Viterbi algorithm）。假设 r_t^j 是在时刻t接收天线j接收的信号，α_{ij}是从发射天线i到接收天线j的路线收益，对标注为 $q_t^1 q_t^2 \cdots q_t^{N_t}$ 的切换的分支度量为

$$\sum_{j=1}^{N_r} \left| r_t^j - \sum_{i=1}^{N_t} \alpha_{ij} q_t^i \right|^2 \tag{7-58}$$

其中N_r是接收天线的数目。

维特比算法用来计算具有最低累积度量的路线。

图7-20展示了一个全速率的M-PSK STTC的通用编码器结构。对于一个具有任意M元信号群的全速率STTC，此编码器结构一般是有效的。此编码器由$m(=\log_2 M)$个前馈移位寄存器构成，此寄存器以m个二进制输入序列 $c^1, c^2, \cdots c^m$ 为输入。第k^{th}移位寄存器的系数乘法集合表示为：

$$g^k = [(g_{0,1}^k, g_{0,2}^k, g_{0,N_t}^k), (g_{1,1}^k, g_{1,2}^k, g_{1,N_t}^k), \cdots, (g_{v_1,1}^k, g_{v_1,2}^k, g_{v_1,N_t}^k)] \tag{7-59}$$

其中

$$g_{j,i}^k, k = 1,2,\cdots,m, j = 1,2,\cdots,v_k, i = 1,2,\cdots,N_t$$

是M-PSK（或者M元）群集合的一个元素，v_k是第k个移位寄存器的记忆深度。所有移位寄存器的乘法器输出还需要做模M的运算，编码器的输出是 $x = (x^1, x^2, \cdots x^{N_t})$。编码器的总共存储深度$v$可以通过下列式子得到：

$$v=\sum_{k=1}^{m} v_k \tag{7-60}$$

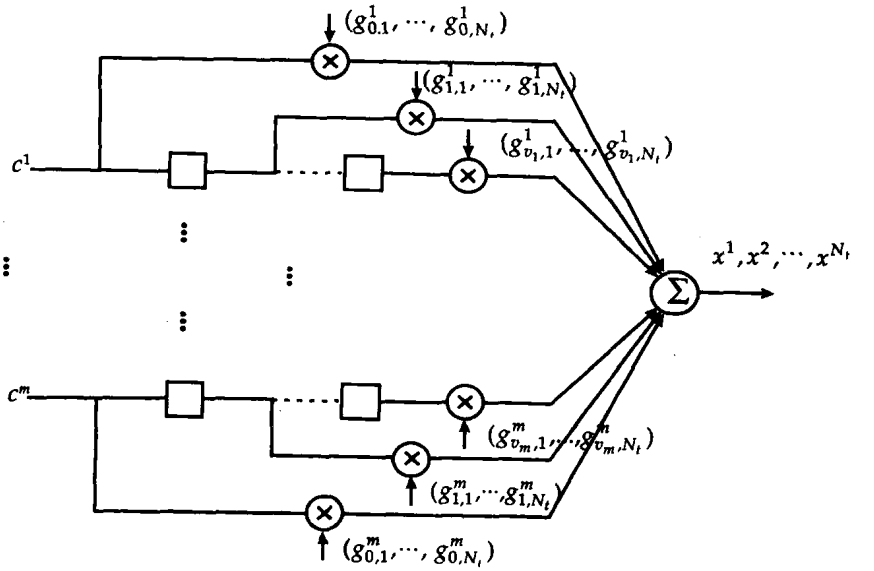


图7-20　STTC的编码器

v_k的值由下列式子确定：

$$v_k=\left\lfloor\frac{v+k-1}{\log_2 M}\right\rfloor \tag{7-61}$$

现在我们分析STTC的性能指标。我们考虑下面两个场景：缓慢雷利衰退（slow Rayleigh Fading）和快速雷利衰退（fast Rayleigh Fading）。

7.9.1　缓慢雷利衰退

对每一个输入符号s_t，空时编码器生成N_t个码符号$c_t^1,c_t^2,\cdots,c_t^{N_t}$，这些码符号将从$N_t$个发射天线中被同时发送。我们定义码向量$\boldsymbol{c}_t=[c_t^1,c_t^2,\cdots,c_t^{N_t}]^T$。假设码向量序列$\boldsymbol{C}=\{\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_J\}$已经传送。考虑ML解码器因合法的码向量序列$\tilde{\boldsymbol{C}}=\{\tilde{\boldsymbol{c}}_1,\tilde{\boldsymbol{c}}_2\cdots,\tilde{\boldsymbol{c}}_J\}$而错误决定的成对错误概率（PEP）。考虑一个长度为J的帧并定义一个$N_t\times N_t$的误差矩阵$\boldsymbol{A}$：

$$\boldsymbol{A}(\boldsymbol{C},\tilde{\boldsymbol{C}})=\sum_{t=1}^{J}(\boldsymbol{c}_t-\tilde{\boldsymbol{c}}_t)(\boldsymbol{c}_t-\tilde{\boldsymbol{c}}_t)^* \tag{7-62}$$

如果接收端具有理想的信道状态信息，错误检验的PEP可以通过下列式子得到：

$$P(\boldsymbol{C}\to\tilde{\boldsymbol{C}})\leqslant\left(\prod_{i=1}^{r}\lambda_i\right)^{-Nr}(E_s/4N_0)^{-rN_r} \tag{7-63}$$

其中E_s是符号能量，N_0是噪音功率谱密度，N_r是接收天线的数量，r是误差矩阵$\boldsymbol{A}$的秩，λ_i $(i=1,\cdots,r)$是$\boldsymbol{A}$的非零特征值。因此可以得到rN_r的分集增益和$\left(\prod_{i=1}^{r}\lambda_i\right)^{-1/r}$的编码增益。将PEP的最大值进行最小化后，我们有下面的设计准则：

Rank-determinant 准则：

(1) Rank准则：为了得到最大的分集优势，矩阵 $A(C,\tilde{C})$ 对所有可能的C和 $\tilde{C}$ 必须是满秩的。

(2) Determinant准则：假设对所有的C和 $\tilde{C}$ ，矩阵 $A(C,\tilde{C})$ 都是满秩的。为了达到最大的编码优势，矩阵 $A(C,\tilde{C})$ 对所有可能的C和 $\tilde{C}$ 的最小值必须进行最大化操作。

7.9.2 快速雷利衰退

记 $\rho(C,\tilde{C})$ 表示使得 $|c_t-\tilde{c}_t|\neq 0$ 的时间实例$1\leqslant t\leqslant J$的集合，n_H 是这种时间实例的数目。对快速雷利衰退而言，下列式子

$$P(C\rightarrow\tilde{C})\leqslant\prod_{t\in\rho(C,\tilde{C})}\left(|c_t-\tilde{c}_t|^2E_s/4N_0\right)^{-N_r} \tag{7-64}$$

导致了下面的对快速衰退的设计准则。

Product-distance 准则：

(1) Distance准则：为了达到分集$n_H N_r$，我们要求对任意两个码字C和 $\tilde{C}$ 的最小符号依符号的汉明距离至少是n_H。

(2) Product准则： 为达到最大编码优势，把积 $\prod_{t\in\rho(C,\tilde{C})}|c_t-\tilde{c}_t|^2$ 对所有的不同的码字C和 $\tilde{C}$ 所取的最小值进行最大化操作。

关于STTC，我们观察到以下几点：

(1) 一个r−STTC的长度限制至少是$r-1$。

(2) 对N_t个发射天线和一个分集增益r，传输率R的上界为$R\leqslant N_t-r+1$。因此，对于完全分集增益($r=N_t$)，最大的速率是对应于2^m信号群的带宽效率为m bit/s/Hz时的全速率。

(3) 如果m是传输率，网格复杂度（状态数量）至少是$2^{m(r-1)}$。

7.10 评注

编码和调制第一次由Massey于1974年作为一个整体进行分析。在此之前，在所有编码数字通信系统中，编码器或译码器和调制/解调器都分别设计和优化。Messay将编码与调制相结合的思想在1982年Ungerboeck的精辟论文中加以具体化。Imai和Hirakawa早在1977年就提出类似的思想，但没有引起充分的关注。TCM的主要优点是在不需要按惯例由编码过程增加带宽的条件下可以得到增强的功率效率。在接下来的几年里由不同的研究人员对TCM理论进化形式化。Calderbank和Mazo证明了非对称一维TCM方案比对称的TCM方案提供更多的编码增益。Wei在1984年提出了旋转不变量TCM方案，它后来由CCITT在新的高速语音带宽调制解调器中采用。STTC是传统的TCM方案用于多天线系统的扩展。这些译码用于提取多样化获得和译码获得。

7.11 小结

- 网格编码调制（TCM）技术允许我们在不扩张带宽或用额外功率的情况下获得更好的性能。
- 网格中任意两个路径间的最小欧几里得距离称为TCM方案的自由欧几里得距离d_{free}。
- 用以得到相同错误概率的编码方案的SNR值与无编码方案的SNR值之差称为编码增益

$g=\mathrm{SNR}|_{\text{无编码}}-\mathrm{SNR}|_{\text{有编码}}$。在高SNR情况下，编码增益可以表示为

$$g_\infty = g|_{SNR\to\infty} = 10\log\frac{(d_{free}^2/E_s)_{\text{有编码}}}{(d_{free}^2/E_s)_{\text{无编码}}}$$

其中g_∞表示渐近编码增益，E_S为平均信号能量。

- 集合分割映射的基础是连续将一个扩展的2^{m+1}重信号集分割成最小欧几里得距离越来越大的子集。每一次我们分割集合时，减少了子集中信号点的数量，但增加了子集中信号点间的最小距离。
- Ungerboeck对AWGN信道的TCM设计准则（基于启发式）为

(1) **准则1**：平行转换，如果存在，必须与集合分割树最底层的子集中的信号相联系。这些信号具有最小欧几里得距离Δ_{m+1}。

(2) **准则2**：源于或汇合于一个状态的转换必须与集合分割第一步的信号相联系。这些信号之间的欧几里得距离至少为Δ_1。

(3) **准则3**：所有信号都以相同频率在网格图中使用。

- 维特比算法可对TCM方案中收到的信号进行译码。译码算法中用到的分支尺度为接收到的信号与网格中对应分支的信号之间的欧几里得距离。
- 那些相距为自由距离的近邻的平均数$N(d_{free})$给出了网格中与传送序列相距为自由欧几里得距离的路径的平均数。这个数字连同d_{free}用于对错误事件概率的评估。
- 错误概率 $P_e \leqslant T(D)|_{D=e^{-1/4N_0}}$，其中 $T(D)=\frac{1}{N}\mathbf{1}^{\mathrm{T}}\boldsymbol{G}\mathbf{1}$，而矩阵$\boldsymbol{G}=\sum_{l=1}^{\infty}\sum_{E_l\neq 0}\prod_{n=1}^{l}\boldsymbol{G}(\mathbf{e}_n)$为标量转换函数。错误事件概率的一个更紧的上界为

$$P_e \leqslant \frac{1}{2}\mathrm{erfc}\left(\sqrt{\frac{d_{free}^2}{4N_0}}\right)e^{\frac{d_{free}^2}{4N_0}}T(D)\Big|_{D=e^{\frac{-1}{4N_0}}}$$

- 对衰退信道，有$P_2(\boldsymbol{s}_l,\hat{\boldsymbol{s}}_l)\leqslant\frac{((1+K)e^{-K})^{l_\eta}}{\left(\frac{1}{4N_0}\right)^{l_\eta}d_p^2(l_\eta)}$，其中$d_p^2(l_\eta)\leqslant\prod_{i\in\eta}|s_i-\hat{s}_i|^2$。这里的项$l_\eta$是错误事件$(\boldsymbol{s}_l,\hat{\boldsymbol{s}}_l)$的有效长度，$K$是Rician参数。因此错误事件概率主要取决于最小有效长度l_η和最小乘积距离$d_p^2(l_\eta)$。
- 在高SNR和小K值的情况下，衰退信道的TCM的设计规则为：

(1) 使码的有效长度L最大化。

(2) 使最小乘积距离$d_p^2(l_\eta)$达到最小。

- 由于最小乘积距离的增加而产生的编码增益的增长可表示为

$$\Delta_g = SNR_1 - SNR_2|_{P_{e1}=P_{e2}} = \frac{10}{L}\log\frac{d_{p2}^2(L)\alpha_1}{d_{p1}^2(L)\alpha_2}$$

其中$a_i(i=1,2)$为有效长度为L的TCM方案i的码序列平均数i。

- 空时网格玛（STTC）是一个为多天线发射的编码技术。STTC是TCM 方案，其中网格的每一个分支均用对应于由N_t发射天线传输的N_t个信号进行标注。
- 假定接收端是理想的信道状态信息，对STTC码进行解码，我们需要使用向量维特比算法。

• 缓慢雷利衰退的设计准则可以通过Rank-determinant准则表示为：

(1) Rank准则：为了得到最大的分集优势，矩阵 $A(C,\tilde{C})$ 对所有可能的C和 $\tilde{C}$ 必须是满秩的。

(2) Determinant准则：假设对所有的C和 $\tilde{C}$，矩阵 $A(C,\tilde{C})$ 都是满秩的。为了达到最大的编码优势，矩阵 $A(C,\tilde{C})$ 对所有可能的C和 $\tilde{C}$ 的最小值必须进行最大化操作。

• 快速雷利衰退的设计准则可以通过Product-distance准则表示为：

(1) Distance准则：为了达到分集$n_H N_r$，我们要求对任意两个码字C和 $\tilde{C}$，这两个码字的最小符号依符号的汉明距离至少是n_H。

(2) Product准则：为达到最大编码优势，把积 $\prod_{i\in p(C,\tilde{C})}|c_i-\tilde{c}_i|^2$ 对所有的不同的码字C和 $\tilde{C}$ 所取的最小值进行最大化操作。

有时候一点点的不精确可以省下大量解释。

——H.H.Munro (Saki)（1870—1916）

习题

7.1 考虑由下列定义的码率为2/3的卷积码：

$$G(D)=\begin{bmatrix}1 & D & D+D^2\\ D^2 & 1+D & 1+D+D^2\end{bmatrix} \tag{7-65}$$

这个码用到使用格雷编码（每个符号被赋值3比特，这样一来两个相连符号的码只在一个比特位不同）的8PSK信号集。该TCM方案的吞吐量为2bit/s/Hz。

(1) 在该编码器的网格图中有多少状态?

(2) 求自由欧几里得距离。

(3) 关于吞吐量为2bit/s/Hz的无编码的QPSK，求渐近编码增益。

7.2 在习题7.1中，假设用的是自然映射而不是格雷编码，即$s_0\rightarrow 000, s_1\rightarrow 001, \cdots, s_7\rightarrow 111$。

(1) 求自由欧几里得距离。

(2) 求关于无编码QPSK（2bit/s/Hz）的渐近编码增益。

7.3 考虑如图7-21所示的编码器：

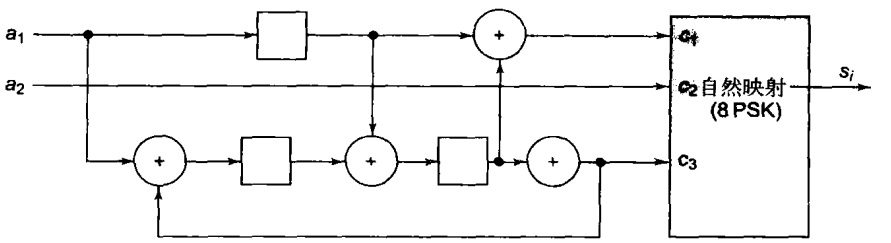


图7-21 习题7.3中的图

(1) 画出该编码器的状态图。

(2) 画出该编码器的网格图。

(3) 求自由欧几里得距离d_{free}。在网格图中找出一对能导致d_{free}的路径。$N(d^2_{free})$是什么?

(4) 接下来用集合分割方法将8PSK中的符号赋值给网格图的分支。现在d_{free}是什么?

(5) 用这个编码器将下列比特流进行编码：1 0 0 1 0 0 0 1 0 1 0 …。对自然映射和使用集合分割的映射给出你的答案。

(6) 比较两种不同的映射的渐近编码增益。

7.4 我们想设计一个TCM方案，使它有后跟一个信号映射码率为2/3的卷积编码器。该映射是根据图7-22所示的非对称星座图的集合分解而成的。它的网格有四个状态，且是全连通的。

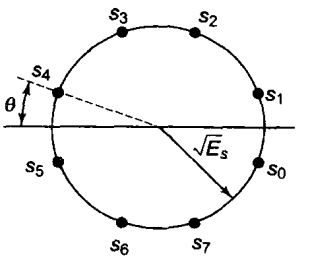


图7-22　习题7.4中的图

(1) 对下面的非对称星座图，给出集合分割。

(2) 该非对称TCM方案的自由欧几里得距离d_{free}是什么？将它同使用标准8PSK信号星座时的d_{free}进行比较。

(3) 你将怎样选择θ的值来改善使用图7-22所示的非对称信号星座的TCM方案的性能？

7.5 考虑图7-23所示的码率为3/4的编码器。该编码器的四个输出比特被映射到如下所示的星座图中16个可能符号之一。用Ungerboek设计规则为AWGN信道设计一个TCM方案。对于无编码8PSK，渐近编码增益是多少？

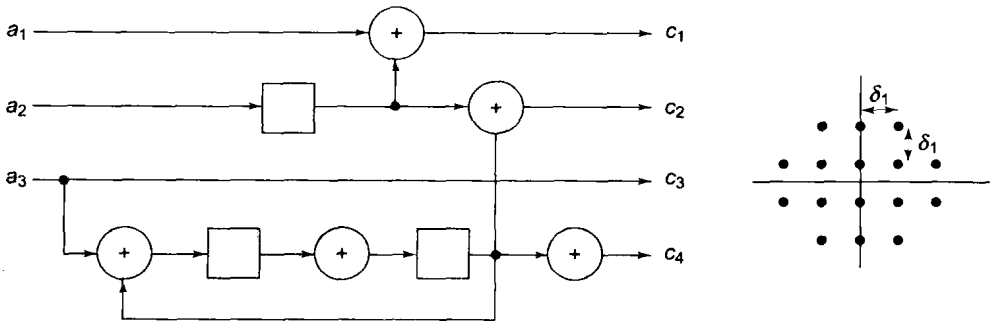


图7-23　习题7.5中的图

7.6 考虑Rician衰退信道的成对事件错误概率表达式

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) \leqslant \prod_{i=1}^{l} \frac{1+K}{1+K+\dfrac{1}{4N_0}|s_i-\hat{s}_i|^2} \exp\left[-\frac{K\dfrac{1}{4N_0}|s_i-\hat{s}_i|^2}{1+K+\dfrac{1}{4N_0}|s_i-\hat{s}_i|^2}\right] \tag{7-66}$$

(1) 证明当K的值很大时上述不等式可简化为

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) \leqslant \prod_{i=1}^{l} \exp\left[-\frac{1}{4N_0}|s_i-\hat{s}_i|^2\right] \tag{7-67}$$

(2) 证明对低SNR的情况，原来不等式可以表示为

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) \leqslant \exp\left[\frac{d_E^2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l)}{4N_0}\right] \tag{7-68}$$

7.7 考虑一个对Rician衰退信道设计的TCM方案，它的有效长度为L，最小乘积距离为$d_p^2(L)$。假设我们想重新设计这个码使对SNR的改善为3dB。

(1) 在$d_p^2(L)$保持不变的情况下，计算所期望的有效长度L。

(2) 在有效长度L保持不变的情况下，计算所期望的乘积距离$d_p^2(L)$。

7.8 假定你不得不设计一个AGWN信道（SNR＝γ）的TCM方案，它期望的BER为P_e。画出你将如何设计这样一个方案的流程图。

(1) 在你的网格中将有多少个状态？

(2) 你将怎样设计卷积编码器？

(3) 在你的设计中有平行路径吗？

(4) 你将选取什么样的调制方案？为什么？

(5) 你将如何用调制方案的符号对分支赋值？

7.9 对维特比译码，所用的尺度形式如下：

$$m(\boldsymbol{r}_l, \boldsymbol{s}_l) = \ln p(\boldsymbol{r}_l|\boldsymbol{s}_l) \tag{7-69}$$

(1) 选择这种尺度的道理是什么？

(2) 给出另一个适用于衰退信道的尺度。对你的答案给出理由。

7.10 一个为Rician衰退信道设计的TCM方案在高SNR环境（SNR＝20dB）中有$L=5$且$d_p^2(L)=2.34E_s^2$。它必须重新设计以得到2dB的改善。

(1) 新码的$d_p^2(L)$是多少？

(2) 对新的d_{free}^2给出你的评论。

7.11 考虑图7-24所示的由一个码率为1/2的卷积编码器结合一个映射的TCM方案。

(1) 画出该编码器的网格图。

(2) 确定标量转换函数$T(D)$。

(3) 确定扩张的生成函数$T(D, L, I)$。

(4) 该码的最小汉明距离（d_{free}^H）是什么？

(5) 有多少路径满足这个d_{free}^H？

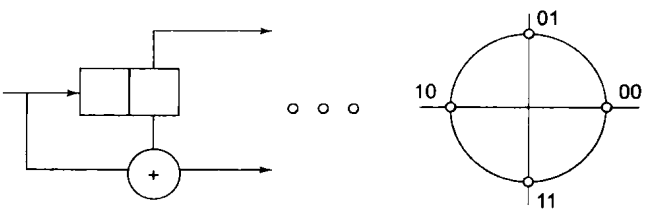


图7-24 习题7.11中的图

7.12 考虑事件对错误概率$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l)$。

(1) 对最大似然译码器，证明

$$P_2(\boldsymbol{s}_l, \hat{\boldsymbol{s}}_l) = \int_r f(\boldsymbol{r}) p_{R|S}(\boldsymbol{r}|\boldsymbol{s}_l)\, d\boldsymbol{r} \tag{7-70}$$

其中r是接收到的向量，$p_{R|S}(r|s_l)$为信道转移概率密度函数，且

$$f(\boldsymbol{r}) = \begin{cases} 1 & \text{如果 } p_{R|S}(\boldsymbol{r}|\hat{\boldsymbol{s}}_l) \geqslant p_{R|S}(\boldsymbol{r}|\boldsymbol{s}_l) \\ 0 & \text{否则} \end{cases} \tag{7-71}$$

(2) 证明

$$f(\boldsymbol{r}) \leqslant \frac{p_{R|S}(\boldsymbol{r}|\hat{\boldsymbol{s}}_l)}{p_{R|S}(\boldsymbol{r}|\boldsymbol{s}_l)} \tag{7-72}$$

$$P_2(s_l, \hat{s}_l) \leq \int_r \sqrt{p_{R|S}(r|\hat{s}_l)p_{R|S}(r|s_l)}dr \tag{7-73}$$

7.13　考虑两个TCM机制（见图7-25）。两个机制均使用4状态完全连接的网格以及8PSK。

(1) 这两种机制中的哪种遵守Ungerboeck的TCM设计准则？

(2) 找出这两个机制的 d^2_{free} 和 $d^2_p(L)$ 。

(3) 哪种机制更适合于纯AWGN信道？哪种机制更适合于衰退信道？试给出分析。

注意，TCM1也称为Wilson Leung码。

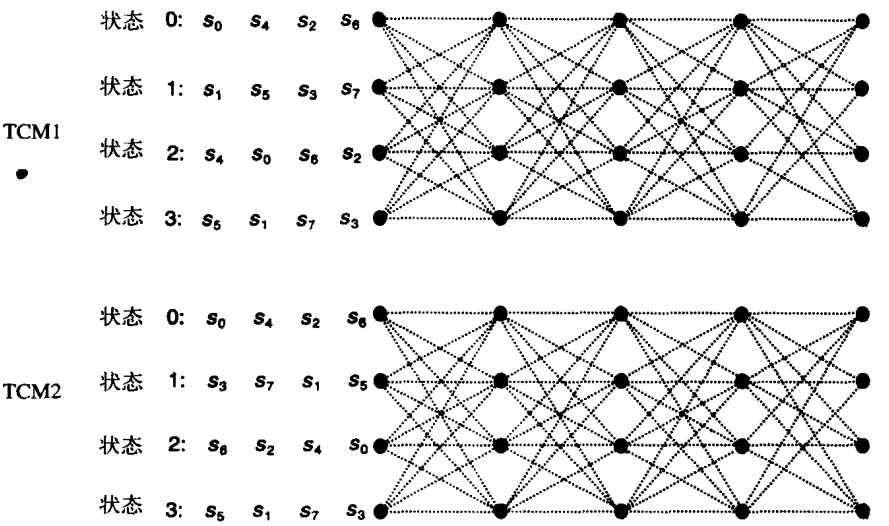


图7-25　练习7.13的图

上机习题

7.14　写一个程序，在给出网格结构和映射规则时它执行网格编码调制。该程序应该接受比特流输入并输出符号序列。该程序的输入可能来自两个方面，一个给出网格状态间的连接性（实质上是网格的结构），另一个给出分支标记。

7.15　写一个程序，当给出网格图及分支标记时它可计算TCM方案的平方自由欧几里得距离 d^2_{free} 、有效长度L和最小乘积距离 $d^2_p(L)$。

7.16　写一个程序对一个输入符号流进行维特比译码。该程序利用给出的网格和网格图中分支的标记信息。

7.17　检验本章中给出的AWGN环境下不同TCM方案的性能。为了能做到这一点，给TCM编码器一个很长的随机比特链作为输入。该编码器将产生一个符号序列（模拟波形）。让AWGN使用不同噪声功率对这些符号进行破坏，即对不同的SNR进行模拟。用维特比算法对接收到的遭到破坏的符号（扭曲了的波形）进行译码。给出BER相对SNR的平面图，将其与理论上预计的错误率进行比较。

7.18　写一个程序来观测维特比译码器译码窗大小的效果。给出一个错误率相对窗大小的平面图及计算量相对窗大小的平面图。

7.19　写一个程序，使之通过穷举搜索来确定一个码率为2/3的为AWGN设计的TCM编码器

(使 d_{free}^2 最大化)。假设网格图是全连通的，其中有四个状态。该网格的分支用一个8PSK信号集中的符号标记。修改程序使之通过穷举搜索找出一个好的TCM方案，它有四个状态的网格，其中可能会有平行分支。

7.20 写一个程序，使之通过穷举搜索来确定一个码率为2/3的为衰退信道设计的TCM编码器(使 $d_p^2(L)$ 最大化)。假设网格图是全连通的，其中有四个状态。该网格的分支用一个8PSK信号集中的符号标记。对在搜索过程中发现的较好的码，列出 $d_p^2(L)$ 和 L。

7.21 针对不同的 K 值 (Rician参数)，画出下列情况下描述 P_e 和 L_{eff} 之间关系的曲线族

(1) 高SNR。

(2) 低SNR。

对这些平面图做出评述。

7.22 编写一个程序，穷尽搜索在下面两个信道上的最优4状态，完全连接的STTC。

(1) 缓慢雷利衰退信道。

(2) 快速雷利衰退信道。

此处的信号群为MPSK。

第三部分 安全通信编码

第8章 密 码 学

不可能感觉不到这些数学公式具有它们自己独立的存在性和智能，它们比我们更聪明，甚至比它们的发现者都更聪明。我们从中得到的多于我们原来给予的。

——Heinrich Hertz（1857—1894）

8.1 密码学简介

密码学是关于设计将信息安全传送，只有应收信人才能恢复该信息的方法的科学。加密建立在一种将信息弄乱成不可读或不可识别的形式的算法上，而解密是将弄乱了的信息重新恢复原状的过程（见图8-1）。

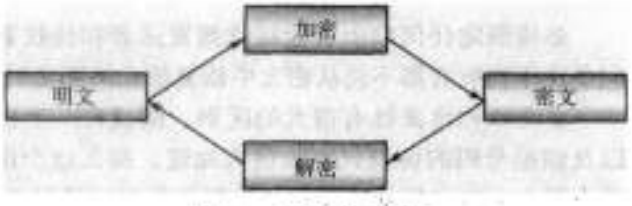


图8-1 加密和解密过程

密码体制（cryptosystem）是一些算法和相关的隐藏信息和显露（不隐藏）信息的程序的集合。**密码分析**（cryptanalysis）是分析一个密码体制的过程（实际上是一种艺术），或者来检验它的完整性，或者为了不可告人的目的而攻破它。**攻击者**（attacker）是进行非法密码分析以破解一个密码体制的个人或系统。攻击者有时又称为黑客、闯入者或偷听者。攻击一个密码体制的过程通常称为**破译**（cracking）。

密码分析者（cryptanalyst）的工作是找到密码体制的弱点。在许多情况下，密码体制的开发者宣布一种公开挑战：任何人如果破译该方案可以得到一大笔奖金。一旦一个密码体制被攻破（且密码分析者泄露了他的技术），该方案的设计者就会试图强化其算法。然而，一个密码体制被攻破并不表明它无用了。黑客们可能在最优条件下用常人没有的设备（快速计算机、专用微处理器等）攻破该体制。一些密码体制用攻破它的时间和计算设备的价格来划分等级。

在过去几十年里，本质上是数学的密码算法发展得非常先进，以至于它们只能用计算机处理。这实际上意味着未编码信息（加密前）是二元形式，因此可以是任何东西：一张图片、一段声音、一份如电子邮件的文件或一段录像。

密码学并不像许多人认为的那样仅用于军事和外交通信。在现实中，密码学具有许多商业用途：保护公司机密信息、保护电话通话、允许人们通过因特网订货而不用担心信用卡号码被截获和滥用。密码学完全是用于保护个人和组织的隐私。例如，密码学常用于防止伪造者假冒获奖彩票。每张彩票上都印了两个号码，一个是明文，另一个是相应的密码。除非伪造者已经分析出了彩票的密码体制，否则他们将不能打印出可以接受的伪造的获奖彩票。

本章组织如下。首先我们一般性地概括不同的加密技术，然后将学习私钥密码学。我们

将详细讨论一些特别的私钥密码技术，尤其详细介绍数据加密标准（DES）和国际数据加密算法（IDEA）。然后介绍公钥密码学。我们将详细讨论两种很流行的公钥密码技术：RSA算法（由Rivest、Shamir和Adelman三人提出）和PGP（一种混合密码算法）。接下来本章将介绍Diffie-Hellman协议和椭圆曲线密码学。本章还将介绍当今使用的一些其他密码技术。密码学领域正在广泛扩展，本章将介绍两种最新进展——量子密码学和生物加密。本章结论部分将讨论密码分析和密码学中的政治因素。

8.2 加密技术概述

密码体制的目标是对网络上交换的信息提供高层次的保密性、完整性、无抵赖性和认证。

消息和存储数据的保密性是通过加密技术隐藏信息来保护的。消息的完整性确保从它产生到被收信人打开期间保持不变。无抵赖性可为某消息来自某人提供一种证明，即使他们试图否认。认证提供两种服务。首先，它排除对信息来源的疑虑，其次，它检查登录到系统的用户的身份，而且持续检查他们的身份以防他人攻入系统。

定义8.1　*要发送的消息称为*明文（plaintext）。*该消息用一个密码算法编码。这个过程称为*加密（encryption）。*加了密的消息称为*密文（ciphertext），*而它由*解密（decryption）*过程变回到明文。*

必须假定任何窃听者能接收到发送者和接收者之间的所有通信。仅当即使在这种完全访问条件下窃听者都不能从密文中恢复原来的明文时，加密方法才算得上是安全的。

安全性和隐匿性有很大的区别。假设在一个机场的锁柜里为某人存放了一个消息，机场以及锁柜号码的详情只有收信人知道，那么这个消息不是安全的，而只是隐匿的。但是如果所有潜在窃听者都知道锁柜的位置但他们还是打不开锁柜拿到消息，那么这个消息是安全的。

定义8.2　*密钥是一个数值，它使一个密码算法以一种特殊方式运行来产生一特别密文。密钥大小通常用比特数来衡量。密钥越大，算法越安全。*

例8.1　假定我们要对下列二元数据流（可能来自于声音、录像、文本或任何其他来源）加密后再发送

$$0\,1\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1\,1\,1\,1\,1\cdots$$

我们可以用4比特长的密钥$x=1011$来加密这个比特流。为了实施加密，明文（二元比特流）首先分成4比特的组：

$$0\,1\,1\,0\quad 0\,0\,1\,0\quad 1\,0\,0\,1\quad 1\,1\,1\,1\cdots$$

每一个子组与密钥$x=1011$进行异或（二元加法）运算。加密后的消息将变为

$$1\,1\,0\,1\quad 1\,0\,0\,1\quad 0\,0\,1\,0\quad 0\,1\,0\,0\cdots$$

收信人必须也拥有密钥的知识才能解密消息。在这种情况下解密过程非常简单。密文（接收到的二元比特流）首先分成4比特的组，每一个子组与密钥$x=1011$进行异或运算。解密后的消息将是原来的明文

$$0\,1\,1\,0\quad 0\,0\,1\,0\quad 1\,0\,0\,1\quad 1\,1\,1\,1\cdots$$

应该注意到在加密和解密中只用到一个密钥。

例8.2 让我们来为文本消息发明一种算法，我们将称其为字符 $+x$。设$x=5$。在这种加密技术中，我们把每一个字符用它后面的第5个字符来取代，即A变成F，B变成G，C变成H，等等。加密消息的接收者只需要知道x的值便可解密消息。密钥必须与要传送的加密消息分开。因为只有一个密钥用于加密和解密，这类技术称为**对称密码学**（symmetric cryptography）或**单密钥密码学**（single key cryptography），也可称为**保密密钥密码学**（secret key cryptography）。这种技术的问题是密钥需要保密。另外，密钥必须不断更换以确保传输的保密性。这表明保密密钥（或密钥集合）要传送给收信人。这可以通过非电子方式解决。

为了避免传递密钥的问题，Difie和Hellman提出了**公钥密码学**（public key cryptography）的概念。该技术又称为**非对称加密**（asymmetric encryption）。这个概念很简单。有两个密钥，一个私自保留，另一个公开。一个密钥可以上锁，另一个密钥可以解锁。

例8.3 假设我们想用公钥加密技术将一个加密消息送给收信人A。要做到这一点，我们将用到收信人A的公钥来加密消息。当消息被收到后，收信人A用他的私钥解密。只有收信人A的私钥能解密用他的公钥加密的消息。类似地，收信人B也只能解密那些用他的公钥加密的消息。所以，不需要再传递私钥，从而我们不需要某个可靠的通信信道来传递密钥。

让我们考虑另一种情况。假设我们要传送给某人一个消息，而且提供证据表明消息确实是从我们这里传送的（传假消息或错消息可造成很大的伤害！）。为了保持消息的保密性并提供认证（它确实是从我们这里传送的），我们用我们的私钥对消息进行特殊的加密，然后再用收信人的公钥加密。收信人用他的私钥打开消息，然后用我们的公钥验证真实性。这种技术称为使用**数字签名**（digital signature）。

还有一个重要的加密技术称为**单向函数**（one-way function）。这是一种不可逆的快速加密方法。加密过程既容易又快，但解密并非如此。假设我们发送一个文件给收信人A，而且想在以后查看文件是否被篡改过。我们可以用一个单向函数来做到这一点。该单向函数产生一个固定长度的值称为**散列**（也叫做**消息摘要**）。该散列是该消息的惟一标识，可以同消息一起传送。收信人A可以运行同一个单向函数来检查文件是否被篡改过。

在消息加密和解密中实际用到的数学函数称为**密码算法**（cryptographic algorithm）或**密码**（cipher）。这只是用来传送和接收安全消息的系统的一部分。当我们详细讨论一些具体的系统时这会变得更清晰。

对于历史上的多数密码，传送的消息的安全性依赖于算法本身也要保密。这种技术称为**受限算法**（restricted algorithm）。它有下面的基本缺点：

(1) 很明显，算法只能限制在那些你希望能对你的消息进行译码的人之间发布。因此，对任何不同组用户都必须发明一个新算法。

(2) 一个大的或常变动的组不能用这些密码，因为每次有用户离开时所有人都要更换算法。

(3) 一旦算法被以任何方式窃取，必须实现一种新算法。

因为有这些缺点，受限算法不再流行，取而代之的是仅依赖密钥的算法。

实际上所有现代密码系统都用到密钥。用到密钥的算法允许算法的全部信息公开，这是因为它的安全性只依赖于密钥。对于只依赖密钥的算法，明文被算法使用某个密钥加密和解密，而所得密文只依赖于密钥，而不是算法。这说明窃听者可以有完整的所使用的算法，但因没有加密消息时使用的特殊密钥，因此他们也无计可施。

本章主要讲述**密码算法**。算法这个词语可以追溯到公元9世纪波斯数学家的名字，Abu

Abdullah Muhammad ibn Musa Al-Khwarizmi（公元780—850）。此人的工作在于介绍了印度的数码和代数思想。引导实用算法设计的两个一般思想是**扩散**和**混淆**。扩散意思是使得明文的单个数字影响多个密文的数字，从而隐藏明文的统计结构。这种思想的一种扩展是让密钥的单个位影响密文中的很多位。混淆意思是通过一定的变换，使得明文和密文之间的统计依赖性变复杂。

8.3　加密算法所用到的运算

尽管计算机出现之后加密/解密方法发生了巨大变化，对一个明文仍然只能执行两种基本运算：替换和转换。真正的区别在于，早期的运算是对字符的，而今天的运算是对二元比特的。

(1) **替换**（Substitution）　替换运算将明文中的比特替换为算法所确定的其他比特以产生密文。这种替换只需要逆转过来就可以从密文产生明文。这样做可能非常复杂。例如一个明文字符可以对应若干密文字符（同音替换），或一个明文字符用另一段文字中对应位置上的字符替代（流动密码）。

例8.4　凯撒是最早在战争中使用替换加密来给部队发送情报的人之一。他发明的替换方法将字符向前移动三个位置。故

$$\text{THIS IS SUBSTITUTION CIPHER} \tag{8-1}$$

变成

$$\text{WKLV LV VXEVWLWXWLRQ FLSKHU} \tag{8-2}$$

(2) **转换**（Transposition）　转换（或置换）并不改变明文中任何比特，但把它们的位置交换。如果上述密文再经过更多转换，那么最后的结果将有更高的安全性。

(3) **异或**（XOR）　XOR就是异或运算。它是一个布尔算子，若两个比特之一为真，则结果为真，若两个都为真或两个都为假，则结果为假。例如：

$$\begin{aligned} &0 \text{ XOR } 0 = 0 \\ &1 \text{ XOR } 0 = 1 \\ &0 \text{ XOR } 1 = 1 \\ &1 \text{ XOR } 1 = 0 \end{aligned} \tag{8-3}$$

数量惊人的商业软件用简单的异或函数来提供安全性，包括美国数字蜂窝电话网和许多办公应用软件，而它很容易破译。我们将在本章后面看到，当处理长比特组，而这些比特组也经过替代和/或转换时，该XOR运算是许多高级密码算法中至关重要的部分。

8.4　对称（保密密钥）密码学

对称算法（或**单密钥算法**或**保密密钥算法**）只有一个在加密和解密中使用的密钥，它因此而得名。为了使收信人能解密消息，他们需要有同样的密钥。这带来一个主要的问题，即密钥分发问题。除非收信人亲自和发信人见面并拿到密钥，否则密钥必须传给收信人，从而易于被窃听者截获。但是，单密钥算法速度快且高效，特别是当有大量数据需要处理时。

在对称密码学中，交换消息的双方用相同的算法。只有密钥需要时常更换。同样的明文在不同密钥下变成不同的密文。加密算法是公开的，因此应该牢固且经过很好的测试。算法越强壮，攻击者越不可能对密码解密。

在产生强密文时，密钥的大小很关键。美国国家安全局（NSA）在20世纪90年代中期声明，他们可以接受40比特长的密钥（即他们可以有效且快速地破译）。不断增长的处理器速度，连同松散耦合的多处理器配置，已经给了潜在黑客攻破这么短的密钥的能力。在1998年，人们建议要使密码有强度，密钥大小需要至少56比特长。早在1996年一个专家组认为90比特是更合适的长度。今天最安全的方案用128比特或更长的密钥。

对称密码学提供了满足消息内容安全性要求的途径，因为没有密钥便不能读它的内容。但是仍然存在暴露秘密的风险，因为任何一方都不能确定对方是否将密钥暴露给第三方（不管意外地还是故意地）。

对称密码学也可以用来提供消息完整性和认证。送信人生成消息的一个摘要，或**消息认证码**（message authentication code，MAC），用保密密钥对其加密，然后同消息一起送出。收信人收到后重新产生MAC，解密发送的MAC，并对二者进行比较。如果它们相同，则接收到的消息必定和发送的消息相同。

如前所述，对称方案的主要困难是双方必须都拥有同一密钥。另外，如果密钥泄露，所有消息传输安全措施就都破坏了。实现密钥产生和传输安全机制的步骤称为**密钥管理**（key management）。

这种技术并没充分强调无抵赖要求，因为通信双方有同样的密钥。因此任何一方都面临对方欺诈和虚假消息的风险，而且任何一方都有理由不承认发送了那样的消息，因为对方可能把密钥泄露了。

有两类对称算法——分组密码和流密码。

定义8.3 分组密码（block cipher）通常作用于多个比特构成的组上，称为组。每一个组经过几次处理。在每一轮密钥以一种单一的方式起作用。迭代的次数越多，加密过程就越长，但产生的密文也越安全。

定义8.4 流密码（stream cipher）对明文的作用是每次一比特。明文以未加工的比特流形式送给加密算法。分组密码使用同一密钥从相同明文产生相同密文，而流密码则不是这样。在同样条件下流密码产生的密文将发生变化。

密钥应该有多长呢？对这个问题没有一个答案。这要看具体情况。要确定我们需要什么样的安全性，必须回答下面的问题：

(1) 要保护的数据价值有多高？

(2) 它需要多长时间的安全性？

(3) 密码分析者/黑客所能获得的资源是什么？

一个顾客名单可能值1 000元，一个广告数据可能值50 000元，而一个数字货币系统的主密钥可能值数百万元。在股票市场上，秘密信息需要保持几分钟。在报业，今天的秘密是明天的头条新闻。一个国家的人口普查数据必须秘密保存几个月（如果不是几年的话）。公司交易秘密是对手公司感兴趣的，而军事秘密是军事上的对手感兴趣的。因此，安全性要求可以用这些条款来描述。例如，我们可能要求密钥长度满足以下条件，假定科技进步的速度为每年25%，那么一个黑客用一百万元的资源有0.000 1%的可能在一年内攻破系统。不同应用环境中对密钥长度的最低要求在表8-1中列出。这个表仅作指导用。

表8-1　不同应用对密钥长度的最低要求

信息种类	生命期	最小密钥长度
军事战术信息	几分钟到几小时	56～64比特
产品公告	几天到几周	64比特
银行利率	几天到几周	64比特
贸易秘密	几十年	112比特
核弹秘密	>50年	128比特
间谍身份	>50年	128比特
个人事务	>60年	>128比特
外交麻烦	>70年	>128比特

未来计算能力是很难预测的。经验方法是计算设备的效率除以价格每18个月翻一番，每五年增长10倍。因此，50年后，最快的计算机将比今天的快一千万倍！这些数字是对通用计算机而言的。我们无法预测在未来几年内会开发出什么样的密码系统破译专用计算机。

本章将介绍两种对称算法，都是分组密码。它们是**数据加密标准**（Data Encryption Standard, DES）和**国际数据加密算法**（International Data Encryption Algorithm, IDEA）。

8.5　数据加密标准（DES）

DES为数据加密标准的缩写，是联邦信息处理标准（FIPS）46-3号的名字，它描述了数据加密算法（DEA）。DEA也是ANSI标准X9.32中定义的。

由IBM设计的DES是在美国国家标准局（NBS）征集满足下列要求的标准密码算法时给出的：

1) 提供高度的安全性。

2) 安全性依赖于密钥，而不依赖于算法的秘密。

3) 安全性是可以评估的。

4) 算法有完整说明且容易理解。

5) 使用效率高且可适应不同用途。

6) 必须对所有用户均可用。

7) 必须可以出口。

DEA实质上是从IBM在20世纪70年代早期设计的“算法Lucifer”改进而来的。美国国家标准局在1975年公布了数据加密标准。尽管算法基本由IBM设计，NSA和NBS（现为NIST）在开发的最后阶段也起了实实在在的作用。DES自从公布以来得到了广泛的研究，它是世界上最著名和用得最广泛的对称算法。

在运行中，DEA的分组长度为64比特，密钥长度为56比特（8个奇偶校验比特从64比特完整密钥中剔除）。DEA是对称密码体制，一种16轮的Feistel密码，最初是为硬件实现设计的。当在通信中应用时，发送者和接收者都必须知道相同的密钥，它可以被用于加密和解密消息，或产生和检验消息验证码（MAC）。DEA也可以用于单用户加密，比如在硬盘上储存加密后的文件。在多用户环境中，安全的密钥分发可能很困难；公钥密码学为这个问题提供了理想的解决办法。

NIST每5年重新批准DES（FIPS46-1、FIPS46-2、FIPS46-3）。FIPS46-3在1999年10月份对DES进行了重新确认，但单一DES只允许在遗留体系中使用。FIPS46-3包括三重DES（对应X9.52中的TDEA）的定义。在几年内，DES和三重DES将被高级加密标准取代。

DES现在已在世界范围内使用20年了，因为它是一个标准，这表明任何实现了DES的系统可以与任何其他使用它的系统通信。全世界的银行和商业部门都使用DES，在网络中（如Kerberos）也用它，还用它来保护UNIX操作系统中的口令文件（如CRYPT）。

(1) **DES加密**。DES是一个对称的分组密码算法，它的密钥长度为64比特，分组长度为64比特（即该算法连续对64比特的明文组进行操作）。由于是对称的，同样的密钥用于加密也用于解密，而且DES的加密和解密都用同一个算法。

首先根据一个表（初始置换）执行转换运算，然后将64比特明文组分为两个32比特组，对每一半要进行16次相同的运算，称为16轮。这两半最后又合在一起进行初始置换的逆运算。第一个转换的目的不明确，因为它并不影响算法的安全性，但可能是为了允许明文和密文以字节大小载入8比特芯片。

在任意一轮，只有原来64比特中的一半参与运算。在不同轮中轮流处理这两半。在DES的一个轮中包含下列功能：

(2) **密钥变换**。64比特密钥通过移去每8比特中的第8个比特（这些比特有时用于错误检测）缩减为56比特，然后产生16个不同的48比特子密钥——每一个子密钥用于一轮加密。这一过程是先把56比特密钥分成两半，然后根据轮数的不同把它们左循环移位一次或两次。在此之后再选出48比特。因为发生移位，每一个子密钥用到密钥中不同的比特组。这一过程称为压缩置换，因为有位的转换和整体大小的缩减。

(3) **扩张置换**。密钥变换后，不管一组中的哪一半参与运算，都要经过一个扩张置换。在这种运算中，通过让第1和第4个比特在输出中出现两次，即第4比特变成第5和第7个比特（见图8-2），我们可以同时得到扩张和转换变换。

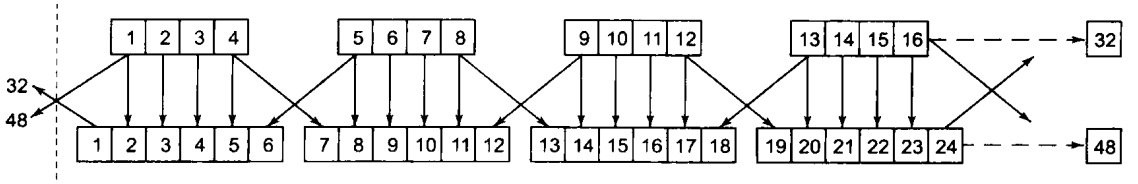


图8-2 扩展置换

扩张置换得到三个结果：首先，它增加了那半个组的大小，从32比特增加到48比特，同压缩后的密钥子集的比特数相同，这是很重要的，因为接下来的运算是把它们异或（XOR）在一起。其次，它为替换运算产生了一个更长的数据串，然后又将其压缩。最后，也是最重要的一点，因为在接下来的替换中第1和第4比特都在两个S盒中（马上给出描述），它们影响到两个替换。这样做的效果是输出比特对输入比特的依赖性大大提高，因此算法的安全性也大大提高。

(4) **XOR**。上述得到的48比特组再与那一轮中的合适子密钥进行XOR运算。

(5) **替换**。接下来的运算是对扩张后的组进行替换。有8个替换盒，称为S盒。第1个S盒对48比特扩张组中的前6比特进行运算，第2个S盒对接下来的6比特进行运算，依此类推。每一个S盒都有一个4行16列的表，表中每一个元素是一个4比特数。S盒的6比特输入以下列方式用于查找表中适当的元素：第1和第6比特构成的2比特数对应行数，第2到第5比特结合在一起形成的4比特数对应一个特定的列。替换阶段的纯结果是8个4比特组，它们结合在一起构成一个32比特组。

S盒的非线性关系提供了DES的安全性，DES算法内所有其他过程都是线性的，这样会很容易分析。

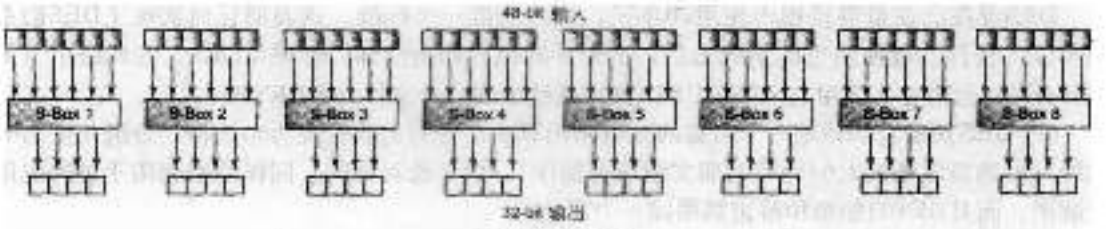


图8-3　S盒替换

(6) 置换。替换阶段得到的32比特输出再用一个表直接转换，这个表有时称为P盒。

当所有轮都完成之后，那两个32比特"半组"再结合成64比特输出，再经过最后置换，而此时得到的64比特组就是DES对输入明文组加密后的密文。

(7) DES解密。解密DES很容易（对有正确密钥的人来说！）。感谢它的设计，解密算法和加密算法一样——惟一的更改是要解密DES密文，在每一轮中用到的那些子密钥要以相反的次序使用，即首先用的是第16个子密钥。

(8) DES的安全性。令人遗憾的是，随着密码分析领域的进步和计算功能的提高，DES不再被认为很安全了。有些算法可以用于减少需要检查的密钥个数，但即使用直接的强力攻击以及检查每一个可能的密钥，有的计算机仍然可以用数分钟就可破译DES。据说美国国家安全局（NSA）可以在3－15分钟内破译DES加密的消息。

如果规定2小时内破译DES加密的文件，则我们需要在两小时内检查所有可能的密钥（2^{56}个），这大约是每秒5万亿个密钥。尽管这看上去是个很大的数，但10美元专用集成电路（ASIC）芯片可以每秒测试2亿个密钥，而且很多芯片可以并行处理。可以推算花一千万美元投资在ASIC上就可以造一个能在6分钟内破译DES加密的消息的计算机。

DES不能再当做充分安全的算法了。如果一个DES加密的消息今天用超大计算机在几分钟内可以破译，那么计算机功能的飞速提高表明在将来破译DES加密将轻而易举（今天加密的消息可能那时还需要保密）。DES有一种扩展，称为DESX，被认为对穷举搜索密钥有本质的免疫力。

8.6　国际数据加密算法（IDEA）

IDEA最早的形式是由Xuejia Lai和James Massey在1990年提出的，当时叫做提议加密标准（PES）。1991年，Xuejia Lai和Massey针对差分密码分析加强了算法，称为改进的PES（IPES）。IPES的名字在1992年改为国际数据加密算法（IDEA）。IDEA以其在PGP（Pretty Good Privacy）中的使用而著称。

(1) IDEA算法。IDEA是个对称分组密码算法，它的密钥长度为128比特，分组长度为64比特，而且也像DES一样，同样的算法提供加密和解密。

IDEA有8轮加密，用到52个子密钥。每一轮用6个子密钥，最后剩下的4个用于输出变换。子密钥的产生方式如下：

首先将128比特密钥分成8个16比特密钥，这是最初的8个子密钥。然后将原来密钥的比特向左移位25比特，然后再分成8个子密钥。这种先移位然后分开的过程重复下去，直到所有52个子密钥（SK1－SK52）都产生出来。

64比特明文组首先分成四个组（B1～B4）。一轮加密由下列步骤组成（OB表示输出组）：

OB1＝B1*SK1（第1子组乘第1子密钥）
OB2＝B2＋SK2（第2子组加第2子密钥）
OB3＝B3＋SK3（第3子组加第3子密钥）
OB4＝B4*SK4（第4子组乘第4子密钥）
OB5＝OB1 XOR OB3（第1步和第3步的结果进行异或）
OB6＝OB2 XOR OB4
OB7＝OB5*SK5（第5步结果乘第5个子密钥）
OB8＝OB6＋OB7（第6步和第7步的结果相加）
OB9＝OB8*SK6（第8步结果乘第6个子密钥）
OB10＝OB7＋OB9
OB11＝OB1 XOR OB9（第1步和第9步的结果进行异或）
OB12＝OB3 XOR OB9
OB13＝OB2 XOR OB10
OB14＝OB4 XOR OB10

进入下一轮的输入为4个子组依次为OB11、OB13、OB12、OB14。

在8轮加密之后，最后4个输出组（F1～F4）要经过最后变换来产生4个密文的4个子组（C1～C4），它们重新连接在一起构成最后的64比特密文。

C1＝F1*SK49
C2＝F2＋SK50
C3＝F3＋SK51
C4＝F4*SK52
密文＝C1&C2&C3&C4

(2) **IDEA的安全性**。IDEA不仅速度约为DES的两倍，它也安全得多。用强力攻击方法，有2^{128}个可能的密钥。如果用10亿个每秒钟可以检测10亿个密钥的芯片来试图攻破IDEA加密过的消息，需要用10^{13}年，这比整个宇宙的年龄都长得多！作为一种较新的算法，人们有可能会发现比强力攻击更好的攻击方法，再结合将来功能更强大的机器，有可能会破译加密的消息。但是，在将来的漫长岁月中，IDEA看来是个非常安全的密码。

8.7 RC密码

RC密码是Ron Rivest为RSA的数据安全设计的。RC表示Ron码或Rivest密码。RC2是为快速取代DES而设计的，它更安全。这是一种分组密码，密钥大小可变，拥有一个适当的算法。RC2是一种可变密钥长度密码。但是，当用微软密码产品时，密钥长度硬性限定为40比特。当用微软增强型密码产品时，密钥长度默认值为128比特，也可以在40比特～128比特间以8比特的增幅变化。

RC4是Ron Rivest在1987年设计的，它是一种变长密钥的流密码。该算法的细节没有正式公布。该算法特别容易描述和编程。同RC2一样，微软产品支持40比特的RC4，而增强型产品允许密钥在40比特～128比特以8比特的增幅变化。

RC5是为提高速度而设计的一种分组密码。分组大小，密钥大小以及迭代次数都是可变的。特别是，密钥大小可以到达2048比特。

到目前为止，我们讨论的所有加密技术都属于对称密码学（DES、IDEA和RC密码）。我

们现在看一下非对称密码技术。

8.8 非对称（公钥）算法

公钥算法是非对称的，也就是说加密所用的密钥与解密所用的密钥不同。加密密钥也称为公钥，是用来加密消息的，但只有拥有解密密钥，也称为私钥的人才能对该消息解码。

这类算法比起传统对称密码来有许多优点。这表明收信人可以把他们的公钥让公众知道——任何人想发送给他们消息时用该算法和收信人的公钥即可做到。窃听者可能有该算法和公钥，但不能解密消息。只有收信人，用他们的私钥可以解密消息。

公钥算法的一个缺点是它们比对称算法在计算上更复杂，因此加密和解密需要更长的时间。这对一个短消息可能不明显，但对长消息（如音频和视频文件）肯定很明显。

公钥密码标准（Public-Key Cryptography Standards, PKCS）是RSA实验室和世界范围内的安全系统开发者联合为加速公钥密码开发而给出的详细说明。自从PKCS公文首次于1991年在一小组早期公钥密码技术采用者的会议后发表以来，它被广泛引用和执行。来自PKCS的贡献已成为许多正式和非正式标准的一部分，包括ANSI X9公文、PKIX、SET、S/MIME和SSL。

下面的两节描述两个流行的公钥密码算法——RSA算法和PGP混合算法。

8.9 RSA算法

用三位发明人Rivest、Shamir和Adleman的名字命名的RSA是第一个被实施的公钥密码算法，而且多年来在全世界密码学分析者的仔细研究下仍立于不败之地。它不像对称密钥算法，只要我们假定算法没缺陷，其安全性就依赖于必须试验所有可能的密钥，公钥密码算法依赖于从公钥恢复私钥在计算上是不可行的。

RSA依赖于这样一个事实：很容易将两个大素数相乘，但从结果中将它们分解却极其困难（即耗时）。分解一个数是指找出它的素因子，也就是那些需要乘在一起以产生要分解的数的那些素数。例如，

$$10 = 2 \times 5$$

$$60 = 2 \times 2 \times 3 \times 5$$

$$2^{113} - 1 = 3391 \times 23279 \times 65993 \times 1868569 \times 1066818132868207$$

(1) **RSA算法**。将两个随机选取的通常有相同长度的大素数相乘在一起：

$$N = A \times B \tag{8-4}$$

$$T = (A-1) \times (B-1) \tag{8-5}$$

然后选取第三个随机数（E）作为公钥，它必须与T没有公共因子（即互素）。则私钥（D）为

$$D = E^{-1} \bmod T \tag{8-6}$$

把一个明文组（M）加密为密文组（C）：

$$C = M^E \bmod N \tag{8-7}$$

解密为：

$$M = C^D \bmod N \tag{8-8}$$

例8.5 考虑下面RSA算法的实现：

第一个素数（A）$=37$

第二个素数（B）$=23$

则

$$N=37\times 23=851$$

$$\mathrm{T}=(37-1)\times(23-1)=36\times 22=792$$

E与792必须除1之外没有其他公共因子。

E（公钥）可以为5。

D（私钥）$=5^{-1} \bmod 792=317$

要加密字符“G”的消息（M）：

若G用7表示（字母表中的第7个字母），则$M=7$。

C（密文）$=7^5 \bmod 851=638$

解密：$M=638^{317} \bmod 851=7$。

(2) **素数**。RSA算法使用大素数。我们是否可以找到任意大的素数？在回答这个问题之前，我们先定义**素数计数函数**（Prime Counting function）。

定义8.5 *素数计数函数$\pi(n)$计算小于等于n的素数个数。*

例8.6 表8-2 给出了一些素数计数函数

表8-2 素数计数函数

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
$\pi(n)$	1	2	2	3	3	4	4	4	4	5	5	6	6	6	…

从上表中可以看出，只有当遇到下一个素数时，$\pi(n)$才会增加。函数$\pi(n)$是单调递增函数。

例8.7 是否有无穷多个素数？我们首先假设只有有限个素数。这些素数表示为$p_1, p_2, \cdots, p_n$。考虑数$m=p_1 p_2, \cdots, p_n+1$。很显然，m比上面所有的素数都大，根据假设，其必然为合数。因此它可以被上面素数中的某些整除。但是m不能被p_1整除，因为除后的余数是1。同理，m也不能被$p_i(i=2,3,\cdots,n)$整除。从而，矛盾产生了。所以假设不成立。因此，素数的个数是无穷的。

定理8.1 *素数密度定理*（Prime Density Theorem）*有下面的式子成立：*

$$\lim_{n\to\infty}\frac{\pi(n)\ln(n)}{n}=1 \tag{8-9}$$

一般地，

$$\pi(n)\approx\frac{n}{\ln(n)}$$

是一个好的估计。这也说明素数有很多。

有两种方法来生成大素数。

(1) 构造可被证明素数；

(2) 随机选择一个大奇数，然后使用素数检验。

定义8.6　素数判断问题（Primality Decision Problem）：给定一个正整数n，判断n是否是素数。

为了说明素数检验的计算复杂性，我们首先需要知道下面的定义。

定义8.7　图灵机是一个产生原始但具有普适意义计算模型的想象中的计算设备。它一般用有限状态机来表示。

定义8.8　一个算法被称为是多项式时间，如果它在最坏情况下运行时间可以表示为输入规模的多项式。任何运行时间不能用多项式来表示的算法称为**超多项式时间算法**。

定义8.9　复杂类*P*是一个可以用判定图灵机在多项式时间内完成的判定问题$D\subseteq\{0,1\}$的类。

定义8.10　复杂类*NP*是这样一个判定问题$D\subseteq\{0,1\}$的类，此类判定问题根据输入的证据（certificate或者witness）信息，可以在多项式时间内判定“yes”的回答。**复杂类*co-NP***是这样一个判定问题$D\subseteq\{0,1\}$的类，此类判定问题根据输入的证据信息，可以在多项式时间内判定“no”的回答。

素数检验算法既可以是确定算法也可以是随机算法。只有少数高效的确定素数检验函数，也就是说，在多项式时间内可以进行判定的函数。随机的素数检验算法，相较而言，更为高效。已知的高效素数检验算法意味着素数判定问题是在复杂类P中。

一个在复杂性理论中很重要的开放问题是NP和P是否相等。一般认为，这两个复杂类是不相等的。这似乎也可以通过直观观察得到：解决一个问题比验证一个答案要更复杂。

注意，如果这两个复杂类同等，那么从数学角度看，就不再存在计算安全的密码系统。

(3) **RSA的安全性**。RSA的安全性取决于黑客分解整数的能力。现在人们不断发现更快、更好的整数分解算法。目前最好的大数分解方法为数域筛选法。十年前人们感到长度不可思议的素数今天可以很容易分解。很明显一个数越长，它越难分解，因此用它的RSA就更安全。随着理论和计算机的进步，我们不得不使用越来越大的密钥。使用极其长的密钥的缺点是在加密和解密时的运算量。这只有在出现一种新的整数分解技术使得对密钥长度增长的要求大大快于使用RSA算法的计算机平均增长速度时才成问题。

在1997年，对512比特RSA密钥的评估显示这样的数可能用不到一百万美元的费用和8个月的工作量就可分解。因此，人们认为512比特密钥只能提供短期需要的不充分安全性。RSA实验室最近建议个人使用768比特大小的密钥，公司使用1024比特长的，极其重要的密钥和权威证明机构所用的根密钥对使用2048比特长的。通过定期更换用户密钥可以提高安全性，典型情况是用户密钥两年后就过期（在更换密钥时也允许选取更长的密钥）。

即使不使用巨大的密钥，RSA的加密和解密也比DES慢1000倍。这导致它不便作为单独的密码体制而广泛使用。但是，它在很多**混合密码体制**中用到，如PGP。混合体制的基本原理是用对称算法（通常为DES或IDEA）加密明文；然后将对称算法的密钥本身用公钥算法（如RSA）加密。然后将RSA加密后的密钥和对称算法加密的消息送给收信人，收信人可以用他的RSA私钥解密对称算法的密钥，再用该密钥解密消息。这比全部用RSA要快得多，而且允许每次使用不同的对称密钥，也加强了对称算法的安全性。

RSA未来的安全性只依赖于整数分解技术的进步。除非整数分解技术的效率或计算能力以天文数字速度增长，2048比特密钥在能预见的将来可以确保提供很安全的保护。例如一个可实现50 000mips（每秒百万次运算）的 Intel Paragon用现代技术需要一百万年才能分解一个2048比特密钥。

8.10 全球电子邮件加密标准

全球电子邮件加密标准（PGP）是由Phil Zimmerman创建的混合密码体制，在1991年作为一个免费软件公布在因特网上。PGP本身不是一个新算法，而是一个由一系列其他算法和精心设计的协议组成的整体。PGP本意是用于电子邮件安全的，但没有理由说它不能用于其他类型的信息传输。

PGP及其源码都可从因特网上免费得到。这说明自诞生以来PGP经历了密码分析者的大量仔细研究，他们想发现其中可以利用的缺陷。

PGP有四个模块：一个对称密码——IDEA用于消息加密；一个公钥算法——RSA用于加密IDEA的密钥和散列值；一个单向散列函数——MD5用于签名和一个随机数生成器。

用一个对称算法（IDEA）来加密消息主体的事实表明PGP生成的电子邮件在加密和解密时比其他用RSA的邮件快得多。用于IDEA模块的密钥作为一次性会话密钥每次都要随机产生，这使得PGP很安全，因为即使一个消息被破译了，所有原来的和后续的消息都保持安全。这个会话密钥再用收信人的RSA公钥加密。因为所用密钥的长度可以到达2048比特，因此是极其安全的。MD5可用来产生消息的散列码，送信人可以用自己的私钥对它签名。PGP安全性的另一个特色是用户的私钥用一个杂乱的口令短语加密，而不是用简单的口令，这使得私钥极其能抗复制攻击，即使能进入用户的计算机也没用。

用计算机产生真正的随机数是极其困难的。PGP试图利用用户敲击键盘时的反应时间来产生随机数。这表明其程序测量击键的间隔时间。初看上去这明显不是随机的，而实际上非常有效——人们击某些键需要比其他键更长的时间，思考时的停顿、犯错误和其他原因（比如对内容的熟悉程度及劳累程度等都会改变打字速度。这些度量结果并不直接被利用，而是用来触发伪随机数生成器。也有其他随机数生成方法，但要得到更好的效果会变得异常复杂。

PGP用一个很聪明但也很复杂的协议来进行密钥管理。每个用户产生并分发他们的公钥。例如，如果詹姆斯相信某个密钥属于它所说的人，则詹姆斯可以对那个人的公开密钥签名，从而詹姆斯的程序将接受来自那个人的消息。用户可以对其他用户有不同程度的信任。例如詹姆斯可能完全相信厄尔对其他人的密钥进行的签名，相当于说“对他的话我完全相信”。这说明若雷切尔有一个密钥被厄尔签过名，她想跟詹姆斯进行通信，需要将她的被签过名的密钥传送给詹姆斯。詹姆斯的程序可以辨认厄尔的签名，并被告知可以相信厄尔签名的密钥，因此可以接受雷切尔的密钥。这等效于厄尔将雷切尔介绍给了詹姆斯。

PGP允许对人们各种不同程度的信任，这一点由图8-4做了最好说明。解释如下：

(1) **第一行。**詹姆斯对厄尔、莎拉、雅各布和凯特的密钥进行了签名。詹姆斯完全相信厄尔对他人的密钥的签名，但一点也不相信莎拉，而仅对雅各布和凯特有一点相信（他对雅各布比对凯特更相信一点）。

(2) **第二行。**尽管詹姆斯没有对山姆的密钥签名，他仍然相信山姆对他人的密钥进行签名，比如对鲍勃的密钥签了名，因为他们实际见面了。因为厄尔对雷切尔的密钥签了名，雷切尔被证实了（但还不相信它对别的密钥的签名）。即使鲍勃的密钥被莎拉和雅各布签名，因为莎拉不是被信任的，而雅各布只是部分被信任，因此鲍勃仍然得不到证实。两个部分信任的人，雅各布和凯特对阿琪的密钥签了名，因此阿琪被得到了证实。

(3) **第三行。**完全被信任的山姆对哈尔的密钥签了名，因此哈尔得到了证实。路易丝的密钥有雷切尔和鲍勃的签名，但他们都不被信任，因此路易丝没有被证实。

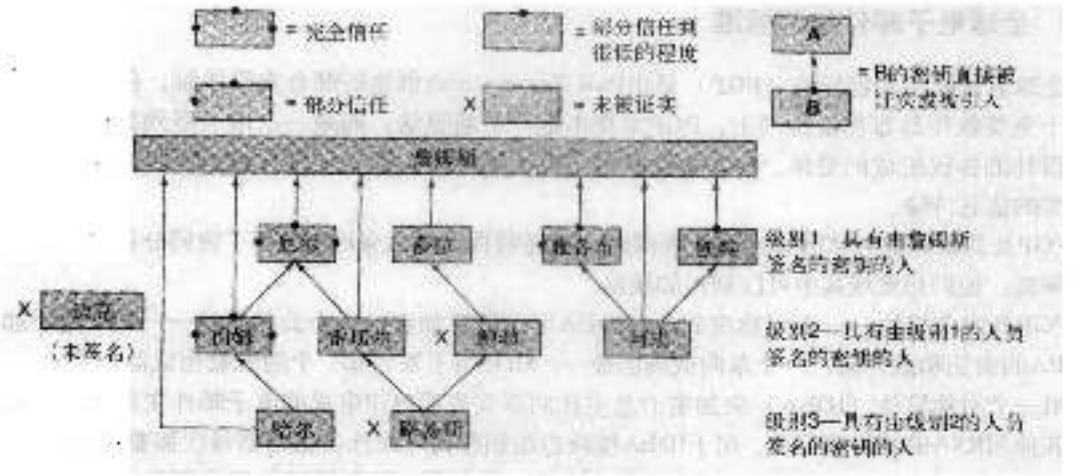


图8-4 一个PGP用户网的例子

(4) 哪里的密钥。迈克的密钥没有被鲁迪斯组中的任何人签名，或许鲁迪斯是在因特网上发现密钥的，但不知其真实性。

PGP从不阻挡用户发送或接收电子邮件，而如果密钥没得到证实，那么它会警告用户，但如何做出决定则完全看用户是否注意到了警告信息。

(5) 密钥撤回。如果一个用户泄露了私钥，则他们可以发送一个密钥撤回证书。令人遗憾的是并不是每个拥有该用户的公钥的人都将收到该证书，因为密钥通常在无组织地进行交换。另外，如果该用户不再拥有私钥，他便不能签署证书，因为它需要私钥来签名。

(6) PGP的安全性。常言道“一个链同它最弱的连接处一样强壮”，它对PGP也是正确的，如果一个用户选取40比特RSA密钥来加密会话密钥而且从不证实任何用户，那么PGP不会很安全。但是如果用2048比特的RSA密钥，而且用户有充分的警惕性，那么PGP是公众所期望得到的最接近军事等级加密水平的东西。

我们将NSA副主任的说法摘录如下：

“如果世界上所有个人计算机，估计有2亿6千万台，对一个PGP加密过的消息进行破译的话，仍然平均需要宇宙年龄1千2百万倍长的时间来完成。”

公钥密码学的一个缺点是任何人都可以用你的公钥给你发送消息，于是有必要证明该消息来自于它所说的发信人。对一个用某人的私钥加密过的消息，任何人都可以用该人的公钥进行解密。这说明如果发信人用其私钥加密一个消息，然后对得到的密文再用收信人的公钥进行加密，则收信人可以首先用他们的私钥进行解密，然后用发信人的公钥解密来恢复消息，从而证明消息来自正确的发信人。

这一过程很花时间，因此很少用。更为常用的对消息的数字签名方法是用一个称为单向散列变换的方法。

8.11 单向散列变换

单向散列函数是一个接受任意长度的消息串并输出一个较小固定长度字符串（散列值）的数学函数。这样的函数设计成不仅从一个消息的散列值中恢复该消息是困难的，而且即使告诉我们所有散列值的长度，我们也不能找到两个具有相同散列值的消息。事实上，要找到

两个具有相同的128比特散列函数值的消息，需要测试2^{64}个散列值。换句话说，一个文件的散列值是该文件很小的惟一“手印”，即使对输入串做很小的更改，都将导致散列值急剧地变化。即使在输入串中有1比特改变了，都将导致散列值大约一半数目的比特发生变化。这种性质称为**雪崩效果**（Avalanche Effect）。

H＝散列值，f＝散列函数，M＝原消息，则

$$H=f(M) \tag{8-10}$$

如果你知道M，那么很容易计算H。但是知道M和f，要计算M是不容易的，而且在计算上是不可行的。

只要有很小的碰撞可能（即两个消息散列到同一值），而且该散列很难求逆，则一个单向散列函数在密码学的许多方面都是特别有用的。

如果你对一个消息做单向散列，结果将短得多，但仍然是惟一数（至少在统计上是这样的）。这可以作为一个消息所有人的证明，而不需要泄露实际消息内容。例如不需要保留关于有版权的文件的数据库，只需要储存每个文件的散列值即可，这样一来不仅节省了储存空间，也提供了很高的安全性。如果需要证明版权，所有人可以给出原文件并证明它散列到那个值。

散列函数也可以用来证明文件没有被更改，因为在文件中即使添加一个字符都会完全改变它的散列值。

至此散列函数最通常的用途是对消息的数字签名。发信人在一个明文消息上施加单向散列变换，并用他的私钥进行加密，然后用收信人的公钥对两个输出进行再加密，并用通常方法发送出去。收信人解密密文后，他可以用发信人的公钥对散列值进行解密，然后他可以对该明文消息施加单向散列变换，将此结果同接收到的进行比较。如果两个散列值相同，则收信人不仅知道消息来自正确发信人，因为散列值是用发信人的私钥加密过的，而且确信明文消息完全可信，因为它散列到相同的值。

微软密码产品供应商支持三个散列算法：MD4、MD5和SHA。MD4和MD5都是Ron Rivest发明的。MD代表**消息摘要**（Message Digest）。两个算法都产生128比特的散列值。MD5是MD4改进的版本。SHA代表安全 散列算法，它是NIST和NSA设计的。SHA产生160比特的散列值，比MS4和MD5的要长些。人们一般认为SHA比其他算法更安全，也是推荐散列算法。

8.12 其他技术

(1) **一次一密**（One Time Pad） 一次一密是由Major Joseph Mauborgne和Gilbert Bernam于1917年发明的，而且是无条件安全的（即不可破译的）算法。一次一密的理论很简单。基垫是一个由字母组成的不重复随机串。基垫上的每个字母在加密对应明文字符时只使用一次。使用后，该基垫必须永远不再使用。只要该基垫保持安全，消息也是安全的。这是因为一个随机密钥与一个不随机消息相加后得到一个完全随机的密文，而且绝对没有什么分析或运算可以改变它。如果两个基垫都被毁坏掉了，则原来消息将永远不能被恢复。有两个主要的缺点：

首先，要生成真正随机的数是特别困难的，而一个哪怕只有一对不随机性质的基垫都在理论上是可破的。其次，因为基垫不管有多么大都不能重复使用，基垫的大小必须要与消息一样大，这对文字不成问题，但对视频则事实上是不可能的。

(2) **隐码学**（Steganography）　隐码学实际不是一种加密消息的方法，而是把它们隐藏在其他东西后面以使它们不被检测到。传统上这通过无色墨水、微胶片或在一个消息中取每一个单词的第一个字母来实现。现在我们把消息藏在图像或声音文件里。例如在一个256色图像中，如果每一字节的最后一个比特用消息的一个比特来取代，那么人的眼睛是不可分辨结果的。窃听者也许没有意识到消息正被传送。这不是密码学，但是尽管它可以蒙骗人的眼睛，计算机却可以很快检测到，而且能将原消息复制出来。

(3) **安全邮件和S/MIME**　安全多用途因特网邮件扩张系统（Secure Multipurpose Internet Mail Extension, S/MIME）是RSA数据安全公司建立在公钥密码学基础上的为发送安全邮件而研发的事实上的标准。MIME是电子邮件的工业标准，它定义了消息主体的结构。S/MIME支持电子邮件的应用，并在其结构上加入了数字签名和加密功能，以确保消息的完整性、数据来源认证和电子邮件的保密性。

当发送一个签了名的消息时，一个PKCS#7格式的签名作为附件连同消息一起发送。签名附件包含用发信人私钥签过名的原消息的散列值和签名人的证书。S/MIME也支持先用发信人私钥签名再用收信人公钥封包的消息。

8.13　椭圆曲线密码学

大部分公钥密码系统的安全性依赖于对单向函数求逆的困难性。但是对单向函数求逆的运算并不是在所有代数结构中都是同样困难的。**椭圆曲线密码学**（ECC）越来越受到重视，正是因为在这些群中还未找到一个亚指数时间的算法来对离散指数函数求逆。因此，为了攻破传统的公钥密码系统，就必须使用标准的指数时间算法。椭圆曲线密码学的根本优势就在于和传统的公钥密码系统（例如 RSA）相比，达到相同等级的安全性所需要的密钥长度更短。

记$\boldsymbol{Z}_p$表示整数集合$\{0, 1, 2, \cdots, p-1\}$，其中p是一个奇素数。如下定义一个在$\boldsymbol{Z}_p$上的椭圆曲线：

$$y^2 \equiv x^3 + ax + b(\mathrm{mod}\ p) \tag{8-11}$$

$a, b \in \boldsymbol{Z}_p$，并且$4a^3 + 27b^2 \neq 0(\mathrm{mod}\ p)$。对任意给定的$a, b \in \boldsymbol{Z}_p$，上面的方程有在$\boldsymbol{Z}_p$上的一对解$x, y$，可以如下表示：

$$\begin{aligned} &E(\boldsymbol{Z}_p)=\{(x, y) | x, y \in \boldsymbol{Z}_p \text{且} \\ &y^2 \equiv x^3+ax+b\ (\mathrm{mod}\ p) \text{且} \\ &4a^3 + 27b^2 \neq 0(\mathrm{mod}\ p)\} \end{aligned} \tag{8-12}$$

集合$E(\boldsymbol{Z}_p)$由所有满足式(8-12)的$(x,y) \in \boldsymbol{Z}_p^2$组成。除了在椭圆曲线上的点，还需要考虑一个特殊的点：无穷远点O。椭圆曲线上的一些性质如下：

（1）对所有的$\boldsymbol{P} \in E(\boldsymbol{Z}_p)$，均有$\boldsymbol{P}+\boldsymbol{O}=\boldsymbol{O}+\boldsymbol{P}$。

（2）如果$\boldsymbol{P}=(x, y) \in E(\boldsymbol{Z}_p)$，则$(x, y)+(x, -y)=\boldsymbol{O}$。点$(x, -y)$又叫做$-\boldsymbol{P}$。注意$-\boldsymbol{P}$永远是椭圆曲线上的点。

（3）记$\boldsymbol{P}=(x_1, y_1) \in E(\boldsymbol{Z}_p)$，$\boldsymbol{Q}=(x_2, y_2) \in E(\boldsymbol{Z}_p)$，并且$\boldsymbol{P} \neq -\boldsymbol{Q}$，则$\boldsymbol{P}+\boldsymbol{Q}=(x_3, y_3)$，其中

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned} \tag{8-13}$$

并且 $\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{如果} \boldsymbol{P} \neq \boldsymbol{Q} \\ \dfrac{3x_1^2 + a}{2y_1} & \text{如果} \boldsymbol{P} = \boldsymbol{Q} \end{cases}$

因此很容易计算$2\boldsymbol{P}$, $3\boldsymbol{P}$, …, $k\boldsymbol{P}$。自相加的计算可以通过一些几何技巧来更高效的实现。椭圆曲线密码学的基础在于给定$\boldsymbol{Q} = k\boldsymbol{P}$，很难找到$k$的值。我们将在下一节使用这个性质来分析椭圆曲线上的Diffie-Hellman协议。

例8.8 令p=23, a=1, b=1。椭圆曲线可以表示为定义在$\boldsymbol{Z}_{23}$上的

$$y^2 \equiv x^3+x+1$$

容易验证$4a^3 + 27b^2 \neq 0 (\text{mod } p)$。在此椭圆曲线上的点有无穷远点$O$和以下各个点：

(0, 1), (0, 22), (1, 7), (1, 16), (3, 10), (3, 13), (4,0),

(5, 4), (5, 19), (6, 4), (6, 19), (7, 11), (7, 12), (9,7), (9, 16), (11, 3), (11, 20), (12, 4),

(12, 19), (13, 7), (13, 16), (17, 3), (17, 20), (18, 3), (18, 20) (19, 5), (19, 18)。

令$\boldsymbol{P}$ = (3,10), $\boldsymbol{Q}$ = (9,7)。为了计算$\boldsymbol{P}+\boldsymbol{Q}$，我们根据式(8-13)首先确定$\lambda$：

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \boldsymbol{Z}_{23}$$

因此，

$$x_3 = 11^2 - 3 - 9 = 17 (\text{mod } 23)$$

$$y_3 = 11(3-(-6)) - 10 = 20 (\text{mod } 23)$$

所以，$\boldsymbol{P}+\boldsymbol{Q}$ = (17, 20)$\in E(\boldsymbol{Z}_{23})$。

接下来，我们计算$\boldsymbol{P}+\boldsymbol{P} = 2\boldsymbol{P}$，同样地，我们根据式(8-13)计算出

$$\lambda = \frac{3(3^2)+1}{20} = \frac{5}{20} = \frac{1}{4} = 6 \in \boldsymbol{Z}_{23}$$

$$x_3 = 6^2 - 3 - 3 \equiv 7 (\text{mod } 23)$$

$$y_3 = 11(3-7) - 10 \equiv 20 (\text{mod } 23)$$

所以，$2\boldsymbol{P}$= (7, 12)$\in E(\boldsymbol{Z}_{23})$。

例8.9 我们探讨一下椭圆曲线的**离散对数问题**。我们考虑这样一条椭圆曲线，其中p=23, a = 9, b = 17。因此，椭圆曲线可以表示为定义在$\boldsymbol{Z}_{23}$上的$y^2 \equiv x^3+9x+17$。

我们试着回答下面的问题：点$\boldsymbol{Q}$ = (4, 5)关于基$\boldsymbol{P}$ = (16, 5)的离散对数k是多少？也就是点$\boldsymbol{P}$进行自加操作多少次才能得到$\boldsymbol{Q}$？有一种暴力方式就是直接算$2\boldsymbol{P}$, $3\boldsymbol{P}$, …, $k\boldsymbol{P}$直到和$\boldsymbol{Q}$点匹配。

例如，$\boldsymbol{P}$ = (16, 5), $2\boldsymbol{P}$= (20, 20), $3\boldsymbol{P}$= (14, 14), $4\boldsymbol{P}$= (19, 20), $5\boldsymbol{P}$= (13, 10), $6\boldsymbol{P}$= (7, 3), $7\boldsymbol{P}$= (8, 7), $8\boldsymbol{P}$= (12, 17), $9\boldsymbol{P}$= (4, 5)。因此k=9。

在实际应用中，k很大，而计算这样的k需要花费很大代价。如前面所说的，椭圆曲线密码学的基础在于给定$\boldsymbol{Q}=k\boldsymbol{P}$，很难找到$k$的值。

8.14 Diffie-Hellman 密钥协商协议

Diffie-Hellman密钥协商协议（又称为指数密钥协商）是由Diffie和Hellman于1976年在具有划时代意义的论文“New Direction in Cryptography”（*IEEE Transaction on Information Theory*）中提出来的。此协议使得处于一个不安全信道上的两个用户在没有预先共享秘密的情况下建立起一个共享密钥。

协议有两个系统参数p和g。它们是公开的，并且被系统中所有用户使用。参数p是一个素数，参数g（通常称为生成元）是一个小于p的整数，并且有以下性质：对任意的$n \in [1, p-1]$，

有一个指数k使得

$$n = g^k \pmod p \tag{8-14}$$

假设Alice和Bob需要使用Diffie-Hellman密钥协商协议建立一个共享密钥。他们需要执行以下步骤：

（1）Alice生成一个随机秘密值a，Bob生成一个随机秘密值b。a和b都是从整数集合中选取。

（2）通过使用参数p, g，他们各自计算公开值：Alice计算$g^a \pmod p$，Bob计算$g^b \pmod p$。

（3）Alice和Bob交互各自计算的公开值。

（4）最后，Alice计算$g^{ab}=(g^b)^a \pmod p$，Bob计算$g^{ba}=(g^a)^b \pmod p$。因为$g^{ab}= g^{ba} = k$，所以共享密钥k就建立起来了。

例8.10　我们在椭圆曲线密码学中实现Diffie-Hellman协议。具体步骤如下：

（1）Alice和Bob先商定好一个公开值$\boldsymbol{P}$。

（2）Alice选择一个随机秘密值k_A，并根据椭圆曲线算法计算k_AP，然后公开此值。

（3）Bob选择一个随机秘密值k_B，并根据椭圆曲线算法计算k_BP，然后公开此值。

（4）Alice接收到k_BP后，根据椭圆曲线算法计算$k_A(k_BP)$。

（5）Bob接收到k_AP后，根据椭圆曲线算法计算$k_B(k_AP)$。

（6）由于$k_A(k_BP)=k_B(k_AP)$，Alice和Bob之间建立起一个共享密钥。

具体过程见图8-5。

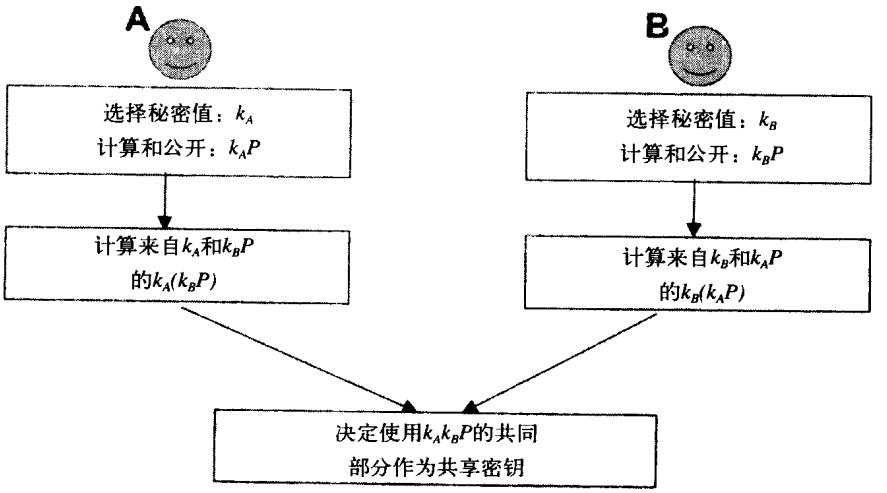


图8-5　使用椭圆曲线的Diffie-Hellman密钥协商协议

协议的安全性依赖于求离散对数的困难性假设，也就是当素数p很大时，给定两个公开的值$g^a \pmod p$和$g^b \pmod p$，输出值$k = g^{ab} \pmod p$是计算困难问题。

Diffie-Hellman密钥协商协议容易遭到**中间人攻击**。在攻击过程中，敌手Carol截获Alice的公开值，然后把他自己计算的公开值发送给Bob。当Bob发送其计算的公开值时，Carol又用他自己的公开值去替换Bob的公开值，然后发送给Alice。此时，Carol和Alice可以共享一个密钥，Carol和Bob同样也可以共享另外一个密钥。因此，Carol可以解密Alice或者Bob传输的消息，或者篡改消息，再用相应的密钥重加密后传送给另一方。造成这种攻击的原因在于Diffie-Hellman协议没有对通信双方进行认证。所以解决方案可以是使用数字签名机制，或者其他类似方案。

8.15 利用混沌理论实现安全通信

混沌函数也用于安全通信和密码学应用。这里的混沌函数指呈现混沌现象的迭代差分方程。如果我们注意到密码学更注重不可预测性而不是随机性这一事实，那么混沌函数是很好的选择，因为他们的不可预测特性。如果一个黑客截获部分序列，他将没有预测下一个是什么的信息。混沌函数的不可预测性使它们成为生成对称密码学的密钥的很好选择。

例8.11 考虑差分方程

$$x_{n+1}=ax_n(1-x_n) \tag{8-15}$$

当$a=4$时，这个函数表现为一个混沌函数，即

(1) 用连续迭代得到的值是不可预测的。

(2) 该函数对初始条件x_0极其敏感。

对任意给定的初始条件，该函数将对每一次迭代产生一个在0和1之间的值x_n。这些值可作为密钥生成的很好的候选值。在单密钥密码学中，密钥用于对消息的加密。该密钥通常为伪噪声（PN）序列。消息可以简单地同密钥进行异或运算以搅乱它。因为x_n的值为正且总是小于1，这些分数的二进制表示可作为密钥。因此，用这些随机不可预测的十进制数来产生密钥的一种方法是直接用它们的二进制表示。这些二元序列的长度将仅受限于这些十进制数的精确度，因此也可以产生很长的二元密钥。收信人必须知道初始条件来产生解密密钥。

对单密钥密码学的应用，下面两个因素需要确定：

(1) 迭代开始的值（x_0）。

(2) 用于计算的十进制数尾数的长度（以避免尾数取舍引起的错误）。

对单密钥密码学，经过一些迭代后的混沌值被转换为二进制小数，它们的前64比特被用于产生PN序列。这些初始迭代使黑客更难猜到初始条件。初始值应该在0与1之间。正确选择初始值可以很大程度地提高性能。

初始值x_0的保密性是该算法成功的关键。因为混沌函数对初始值（x_0）哪怕10^{-30}的错误都极其敏感，这说明我们可以有10^{30}个不同的初始组合。因此一个知道该混沌函数和加密算法的黑客也需要试验10^{30}个不同的初始组合。在DES算法中，黑客需要试验大约10^{19}个不同的密钥值。

8.16 量子密码学

传统密码学利用一些数学技巧阻止敌手获得密文消息，而**量子密码学**则是利用物理法则来达到对消息的机密性保护。传统密码学中不能保证绝对的安全。量子密码学却可以提供一个绝对通信安全的专用信道，使得通过此信道通信双方可以交互一个加密密钥。在量子密码学中，安全通信系统利用了海森堡测不准原理和量子纠缠原理。

一个量子密码系统的描述如下。假设密码系统包含一个发射器和接收器。发送方利用发射器发射偏振（polarisation）为0°、45°、90°、135°这四种之一的光量子。接收方在另一端利用接收器对偏振进行测量。根据量子理论，接收器可以在直线偏振（0°和90°）中，或者在对角偏振（45°和135°）中进行分辨。但是接收器不能在直线偏振和对角偏振之间进行分辨。密钥分发过程的步骤如下。发送者随机选择四种偏振中的一种用来发送光量子。对每一个接收

到的光量子，接收方随机选择一种偏振类型（直线偏振或者对角偏振）来进行测量。接收放记录测量结果，并把结果秘密保存。接下来，接收方公开所有其选择用来测量的偏振类型（但不公开测量结果），发送方通知接收方哪些选择的偏振类型是正确的。通信双方（发送方和接收方）保留所有当接收方选择正确偏振类型时的事件记录。这些记录然后转换成为比特串（1和0），这个比特串可以用来当做密钥。

由于窃听者预先不知道每一个光量子偏振的类型，所以窃听者必然会在传输中引入误差。量子机制同时防止了他/她获得两个非交换的观察量（这里是直线偏振和对角偏振）。量子信道上的通信双方通过揭示密钥的随机子集以及公开检查误差率来检测偷听。虽然他们不能阻止偷听，但是他们经常可以检测出偷听者的存在。这是因为，无论多么精巧细致的对信道的窃听行为都会被发现。当通信双方对信道的安全性不满意时，他们可以重新再次建立密钥。因此，量子密码学使得通信双方可以在一条达到完美安全性的信道上进行通信。

例8.12 假设A和B希望在一条不安全的信道上使用量子密码机制建立一个密钥。记+表示直线基（rectilinear basis），×表示对角基（diagonal basis）。直线偏振（0°和90°）用↑和→表示，对角偏振（45°和135°）用/和\表示。通信步骤如下：

A选择的随机比特	1	0	0	1	0	1	1	0
A选择的随机基	+	×	×	×	+	×	+	+
A发射的偏振	→	/	/	\	↑	\	→	↑
B选择的随机基	+	+	×	+	×	×	×	+
B用来测量的偏振	→	→	/	→	/	\	/	↑
公开的关于基的讨论								
共享的密钥	1		0			1		0

现在，A和B揭示密钥的一个子集，并且比较是否所有比特都相等。假设他们比较第2个比特和第4个比特。在此例中，所有比特都匹配，因此A和B可以使用剩下的两个比特（第1个和第3个）作为他们的密钥。如果有敌手窃听，则B所得到的密钥中的一些比特将不能匹配A所得到密钥中的相应比特。

8.17 生物加密

传统密码学使用的加密密钥是一个足够长的比特串。这些密钥，无论是公钥，还是私钥，都是密码学系统重要的部分。这些密钥由于长度比较长，所以这种使用密码系统的用户很难记住密钥。因此，这些密钥可以由一些易于记忆的口令或者PIN码经过一定的变化生成。但由于口令长度短，所以易于被穷举攻击或者偷窃。因此口令的管理在密码系统中是一个弱点。此节中我们学习生物密码学。

定义8.11 *生物统计学是这样一个自动系统。此系统使用度量的、物理的或者生理的特征或者行为的特性去识别身份，或者验证/认证某个实体所声明的身份。*

用于自动识别的生物特征的例子有：指纹、虹膜、脸部、手或者手指握法、视网膜、声音、签名、击键频率、体味和DNA。这些生物特征能不能被直接用为密码系统中的密钥呢？遗憾的是，答案是否定的：生物图像或者模板是会改变的，也就是每一个生物样本通常是不同的。而我们回顾一下，在传统密码学中，密码系统不能容忍密钥中任何一比特的错误。

一个生物统计系统通常可以输出“Yes/No”的回答，这个回答代表了信息的一个比特。

因此，生物学的作用在于对传统密码学中口令的保护。当收到“Yes”回答后，系统开启口令或者密钥。密码必须存放在一个安全地方（也称为“可信任”设备）。由于生物统计系统和业务应用只是通过一个比特来进行联系，所以此方案仍旧很容易遭到攻击。生物模板或者图像存放在某一个数据库中，而此数据库通过传统密码学方法进行加密。此时由于敌手需要先获得加密密钥，所以系统的安全性大大提高。由于密钥和生物统计数据由数据库管理者控制，所以，数据库相关的隐私问题依然存在。

很明显，由于自身经常变化，所以生物图像或者模板不能直接作为密钥使用。但是，生物图像包含的信息量非常大：例如，一个300 × 300像素的图像，每一个像素用8比特编码，所以此图像一个300 × 300 × 8=720 000比特的信息。当然，在此数据中还有不少冗余。我们试着问以下问题：

（1）有没有可能始终从这个720 000比特中抽取长度较短（比如128位）的比特？

（2）有没有可能把128比特的密钥和生物统计信息绑定在一起，使得此密钥始终可以新生成？

第一个问题的答案很明显，但是第二个问题却可以成为一个研究的新领域，称为生物加密。

定义8.12 *生物加密是通过把一个PIN或者一个密钥和某种生物统计信息绑定的过程，从而使得从所存储的的模板内既不能把密钥也不能把生物统计信息恢复。*

数字密钥（口令或者PIN等）通常是在注册时随机生成的，用户可能不知道此密钥。此密钥本身和生物统计信息完全独立，可以随时改变或者更新。当获得一个生物统计样本后，生物统计算法安全地并且持续地把密钥和生物统计信息进行绑定，从而得到一个受保护的生物加密后的模板，也被称为“私有模板”（private template）。实质上，密钥是用生物信息进行加密的。生物加密模板可以非常好地提供隐私保护，并且可以存储在数据库或者本地（智能卡、token、笔记本、手机等）。注册完成后，密钥和生物统计信息被删除。

为了进行生物解密，用户需要出示新鲜的生物信息统计样本。这可以保证当应用于一个合法的生物加密模板时，生物加密算法可以恢复出相同的密钥/口令。换句话来说，生物统计信息就是起到解密密钥的作用。在验证的最后，生物信息统计样本再次被删除。 生物加密算法的设计还需要考虑输入生物统计信息的可变性（即能够允许生物信息一定程度的改变）。换句话说，如果生物信息统计样本的差距太大，敌手就不能恢复口令。

8.18 密码分析

密码分析是不需要密钥而从密文中恢复出明文消息的科学（或黑艺术）。在密码分析中，我们总假设密码分析者掌握算法的全部情况。一次密码分析尝试称为一次攻击，它有五种主要类型：

- **强力攻击**：这种技术需要大量计算能力和很长运行时间。它包括在逻辑上测试所有可能性直到发现正确的解为止。对绝大多数加密算法来说，强力攻击是不现实的，因为有太多的可能性。
- **仅知密文攻击**：密码分析者得到的惟一信息是用同一算法加密的一些密文。
- **已知明文攻击**：在这种情况下，密码分析者不仅知道一些消息的密文，也知道它们对应的明文。
- **选择明文攻击**：密码分析者有已知明文攻击条件下的信息，但这次可以选择要加密的明文。这种攻击更有力，因为某些特殊的明文组可以被选取，它们可能会泄露关于密钥的更多信息。一种适应性选择明文攻击仅仅是密码分析者可以重复加密明文，因此根据前

面的加密结果可以修改新的输入。

- **选择密文攻击**：密码分析者用一种较新的称为差分分析的方法，这是一种交互的迭代过程。它要经过很多轮的运算，每次用到前一轮的结果，直到找到密钥。密码分析者重复选择要解密的密文，而且能得到解密后的明文。在这些条件下他们试图推导出密钥。

只有一种完全安全的算法，那就是一次一密。所有其他算法在给定无限时间和资源条件下都是可破译的。这表明，尽管在理论上可能，但需要的时间和资源使得它完全不现实。

假设一个算法是完备的，则破译它的惟一方法是试验所有可能的密钥，直到最后的明文有意义。如前面所述，这类攻击称为强力攻击。平行计算领域完全适合强力攻击，这样每一个处理器可以测试一些可能的密钥，而且它们不需要相互作用，只需要宣布结果即可。一种变得越来越流行的技术是用连接在因特网上的成千上万的计算机进行并行处理，这称为分布式计算。许多密码学家认为当使用长的密钥时强力攻击是无效的。一个用长密钥（100比特以上）的加密算法即使用今天强大的连网的计算机来攻击都需要几百万年的时间。另外，添加一个额外的密钥将使强力密码分析的代价翻倍。

关于强力攻击，有另外几个中肯的问题。如果原来的明文本身是个密码会怎么样？在那种情况下，黑客怎么知道他是否找到了正确的密钥？另外，密码分析者坐在计算机旁边观察每个被测试的密钥吗？因此我们可以设想当使用充分长的密钥，强力攻击是不可能的。

这里是一些密码分析者在攻击密文时使用的技术：

- **差分密码分析**：如前面提到的，这种技术用一种迭代过程来估算由一种迭代分组算法（如DES）产生的密码。相关的明文用同一密钥加密，然后分析其差别。这种技术被证明对DES和某些 散列函数是成功的。
- **线性密码分析**：在这种方法中，我们分析明文和密文对之间的关系，并用一种线性近似技术来确定分组密码的性能。这种技术也成功地用于对DES的分析。
- **代数攻击**：这种技术利用了分组密码的代数结构。如果这种结构存在，那么用一个密钥的一次加密可能和用两个不同密钥的两次加密得到同样的结果，因此可以缩短搜索时间。

但是，不管加密所用的算法是强还是弱，如果用来恢复明文所消耗的时间和/或资源远远超出得到消息内容所获的收益，我们就认为消息是安全的。这可能是因为花费的代价远高于消息本身的经济价值，或者是当恢复消息时内容已经过时了。

8.19　密码学中的政治因素

密码体制的广泛应用对大多数政府来说都不是很受欢迎的事——更精确地说是因为它给予个人更多的隐私，包括罪犯。许多年来，警察部门一直能在电话线上搭线偷听和截获邮件，但是在有加密的未来，那将变为不可能。

这导致了政府部门的一些奇怪的决定，特别是美国政府。在美国，密码学被归类为军用品，而含有密码体制的软件的出口受到严格控制。在1992年，软件出版商协会与美国国务院达成协议，允许出口的软件含有RSA公司的RC2和RC4加密算法，但必须限制密钥长度为40比特，而在美国境内所用的是128比特的密钥。这严重降低了它所提供的保密性。在1993年，美国国会请求国家研究理事会进一步研究美国密码学政策。在1996年的报告中，两年的工作结果提供了下述结论和建议：

- “综合考虑，密码学的广泛应用优点大于缺点。”
- “不应该用法律来禁止在美国境内制造、销售以及对任何形式加密的使用。”

- “对密码产品的出口控制应逐渐放松，但不能没有。”

在1997年，对密钥长度的限制增加到56比特。美国政府提出了几种方法来允许更强加密方法的出口，所有这些都是基于在必要情况下美国政府能得到密钥的系统的，例如*clipper* 芯片。最近有很多密码学界人士对美国政府限制密码技术发展进行了抗议。MIT的Ronald L. Rivest教授在1988年*Scientific American*（116～117页）的一篇题为“加密技术管制的现状”的文章就是这种抗议的一个例子。该问题的解决被认为是对未来电子商务最有影响的事件之一。

8.20 评注

在本节中我们将介绍密码学的简单历史。自从发明文字以来，人们曾试图以书写形式来隐藏信息。有尚存的石刻记录和草纸记录的例子都表明许多古代文明包括埃及人、希伯来人和亚述人都开发了密码体系。最早使用密码进行通信的是斯巴达人，他们（早在公元前400年）利用一种称为scytale的密码装置在军事指挥官之间进行秘密通信。

该scytale是在一个指挥棒上缠有羊皮纸，然后在羊皮纸上刻上消息。一旦从指挥棒上解下来，羊皮纸上的文字看上去无法理解。但当重新缠到另一个同样大小的指挥棒上时，就出现了原来的文字。

希腊人最早发明了转换密码。在公元前四世纪，在这方面最早的论文是希腊人Aeneas Tacticus写的，它是题为《防御工事的护卫》的著作的一部分。另一个希腊人Polybius后来发明了一种将字母编码成一对符号的方法，所用的工具叫做*波力比阿棋盘*，它包含许多以后的加密体系常见的东西。除希腊人之外，也有包括罗马人等其他文明所使用的类似的基本替代和转换的例子。波力比阿棋盘有五行五列的方格，包含字母集中的所有字母。每一个字母转换成两个数字，第一个是可以找到该字母的行，第二个是列。从而字母A变成11，而字母B变成12，依此类推。

阿拉伯人是最早明白理解密码学原理的人。他们发明并使用了替代和转换密码，并发现了利用字母频率分布进行密码分析。这样一来，大约在1412年，al-Kalkashandi在他的百科全书《Subh al-a'sha》中包括一个如果算比较初步，但也很受推崇的对许多密码体系的处理方法。他还给出怎样利用字母的频率计数进行密码分析的具体指导，并给出了例子说明这种方法。

欧洲密码学的历史可以追溯到中世纪，那时是由罗马教皇和意大利城市政府研发的。最早的密码只包括元音字母的替换（保留辅音字母不变）。包括对密码的编辑的欧洲第一本密码学指南Circa（1379）是帕尔马人Gabriele de Lavinde写的。他服务于罗马的克莱孟七世。该指南包括一个用于通信的密钥集合，并用符号代替字母和零，用许多二字母码来等价一些单词和名字。第一个简单的码字典，称为词汇手册，在几个世纪内得到逐渐扩展，成为几乎所有欧洲政府的外交通信主流方式。在1470年，Leon Battista Alberti在《Trattati in cifra》中描述了第一个密码圆盘，而Blaise de Vigernère在1586年出版的书《Traicté de chiffres》中包括了一个他发明的正方表和第一个明文密文自密钥系统的描述。

到1860年，大型码在外交通信中普遍使用，而密码系统在这种应用中变得很少见。但是，密码系统在军事通信中（除高级首领的通信外，因为俘虏和泄露等原因很难保护密码本）变得很盛行。在美国南北战争期间，联邦军队大量使用转换密码，而同盟军主要用的是Vigenère密码，也偶尔使用单字母替换。联邦政府的密码分析者破译了截获到的多数同盟军密码，而同盟军却很绝望，有时将联邦密码发表在报纸上寻求读者帮他们分析破译。

在第一次世界大战期间，双方都使用了密码系统，它们对战术通信是惟一的，尽管码系

统仍然主要在高级指挥和外交通信中使用。虽然域密码系统，如美国信号军团密码圆盘不够灵活，在战争末期一些复杂的密码系统也在高层次的通信中得以使用。它们中最著名的就是德国ADFGVX分馏密码。

在20世纪20年代，机械和电子机械的成熟带来了对电报和无线电的需求，也带来密码元件的革命——发明了转子密码机器。转子的概念已经在老式的机械密码转盘中预见到了，但是，美国人Edward Hebern发现，通过单字母表替换电子转子的一面同另一面连接物理接线，并把这样的转子进行级联，可以产生几乎达到任何复杂程度的多字母表替换。从1921年后的十年时间，Hebern构造了一系列稳定改良的转子机器，它们得到美国海军的评估。毫无疑问，这项工作使美国在第二次世界大战中在密码的使用上占绝对优势地位。几乎在Hebern在美国发明转子密码的同时，欧洲工程师如Hugo Koch（荷兰）和Arthur Scherbius（德国）独立发现了转子的观念，而且设计了历史上最著名的密码机器的先驱，即在第二次世界大战中使用的德国谜惑机。这些机器也受到了英国在第二次世界大战中使用的密码机器TYPEX的启发。

在二战期间，美国引入了M-134-C（SIGABA）密码。日本在二战中的密码机器有着有趣的历史，它们与Hebern和谜惑机有关。继Herbert Yardley之后，一个在第一次世界大战期间及之后组织和指挥了美国政府的第一个正规破译码工作组的美国人出版了《The American Black Chamber》一书，在该书中他给出了美国成功分析日本密码的详细情况，日本政府也下决心发展最好的密码机器。在这种思想指导下，日本从Hebern那里购买了转子机和商业谜惑机以及一些其他的现代机器供研究。在1930年，日本的第一个被美国密码分析人员给予代号RED的转子机服务于日本外交部门。但是，根据对Hebern转子机所产生的密码的分析经验，美国陆军信号情报队的密码分析者们成功地分析了RED密码。在1939年，日本引入了一种新的密码机器，被美国密码分析者们给予代号PURPLE，在该机器中转子用电话分级开关取代了。在第二次世界大战中密码分析者们取得的最大胜利是波兰和英国破译了谜惑机密码和美国密码分析者们破译了日本的RED、ORANGE和PURPLE密码。这些发展对盟军在二战中的战绩起了主要作用。

二战后用于支持雷达而发展起来的电子学被密码机器采用了。第一台电力密码机与转子机相差不大，只是把转子用电子置换替代了。这些电子转子机的惟一优点是它们的运转速度，但它们仍然受到机械转子机内在弱点的影响。1949年，香农发表了论文“Communication Theory of Secrecy Systems”，使得研究进入了密钥密码学时代。

计算机和电子时代给密码设计者的精心设计带来了空前的自由，这些设计如果用笔和纸完成的话会太容易出错，而用电子机械密码机又会造价太高。发展中最大的推进是分组密码的开发，开始于IBM的LUCIFER，它是DES（数据加密标准）的直接先驱。

密码学技术发展中一个划时代的工作是W. Diffie和M. E. Hellman在1976年写的论文“New Direction in Cryptograhy”。他们首次指出了即使在发送方和接收方之间不需要先传输秘密密钥，两者之间依然可以进行安全通信。从而密码学的发展进入到公钥密码学时代。

在现代密码学中可以同时用到对称和公钥密码算法。混合密码体制成功结合了两种算法，看上去安全且速度快。尽管PGP及其他复杂的协议是为因特网用户设计的，显然其背后的加密是很强的，而且可以用于很多方面。可能有些情况下有必要使用简单的算法，但有了像IDEA这样的算法所提供的安全性，绝对没有理由认为这些会不安全。椭圆曲线密码学（ECC）由Victor Miller（IBM公司）和Neil Koblitz（华盛顿大学）在1985年提出的，它可以作为实现公钥密码的另外一种选择。和类似于RSA等一些通常算法不同，在相同密钥长度的条件下，ECC所基于的离散对数问题更难解决。

量子密码学源于Stephen Weisner在20世纪70年代初期提出的“共轭编码”(Conjugate Coding)。直到1983年，这个思想才正式发表。当时，由于熟悉Weisner的理论，Bennett和Brassard已经准备提出他们自己的想法。他们在1984年提出了“BB84”——第一个量子密码协议。直到1991年，基于这个思想的第一个实验原型系统才得以实现（距离是32厘米）。最近一段时期，跨越千米距离的光纤电缆的若干系统被测试成功。

混沌理论，非线性动力系统理论的一个分支，作为密码学领域的一个新方向，其相关研究越来越多。低维度动力系统有非常复杂并且不可预测的性质，而这些性质对于信息的扩散和混淆非常有利。此方向最近的动态是由Baptista、Kocarev和Bose提出的。

从20世纪70年代早期开始，基于大量广泛的生物信息模板的身份识别系统引起了学术界、工业界以及科幻电影的广泛兴趣。当前的研究使用各种不同的生物统计信息：(1) 传统的生物统计信息（例如指纹、掌形、虹膜、视网膜）；(2) 最近的生物信息模板（例如声音、签名、掌纹和脸）；(3) 新方法（例如耳形、DNA、击键节奏、脸的不对称性和体味）。由于生物信息不是一成不变的，所以生物模板不能作为密钥。基于生物统计信息加密的研究源于1990年，并且成为当今一个热点研究方向。

在挑选锁的方面有一篇因特网上的文章说：“窃贼撬门最有效的工具一直是铁橇。”这也适用于密码分析——直接的行动往往是最有效的。用128比特的IDEA传递你的消息是绝对没问题的，但如果用一张软盘插入到加密所用的一台计算机去就能得到密钥，那么加密就失去了意义。换句话说，超强的算法也是不充分的。要使一个系统有效，必须同时有有效的管理协议。最后，Edgar Allen Poe先生说过：“人类的聪明无法编造一个人类的智慧不能解决的密码。”

8.21 小结

- 密码体制是一些用于隐藏和再现信息的算法和相关程序的总称。密码分析是分析密码体制的过程，或者检查它的完整性，或者为将来的动机破译它。攻击者是为了破译一个密码体制而进行分析的个人或系统。攻击一个密码体制的过程通常称为破译。密码分析者的工作就是发现密码体制中的弱点。
- 一个要发送的消息称为明文，该消息用密码算法进行编码，这一过程称为加密。加密后的消息称为密文，它由解密过程重新恢复到明文。
- 密钥是一个值，它使密码算法以一种特别的方式运转来产生一个特别的密文作为输出。密钥的大小通常用比特数来度量。密钥越大，算法将越安全。
- 对称算法（或单密钥算法或保密密钥算法）在加密和解密一个消息时用同一个密钥，因而得名。为了使收信人能解密消息，他们需要有同样的密钥。这带来了一个主要问题，即密钥分配问题。
- 分组密码通常在一个称为组块的比特组上进行操作，每一组都需要处理很多次。在每一轮中密钥的使用方式是惟一的。加密迭代次数越多，加密过程需要的时间也越长，最后的密文也越安全。
- 流密码在明文的操作上是每次一比特。明文以原始比特流的方式进入加密算法。分组密码用同一密钥对相同明文产生相同的密文，而流密码不是这样。在同样条件下流密码产生的密文会发生变化。
- 要确定我们需要多么高的安全程度，必须回答下列问题：

  1) 要处理的数据的价值是多少?

2) 需要安全保护的时间是多长？

3) 密码分析者/黑客拥有的资源是什么？

- 本章讨论了两种对称算法，都是分组密码。它们是数据加密标准（DES）和国际数据加密算法（IDEA）。
- 公钥算法是非对称的，也就是说用于加密一个消息的密钥与用于解密该消息的密钥是不同的。加密密钥，也称为公开密钥，是用来加密消息的，但这个消息只能被拥有称为私钥的解密密钥的人解码。RSA算法和PGP是两个很流行的公钥加密技术。
- 混合算法全球电子邮件加密标准（PGP）有四个模块：一个对称密码——IDEA用于消息加密，一个公钥算法——RSA用于加密IDEA的密钥和散列值，一个单向散列函数——MD5用于签名和一个随机数生成器。
- RSA的根据是将两个大的素数相乘很简单，但若将结果分解回去则极其困难（耗时）。分解一个数是指找出它的素因子，这些素因子乘起来得到该数。
- 单向散列函数是可以接受任意长度的数据串而输出一个短的固定长度的串（散列值）的数学函数。这些函数设计为不仅从散列值很难找到原来的消息，而且在已知所有散列值长度的条件下，要找到两个具有相同散列值的消息也是极其困难的。
- 椭圆曲线密码学（ECC）的基础在于给定mP，很难找到P的值。和RSA等一些普遍的算法不同，在相同密钥长度的条件下，ECC所基于的离散对数问题更难解决。
- Diffie-Hellman密钥协商协议（又称为指数密钥协商）使得处于一个不安全信道上的两个用户在没有预先共享秘密的情况下建立起一个共享密钥。
- 混沌函数可用于安全通信和密码应用。混沌函数主要用于基本不可预测的密钥的产生。
- 量子密码学是利用物理法则来达到对消息的机密性保护和提供一个绝对通信安全的专用信道，使得通过此信道通信双方可以交互一个加密密钥。在量子密码学中，安全通信系统利用了海森堡测不准原理和量子纠缠原理。
- 生物加密是通过把一个PIN或者一个密钥和某种生物统计信息绑定的过程，从而使得从所存储的模板内既不能把密钥也不能把生物统计信息恢复。
- 非法密码分析的尝试称为一次攻击，它的类型有五种：强力攻击、仅知密文攻击、已知明文攻击、选择明文攻击和选择密文攻击。
- 用来攻击密文的密码分析方法中，常见的手段是差分密码分析、线性密码分析和代数攻击。
- 多数政府不欢迎密码体制的广泛使用，因为它给了包括罪犯在内的个人更多的隐私而可能带来威胁。
- 密码体制的广泛使用是多数政府不欢迎的，因为它给了包括罪犯在内的个人更多的隐私而带来威胁。

想像力比知识更重要。

——爱因斯坦（1879—1955）

习题

8.1　我们想要测试加密技术字符$+x$的安全性，其中每个明文字符移动x个位置来产生密文。

(1) 假设用强力攻击，需要实验多少次才能破译这个码？

(2) 假设一个计算机需要1ms来测试一个移位，那么要破译这个码需要多长时间？

8.2 对下面的密文序列（加密方法是*character+x*）进行解密（明文是从26个英文单词中选择的）。注意，此题可以不需要计算机的协助。

JXU SQJ YI EKJ EV JXU RQW

8.3 转换密码重新安置明文中的字母而不去改变这些字母。例如一个非常简单的转换密码是*铁轨栅栏*，它把明文两两行交错然后读出密文。在一个有两个行的铁轨栅栏中，消息MERCHANT TAYLORS' SCHOOL变成

M R H N T Y O S C O L

E C A T A L R S H O

这样读出来就是：MRHNTYOSCOLECATALRSHO。

(1) 如果密码分析者想破译铁轨栅栏密码，假设知道密文长度为n，他需要试验多少不同的攻击？

(2) 给出该铁轨栅栏密码的一种解密算法。

8.4 最著名的域密码是分馏系统——即ADFGVX密码，它是德国军队在第一次世界大战中使用的密码。该系统这样命名的原因是因为它用了一个6×6阶矩阵来替换加密26个字母和10个数字，使之成为A、D、F、G、V和X的符号对。所得到的两个字母组成的密码只是中间结果，它再被写进一个长方形矩阵，再转置后产生最后发送的密码。这里给出一个用此密码将短语“Merchant Taylors”加密的例子，所用密钥为单词“Subject”。

	A	D	F	G	V	X
A	S	U	B	J	E	C
D	T	A	D	F	G	H
F	I	K	L	M	N	O
G	P	Q	R	V	W	X
V	Y	Z	0	1	2	3
X	4	5	6	7	8	9

明文：M E R C H A N T T A Y L O R S

密文：FG AV GF AX DX DD FV DA DA DD VA FF FX GF AA

这种中间密码可以根据不同的密钥放到一个转置矩阵。

C	I	P	H	E	R
I	4	5	3	2	6
F	G	A	V	G	F
A	X	D	X	D	D
F	V	D	A	D	A
D	D	V	A	F	F
F	X	G	F	A	A

因此最后的密码为：FAFDFGDDFAVXAAFGXVDXADDVGFDAFA。

(1) 如果一个密码分析者想破译ADFGVX密码，在知道密文长度n的情况下需要试验多少不同的攻击？

(2) 给出ADFGVX密码的解密算法。

8.5 考虑由XEROX公司的Ralph Merkle和斯坦福大学的Martin Hellman在1976年提出的背包加密技术。他们提出用一个背包或子集合问题为基础建立公钥密码体制。这个问题归结为确定一个给定的数是否可以表示为一个给定数列的某子集的和，更重要的是哪个集合有这样的和。

给定一个数列A，其中$A=(a_1, \cdots, a_n)$，和一个数C，背包问题就是找到$a_1, \cdots, a_n$的子集合使其和为C。

考虑下面的例子：

$$n=5,\ C=14,\ A=\ (1,\ 10,\ 5,\ 22,\ 3)$$

答案$=14=1+10+3$。

一般地，所有子集的可能和可以表示为：

$m_1a_1+m_2a_2+m_3a_3+\cdots+m_na_n$，其中每个$m_i$为0或1。

因此答案为二元向量$M=(1,\ 1,\ 0,\ 0,\ 1)$。

这样的向量共有2^n个（在本例子中$2^5=32$）。

显然并不是C的所有值都可以表示为一个子集合，而有些可能有不止一种表示方法。例如，当$A=(14,\ 28,\ 56,\ 82,\ 90,\ 132,\ 197,\ 284,\ 341,\ 455,\ 515)$时，数字515可以有三种不同的表示方式，但516不能以任何方式表示。

(1) 如果一个密码分析者想破译背包密码，他需要试验多少种不同的攻击？

(2) 给出背包密码的解密算法。

8.6 考虑一个3D密码系统：一个立方体的每一条边有3个区，因此在立方体中共有27个区。每一个区表示为字母表中的一个字母或者一个空格（27个区对应了27个字符）。因此，每一个字符有三个坐标(α, β, γ)，这三个坐标都是从$\{1, 2, 3\}$中选择的。一种可能的密码系统是："a"$=(1, 1, 1)$，"b"$=(1, 1, 2)$，$\cdots$，"z"$=(3, 3, 2)$，"<space>"$=(3, 3, 3)$。假设有一个敌手进行穷举攻击，为破解一个未知的3D密码，他需要尝试多少次？

8.7 (1) 用素数29和61生成RSA算法的密钥。

(2) 将字母"RSA"用ASCII码表示，然后用上述生成的密钥将它们加密。

(3) 接下来用素数对37和67生成密钥。步骤 (1) 还是步骤 (2) 中的密钥更安全？为什么？

8.8 在10^{90}和10^{100}之间共有多少个素数？

8.9 考虑一个由n个顶点组成的完全连通的网络。如果所有的顶点均被要求相互之间通信，则使用下面两种加密系统，各需要多少个密钥？

(1) 秘密钥加密。

(2) 公钥加密。

8.10 考虑一个定义在$\mathbf{Z}_{23}$上的椭圆曲线$y^2 \equiv x^3+x+1$

(1) 我们希望利用此曲线实现重复自加操作。对于$\boldsymbol{P}=(3, 10)$，计算值$2^{10^{74}}+1$。

(2) 计算概率$\mathrm{P}_{\mathrm{rob}}\left[\lim_{m\to\infty}\{2^m\boldsymbol{P}\}=(5,19)\right]$。

8.11 我们希望确定图像加密的Jigsaw转换的安全性。假设，原始图像（大小是$p\times p$）分成若干个子分区，每一个大小是$q\times q$，从而使得$p=mq$。这些子分区随机排列，因此得到了

Jigsaw图像。

(1) 计算对此jigsaw编码进行暴力攻击的数学复杂度。

(2) 此种情况下，密钥的比特数是多少?

上机习题

8.12 写一个程序实现DES加密。

8.13 写一个程序用IDEA加密和解密。对同样密钥大小，比较DES和IDEA在加密一个明文时所需要的运算数。

8.14 写一个对给定的数进行分解的一般程序。画出所需要的浮点运算平均次数相对于被分解数的位数的图。

8.15 写一个程序用RSA算法进行加密和解密。画出程序所执行的浮点运算数相对密钥大小的平面图。

8.16 写一段实现Diffie-Hellman协议的程序。

8.17 考虑差分方程

$$x_{n+1}=ax_n(1-x_n)$$

当$a=4$时，该函数表现为一个混沌函数。

(1) 画出通过迭代该差分函数所产生的100个值。如果开始值$x_0=0.5$，会发生什么现象?

(2) 取两个差值为Δx的初始条件（即两个不同的开始值x_{01}和x_{02}）。用上述差分方程对每个开始值迭代n次得到最后的值y_{01}和y_{02}，它们的差为Δy。对给定的Δx，画出Δy相对n的平面图。

(3) 对给定的值n（比如$n=500$），画出Δx相对Δy的平面图。

(4) 对$a=3.7$和$a=3.9$重复步骤（1）、（2）和（3）。比较并给出说明。

(5) 用下面的混沌函数给出一个基于混沌的加密程序，它为单钥加密生成密钥：

$$x_{n+1}=4x_n(1-x_n)$$

(6) 比较上面基于混沌的加密程序和IDEA用128比特长密钥的加密速度。

(7) 比较上面基于混沌的加密算法和IDEA用128比特长密钥的安全性。